Brazil project Developer Documentation

Stephen A. Uhler

Release 2.2

Copyright (c) 2006, Sun Microsystems

Table of Contents

Welcome to the Brazil server developer documentation . . . 1
- Introduction and Background . . . 1
- Architectural overview . . . 1

Package sunlabs.brazil.asterisk . . . 3
- Class AsteriskAGIHandler . . . 4
- Class AsteriskHandler.AmiStringMap . . . 8
- Class AsteriskHandler.EventItem . . . 11
- Class AsteriskHandler.Events . . . 14
- Class AsteriskHandler . . . 16

Package sunlabs.brazil.beanshell . . . 22
- Class BeanShellHandler . . . 23
- Class BeanShellServerTemplate . . . 26
- Class BeanShellTemplate . . . 30

Package sunlabs.brazil.derby . . . 33
- Class DerbyServer . . . 34

Package sunlabs.brazil.email . . . 37
- Class EmailTemplate . . . 38

Package sunlabs.brazil.filter . . . 49
- Class CopyContentFilter . . . 50
- Class ExecFilter . . . 54
- Interface Filter . . . 58
- Class FilterHandler . . . 60
- Class MD5Filter . . . 64
- Class PlainFilter . . . 67
- Class ReFilter . . . 70
- Class ReplaceFilter . . . 74
- Class SessionFilter . . . 78
- Class TemplateFilter . . . 83
- Class UrlMapFilter . . . 87
- Class UrlSessionFilter . . . 90

Package sunlabs.brazil.groovy . . . 94
- Class GroovyServerTemplate . . . 95

Package sunlabs.brazil.handler . . . 98
- Class AclSwitchHandler . . . 99
- Class BasicAuthHandler . . . 102
- Class CgiHandler . . . 109
- Class ChainSawHandler . . . 112
- Class ChownHandler . . . 116
- Class ConfigFileHandler . . . 119
- Class CookieSessionHandler . . . 122
- Class DefaultFileHandler . . . 126
- Class DeferredHandler . . . 129
- Class DialogHandler . . . 132
- Class DigestAuthHandler . . . 138

Class DirectoryHandler 145
Class DynamicConfigHandler 148
Class GenericProxyHandler 152
Class HomeDirHandler 157
Class HtmlRewriter 160
Class LogHandler 172
Class MapPage 177
Class MatchString 182
Class MultiHostHandler 185
Class MultiProxyHandler 189
Class NotFoundHandler 192
Class PollHandler 195
Class PropertiesHandler 200
Class ProxyPropertiesHandler 203
Class PublishHandler 207
Class PushHandler.Split 210
Class PushHandler 213
Class RePollHandler.Extract 217
Class RePollHandler 220
Class ReflectHandler 225
Class ResourceHandler 228
Class ResourceLimitHandler 232
Class RestartingMultiHostHandler 235
Class RestrictClientHandler 237
Class RolesHandler 240
Class SMTPHandler 244
Class SimpleSessionHandler 250
Class SupplyHandler 254
Class UrlMapperHandler 257
Class VirtualHostHandler 260
Package sunlabs.brazil.javascript 263
Class JavaScriptTemplate 264
Package sunlabs.brazil.ldap 268
Class LDAPTemplate 269
Package sunlabs.brazil.properties 273
Class ExprProps 274
Class ExprPropsHandler 278
Class PropertiesList 281
Interface SubstPropsHandler.Convert 294
Class SubstPropsHandler.Html 295
Class SubstPropsHandler.LowerCase 297
Class SubstPropsHandler.Resub 299
Class SubstPropsHandler.SubstProps 301
Class SubstPropsHandler.Url 303
Class SubstPropsHandler 305
Package sunlabs.brazil.proxy 309
Class CookieFilter 310

Class HistoryFilter 315
Class JunkBusterHandler 320
Class ProxyHandler 323
Interface UseProxy 328
Package sunlabs.brazil.python 329
Class PythonServerTemplate 330
Package sunlabs.brazil.server 333
Class ChainHandler 334
Class FileHandler 339
Interface Handler 345
Class Main 347
Class Request.HttpOutputStream 351
Class Request 354
Class Server 372
Class TestRequest 381
Package sunlabs.brazil.servlet 384
Class BServletRequest.HttpOutputStream 386
Class BServletRequest 389
Class BServletServerSocket 393
Class BServletSocket 398
Class BrazilServlet 407
Package sunlabs.brazil.session 411
Class CacheManager 412
Class PJamaSessionManager 415
Interface PropertiesCacheManager.Saveable 416
Class PropertiesCacheManager 418
Class SessionManager 422
Package sunlabs.brazil.sql 425
Class SqlTemplate 426
Package sunlabs.brazil.ssl 430
Class BasicSSLHandler 431
Class CertHandler 434
Class SslHandler 438
Package sunlabs.brazil.sunlabs 441
Class DateTemplate 442
Class DelayHandler 444
Class DigestTemplate 447
Class ExecTemplate 449
Class ExpContentTemplate 452
Class ListTemplate.MyList 455
Class ListTemplate 462
Class MiscTemplate.GlobProperties 466
Class MiscTemplate 468
Class PlainTemplate 475
Class RemoteHostTemplate 477
Class RestartHandler 479
Class SnarfTemplate 482

Class SourceTemplate 485
Class StunnelHandler 487
Class SubstAllTemplate 490
Class XmlTree.DefaultNodeName 492
Class XmlTree.IllegalXmlException 494
Class XmlTree.Node 497
Interface XmlTree.NodeName 501
Class XmlTree.XmlErrorInfo 502
Class XmlTree 504
Class XmlTreeTemplate 513
Package sunlabs.brazil.tcl 516
Class TclFilter 517
Class TclHandler 521
Class TclRePollHandler 524
Class TclServerTemplate 527
Package sunlabs.brazil.template 531
Class AddHeaderTemplate 532
Class AllowGtTemplate 534
Class BSLTemplate 536
Class ChangedTemplate 547
Class ContentTemplate 551
Class DeCommentTemplate 556
Class DebugTemplate 558
Class DirectoryTemplate 560
Class FormClientTemplate 564
Class FormTemplate 568
Class HighlightTemplate 572
Class IncludeTemplate 577
Class MacroTemplate 580
Class ModifiedTemplate 584
Class MultipartSetTemplate 587
Class NoImageTemplate 592
Class PropsTemplate 594
Class QueueTemplate.Queue 597
Class QueueTemplate.QueueItem 602
Class QueueTemplate 605
Class RedirectTemplate 611
Class RewriteContext 613
Class ScriptEvalTemplate 621
Class SetTemplate 623
Class TOCTemplate 632
Class Template 634
Class TemplateHandler 637
Interface TemplateInterface 641
Class TemplateRunner 642
Class TemplateTemplate 646
Class UrlNavBarTemplate 648

Package sunlabs.brazil.util . . . 650
Class Base64 . . . 651
Class Calculator . . . 653
Class ClockFormat . . . 657
Class ClockScan . . . 661
Class Format . . . 663
Class Glob . . . 667
Class Guid . . . 669
Class LexHTML . . . 671
Class LexML . . . 674
Interface SocketFactory . . . 681
Interface Sort.Compare . . . 683
Class Sort . . . 684
Class StringMap . . . 686
Package sunlabs.brazil.util.http . . . 694
Class HttpInputStream . . . 695
Class HttpRequest . . . 701
Class HttpSocket . . . 716
Interface HttpSocketPool . . . 719
Class HttpUtil . . . 721
Class MimeHeaders . . . 726
Package sunlabs.brazil.util.regexp . . . 732
Interface Regexp.Filter . . . 733
Class Regexp . . . 735
Class Regsub . . . 744
Package sunlabs.brazil.velocity . . . 748
Class VelocityFilter.Vrequest . . . 749
Class VelocityFilter.Vserver . . . 751
Class VelocityFilter . . . 753
Configuration Parameter Summary . . . 758
AclSwitchHandler . . . 758
AsteriskHandler . . . 758
BSLTemplate . . . 759
BasicAuthHandler . . . 759
BasicSSLHandler . . . 760
BeanShellHandler . . . 761
BeanShellServerTemplate . . . 761
BrazilServlet . . . 761
CacheManager . . . 762
CertHandler . . . 762
CgiHandler . . . 762
ChainHandler . . . 763
ChainSawHandler . . . 763
ChangedTemplate . . . 763
ChownHandler . . . 764
ConfigFileHandler . . . 764
ContentTemplate . . . 765

CookieFilter 765
CookieSessionHandler 766
CopyContentFilter 766
DeCommentTemplate 766
DebugTemplate 766
DefaultFileHandler 767
DeferredHandler 767
DelayHandler 767
DerbyServer 767
DialogHandler 768
DigestAuthHandler 768
DirectoryHandler 768
DirectoryTemplate 769
DynamicConfigHandler 770
ExecFilter 770
ExecTemplate 770
ExpContentTemplate 771
ExprPropsHandler 771
FileHandler 771
FilterHandler 772
GenericProxyHandler 772
GroovyServerTemplate 773
HighlightTemplate 773
HistoryFilter 774
HomeDirHandler 774
JavaScriptTemplate 774
JunkBusterHandler 775
LDAPTemplate 775
LogHandler 777
MD5Filter 778
MacroTemplate 778
MatchString 778
ModifiedTemplate 779
MultiHostHandler 779
MultipartSetTemplate 779
NoImageTemplate 780
NotFoundHandler 780
PlainFilter 780
PollHandler 780
PropertiesCacheManager 782
PropertiesHandler 782
PropsTemplate 782
ProxyHandler 783
ProxyPropertiesHandler 783
PublishHandler 783
PushHandler 784
PythonServerTemplate 784

QueueTemplate 784
ReFilter 785
RePollHandler 785
ReplaceFilter 786
ResourceHandler 786
ResourceLimitHandler 786
RestartHandler 787
RestrictClientHandler 787
RolesHandler 787
SMTPHandler 788
SessionFilter 788
SetTemplate 788
SimpleSessionHandler 789
SnarfTemplate 790
SqlTemplate 790
SslHandler 791
StunnelHandler 791
SubstPropsHandler 791
SubstPropsHandler.Resub 792
SupplyHandler 792
TclFilter 792
TclHandler 793
TclRePollHandler 793
TclServerTemplate 793
TemplateFilter 794
UrlMapperHandler 794
UrlNavBarTemplate 795
VelocityFilter 795
VirtualHostHandler 795
Template Tag Summary by class 797
Tag Index by tag 803

Welcome to the Brazil server developer documentation

.

Introduction and Background

The Brazil server began as an extremely small footprint http stack, originally designed to provide a URL based interface to smartcards, so the smartcards could be accessed more easily from an ordinary web browser.

Along the way it grew to provide a more flexible architecture for adding URL based interfaces to arbitrary applications and devices.

Over time it has become a modular, full function web application development system.

Supported Platforms

The Brazil Framework will run on any Java VM version 1.1 or greater, although some of the optional features available from external sources require more recent VM's.

Architectural overview

Introduction

Typical applications of the Brazil system combine one or more existing components together with custom additions, consisting of one or more implementations of:

`handlers`
: The primary extension interface, which allows for the custom handling of URL's.

`filters`
: A special type of handler, used in conjunction with the `FilterHandler` that permits content obtained from other handlers to be rewritten.

`templates`
: Are classes that work in conjunction with the `TemplateFilter` or `TemplateHandler` that allow html/XML content to be processed on a tag by tag basis.

These custom, or domain specific extensions, interoperate with the supplied Handlers, Filters, and Templates that provide generic functionality to produce complete web applications.

How to use this manual

The following pages consist of an alphabetized listing (by package) of the *JavaDoc* documentation for the Brazil system, followed by a summary of the supplied *Handlers* and their configuration options. There are different types of classes documented here; their function determines which portions of the javadoc documentation is most relevant.

- Classes whose names end in `Handler`, `Filter` or `Template` provide functionality that may be used as-is with an appropriate server configuration. The most important part of the documentation for these classes are the configuration parameters; these classes are rarely accessed directly from your Java code. Many applications of the Brazil framework have been implemented using only these classes, with no custom Java code required at all.

- Classes that define public interfaces, such as `Handler`, `Filter` and, `TemplateInterface`, as well as classes whose instances are passed as parameters to those interfaces, such as `Server`, and `RewriteContext` define the primary mechanisms for extending the system, by writing *Handlers*, *Filters*, or *Template*s. The descriptions of the public methods and fields are most useful, as they will be referenced by your code.

- Many of the core capabilities of the system are accessed by class factories. New implementations of these capabilities may be written and installed by sub-classing the default implementations of these capabilities, and installing your own implementations as described in the documentation of the provided implementations. Examples include: `SesionManager`, used to control session state, `Guid`, to install your own unique ID generator, or `SocketFactory` to change the underlying client socket protocol.

  When writing new implementations of core services, perusing the source code (which is included in the distribution) is likely to be helpful as well, as all of the some times subtle semantics of these classes are not properly captured in the javadocs.

- Classes in the `util` package, as well as classes in the `handler` or `template` packages that are not themselves *handlers*s or *templates*s are utility or helper classes, they provide commonly used capabilities; the descriptions of their public methods and fields would be most useful. Some of the classes duplicate functionality that is now included in recent versions of the JDK, but was not previously available. They are retained (and used by the core system) to maximize the portability of the Brazil system across different Java versions.

To get a better idea how the pieces fit together, see the example applications included with the distribution.

Package sunlabs.brazil.asterisk

Provide ways of integrating the the Brazil project server into the Asterisk PBX.

Asterisk is an open source PBX that runs on Linux and Solaris (see asterisk.org). in this package provide mechanisms for interacting with the Asterisk server (or servers) using the Asterisk public wire-protocols.

Class AsteriskAGIHandler

sunlabs.brazil.asterisk
Class AsteriskAGIHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.asterisk.AsteriskAGIHandler
```

All Implemented Interfaces:
Handler, Runnable, TemplateInterface

---

public class **AsteriskAGIHandler**
extends Template
implements Handler, Runnable

FAGI (fast AGI) handler and template for Asterisk. This handler/template starts a server listening on the * FAGI port. Anytime it gets an agi request from * it creates a dummy request object (sort of like TestRequest) to simulate an http request, reads a file implied by the request agi:... string, and processes the file through the template runner. The template can be used to interact with * via standard agi commands. The template output is discarded; everything is done via side effect. This allows us to interact with the ordinary template variables and namespaces. I'm still not sure how to deal with sessions, so we'll use a different one for each uniqueid in the agi request.

(Implementation notes)
Just for fun use this class for all 4 different threads: - handler/init: to get the config params and start the listening socket - The thread that listens and accepts connections - the threads that handle the incoming agi requests - the threads that do the template stuff

---

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

`AsteriskAGIHandler()`	

Method Summary

boolean	**done**(RewriteContext hr) Close the socket connection.
boolean	**init**(RewriteContext hr) Open the socket's streams at top of page.
boolean	**init**(Server server, String prefix) Start a Listening socket thread, and wait for AGI connections.
boolean	**respond**(Request request) We don't handle any "normal" requests.
void	**run**() Either start a listening socket or handle an AGI request.
void	**tag_agi**(RewriteContext hr) Provide the 'agi' tag.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

AsteriskAGIHandler

```
public AsteriskAGIHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Start a Listening socket thread, and wait for AGI connections.

Specified by:

init in interface Handler

Tags copied from interface: Handler

Parameters:

`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.

`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
                throws IOException
```

We don't handle any "normal" requests.

Specified by:

respond in interface Handler

Returns:

always false

---

init

```
public boolean init(RewriteContext hr)
```

Open the socket's streams at top of page. This will be used by the <agi> calls.

Overrides:

init in class Template

---

done

```
public boolean done(RewriteContext hr)
```

Close the socket connection.

Overrides:

done in class Template

---

tag_agi

```
public void tag_agi(RewriteContext hr)
```

Provide the 'agi' tag. <agi command="agi command"> The result is placed in "agi_result". NOTE: the thread running this instance doesn't set any of the instance variables. We get everything from "hr".

---

run

```
public void run()
```

Either start a listening socket or handle an AGI request.

Specified by:

run in interface Runnable

Class AsteriskHandler.AmiStringMap

sunlabs.brazil.asterisk
Class AsteriskHandler.AmiStringMap

```
java.lang.Object
  |
  +--java.util.Dictionary
        |
        +--sunlabs.brazil.util.StringMap
              |
              +--sunlabs.brazil.asterisk.AsteriskHandler.AmiStringMap
```

Enclosing class:
AsteriskHandler

---

public static class **AsteriskHandler.AmiStringMap**
extends StringMap

This class is built on top of the StringMap class and adds methods for reading Asterisk ManagerInterface replies.

AMI responses are either:

- Mime-style headers, followed by a blank line, or
  The header "Response: follows", followed by zero or more additional headers, followed by one or more lines of output, followed by the line:

  ```
  --END COMMAND--
  ```

  Unfortunately, the first line of the following response can have a ":" in it, making it indistinguishable from another header [they should'a added a blank line after the last header]. We need to use some heuristics to figure out if it's a header or data. grumph!

  XXX to do:
  Any time data follows, the "ActionID" key (if present) will always be the last key before the data starts. We could use that, or if the data consists of what looks like headers, then just make them headers, and don't stuff them into "data", which is sort-of what happens now.
  In the second case, all the response data is put in a header called: data:

  This is modelled after MimeHeaders.

Constructor Summary

AsteriskHandler.AmiStringMap()
AsteriskHandler.AmiStringMap(StringMap map)

Method Summary

String	**commandify**(String id) Turn an AmiMap into an asterisk command.
static String	**commandify**(StringMap map, String id)
void	**read**(HttpInputStream in)

Methods inherited from class sunlabs.brazil.util.StringMap
add, append, clear, elements, get, get, get, getKey, isEmpty, keys, put, put, put, remove, remove, remove, size, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

AsteriskHandler.AmiStringMap

public **AsteriskHandler.AmiStringMap**()

AsteriskHandler.AmiStringMap

```
public AsteriskHandler.AmiStringMap(StringMap map)
```

Method Detail

read

```
public void read(HttpInputStream in)
          throws IOException
```

---

commandify

```
public String commandify(String id)
```

Turn an AmiMap into an asterisk command. Make sure the "Action" keyword exists and is first.

Returns:

null if there wasn't a valid command

---

commandify

```
public static String commandify(StringMap map,
                                String id)
```

Class AsteriskHandler.EventItem

sunlabs.brazil.asterisk
Class AsteriskHandler.EventItem

```
java.lang.Object
  |
  +--sunlabs.brazil.asterisk.AsteriskHandler.EventItem
```

Enclosing class:
AsteriskHandler

---

public static class **AsteriskHandler.EventItem**
extends Object

Keep track of an event listener entry. [I'm not sure what this should do yet.] Each time an event arrives, we traverse the list checking for each regexp match. When a match is found, we send the event to all the listening Q's.

Constructor Summary

AsteriskHandler.EventItem(String queue, String key, String exp, String context, String serverName)

Method Summary

boolean	**addQ2Event**(String queue, String key, String exp, String context, String serverName) Add a new queue to an existing event.
boolean	**remQEvent**(String queue, String key, String exp) Remove an exp/Q pair.
boolean	**send2Q**(Dictionary event) Send an event to the q's if there is a match.
int	**size**()
String	**toString**()

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Constructor Detail

AsteriskHandler.EventItem

```
public AsteriskHandler.EventItem(String queue,
                                 String key,
                                 String exp,
                                 String context,
                                 String serverName)
```

Method Detail

addQ2Event

```
public boolean addQ2Event(String queue,
                          String key,
                          String exp,
                          String context,
                          String serverName)
```

Add a new queue to an existing event.

Parameters:

`queue` - The destination Q

`exp` - The regular expression

Returns:

true if there is now an event/q match

---

remQEvent

```
public boolean remQEvent(String queue,
                         String key,
                         String exp)
```

Remove an exp/Q pair. Return true if removed.

Parameters:

`queue` - The destination Q to remove (or all if null)

`key` - The event key to match on (null for all keys)

`exp` - The event re (or null for all re's)

Returns:

true if something was removed

---

size

```
public int size()
```

---

send2Q

```
public boolean send2Q(Dictionary event)
```

Send an event to the q's if there is a match. XXX need to think about event format. XXX if key contains '*' or '?' do globbing

---

toString

```
public String toString()
```

Overrides:
toString in class Object

Class AsteriskHandler.Events

sunlabs.brazil.asterisk
Class AsteriskHandler.Events

```
java.lang.Object
  |
  +--sunlabs.brazil.asterisk.AsteriskHandler.Events
```

Enclosing class:
AsteriskHandler

---

public static class **AsteriskHandler.Events**
extends Object

Class to manage the set of events. This implementation maintains a vector of eventItems. XXX We need to detect when the requester of an event goes away XXX without unregistering the event, so we can remove it for them.

Constructor Summary

AsteriskHandler.Events()	

Method Summary

void	**addEvent**(String queue, String key, String exp, String context, String serverName) Add an event to the current set of events.
boolean	**processEvents**(Dictionary event) Send the event to all the proper Q's.
int	**removeEvents**(String queue, String key, String exp) Remove events.
String	**toString**()

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

AsteriskHandler.Events

```
public AsteriskHandler.Events()
```

Method Detail

addEvent

```
public void addEvent(String queue,
                     String key,
                     String exp,
                     String context,
                     String serverName)
```

Add an event to the current set of events. If the event expression already exists, add the queue name to the existing event, otherwise create a new event item.

---

removeEvents

```
public int removeEvents(String queue,
                        String key,
                        String exp)
```

Remove events. If a parameter is null, it matches everything. Warning: if all three parameters are null, then all events will be removed.

---

processEvents

```
public boolean processEvents(Dictionary event)
```

Send the event to all the proper Q's.

---

toString

```
public String toString()
```

Overrides:
toString in class Object

Class AsteriskHandler

sunlabs.brazil.asterisk
Class AsteriskHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.asterisk.AsteriskHandler
```

All Implemented Interfaces:
Handler, TemplateInterface

---

public class **AsteriskHandler**
extends Template
implements Handler

Connect to asterisk manager api. There is one connection per server. This is used both to issue Commands to Asterisk via the manager interface, and to register for, and receive asynchronous event notifications.

Usage:

```
<register queue=xxx key=xxx pattern=xxx [context=xxx server=xxx]>
 - register interest in a manager event (or events).  All generated events
   will be available with <dequeue name=xxx ...>, where the "name"
   parameter of the <dequeue> matches the "queue" parameter of
   <register>.
<unregister queue=xxx key=xxx>
 - unregister interest in a previously registered event.
<enqueue name="queue" from="my-q" ...>
 - Send an command to the manager interface. "queue" is the name
   specified in the handler "queue" parameter.  The enqueue'd data
   must have an "action" key.  The result of the command is obtained by:
   <dequeue name="my-q" ...>  where "my-q" is the "from" attribute of
   the corrosponding <enqueue>
<amicommand server=xxx action=xxx ...>
 - A syncronous version of the <enqueue> ... <dequeue> above.
   See below for details.
```

Inner Class Summary	
static class	**AsteriskHandler.AmiStringMap** This class is built on top of the StringMap class and adds methods for reading Asterisk ManagerInterface replies.
static class	**AsteriskHandler.EventItem** Keep track of an event listener entry.
static class	**AsteriskHandler.Events** Class to manage the set of events.

Fields inherited from class sunlabs.brazil.template.Template
debug

Constructor Summary	
AsteriskHandler()	

Method Summary	
static void	**addEvent**(String queue, String key, String exp, String context, String serverName) Java access to adding event registrations.
boolean	**init**(Server server, String prefix) Remember the host, port, id, and password for an asterisk manager connection.
static int	**removeEvents**(String queue, String key, String exp) java access to removing event registrations.
boolean	**respond**(Request request) The handler only registers * servers.
void	**tag_amicommand**(RewriteContext hr) Issue a synchronous command to the Asterisk AMI interface.
void	**tag_asterisk**(RewriteContext hr) This only emits diagnostic information to stdout.
void	**tag_register**(RewriteContext hr) Register an event.
void	**tag_unregister**(RewriteContext hr) Unregister an event (or events).

Methods inherited from class sunlabs.brazil.template.Template
done, init

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

AsteriskHandler

public **AsteriskHandler**()

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Remember the host, port, id, and password for an asterisk manager connection. We don't require that the asterisk server actually be running at server startup time. We will try hard to reconnect to the server if it goes away.

queue
: The name of the Q to send manager commands to using the <enqueue name="queue" ...>. If not specifies, the "host:port" combination is used.

server
: The server:port to use to contact the asterisk server

userid, password
: The Manager credentials

debug=true|false
: turn on more diagnostics on the console, depending on the current server logging level. at "3", keep alives are logged, at "4", all events are logged, and at "5" even more stuff is logged.

keepalive=n
: If set, this handler will issue a keep-alive every "n" seconds to the Asterisk Server. If the keep-alive fails, a new connection will be attempted with the Asterisk server.

retry=n
: Set the number of seconds to wait before retrying a broken connection to an asterisk server (defaults to 10).

To communicate with multiple asterisk servers, each should have their own handler instance. Servers are distinguished when sending commands to them via the "queue" parameter, which should be different for each server. Alternately, the <amicommand> server parameter (which is the same as the server queue name) may be used.

When using multiple servers, only one of the server handler configurations should be listed as a template (there can only be one template instance for a given entity) which server doesn't matter, as events get registered for all servers (the "server" attribute of the response determines where it came from). As above, commands to a server are distinquished with either the "queue" or "server" attributes, depending on whether the command Qs are used directly, or the <amicommand> template is used.

To Do
Figure out where to send unregistered events, such as "reload".

Specified by:
: init in interface Handler

Tags copied from interface: Handler

Parameters:
: `server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
: `prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and

`ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
               throws IOException
```

The handler only registers * servers. No requests are handled. Use <register> in a template instead.

Specified by:

respond in interface Handler

Tags copied from interface: Handler

Parameters:

`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

tag_asterisk

```
public void tag_asterisk(RewriteContext hr)
```

This only emits diagnostic information to stdout.

---

tag_amicommand

```
public void tag_amicommand(RewriteContext hr)
```

Issue a synchronous command to the Asterisk AMI interface. This is a convenience for using <enqueue> and <dequeue> directly.

Attributes:

server

The Asterisk server's Q (required).

action:
The action to perform.
variable=value
one of the variables needed by the action. (There is a fixed list, See the manager docs for more detail).
timeout
The max time to wait for a response
prepend
What to prepend all the results too.

---

tag_register

```
public void tag_register(RewriteContext hr)
```

Register an event.

```
<register queue=xxx key=xxx exp=xxx [context=xxx server=xxx]>
  queue:  The Q name to send the results to.
  key:    The manager response key to match on.  Use "*" for all keys.
  exp:  A regular expression that matches a key value
  context: If specified, only events with this context are considered.
  server: If specified, only events from this server are considered.
        The server matches the "Server" item in the event, and is the server
  name, followed by a ":", then the port number (e.g. pbx.com:5038).
```

---

addEvent

```
public static void addEvent(String queue,
                            String key,
                            String exp,
                            String context,
                            String serverName)
```

Java access to adding event registrations.

---

tag_unregister

```
public void tag_unregister(RewriteContext hr)
```

Unregister an event (or events). Match on "queue", "exp" and "key"

---

removeEvents

```
public static int removeEvents(String queue,
                               String key,
                               String exp)
```

java access to removing event registrations.

Package sunlabs.brazil.beanshell

Provide ways of integrating the Bean Shell scripting language into the Brazil project server.

Files in this package depend upon the `beanshell.jar` files, which may be downloaded from www.beanshell.org

Class BeanShellHandler

sunlabs.brazil.beanshell
Class BeanShellHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.beanshell.BeanShellHandler
```

All Implemented Interfaces:
Handler

---

public class **BeanShellHandler**
extends Object
implements Handler

The `BeanShellHandler` permits handlers to be written in "beanshell".

The reason one would write a BeanShell handler, instead of coding the handler directly in Java, is for ease of handler development and maintanence; a BeanShell handler may be modified at will while the server is running, permitting rapid development. Once functional, the code is easily converted into a traditional handler.

The beanshell script is expected to contain both `init` and `respond` methods, which are invoked by the server just like an ordinary handler.

script
: The name of the BeanShell script to use as the handler. Normally, the script is read only once. (defaults to *prefix*.bsh)

root
: The script directory, if the script is a relative file name. If the "root" property under the prefix is not found, the global "root" property is used. If the global "root" property is not found, the current directory is used.

debug
: If this configuration parameter is present, the `script` is re-read on each request, and a new interperter is created and initialized. The call to `init` is deferred until *request* time, and called before each call to `respond`.

  This allows beanshell scripts to be debugged interatively from scratch.

Constructor Summary

`BeanShellHandler()`	

Method Summary	
boolean	**init**(Server server, String prefix) Initializes the handler.
boolean	**respond**(Request request) Responds to an HTTP request.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

BeanShellHandler

```
public BeanShellHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.
Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

respond

```
public boolean respond(Request request)
               throws IOException
```

Description copied from interface: Handler
Responds to an HTTP request.
Specified by:
respond in interface Handler
Tags copied from interface: Handler
Parameters:
`request` - The `Request` object that represents the HTTP request.
Returns:
`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.
Throws:
IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

Class BeanShellServerTemplate

sunlabs.brazil.beanshell
Class BeanShellServerTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.beanshell.BeanShellServerTemplate
```

All Implemented Interfaces:
TemplateInterface
Direct Known Subclasses:
BeanShellTemplate

---

public class **BeanShellServerTemplate**
extends Template

The `BeanShellServerTemplate` looks for one of the starting tags `<server language="beanshell">`, `<beanshell>`, or `<bsh>` in an HTML page and treats the following data up to the corresponding ending tag (`</server>`, `</beanshell>`, or `</bsh>`) as a BeanShell script to evaluate. For information on BeanShell, see www.beanshell.org.

The reason that BeanShell scripts are included in an HTML page is usually to generate dynamic, server-side content. After running this template, everything between and including the starting tag and and the ending tag is replaced by all output written to the BeanShell output stream (if any).

All BeanShell fragments within a given page are evaluated in the same BeanShell interpreter. The BeanShell interpreter actually lives for the entire duration of this `Template` object, so the user can implement persistence across requests.

The following configuration parameters are used to initialize this template.

script
: The name of the BeanShell script to evaluate when the interpreter is created. This script is only evaluated when the interpreter is created, not on every request. The variables `prefix` and `server` are set before this file is evaluated, and are references to the parameters passed to a `handler` init method.

root
: The document root, if the script is a relative file name. If the "root" property under the template prefix is not found, the global "root" property is used. If the global "root" property is not found, the current directory is used.

debug
: If this configuration parameter is present, this class replaces the starting and ending tags with comments, so the user can keep track of where the dynamically generated content is coming from by examining the comments in the resultant HTML document. By default, the starting and ending tags are completely eliminated from the HTML document rather than changed into comments.

Before evaluating each HTML document, this class sets variables in the BeanShell interpreter, which can be used to interact back with Java to do things like set the response headers:

request
: Exposes the `Request` Java object. It is set anew at each request.

prefix
: Exposes the handler prefix String.

server
: Exposes the handler `Server` object.

If the attribute `eval` is present as an attribute, all constructs off the form ${...} are substituted before processing the script.

Here's a simple example of a BeanShell template:

```
<html>
<head>
<title>BeanShell Examples</title>
</head>
<body>
The global variables <code>request</code>,
<code>prefix</code>, and <code>server</code>
are already defined.  Here's how to add a new property:
<bsh>
sum = 3 + 4 + 5;
request.props.put("sum", Integer.toString(sum));
</bsh>
And here's a way to list the properties contained in the request:
<table>
<bsh>
e = request.props.propertyNames();
while (e.hasMoreElements()) {
  name = e.nextElement();
  value = request.props.getProperty(name);
  print("<tr><td>" + name + "</td><td>"
        + value + "</td></tr>");
}
</bsh>
</table>
</body>
</html>
```

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

`BeanShellServerTemplate()`	

Method Summary	
boolean	**init**(RewriteContext hr) Called at the beginning of each HTML document that this BeanShellServerTemplate is asked to process.
void	**tag_beanshell**(RewriteContext hr) Processes the <beanshell> tag.
void	**tag_bsh**(RewriteContext hr) Processes the <bsh> tag.
void	**tag_server**(RewriteContext hr) Processes the <server> tag.

Methods inherited from class sunlabs.brazil.template.Template
done

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

BeanShellServerTemplate

public **BeanShellServerTemplate**()

Method Detail

init

public boolean **init**(RewriteContext hr)

Called at the beginning of each HTML document that this BeanShellServerTemplate is asked to process.

The first time this method is called, the initialization script is sourced into the interpreter, based on the configuration properties in the `Request`

Overrides:

init in class Template

Parameters:

`hr` - The request and associated HTML document that will be processed.

Returns:

`true` interpreter was successfully initialized `false` otherwise. About the only way that the initialization could fail would be due to an error sourcing the initialization script. If `false` is returned, an error message is logged.

---

tag_server

```
public void tag_server(RewriteContext hr)
```

Processes the `<server>` tag. Substitutes the result of evaluating the following BeanShell script into the resultant HTML document.

Parameters:

`hr` - The request and associated HTML document that will be processed.

---

tag_beanshell

```
public void tag_beanshell(RewriteContext hr)
```

Processes the `<beanshell>` tag. Substitutes the result of evaluating the following BeanShell script into the resultant HTML document.

Parameters:

`hr` - The request and associated HTML document that will be processed.

---

tag_bsh

```
public void tag_bsh(RewriteContext hr)
```

Processes the `<bsh>` tag. Substitutes the result of evaluating the following BeanShell script into the resultant HTML document.

Parameters:

`hr` - The request and associated HTML document that will be processed.

Class BeanShellTemplate

sunlabs.brazil.beanshell
Class BeanShellTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.beanshell.BeanShellServerTemplate
              |
              +--sunlabs.brazil.beanshell.BeanShellTemplate
```

All Implemented Interfaces:
TemplateInterface

---

public class **BeanShellTemplate**
extends BeanShellServerTemplate

Template to define new templates in beanshell, and use them later on in the current session (or page). This adds to the capability of the template in the following way:
The variable "tagMap", a Properties object, may be used to associate bsh code with a tag.

```
tagMap.put("mytag","do_mytag();");
```

will cause the method `do_mytag()` to be called in response to: <mytag...>. The variable "rewriteContext" will be available to access the attributes of the tag.

Field Summary	
Properties	**tagMap**

Fields inherited from class sunlabs.brazil.template.Template
debug

Constructor Summary	
BeanShellTemplate()	

Method Summary

void	**defaultTag**(RewriteContext hr) if a mapping exists for this tag, run the code

Methods inherited from class sunlabs.brazil.beanshell.BeanShellServerTemplate
init, tag_beanshell, tag_bsh, tag_server

Methods inherited from class sunlabs.brazil.template.Template
done

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

tagMap

public Properties **tagMap**

Constructor Detail

BeanShellTemplate

public **BeanShellTemplate**()

Method Detail

defaultTag

```
public void defaultTag(RewriteContext hr)
```

if a mapping exists for this tag, run the code

Package sunlabs.brazil.derby

Provide ways of integrating the Derby (aka cloudscape) database into the Brazil project server.

Files in this package depend upon the Derby database system, and require one or both of `derby.jar` and `derbynet.jar`. Although Brazil will run with versions of the Java JDK that are at least version 1.1, Derby requires at least 1.4 (or 1.3 with additional jar files).

Class DerbyServer

sunlabs.brazil.derby
Class DerbyServer

```
java.lang.Object
  |
  +--sunlabs.brazil.derby.DerbyServer
```

All Implemented Interfaces:
Handler

---

public class **DerbyServer**
extends Object
implements Handler

Template for starting derby in network server mode.
port
The port the server will listen on (defaults to 1527)
log
If true, log connections

Constructor Summary

DerbyServer()	

Method Summary

boolean	**init**(Server server, String prefix) Initializes the handler.
static boolean	**isReady**(org.apache.derby.drda.NetworkServerControl nsc)
boolean	**respond**(Request request) Responds to an HTTP request.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

DerbyServer

```
public DerbyServer()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.
Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

isReady

```
public static boolean isReady(org.apache.derby.drda.NetworkServerControl nsc)
```

---

respond

```
public boolean respond(Request request)
                throws IOException
```

Description copied from interface: Handler
Responds to an HTTP request.
Specified by:
respond in interface Handler
Tags copied from interface: Handler

Parameters:

`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

Package sunlabs.brazil.email

Provide ways of integrating Email services into the Brazil project server.

Files in this package depend upon the Java Mail API, a sample implementation may be downloaded from www.javasoft.com

Class EmailTemplate

sunlabs.brazil.email
Class EmailTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.email.EmailTemplate
```

All Implemented Interfaces:
Filter, Handler, TemplateInterface

---

public class **EmailTemplate**
extends Template
implements Filter

The `EmailTemplate` takes an HTML document with embedded "Email" markup tags in it and evaluates those special tags to produce resulting properties that allow email reading/composing functionality to be embedded within the document.

The email functionality is implemented using four tags:

- `<email>`
- `<folder>`
- `<message>`
- `<sendmail>`

The `<email>` tag handles 'state control'. The `<folder>` tag provides a means to manipulate mail folders. The `<message>` tag provides viewing capabilities for email messages. The `<sendmail>` tag allows outgoing messages to be sent via SMTP. There is an additional tag (`<forcetimeout>`) that is used mainly for debugging.

<email> TAG

The `<email>` tag provides 'state control' for the connection to the email server. It handles opening and closing the connection, as well as providing authentication parameters to the server. The general format of the `<email>` tag is as follows:

```
<email server=servername user=username password=userpassword
 action=[open,close] onError="somefile.html" connhandle=xyz>
```

The *connhandle* parameter is optional, and is used to be able to identify a unique server connection if more than one `<email>` tag is present in an HTML page (for example, an email client that connects to more than one email server to retrieve messages). The *connhandle* parameter is also used as the prefix for properties returned from this and all of the email tags. See `PropsTemplate` for more information on the `<property>` tag. If no *connhandle* is present, the properties returned are prefixed with the identifier used for the `EmailTemplate` in the configuration file. The string *prefix* will be used throughout this document

to refer to this prepended string.

Note that although *server*, *user*, and *password* are not listed as optional, they can be left out after their first use. For example, after performing an *action=open* command, the connection to the server can be closed with the following command:

```
<email action=close>
```

The *onError* parameter is also optional, but if present, will give a location that the HTML client expects to be redirected to in case of a fatal error within the processing for this tag. In an error condition, this parameter does an implicit abort on all further processing of the HTML page.

<folder> TAG

The `<folder>` tag provides functionality necessary to manipulate email folders. It allows for the selection of a particular folder, listing of all folders within a folder directory, as well as for other utility functions. The general format of the `<folder>` tag is as follows:

```
<folder dir=/xyz foldername=[INBOX,anyfoldername]
 action=[list,purge,msgcount,create,delete] onError="somefile.html"
 connhandle=xyz>
```

The *onError* parameter is optional, but if present, will give a location that the HTML client expects to be redirected to in case of a fatal error within the processing for this tag. In an error condition, this parameter does an implicit abort on all further processing of the HTML page.

The *connhandle* parameter is also optional, and its use is specified in the documentation for the `<email>` tag.

The *dir* parameter is used to establish a base directory where a user's email folders reside. For example, on some UNIX systems, this is /home/username/Mail.

The *foldername* parameter is used to specify a foldername to operate on for the various operations (except *action=list*).

The *action=list* operation will create a list all mail folders within the directory specified by *dir*. This list is returned in property: *`prefix`*`.folders` - where each folder is delimited by the `delim` character, which defaults to "/". An optional `glob` attribute restircts the folders to those matching the glob pattern.

The *action=msgcount* operation returns the total count of messages and the count of new messages in the selected folder in the following properties:

`prefix``.totalmsgcount`
`prefix``.newmsgcount`

This operation works on either a specified folder name (if provided), or the last selected folder.

The *action=create* operation creates a new folder with the name given. The name of the folder is required, but the dir parameter is optional. If not specified, the current default directory will be used as the root location of the new folder.

The *action=delete* operation deletes the named folder. The name of the folder is required, but the dir parameter is optional. If not given, the tag assumes that the folder to be deleted is located in the default directory.

<message> TAG

The `<message>` tag is the workhorse of the Brazil email suite. It provides the ability to get message header listings, and to retrieve/delete/refile individual messages and their attachments. The general format of the `<message>` tag is as follows:

```
<message action=[getheaders,getnewheaders,getmsg,deletemsg,undeletemsg,refilemsg]
 startmsg=nnn msglimit=nnn msgnum="n n n" headerlist="From, To, Date"
 foldername=xyz refilefolder=xyz onError="somefile.html" connhandle=xyz>
```

The *onError* parameter is optional, but if present, will give a location that the HTML client expects to be redirected to in case of a fatal error within the processing for this tag. In an error condition, this parameter does an implicit abort on all further processing of the HTML page.

The *connhandle* parameter is also optional, and its use is specified in the documentation for the `<email>` tag.

The *foldername* parameter is required for all operations, so that multiple open folders can be supported in seperate browser windows.

The *action=getheaders* operation uses the *headerlist* parameter to determine which header values to retrieve for each message. The header names in this list are case-sensitive. If *headerlist* isn't present, **ALL** headers will be retrieved. This operation takes the optional parameters *msglimit* and/or *startmsg*. If *msglimit* is specified without *startmsg*, the headers returned will be the last *msglimit* number.

The *action=getnewheaders* operation takes all of the same parameters as the *action=getheaders* operation, except *msglimit*, and returns values in the same properties listed below. As its name suggests, however, it only returns header information for the newly arrived messages in the folder.

The header listing information is returned in the following properties:

`prefix``.n.msgnum` - where 'n' = 1,2,3... (this is the msg number as defined by the mail server)

`prefix``.n.msgstatus` (this is either a 'U' for unread, an 'N' for new, a 'R' for read, or a 'D' for deleted)

`prefix``.n.Headername` - where 'Headername' = one of the header names returned or one of those within the *headerlist* parameter (for example *`prefix`*`.1.From`)

`prefix``.n.Headername.n` - where 'Headername' = one of the header names returned or one of those within the *headerlist* parameter and 'n' = msg number. (This format is only returned for headers that occur multiple times in an email message with unique values (such as 'Received'). Also returned in this case is a

property named: *prefix*.n.Headername.count which contains the enumerated count of the multivalue header.

prefix.n.[To,From,Cc] - This is a special case property that returns a comma seperated list of rfc822 style (user@domain) addresses for the particular header (To,From,Cc) that is used to name the property.

prefix.n.[To,From,Cc].address.n - This is a special case property that is returned for headers that are used for addresses (To,From,Cc). This property contains the rfc822 'user@domain' portion of the header. Since it is a multivalue header, (like Received), it also has a count portion (the last 'n') representing an enumeration of which address it was in the header (e.g. '0 1 2').

prefix.n.[To,From,Cc].personal.n - This is a special case property that corresponds to the 'address' property above. It contains the 'Joe A. User' portion of the address. It is also a multivalue property like address. Also returned in these cases is a property named: *prefix*.n.To.count which contains the enumerated count of the multivalue header (To,From,Cc).

prefix.n.Date - This is a special case property that returns the message sent date as a string representation.

prefix.n.Datestamp - This is a special case property that returns the message sent date as a string representation of the number of seconds elapsed since the epoch (Jan 1, 1970). This is useful in providing additional formatting via the Brazil DateTemplate.

Both of the above properties are returned for every 'Date' header requested in the *action=getheaders* operation.

prefix.n.msgindex (this contains a list of the msg numbers - for example '1 2 3 4 ...' - this is useful in iterating over the properties returned here with the BSLTemplate <foreach> tag)

The *action=getmsg* operation requires the *msgnum* parameter and returns the following properties:

prefix.msgnum (contains the message number)

prefix.n.Headername - where 'Headername' = one of the names within the *headerlist* parameter and 'n' represents the msgnum (multi-value headers like 'Received' are handled in the same manner as described above for *action=getheaders*).

prefix.n.body (contains the text of the email msg (if available))

prefix.n.msgmimetype (contains the string that specifies what MIME type the message is - useful for deciding if you'll need to convert plaintext to HTML before displaying it)

prefix.n.j.partname - where 'n' is the msgnum, and 'j' is a part number (for multipart messages/attachments) (this contains the text string name of the multipart attachment)

prefix.n.j.parturl - where 'n' is the msgnum, and 'j' is a part number (for multipart messages/attachments) (this contains the URL that is needed to retrieve the multipart attachment - this URL can be used in an <href> or <form> tag)

prefix.n.partcount (this contains a list of the part numbers and is useful for iterating over the parts properties with the BSLTemplate <foreach> tag)

The *action=deletemsg* and *action=undeletemsg* operations requires the *msgnum* parameter, which can contain either a single message number or a space seperated list (e.g. '1 2 3 4') of numbers.

The *action=refilemsg* operation requires the *msgnum* and *refilefolder* parameters. The value of *refilefolder* is a relative path from the *dir* parameter value provided in the <folder> tag. The *msgnum* parameter here can also accept a space seperated list (as in the *action=deletemsg* tag).

<sendmail> TAG

The <sendmail> tag provides the ability to send outgoing email messages using the SMTP protocol. The simplest form of the <sendmail> tag is as follows:

```
<sendmail to=user@domain cc=user@domain bcc=user@domain from=user@domain subject="test subject"
 body="msg text" onError="somefile.html" connhandle=xyz>
```

This form of the tag provides basic functionality (no attachments) to send simple messages. The complete form of the tag is as follows:

```
<sendmail to=user@domain cc=user@domain from=user@domain subject="test subject"
 body="msg text" onError="somefile.html" connhandle=xyz>
    <part body=partdata name=bodypartfilename encoding=none|qp|base64
     content-type=mimetype>
    <part ....>
</sendmail>
```

This form of the tag allows attachments to be added to outgoing messages (via the <part> tag. The <part> tag requires parameters for the actual data for each part, a filename (optional), an encoding type and a content-type (optional).

The *onError* parameter is optional, but if present, will give a location that the HTML client expects to be redirected to in case of a fatal error within the processing for this tag. In an error condition, this parameter does an implicit abort on all further processing of the HTML page.

The *connhandle* parameter is also optional, and its use is specified in the documentation for the <email> tag.

The *cc* parameter is also optional.

The *from*, *subject*, and *body* parameters should all be set to their respective values.

In the event of an error during an attempt to send a message (as denoted by a 'mailError' property being set (see Error Handling)), the following property will be set:

prefix.sendmailerror - contains the specific reason that the sending of this message failed.

<forcetimeout> TAG

This tag provides the ability to 'force' a timeout of the connection to the email server. It is generally only used for debugging purposes. The complete form of the tag is as follows:

```
<forcetimeout connhandle=xyz>
```

The *connhandle* parameter is also optional, and its use is specified in the documentation for the `<email>` tag.

After this tag is processed, calls to any of the other tags will result in errors being generated (see Error Handling below).

Error Handling

In the case of a fatal error (one that requires re-authentication with the email server) during any email operation, the following property is set (unless you used the per tag 'onError' parameter for error handling):

prefix`.fatalMailError` - this contains a string with the error message returned from the email server.

Some non-fatal errors may be returned during the processing of any of the tags and these are returned in the following property (note that these are returned regardless of the existence of the 'onError' parameter in a particular tag):

prefix`.mailError` - this contains a string with the non-fatal error message returned from the email server.

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary	
`EmailTemplate()`	

Method Summary	
boolean	**done**(RewriteContext hr) If we run off the end of the page, but there is email pending to be sent, send it anyway.
byte[]	**filter**(Request request, MimeHeaders headers, byte[] content) Filters the content generated by the wrapped Handler.
boolean	**init**(RewriteContext hr) Called before this template processes any tags.
boolean	**init**(Server server, String prefix) Initializes the handler.
boolean	**respond**(Request request) Responds to an HTTP request.
boolean	**shouldFilter**(Request request, MimeHeaders headers) Gives this Filter the chance to examine the HTTP response headers from the wrapped Handler, before the content has been retrieved.
void	**tag_email**(RewriteContext hr) Handles the <email> tag.
void	**tag_folder**(RewriteContext hr) Handles the <folder> tag.
void	**tag_forcetimeout**(RewriteContext hr) The <forcetimeout> tag will cause an immediate timeout of the connection to the email server.
void	**tag_message**(RewriteContext hr) Handles the <message> tag.
void	**tag_part**(RewriteContext hr) Handles the <part> tag.
void	**tag_sendmail**(RewriteContext hr) Handles the <sendmail> tag.
void	**tag_slash_sendmail**(RewriteContext hr) Handles the </sendmail> tag.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

EmailTemplate

```
public EmailTemplate()
```

Method Detail

shouldFilter

```
public boolean shouldFilter(Request request,
                            MimeHeaders headers)
```

Description copied from interface: Filter
Gives this `Filter` the chance to examine the HTTP response headers from the wrapped `Handler`, before the content has been retrieved.

If this `Filter` **does** want to examine and possibly rewrite the content, it should return `true`; once the content is available, the `filter` method will be invoked. For instance, if this `Filter` is only interested in rewriting "text/html" pages, it should return `false` if the "Content-Type" is "image/jpeg". If all filters return `false` for the `shouldFilter` method, the `FilterHandler` can switch to a more effient mechanism of delivering content to the client.

The MIME headers may also be modified by this `Filter`, for instance, to change the "Content-Type" of a web page. The "Content-Length" will automatically be computed.
Specified by:
shouldFilter in interface Filter
Tags copied from interface: Filter
Parameters:
`request` - The in-progress HTTP request.
`headers` - The MIME headers generated by the wrapped `Handler`.
Returns:
`true` if this filter would like to examine and possibly rewrite the content, `false` otherwise.

---

filter

```
public byte[] filter(Request request,
                     MimeHeaders headers,
                     byte[] content)
```

Description copied from interface: Filter
Filters the content generated by the wrapped `Handler`. The content may be arbitrarily rewritten by this method.

The MIME headers may also be modified by this `Filter`, for instance, to change the "Content-Type" of a web page. The "Content-Length" will automatically be computed by the `FilterHandler`.

Specified by:

filter in interface Filter

Tags copied from interface: Filter

Parameters:

`request` - The finished HTTP request.
`headers` - The MIME headers generated by the `Handler`.
`content` - The output from the `Handler` that this `Filter` may rewrite.

Returns:

The rewritten content. The `Filter` may return the original `content` unchanged. The `Filter` may return `null` to indicate that the `FilterHandler` should stop processing the request and should not return any content to the client.

---

init

```
public boolean init(RewriteContext hr)
```

Description copied from class: Template

Called before this template processes any tags.

Overrides:

init in class Template

---

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler

Initializes the handler.

Specified by:

init in interface Handler

Tags copied from interface: Handler

Parameters:

`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
               throws IOException
```

Description copied from interface: Handler
Responds to an HTTP request.
Specified by:
respond in interface Handler
Tags copied from interface: Handler
Parameters:
`request` - The `Request` object that represents the HTTP request.
Returns:
`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.
Throws:
IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

tag_forcetimeout

```
public void tag_forcetimeout(RewriteContext hr)
```

The <forcetimeout> tag will cause an immediate timeout of the connection to the email server. This tag is mainly intended for debugging.

---

tag_email

```
public void tag_email(RewriteContext hr)
```

Handles the <email> tag.

---

tag_folder

```
public void tag_folder(RewriteContext hr)
```

Handles the <folder> tag.

---

tag_message

```
public void tag_message(RewriteContext hr)
```

Handles the <message> tag.

---

tag_sendmail

```
public void tag_sendmail(RewriteContext hr)
```

Handles the <sendmail> tag.

---

tag_slash_sendmail

```
public void tag_slash_sendmail(RewriteContext hr)
```

Handles the </sendmail> tag.

---

tag_part

```
public void tag_part(RewriteContext hr)
```

Handles the <part> tag.

---

done

```
public boolean done(RewriteContext hr)
```

If we run off the end of the page, but there is email pending to be sent, send it anyway. Also, check to see if we have pending message refiles that need to happen.

Overrides:

done in class Template

Package sunlabs.brazil.filter

Filters are a type of handler, used by the `sunlabs.brazil.filter.FilterHandler` that can modify content after it has been obtained by another handler.

A filter is a special type of handler that has two additional methods, `filter` and `shouldFilter`. Normally, once a handler generates the content for a request, the content is delivered to the client, and the request is finished. Although this allows for a choice in the manner in which the content is obtained, nothing can be done to modify that content before it is transmitted to the client.

The `filterHandler` arranges for the content generated by another handler (or handlers) to be captured, allowing it to be *filtered* before it is sent to the client.

Class CopyContentFilter

sunlabs.brazil.filter
Class CopyContentFilter

```
java.lang.Object
  |
  +--sunlabs.brazil.filter.CopyContentFilter
```

All Implemented Interfaces:
Filter, Handler

---

public class **CopyContentFilter**
extends Object
implements Filter

Filter to save content (of an entire site) to a disk file. This is used to "steal" other sites. It is expected to be used in conjunction with a `GenericProxyHandler`. Only files that don't already exist on the local file system are saved. Properties:
directoryName
The root in the file system to save the content in

Constructor Summary

`CopyContentFilter``()`	

Method Summary

`byte[]`	**`filter`**`(Request request, MimeHeaders headers, byte[] content)` Grab the contents, and save as a file (if file doesn't already exist).
`boolean`	**`init`**`(Server server, String prefix)` Initializes the handler.
`boolean`	**`respond`**`(Request request)` This is the request object before the content was fetched
`boolean`	**`shouldFilter`**`(Request request, MimeHeaders headers)` Watch every document that passes by.

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

CopyContentFilter

```
public CopyContentFilter()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.
Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
```

This is the request object before the content was fetched
Specified by:
respond in interface Handler
Tags copied from interface: Handler
Parameters:
`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

shouldFilter

```
public boolean shouldFilter(Request request,
                            MimeHeaders headers)
```

Watch every document that passes by. If the HTTP rerun code is "200", plan to save the content on the local file system.

Specified by:

shouldFilter in interface Filter

Tags copied from interface: Filter

Parameters:

`request` - The in-progress HTTP request.
`headers` - The MIME headers generated by the wrapped `Handler`.

Returns:

`true` if this filter would like to examine and possibly rewrite the content, `false` otherwise.

---

filter

```
public byte[] filter(Request request,
                     MimeHeaders headers,
                     byte[] content)
```

Grab the contents, and save as a file (if file doesn't already exist). The URL is mapped into a pathname starting from `directoryName` .

Specified by:

filter in interface Filter

Tags copied from interface: Filter

Parameters:

`request` - The finished HTTP request.
`headers` - The MIME headers generated by the `Handler`.
`content` - The output from the `Handler` that this `Filter` may rewrite.

Returns:

The rewritten content. The `Filter` may return the original `content` unchanged. The `Filter` may return `null` to indicate that the `FilterHandler` should stop processing the request and

should not return any content to the client.

Class ExecFilter

sunlabs.brazil.filter
Class ExecFilter

```
java.lang.Object
  |
  +--sunlabs.brazil.filter.ExecFilter
```

All Implemented Interfaces:
Filter, Handler

---

public class **ExecFilter**
extends Object
implements Filter

Filter to Run all content through an external process filter. The content is provided as the standard input to a command, that is expected to return the new content.

The following server properties are used:

command
: The command to exec. The content is supplied as stdin, and the filtered output is expected on stdout. ${...} substitutions Are done at each filter invocation.

types
: A regular expression that matches the content types for the content we wish to filter

type
: This property is set to the content type of the content just before *command* is evaluated.

newType
: This property, if set, is used as the new content type. It is evaluated for ${...} at each conversion.

error
: If the command failed, this property will contain the error message. If the command generated output on stderr, this property will contain the output.

The following configuration will halve the size of all jpeg and gif images, using the *convert* program from ImageMagic.

```
prefix.types=image/(jpeg|gif)
prefix.command= convert -sample 50% \
    ${map.${prefix.type}}:- ${map.${prefix.type}}:-
map.image/gif=GIF
map.image/jpeg=JPEG
```

Constructor Summary	
ExecFilter()	

Method Summary	
byte[]	**filter**(Request request, MimeHeaders headers, byte[] content) Run content through filter.
boolean	**init**(Server server, String prefix) Initializes the handler.
boolean	**respond**(Request request) This is the request object before the content was fetched
boolean	**shouldFilter**(Request request, MimeHeaders headers) Only filter content types that match

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ExecFilter

```
public ExecFilter()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it

uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
```

This is the request object before the content was fetched

Specified by:

respond in interface Handler

Tags copied from interface: Handler

Parameters:

`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

shouldFilter

```
public boolean shouldFilter(Request request,
                            MimeHeaders headers)
```

Only filter content types that match

Specified by:

shouldFilter in interface Filter

Tags copied from interface: Filter

Parameters:

`request` - The in-progress HTTP request.

`headers` - The MIME headers generated by the wrapped `Handler`.

Returns:

`true` if this filter would like to examine and possibly rewrite the content, `false` otherwise.

---

filter

```
public byte[] filter(Request request,
                     MimeHeaders headers,
                     byte[] content)
```

Run content through filter. Process ${...}

Specified by:

filter in interface Filter

Tags copied from interface: Filter

Parameters:

`request` - The finished HTTP request.

`headers` - The MIME headers generated by the `Handler`.

`content` - The output from the `Handler` that this `Filter` may rewrite.

Returns:

The rewritten content. The `Filter` may return the original `content` unchanged. The `Filter` may return `null` to indicate that the `FilterHandler` should stop processing the request and should not return any content to the client.

Interface Filter

sunlabs.brazil.filter
Interface Filter
All Superinterfaces:
Handler
All Known Implementing Classes:
CopyContentFilter, EmailTemplate, ExecFilter, MD5Filter, PlainFilter, ReFilter, ReplaceFilter, SessionFilter, TemplateFilter, UrlMapFilter, UrlSessionFilter, CookieFilter, HistoryFilter, TclFilter, VelocityFilter

---

public interface **Filter**
extends Handler

The `Filter` interface is used by the `FilterHandler` to examine and dynamically rewrite the contents of web pages obtained from some source before returning that page to the client.

A chain of filters can be established in the manner of a pipeline. The `FilterHandler` sends the output of a `Filter` to the input of the next `Filter`.

The `init` and `respond` methods inherited from the `Handler` interface are called as for ordinary handlers:

`Handler.init(sunlabs.brazil.server.Server, java.lang.String)`
is called when the server starts, to obtain run-time configuration information.

`Handler.respond(sunlabs.brazil.server.Request)`
is called when the request comes in, before the request is sent to the wrapped handler. This method returns true to indicate that the request has been completely handled by this `Filter`, and no further processing filtering takes place.

Method Summary	
byte[]	**filter**(Request request, MimeHeaders headers, byte[] content) Filters the content generated by the wrapped `Handler`.
boolean	**shouldFilter**(Request request, MimeHeaders headers) Gives this `Filter` the chance to examine the HTTP response headers from the wrapped `Handler`, before the content has been retrieved.

Methods inherited from interface sunlabs.brazil.server.Handler
init, respond

Method Detail

shouldFilter

```
public boolean shouldFilter(Request request,
                            MimeHeaders headers)
```

Gives this `Filter` the chance to examine the HTTP response headers from the wrapped `Handler`, before the content has been retrieved.

If this `Filter` **does** want to examine and possibly rewrite the content, it should return `true`; once the content is available, the `filter` method will be invoked. For instance, if this `Filter` is only interested in rewriting "text/html" pages, it should return `false` if the "Content-Type" is "image/jpeg". If all filters return `false` for the `shouldFilter` method, the `FilterHandler` can switch to a more effient mechanism of delivering content to the client.

The MIME headers may also be modified by this `Filter`, for instance, to change the "Content-Type" of a web page. The "Content-Length" will automatically be computed.

Parameters:

`request` - The in-progress HTTP request.
`headers` - The MIME headers generated by the wrapped `Handler`.

Returns:

`true` if this filter would like to examine and possibly rewrite the content, `false` otherwise.

---

filter

```
public byte[] filter(Request request,
                     MimeHeaders headers,
                     byte[] content)
```

Filters the content generated by the wrapped `Handler`. The content may be arbitrarily rewritten by this method.

The MIME headers may also be modified by this `Filter`, for instance, to change the "Content-Type" of a web page. The "Content-Length" will automatically be computed by the `FilterHandler`.

Parameters:

`request` - The finished HTTP request.
`headers` - The MIME headers generated by the `Handler`.
`content` - The output from the `Handler` that this `Filter` may rewrite.

Returns:

The rewritten content. The `Filter` may return the original `content` unchanged. The `Filter` may return `null` to indicate that the `FilterHandler` should stop processing the request and should not return any content to the client.

Class FilterHandler

sunlabs.brazil.filter
Class FilterHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.filter.FilterHandler
```

All Implemented Interfaces:
Handler

---

public class **FilterHandler**
extends Object
implements Handler

The `FilterHandler` captures the output of another `Handler` and allows the ouput to be modified. One or more `Filters` may be specified to change that output before it is returned to the client.

This handler provides one of the core services now associated with the Brazil Server: the ability to dynamically rewrite web content obtained from an arbitrary source.

For instance, the `FilterHandler` can be used as a proxy for a PDA. The wrapped `Handler` would go to the web to obtain the requested pages on behalf of the PDA. Then, a `Filter` would examine all "text/html" pages and rewrite the pages so they fit into the PDA's 200 pixel wide screen. Another `Filter` would examine all requested images and dynamically dither them to reduce the wireless bandwidth consumed by the PDA.

The following configuration parameters are used to initialize this `Handler`:

prefix, suffix, glob, match
: Specify the URL that triggers this handler. (See `MatchString`).

`handler`
: The name of the `Handler` whose output will be captured and then filtered. This is called the "wrapped handler".

`filters`
: A list of `Filter` names. The filters are applied in the specified order to the output of the wrapped handler.

`exitOnError`
: If set, the server's `initFailure` will set any of the filters fail to initialize. No handler prefix is required.

A sample set of configuration parameters illustrating how to use this handler follows:

```
handler=filter
port=8081

filter.class=sunlabs.brazil.filter.FilterHandler
filter.handler=proxy
filter.filters=noimg

proxy.class=sunlabs.brazil.proxy.ProxyHandler

noimg.class=sunlabs.brazil.filter.TemplateFilter
noimg.template=sunlabs.brazil.template.NoImageTemplate
```

These parameters set up a proxy server running on port 8081. As with a normal proxy, this proxy server forwards all HTTP requests to the target machine, but it then examines all HTML pages before they are returned to the client and strips out all `<img>` tags. By applying different filters, the developer could instead build a server

- to automatically dither embedded images down to grayscale (instead of simply stripping them all out)
- to apply pattern recognition techniques to strip out only the advertisements
- to examine and change arbitrary URLs on the page
- to extract the content from an HTML page and dynamically combine it with another file to produce a different look-and-feel.

See the description under `responnd` for a more detailed explaination.

Field Summary

`Filter[]`	**`filters`**
`Handler`	**`handler`**

Constructor Summary

`FilterHandler``()`	

Method Summary

`boolean`	**`init`**`(Server server, String prefix)` Start the handler and filter classes.
`boolean`	**`respond`**`(Request request)` Responds to an HTTP request by the forwarding the request to the wrapped `Handler` and filtering the output of that `Handler` before sending the output to the client.

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

handler

```
public Handler handler
```

---

filters

```
public Filter[] filters
```

Constructor Detail

FilterHandler

```
public FilterHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Start the handler and filter classes.

Specified by:

init in interface Handler

Tags copied from interface: Handler

Parameters:

`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.

`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this

`Handler` should not be used.

---

respond

```
public boolean respond(Request request)
                throws IOException
```

Responds to an HTTP request by the forwarding the request to the wrapped `Handler` and filtering the output of that `Handler` before sending the output to the client.

At several stages, the `Filters` are given a chance to short-circuit this process:

1. Each `Filter` is given a chance to examine the request before it is sent to the `Handler` by invoking its *respond()* method. The `Filter` may decide to change the request's properties. A `Filter` may even return some content to the client now, by (calling `request.sendResponse()` and returning *true*), in which case, neither the `Handler` nor any further `Filter`s are invoked at all.
2. Assuming the *respond()* methods of all the filters returned *false* (which is normally the case), the `handler's respond()` method is called, and is expected to generate content. If no content is generated at this step, this handler returns *false*.
3. After the `Handler` has generated the response headers, but before it has generated any content, each `Filter` is asked if it would be interested in filtering the content. If no `Filter` is, then the subsequent content from the `Handler` will be sent directly to the client.
4. On the other hand, if any `Filter` **is** interested in filtering the content, then the output of the `Handler` will be sent to each of the interested `Filter`s in order. The output of each interested `Filter` is sent to the next one; the output of the final `Filter` is sent to the client.
5. At this point, any one of the invoked `Filter`s can decide to reject the content completely, instead of rewriting it.

See `Filter` for a description of how to cause filters to implement the various behaviors defined above.

Specified by:

respond in interface Handler

Parameters:

`request` - The HTTP request to be forwarded to one of the sub-servers.

Returns:

`true` if the request was handled and content was generated, `false` otherwise.

Throws:

IOException - if there was an I/O error while sending the response to the client.

Class MD5Filter

sunlabs.brazil.filter
Class MD5Filter

```
java.lang.Object
  |
  +--sunlabs.brazil.filter.MD5Filter
```

All Implemented Interfaces:
Filter, Handler

---

public class **MD5Filter**
extends Object
implements Filter

Filter to compute the MD5 checksum of the content, and generate the appropriate "Content-MD5" http header. As md5 checksum generation can be expensive, care should be taken as to which types of content are digested.

The following server properties are used:
prefix, suffix, glob, match
Specify the URLs that trigger this filter (See `MatchString`).

Constructor Summary

`MD5Filter()`	

Method Summary

`byte[]`	**`filter`**`(Request request, MimeHeaders headers, byte[] content)` Compute digest, add to header.
`boolean`	**`init`**`(Server server, String prefix)` Make sure MD5 is available in this VM, or don't start.
`boolean`	**`respond`**`(Request request)` This is the request object before the content was fetched.
`boolean`	**`shouldFilter`**`(Request request, MimeHeaders headers)` Only filter url's that match.

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

MD5Filter

```
public MD5Filter()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Make sure MD5 is available in this VM, or don't start.

Specified by:

init in interface Handler

Tags copied from interface: Handler

Parameters:

`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.

`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
```

This is the request object before the content was fetched.

Specified by:

respond in interface Handler

Tags copied from interface: Handler

Parameters:

`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

shouldFilter

```
public boolean shouldFilter(Request request,
                            MimeHeaders headers)
```

Only filter url's that match.

Specified by:

shouldFilter in interface Filter

Tags copied from interface: Filter

Parameters:

`request` - The in-progress HTTP request.

`headers` - The MIME headers generated by the wrapped `Handler`.

Returns:

`true` if this filter would like to examine and possibly rewrite the content, `false` otherwise.

---

filter

```
public byte[] filter(Request request,
                     MimeHeaders headers,
                     byte[] content)
```

Compute digest, add to header.

Specified by:

filter in interface Filter

Tags copied from interface: Filter

Parameters:

`request` - The finished HTTP request.

`headers` - The MIME headers generated by the `Handler`.

`content` - The output from the `Handler` that this `Filter` may rewrite.

Returns:

The rewritten content. The `Filter` may return the original `content` unchanged. The `Filter` may return `null` to indicate that the `FilterHandler` should stop processing the request and should not return any content to the client.

Class PlainFilter

sunlabs.brazil.filter
Class PlainFilter

```
java.lang.Object
  |
  +--sunlabs.brazil.filter.PlainFilter
```

All Implemented Interfaces:
Filter, Handler

---

public class **PlainFilter**
extends Object
implements Filter

Filter to turn text/plain into html. This allows plain text to be processed by other filters that only deal with html.

The following server properties are used:
template
The string to use as an html template. The string should contain a single "%", which is replaced by the text/plain content. The default stuff the content between <pre>...</pre>.

Constructor Summary

PlainFilter()	

Method Summary

byte[]	**filter**(Request request, MimeHeaders headers, byte[] content) Wrap html around text/plain, converting it to html.
boolean	**init**(Server server, String prefix) Initializes the handler.
boolean	**respond**(Request request) This is the request object before the content was fetched
boolean	**shouldFilter**(Request request, MimeHeaders headers) Only filter text/plain documents

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

PlainFilter

```
public PlainFilter()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.
Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
```

This is the request object before the content was fetched
Specified by:
respond in interface Handler
Tags copied from interface: Handler
Parameters:
`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

shouldFilter

```
public boolean shouldFilter(Request request,
                            MimeHeaders headers)
```

Only filter text/plain documents

Specified by:

shouldFilter in interface Filter

Tags copied from interface: Filter

Parameters:

`request` - The in-progress HTTP request.

`headers` - The MIME headers generated by the wrapped `Handler`.

Returns:

`true` if this filter would like to examine and possibly rewrite the content, `false` otherwise.

---

filter

```
public byte[] filter(Request request,
                     MimeHeaders headers,
                     byte[] content)
```

Wrap html around text/plain, converting it to html. Change the content-type to text/html.

Specified by:

filter in interface Filter

Tags copied from interface: Filter

Parameters:

`request` - The finished HTTP request.

`headers` - The MIME headers generated by the `Handler`.

`content` - The output from the `Handler` that this `Filter` may rewrite.

Returns:

The rewritten content. The `Filter` may return the original `content` unchanged. The `Filter` may return `null` to indicate that the `FilterHandler` should stop processing the request and should not return any content to the client.

Class ReFilter

sunlabs.brazil.filter
Class ReFilter

```
java.lang.Object
  |
  +--sunlabs.brazil.filter.ReFilter
```

All Implemented Interfaces:
Filter, Handler

---

public class **ReFilter**
extends Object
implements Filter

Filter to replace text content via a reqular expression substitution. See `Regexp.sub(java.lang.String, java.lang.String)` and `Regexp.subAll(java.lang.String, java.lang.String)`.

Note: The regular expression processing should be consolodated with the RePollHandler, and the tag_extract... processing.

The following server properties are used:
re
The regular expression to match the content
sub
The replacement expression. If not specified, the matched content is deleted.
oneOnly
If set, only replace the first match. by default, all matches are replaced.
noCase
If set, case-insensitive matchins is performed.

Constructor Summary	
ReFilter()	

Method Summary	
byte[]	**filter**(Request request, MimeHeaders headers, byte[] content) If the content matches the regular expression, do the substitution.
boolean	**init**(Server server, String prefix) Initializes the handler.
boolean	**respond**(Request request) This is the request object before the content was fetched
boolean	**shouldFilter**(Request request, MimeHeaders headers) Only filter text documents

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ReFilter

```
public ReFilter()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
```

This is the request object before the content was fetched
Specified by:
respond in interface Handler
Tags copied from interface: Handler
Parameters:
`request` - The `Request` object that represents the HTTP request.
Returns:
`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.
Throws:
IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

shouldFilter

```
public boolean shouldFilter(Request request,
                            MimeHeaders headers)
```

Only filter text documents
Specified by:
shouldFilter in interface Filter
Tags copied from interface: Filter
Parameters:
`request` - The in-progress HTTP request.
`headers` - The MIME headers generated by the wrapped `Handler`.
Returns:
`true` if this filter would like to examine and possibly rewrite the content, `false` otherwise.

---

filter

```
public byte[] filter(Request request,
                     MimeHeaders headers,
                     byte[] content)
```

If the content matches the regular expression, do the substitution. Otherwise, return the original content un-changed.

Specified by:

filter in interface Filter

Tags copied from interface: Filter

Parameters:

`request` - The finished HTTP request.

`headers` - The MIME headers generated by the `Handler`.

`content` - The output from the `Handler` that this `Filter` may rewrite.

Returns:

The rewritten content. The `Filter` may return the original `content` unchanged. The `Filter` may return `null` to indicate that the `FilterHandler` should stop processing the request and should not return any content to the client.

Class ReplaceFilter

sunlabs.brazil.filter
Class ReplaceFilter

```
java.lang.Object
  |
  +--sunlabs.brazil.filter.ReplaceFilter
```

All Implemented Interfaces:
Filter, Handler

---

public class **ReplaceFilter**
extends Object
implements Filter

Filter to replace current content with a static form, or template. This should be called the TemplateFilter, but that name's already taken. The content is replaced by the template lock-stock-and-barrel. Typically, an upstream filter has extracted the relevent parts of the content, and a down-stream filter will combine it with the template. The filename to use for the template is computed at each request, so it may be modified dynamically. The following server properties are used:

type
: Text subtype of content to filter. Defaults to "html"

debug
: If set, the template is re-read each time. Otherwise a cached copy is used.

fileName
: Name of the file to use as the form or template. The file is searched for as a *Resource* if not found in the filesystem.

root
: The document root used to find the template file. If not found, "root" with no prefix is used instead.

Constructor Summary	
ReplaceFilter()	

Method Summary	
byte[]	**filter**(Request request, MimeHeaders headers, byte[] content) Grab the template file name, Read in the file, and deliver it as content.
boolean	**init**(Server server, String prefix) Initializes the handler.
boolean	**respond**(Request request) This is the request object before the content was fetched
boolean	**shouldFilter**(Request request, MimeHeaders headers) Only replace text documents

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ReplaceFilter

```
public ReplaceFilter()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
```

This is the request object before the content was fetched
Specified by:
respond in interface Handler
Tags copied from interface: Handler
Parameters:
`request` - The `Request` object that represents the HTTP request.
Returns:
`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.
Throws:
IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

shouldFilter

```
public boolean shouldFilter(Request request,
                            MimeHeaders headers)
```

Only replace text documents
Specified by:
shouldFilter in interface Filter
Tags copied from interface: Filter
Parameters:
`request` - The in-progress HTTP request.
`headers` - The MIME headers generated by the wrapped `Handler`.
Returns:
`true` if this filter would like to examine and possibly rewrite the content, `false` otherwise.

---

filter

```
public byte[] filter(Request request,
                     MimeHeaders headers,
                     byte[] content)
```

Grab the template file name, Read in the file, and deliver it as content.

Specified by:

filter in interface Filter

Tags copied from interface: Filter

Parameters:

`request` - The finished HTTP request.

`headers` - The MIME headers generated by the `Handler`.

`content` - The output from the `Handler` that this `Filter` may rewrite.

Returns:

The rewritten content. The `Filter` may return the original `content` unchanged. The `Filter` may return `null` to indicate that the `FilterHandler` should stop processing the request and should not return any content to the client.

Class SessionFilter

sunlabs.brazil.filter
Class SessionFilter

```
java.lang.Object
  |
  +--sunlabs.brazil.filter.SessionFilter
```

All Implemented Interfaces:
Filter, Handler

---

public class **SessionFilter**
extends Object
implements Filter

Filter to manage browser sessions using browser cookies or URL rewriting as needed. This should be used as the last filter in the filter chain. It attempts to use browser cookies. If they don't work, it rewrites the URL's instead, tacking the session info onto the end of the URL.

This Filter works by first examining the request as a *handler*. If the request contains an ID, either in the "browser cookie" or written into the URL, the session ID is extracted. In the id-in-the-url case, the ID is removed from the URL. When called later as a filter, the SessionFilter rewrites all relevent URL's in the page to incorporate the ID.

If an ID can't be found either in the *cookie* or URL, a couple the session creation sequence starts. First, the browser is send a "set-cookie" request along with a redirect that contains the cookie value encoded into the redirected URL. When the browser follows the redirect, the request is examined to se if the cookie value was sent. If so, the browser is redirected back to the original URL, and normal "cookie" processing takes place. If no cookie is found, the browser is redirected back to the original URL, modified to embed the ID into it, and normal URL session rewriting takes place.

The following server properties are used:
cookie
: The name of the cookie to use (defaults to "cookie"). If the name is "none", then no cookies are used. Instead, session rewriting will occur for every session.
session
: The name of the request property that the Session ID will be stored in, to be passed to downstream handler. The default value is "SessionID". If the session property is set, and not empty, then no processing is done.
persist
: If set, cookies persist across browser sessions. If cookies are disabled, no persistence is available.
cookiePrefix
: The URL prefix for which the cookie applies. Defaults to "/".
suffix
: A regular expression that matches url suffix we process. Defaults to `html|xml|txt`.
The Following request properties are set:

gotCookie
: An id was retrieved out of a cookie header

UrlID
: Set to the string tacked onto the end of each URL, if session ID's are managed by URL rewriting. If cookies are used, this is set to the empty string.

Field Summary

String	**cookieName**
String	**encoding**
boolean	**persist**
String	**redirectToken**
String	**session**
String	**urlSep**

Constructor Summary

SessionFilter()	

Method Summary

byte[]	**filter**(Request request, MimeHeaders headers, byte[] content) Rewrite all the url's, adding the session id to the end
boolean	**init**(Server server, String propsPrefix) Initializes the handler.
boolean	**respond**(Request request) This is called by the filterHandler before the content generation step.
boolean	**shouldFilter**(Request request, MimeHeaders headers) We have the results, only filter if html and we're rewriting

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

session

`public String` **`session`**

---

cookieName

`public String` **`cookieName`**

---

urlSep

`public String` **`urlSep`**

---

redirectToken

`public String` **`redirectToken`**

---

encoding

`public String` **`encoding`**

---

persist

`public boolean` **`persist`**

Constructor Detail

SessionFilter

`public` **`SessionFilter`**`()`

Method Detail

init

```
public boolean init(Server server,
                    String propsPrefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.
Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
               throws IOException
```

This is called by the filterHandler before the content generation step. It is responsible for extracting the session information, then (if required) restoring the URL's to their original form. It tries relatively hard to use cookies if they are available through a series or redirects.
Specified by:
respond in interface Handler
Tags copied from interface: Handler
Parameters:
`request` - The `Request` object that represents the HTTP request.
Returns:
`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.
Throws:
IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

shouldFilter

```
public boolean shouldFilter(Request request,
                            MimeHeaders headers)
```

We have the results, only filter if html and we're rewriting

Specified by:

shouldFilter in interface Filter

Tags copied from interface: Filter

Parameters:

`request` - The in-progress HTTP request.

`headers` - The MIME headers generated by the wrapped `Handler`.

Returns:

`true` if this filter would like to examine and possibly rewrite the content, `false` otherwise.

---

filter

```
public byte[] filter(Request request,
                     MimeHeaders headers,
                     byte[] content)
```

Rewrite all the url's, adding the session id to the end

Specified by:

filter in interface Filter

Tags copied from interface: Filter

Parameters:

`request` - The finished HTTP request.

`headers` - The MIME headers generated by the `Handler`.

`content` - The output from the `Handler` that this `Filter` may rewrite.

Returns:

The rewritten content. The `Filter` may return the original `content` unchanged. The `Filter` may return `null` to indicate that the `FilterHandler` should stop processing the request and should not return any content to the client.

Class TemplateFilter

sunlabs.brazil.filter
Class TemplateFilter

```
java.lang.Object
  |
  +--sunlabs.brazil.filter.TemplateFilter
```

All Implemented Interfaces:
Filter, Handler

---

public class **TemplateFilter**
extends Object
implements Filter

The `TemplateFilter` sends HTML content through an Html/XML parser to a set of `Template`s. Each Html/XML tag may dynamically invoke a Java method present in the `Template`s. The dynamically-generated content from evaluating the Html/XML tags is returned to the caller.

The following configuration parameters are used to initialize this `Filter`.

`templates`
: A list of template names. For each name in the list, the property *`name`*`.class` is examined to determine which class to use for each template. Then `name` is used as the prefix for other template specific properties if any. If *`name`*`.class` does not exist, then `name` is assumed to be the class name, and there are no template specific properties for the template. Methods in the template classes will be invoked to process the XML/HTML tags present in the content.

`session`
: The request property that contains the session ID. If no "session" property is found with the supplied prefix, then the global "session" property is used instead. The default value is "SessionID".

`subtype`
: Restrict this template to only handle specified text sub-types. defaults to the empty string, which implies any text sub-type.

`encoding`
: The charset encoding to use to represent the content as text. If none is specified, the default encoding is used.

`outputEncoding`
: The character encoding to use to interpret the template results. If no "outputEncoding" is specified, then "encoding" is used. Once template processing is complete, the results are converted into a byte stream for transmission to the next filter, using "outputEncoding", if specified. If not specified then the default encoding is used.

`tagPrefix`
: If specified, all tag names defined for each template class are prefixed with *tagPrefix*. This parameter only takes effect if the `tagPrefix` option is not specified for an individual template.

The `TemplateHandler` class is similar, but not identical to running a `FilterHandler` with the `FileHandler` and the `TemplateFilter`. The differences between the two should be resolved.

Note: The `templates` property accepts a list of class names or *tokens* that could be used to represent class names. If class *names* are used, all template classes share the TemplateHandler's properties prefix.

Constructor Summary

`TemplateFilter``()`	

Method Summary

`byte[]`	**`filter`**`(Request request, MimeHeaders headers, byte[] content)` Evaluates the content as html/XML tags, if the file is (or has now been converted to) "text/html".
`boolean`	**`init`**`(Server server, String prefix)` Initializes the handler.
`boolean`	**`respond`**`(Request request)` No action before request is made
`boolean`	**`shouldFilter`**`(Request request, MimeHeaders headers)` Filters all HTML files, or files that are likely to be html files, specifically, those whose "Content-Type" starts with "text/".

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

TemplateFilter

`public` **`TemplateFilter`**`()`

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.
Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
```

No action before request is made
Specified by:
respond in interface Handler
Tags copied from interface: Handler
Parameters:
`request` - The `Request` object that represents the HTTP request.
Returns:
`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.
Throws:
IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

shouldFilter

```
public boolean shouldFilter(Request request,
                            MimeHeaders headers)
```

Filters all HTML files, or files that are likely to be html files, specifically, those whose "Content-Type"

starts with "text/".

Specified by:

shouldFilter in interface Filter

Tags copied from interface: Filter

Parameters:

`request` - The in-progress HTTP request.

`headers` - The MIME headers generated by the wrapped `Handler`.

Returns:

`true` if this filter would like to examine and possibly rewrite the content, `false` otherwise.

---

filter

```
public byte[] filter(Request request,
                     MimeHeaders headers,
                     byte[] content)
```

Evaluates the content as html/XML tags, if the file is (or has now been converted to) "text/html".

Specified by:

filter in interface Filter

Tags copied from interface: Filter

Parameters:

`request` - The finished HTTP request.

`headers` - The MIME headers generated by the `Handler`.

`content` - The output from the `Handler` that this `Filter` may rewrite.

Returns:

The rewritten content. The `Filter` may return the original `content` unchanged. The `Filter` may return `null` to indicate that the `FilterHandler` should stop processing the request and should not return any content to the client.

Class UrlMapFilter

sunlabs.brazil.filter
Class UrlMapFilter

```
java.lang.Object
  |
  +--sunlabs.brazil.filter.UrlMapFilter
```

All Implemented Interfaces:
Filter, Handler

---

public class **UrlMapFilter**
extends Object
implements Filter

Filter to Map url's from any proxied content. The `MultiProxyHandler` is used to virtually mount other web sites into the local document root. In the process, it rewrites all the local url's found in the proxied content to point to the locally mounted version.

This filter examimes content derived from non-proxied sources for absolute url's to proxied sites, and rewrites them to point to the virtual mount point on the local machine instead of directly to the mounted site.

Constructor Summary

`UrlMapFilter()`	

Method Summary

`byte[]`	**`filter`**`(Request request, MimeHeaders headers, byte[] content)` Rewrite all absolute links, if there are any left
`boolean`	**`init`**`(Server server, String prefix)` Initializes the handler.
`boolean`	**`respond`**`(Request request)` This is the request object before the content was fetched
`boolean`	**`shouldFilter`**`(Request request, MimeHeaders headers)` Only filter text documents if the MultiProxyHandler was called

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

UrlMapFilter

```
public UrlMapFilter()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.
Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
```

This is the request object before the content was fetched
Specified by:
respond in interface Handler
Tags copied from interface: Handler
Parameters:
`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

shouldFilter

```
public boolean shouldFilter(Request request,
                            MimeHeaders headers)
```

Only filter text documents if the MultiProxyHandler was called

Specified by:

shouldFilter in interface Filter

Tags copied from interface: Filter

Parameters:

`request` - The in-progress HTTP request.

`headers` - The MIME headers generated by the wrapped `Handler`.

Returns:

`true` if this filter would like to examine and possibly rewrite the content, `false` otherwise.

---

filter

```
public byte[] filter(Request request,
                     MimeHeaders headers,
                     byte[] content)
```

Rewrite all absolute links, if there are any left

Specified by:

filter in interface Filter

Tags copied from interface: Filter

Parameters:

`request` - The finished HTTP request.

`headers` - The MIME headers generated by the `Handler`.

`content` - The output from the `Handler` that this `Filter` may rewrite.

Returns:

The rewritten content. The `Filter` may return the original `content` unchanged. The `Filter` may return `null` to indicate that the `FilterHandler` should stop processing the request and should not return any content to the client.

Class UrlSessionFilter

sunlabs.brazil.filter
Class UrlSessionFilter

```
java.lang.Object
  |
  +--sunlabs.brazil.filter.UrlSessionFilter
```

All Implemented Interfaces:
Filter, Handler

---

public class **UrlSessionFilter**
extends Object
implements Filter

Sample filter to use url's instead of cookies for sessions. When html files are delivered, all URL's back to this host are changed to add in the session information. When requests are made, the session info is stripped off the URL, which is passed to the rest of the handlers.

Note: This fiter has been superceded by the `SessionFilter`. It is included for illustrative purposes only.

Field Summary

String	**session**

Constructor Summary

UrlSessionFilter()	

Method Summary	
byte[]	**filter**(Request request, MimeHeaders headers, byte[] content) Rewrite all the url's, adding the session id to the end
boolean	**init**(Server server, String propsPrefix) Initializes the handler.
boolean	**respond**(Request request) Extract the cookie out of the URL, rewriting the url as needed.
boolean	**shouldFilter**(Request request, MimeHeaders headers) We have the results, only filter if html

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

session

```
public String session
```

Constructor Detail

UrlSessionFilter

```
public UrlSessionFilter()
```

Method Detail

init

```
public boolean init(Server server,
                    String propsPrefix)
```

Description copied from interface: Handler
Initializes the handler.

Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.
Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
```

Extract the cookie out of the URL, rewriting the url as needed. Add the session info at the end of the url: /a/b.html -> /a/b.html$xxxx where xxx is the session This gets called before the original request is made.
Specified by:
respond in interface Handler
Tags copied from interface: Handler
Parameters:
`request` - The `Request` object that represents the HTTP request.
Returns:
`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.
Throws:
IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

shouldFilter

```
public boolean shouldFilter(Request request,
                            MimeHeaders headers)
```

We have the results, only filter if html
Specified by:
shouldFilter in interface Filter

Tags copied from interface: Filter
Parameters:
`request` - The in-progress HTTP request.
`headers` - The MIME headers generated by the wrapped `Handler`.
Returns:
`true` if this filter would like to examine and possibly rewrite the content, `false` otherwise.

---

filter

```
public byte[] filter(Request request,
                     MimeHeaders headers,
                     byte[] content)
```

Rewrite all the url's, adding the session id to the end
Specified by:
filter in interface Filter
Tags copied from interface: Filter
Parameters:
`request` - The finished HTTP request.
`headers` - The MIME headers generated by the `Handler`.
`content` - The output from the `Handler` that this `Filter` may rewrite.
Returns:
The rewritten content. The `Filter` may return the original `content` unchanged. The `Filter` may return `null` to indicate that the `FilterHandler` should stop processing the request and should not return any content to the client.

Package sunlabs.brazil.groovy

Provide ways of integrating the Groovy scripting language into the Brazil project server.

See groovy.codehaus.org for more information about Groovy. Files in this package require `groovy.jar` as well as other `jar`s that are part of the Groovy system. This package has been tested using a command line like:

```
java -classpath brazil.jar:groovy-all-1.0-jsr-04.jar ...
```

Class GroovyServerTemplate

sunlabs.brazil.groovy
Class GroovyServerTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.groovy.GroovyServerTemplate
```

All Implemented Interfaces:
TemplateInterface

---

public class **GroovyServerTemplate**
extends Template

The `GroovyServerTemplate` looks for each `<groovy>` tag in an HTML page and treats the following data up to the next `</groovy>`) tag as a groovy script to evaluate.

All Groovy fragments within a given page are evaluated in the same Groovy interpreter. The Groovy interpreter actually lives for the entire duration of this `Template` object, so the user can implement persistence across requests: Each session gets its own interpreter.

The following configuration parameters are used to initialize this template.

script
: The name of the Groovy script to evaluate when the interpreter is created. This script only evaluated when the interp is created, not on every request. The variables `prefix` and `server` are set before this file is evaluated, and are references to the parameters passed to a `handler` init method.

root
: The document root, if the script is a relative file name. If the "root" property under the template prefix is not found, the global "root" property is used. If the global "root" property is not found, the current directory is used.

debug
: If this configuration parameter is present, this class replaces the `<groovy>` and `</groovy>` tags with comments, so the user can keep track of where the dynamically generated content is coming from by examining the comments in the resultant HTML document. By default, the `<groovy>` and `</groovy>` are completely eliminated from the HTML document rather than changed into comments.

Before evaluating each HTML document, this class sets variables in the Groovy interpreter, which can be used to interact back with Java to do things like set the response headers:

request
: Exposes the `Request` Java object. It is set anew at each request.

prefix
: Exposes the handler prefix String.

server
Exposes the handler `Server` object.

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

`GroovyServerTemplate``()`	

Method Summary

`boolean`	**`init`**`(RewriteContext hr)` Initialize the groovy interperter (1st time only) and set the current request object.
`void`	**`tag_groovy`**`(RewriteContext hr)`

Methods inherited from class sunlabs.brazil.template.Template
`done`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

GroovyServerTemplate

```
public GroovyServerTemplate()
```

Method Detail

init

```
public boolean init(RewriteContext hr)
```

Initialize the groovy interperter (1st time only) and set the current request object.

Overrides:

init in class Template

---

tag_groovy

```
public void tag_groovy(RewriteContext hr)
```

Package sunlabs.brazil.handler

This package contains a collection classes that implement the `sunlabs.brazil.server.Handler` interface for use with the `sunlabs.brazil.server.Server` package, along with several support classes.

Just about the only thing they have in common is their lack of dependencies on any other packages. Some provide generic capabilities, such as standard CGI interfaces or template processing, others are either special purpose, provided to demonstrate how to write handlers, and others are skeleton handlers, designed to be *finished* to provide application specific functionality.

You can find a summary of the handlers' options here.

Class AclSwitchHandler

sunlabs.brazil.handler
Class AclSwitchHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.AclSwitchHandler
```

All Implemented Interfaces:
Handler

---

public class **AclSwitchHandler**
extends Object
implements Handler

Simple access control hander based on url prefixes or regexps. Looks up list of valid prefixes or regular expressions in `Request.props` , and allows/denies access based on those prefixes. This is expected to work in conjunction with an upstream handler, such as `RolesHandler` or `BasicAuthHandler` that examines the request, and place credentials into the request object. The credentials consist of url prefixes or regular expressions that match classes of url's. Documents whose URL prefix don't match a credential are rejected. If a credential does not begin with a slash (/), the `init(sunlabs.brazil.server.Server, java.lang.String)` prefix for this handler is prepended.

Properties:
prefix, suffix, glob, match
Sepcify the URL that triggers this handler. (See `MatchString`).
authName
The name of the request.props entry to find a white-space delimited list of url prefixes or regular expression patterns. (defaults to "roles"). If the items in the list don't start with "/", then the url prefix is prepended (only for prefix matching).
redirect
Name of the url to re-direct to if permission is denied. If not specified, a simple message is sent to the client.
useRegexp
If provided, the list of credentials is interpreted as regular expressions, otherwise url prefixes are used.

Constructor Summary

`AclSwitchHandler()`	

Method Summary

boolean	**init**(Server server, String prefix) Initializes the handler.
boolean	**respond**(Request request) Responds to an HTTP request.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

AclSwitchHandler

```
public AclSwitchHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.
Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

respond

```
public boolean respond(Request request)
               throws IOException
```

Description copied from interface: Handler
Responds to an HTTP request.
Specified by:
respond in interface Handler
Tags copied from interface: Handler
Parameters:
`request` - The `Request` object that represents the HTTP request.
Returns:
`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.
Throws:
IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

Class BasicAuthHandler

sunlabs.brazil.handler
Class BasicAuthHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.BasicAuthHandler
```

All Implemented Interfaces:
Handler

---

public class **BasicAuthHandler**
extends Object
implements Handler

The `BasicAuthHandler` obtains a Session ID by performing "basic" authentication, using either the "Authorization" or the "Proxy-Authorization" headers. This handler prevents subsequent downstream handlers from being accessed unless the proper authentication was seen in the request. The Session ID obtained by this handler is meant to be used by those downsteams handlers to access whatever session-dependent information they need.

If the request does not contain the authentication headers or the authentication information is not valid, this handler sends an HTTP error message along with the "WWW-Authenticate" or "Proxy-Authenticate" header, as appropriate. See `code [p ??]` , `authorization [p ??]` , `authenticate [p ??]`

If the request does contain valid authentication information, the Session ID associated with the authentication information is inserted into the request properties, for use by downstream handlers. After inserting the Session ID, this handler returns `false` to allow the downstream handlers to run. IF the Session ID in empty (e.g. ""), then, although authenticateion succeeds, no Session Id property is set.

The set of valid Session IDs is contained in a globally accessible table managed by the `SessionManager`, which may be initialized with a static table (see `mapFile [p ??]`).

The format of the initialization table (if any) described above is a Java properties file where keys are the Base64 encoded strings obtained from the Authentication header and the values are the associated Session IDs. Base64 strings can contain the ’=’ character, but the keys in a Java properties file cannot contain an ’=’ character, so all ’=’ characters in the Base64 strings must be converted to ’!’ in the properties file, as shown in the following sample properties file:

```
bXIuIGhhdGU6a2ZqYw!! = radion
Zm9vOmJhcg!! = foo
```

The data in the SessionManager table doesn’t use the ’!’s, only =’s.

There are several different types of authentication possible. All authentication handlers should follow these basic principles:

- The authentication handler examines some aspect of the request to decide if the appropriate

authentication is present.

- If the request is acceptable, the authentication handler should insert the extracted Session ID into a request property and then return `false`, to allow subsequent handlers to run and perhaps use the Session ID.
- If the request is not acceptable, the authentication handler can return an error message or do some other thing to try to obtain a valid authentication.
- Handlers wishing to be protected by authentication should not subclass an authentication handler. Instead, such handler should be written to assume that authentication has already been performed and then just examine the Session ID present. The web developer is then responsible for choosing which one (of possibly many) forms of authentication to use and installing those authentication handlers before the "sensitive" handler.
- Handlers that are protected by an authentication handler can use the Session ID stored in the request properties regardless of the specifics of the authentication handler.

```
handlers=auth history file

auth.class=BasicAuthHandler
auth.session=account
auth.message=Go away, you're not allowed here!

history.class=HistoryHandler
history.session=account

file.class=FileHandler
file.root=htdocs
```

In the sample pseudo-configuation file specified above, the `BasicAuthHandler` is first invoked to see if the HTTP "basic" authentication header is present in the request. If it isn't, a nasty message is sent back. If the "basic" authentication header is present and corresponds to a user that the `BasicAuthHandler` knows about, the Session ID associated with that user is stored in the specified property named "account".

Subsequently, the `HistoryHandler` examines its specified property (also "account") for the Session ID and uses that to keep track of which session is issuing the HTTP request.

Each handler that needs a Session ID should have a configuration parameter that allows the web developer to specify the name of the request property that holds the Session ID. Multiple handlers can all use the same request property as each other, all protected by the same authentication handler.

This handler uses the following configuration properties:

prefix, suffix, glob, match

Sepcify the URL that triggers this handler.

`code`

The type of authentication to perform. The default value is 401.

The value 401 corresponds to standard "basic" authentication. The "Authorization" request header is supposed to contain the authentication string. If the request was not authenticated, the "WWW-Authenticate" header is sent in the HTTP error response to cause the browser to prompt the client to authenticate.

The value 407 corresponds to "basic" proxy/firewall authentication. The "Proxy-Authorization" request header is supposed to contain the authentication string. If the request was not authenticated, the "Proxy-Authenticate" header is sent in the HTTP error response to cause the browser to prompt the client to authenticate.

Any other value may also be specified. Whatever the value, it will be returned as the HTTP result code of the error message.

`authorization`
: If specified, this is the request header that will contain the "basic" authentication string, instead of the "Authorization" or "Proxy-Authorization" header implied by `code`.

`authenticate`
: If specified, this is the response header that will be sent in the HTTP error response if the user is not authenticated.

If this string is "", then this handler will authenticate the request if the authorization header is present, but **will not** send an HTTP error message if the request could not be authenticated. This is useful if the web developer wants to do something more complex (such as invoking an arbitrary set of handlers) instead of just sending a simple error message if the request was not authenticated. In this case, the web developer can determine that the request was not authenticated because **no** Session ID will be present in the request properties.

`realm`
: The "realm" of the HTTP authentication error message. This is a string that the browser is supposed to present to the client when asking the client the authenticate. It provides a human-friendly name describing who wants the authentication.

`message`
: The body of the HTTP authentication error message. This will be displayed by the browser if the client chooses not to authenticate. The default value is "". Patterns of the form *${xxx}* are replaced with the value of the *xxx* entry of `request.props`.

`mapFile`
: If specified, this is the initial Session ID file. This is expected to be a java properties file, whose keys are the authentication tokens, and whose values are the Session IDs that are inserted into the request properties.

The keys in the file are basic authentication (base64) tokens with any trailing `"="` characters changed to `"!"`.

`session`
: The name of the request property that the Session ID will be stored in, to be passed to downstream handlers. The default value is "SessionID".

`ident`
: The `ident` argument to `SessionManager.getSession(java.lang.Object, java.lang.Object, java.lang.Class)` to get the table of valid sessions. The default value is "authorized". If `ident` is of the form `ident:session`, then the `session` portion is used as the `session` argument to SessionManager.get(). Otherwise the `session` argument is NULL. This table may be manipulated with the SetTemplate, using the "ident" namespace and "session" for the SetTemplate "sessionTable" parameter.

Field Summary

String	**authenticate**
String	**authorization**
int	**code**
String	**ident**
String	**mapFile**
String	**message**
String	**realm**
String	**session**
String	**sessionTable**

Constructor Summary

BasicAuthHandler()	

Method Summary

boolean	**complain**(Request request, String reason) Authentication failed.
boolean	**init**(Server server, String propsPrefix) Initializes this handler.
boolean	**respond**(Request request) Looks up the credentials for this request, and insert them into the request stream.

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

code

`public int` **`code`**

---

authorization

`public String` **`authorization`**

---

authenticate

`public String` **`authenticate`**

---

realm

`public String` **`realm`**

---

message

`public String` **`message`**

---

mapFile

`public String` **`mapFile`**

---

session

`public String` **`session`**

---

ident

```
public String ident
```

---

sessionTable

```
public String sessionTable
```

Constructor Detail

BasicAuthHandler

```
public BasicAuthHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String propsPrefix)
```

Initializes this handler. It is an error if the `mapFile` parameter is specified but that file cannot be loaded.

Specified by:

init in interface Handler

Parameters:

`server` - The HTTP server that created this handler.

`prefix` - A prefix to prepend to all of the keys that this handler uses to extract configuration information.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise.

---

respond

```
public boolean respond(Request request)
                throws IOException
```

Looks up the credentials for this request, and insert them into the request stream. If no credentials are found, prompt the user for them.

Specified by:

respond in interface Handler

Tags copied from interface: Handler

Parameters:

`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

complain

```
public boolean complain(Request request,
                        String reason)
                 throws IOException
```

Authentication failed. Send the appropriate authentication required header as a response.

Parameters:

`request` - The request to respond to

`reason` - The reason for failure (for diagnostics)

Returns:

True

Class CgiHandler

sunlabs.brazil.handler
Class CgiHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.CgiHandler
```

All Implemented Interfaces:
Handler

---

public class **CgiHandler**
extends Object
implements Handler

Handler for implementing cgi/1.1 interface. This implementation allows either suffix matching (e.g. .cgi) to identify cgi scripts, or prefix matching (e.g. /bgi-bin). Defaults to "/". All output from the cgi script is buffered (e.g. chunked encoding is not supported).
NOTE: in versions of Java prior to release 1.3, the ability to set a working directory when running an external process is missing. This handler automatically checks for this ability and sets the proper working directory, but only if the underlying VM supports it.

The following request properties are used:

root
: The document root for cgi files

suffix
: The suffix for cgi files (defaults to .cgi)

prefix
: The prefix for all cgi files (e.g. /cgi-bin)

url
: "o(riginal)" or "c(urrent)". If an upstream handler has changed the URL, this specifes which url to look for the cgi script relative to. The default is to use the original url.

custom
: set to "true" to enable custom environment variables. If set, all server properties starting with this handler's prefix are placed into the environment with the name: `CONFIG_`*`name`*, where *name* is the property key, in upper case, with the prefix removed. This allows cgi scripts to be customized in the server's configuration file.

Constructor Summary

`CgiHandler()`	

Method Summary

boolean	**init**(Server server, String prefix) One time initialization.
boolean	**respond**(Request request) Dispatch and handle the CGI request.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

CgiHandler

```
public CgiHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

One time initialization. The handler configuration properties are extracted and set in `respond(Request)` to allow upstream handlers to modify the parameters.

Specified by:

init in interface Handler

Tags copied from interface: Handler

Parameters:

`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.

`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

respond

```
public boolean respond(Request request)
```

Dispatch and handle the CGI request. Gets called on ALL requests. Set up the environment, exec the process, and deal appropriately with the input and output. In this implementation, all cgi script files must end with a standard suffix, although the suffix may omitted from the url. The url /main/do/me/too?a=b will look, starting in DocRoot, for main.cgi, main/do.cgi, etc until a matching file is found.

Input parameters examined in the request properties:
Suffix
The suffix for all cgi scripts (defaults to .cgi)
DocRoot
The document root, for locating the script.

Specified by:
respond in interface Handler

Tags copied from interface: Handler

Parameters:
`request` - The `Request` object that represents the HTTP request.

Returns:
`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:
IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

Class ChainSawHandler

sunlabs.brazil.handler
Class ChainSawHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.server.ChainHandler
        |
        +--sunlabs.brazil.handler.ChainSawHandler
```

All Implemented Interfaces:
Handler

---

public class **ChainSawHandler**
extends ChainHandler

Variant of the chain handler for doing standard logging. Don't use on fine furniture.

Output is a variant of the common logfile format. The common logfile format is as follows:

```
remotehost rfc931 authuser [date] "request" status bytes
```

remotehost
: Remote hostname (or IP number if DNS hostname is not available, or if DNSLookup is Off.

rfc931
: The remote logname of the user.

authuser
: The username as which the user has authenticated himself.

[date]
: Date and time of the request.

"request"
: The request line exactly as it came from the client.

status
: The HTTP status code returned to the client.

bytes
: The content-length of the document transferred.

"referrer" (optional)
: the referring url

"user agent" (optional)
: "The user agent making the request

Additional Configuration options:

logFile
: The name of the file to write the logs to.

flush
: The number of requests between flushes to the file

If the logFile is removed, the server creates a new one. Thus logs may be truncated by periodically moving them to another name (at least on unix).

See the `LogHandler` handler for generating logs whose contents are configurable.

Field Summary	
static int	**BUFSIZE**
static String	**FLUSH**
static String	**LOG**

Fields inherited from class sunlabs.brazil.server.ChainHandler
exitOnError, handlers, names, prefix, report, urlPrefix

Constructor Summary	
ChainSawHandler()	

Method Summary	
boolean	**init**(Server server, String prefix) Initializes this `ChainHandler` by initializing all the "wrapped" handlers in the list of handlers.
boolean	**respond**(Request request) Run the chain-handler, counting the # of bytes of output generated by its chained handlers.

Methods inherited from class sunlabs.brazil.server.ChainHandler
initHandler

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

LOG

```
public static final String LOG
```

FLUSH

```
public static final String FLUSH
```

BUFSIZE

```
public static final int BUFSIZE
```

Constructor Detail

ChainSawHandler

```
public ChainSawHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from class: ChainHandler
Initializes this `ChainHandler` by initializing all the "wrapped" handlers in the list of handlers. If a wrapped handler cannot be initialized, this method logs a message and skips it. If no handlers were specified, or no handlers were successfully initialized, then the initialization of this `ChainHandler` is considered to have failed.
Overrides:
init in class ChainHandler
Tags copied from class: ChainHandler

Parameters:

`server` - The HTTP server that created this `ChainHandler`.

`prefix` - The prefix for this `ChainHandler`'s properties.

Returns:

`true` if at least one of the wrapped handlers was successfully initialized.

---

respond

```
public boolean respond(Request request)
                throws IOException
```

Run the chain-handler, counting the # of bytes of output generated by its chained handlers.

Overrides:

respond in class ChainHandler

Tags copied from class: ChainHandler

Parameters:

`request` - The HTTP request.

Returns:

`true` if one of the `Handler`s returns `true`, `false` otherwise.

Throws:

IOException - if one of the `Handler`s throws an `IOException` while responding.

Class ChownHandler

sunlabs.brazil.handler
Class ChownHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.ChownHandler
```

All Implemented Interfaces:
Handler

---

public class **ChownHandler**
extends Object
implements Handler

Handler for changing the group and owner of the server. This handler expects the "server.so" file used by the Java Webserver 2.0 Make sure you rename the file lib`server.so` in the distribution to lib`com_sun_server_ServerProcess.so`, and put it where it will be found by **System.loadLibrary**. Note: If the native library is unalvailable on your platform, try the *RunAs* handler, that includes the native source code.

Properties:
userName
name of the user to run as
groupName
The name of the group to run as

Constructor Summary

`ChownHandler()`	

Method Summary

`boolean`	**`init`**`(Server server, String prefix)` set up the Unix user and group.
`boolean`	**`respond`**`(Request request)` Nothing to respond to

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

ChownHandler

```
public ChownHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

set up the Unix user and group. We could return false, so our respond method would never be called, but some containers cause the server to exit on false returns.

Specified by:

init in interface Handler

Tags copied from interface: Handler

Parameters:

`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.

`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
                throws IOException
```

Nothing to respond to

Specified by:

respond in interface Handler

Tags copied from interface: Handler

Parameters:

`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

Class ConfigFileHandler

sunlabs.brazil.handler
Class ConfigFileHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.ConfigFileHandler
```

All Implemented Interfaces:
Handler

---

public class **ConfigFileHandler**
extends Object
implements Handler

Handler for manipulating per-user session state that can be represented as ASCII name/value pairs. The state for each session is stored in a file, which is expected to be in java properties format. If "prefix" is matched, the contents of the (usually cached) config file for the current session is added to the request properties. If the url matches the "set" property, the contents of the config file are changed based on the supplied query parameters (either GET of POST). If no config file exists for the session, one is created from a default properties file. Only properties already in the config file may be changed using the "set" method. If a "%" is specified in the file name, it is replaced by the SessionID property, if any, or "common" if sessions aren't used. This should be replaced with something more general, so we can have more arbitrary mappings between request and the session info.

The following request properties are used:
prefix
: The URL prefix required for all documents
set
: The url pattern to match setting properties. Currently, it must also match "prefix".
noContent
: a url, matching the "set" pattern that causes a "204 no content" to be returned to the client (experimental).
name
: The name of the config file. the first "%" is replaced by the current SessionID.
default
: The default properties file to "seed" session properties
glob
: Properties that match this "glob" pattern may be set using the "set" pattern. If this property is specified, the "default" property is optional.
root
: The document root (no properties prefix required). If the "name" or "default" properties don't start with a "/", this is used as the current directory.

If "%" is specified in the file name, a new session file is created only if 1) a property is changed from the default, and 2) A cookie was received by the browser.

See also: `SetTemplate` which is preferrable in most cases, providing a templated based (instead of URL based) mechanism for maintaining persistent properties.

Constructor Summary

`ConfigFileHandler``()`	

Method Summary

`boolean`	**`init`**`(Server server, String prefix)` Make sure default properties exist before starting this handler, or that "match" is specified".
`boolean`	**`respond`**`(Request request)` Extract the session state into the request object, optionally modifying the properties.

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

ConfigFileHandler

```
public ConfigFileHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Make sure default properties exist before starting this handler, or that "match" is specified".

Specified by:

init in interface Handler

Tags copied from interface: Handler

Parameters:

`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.

`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
                throws IOException
```

Extract the session state into the request object, optionally modifying the properties. If the properties are modified, they are stored in a file for safe keeping.

Specified by:

respond in interface Handler

Tags copied from interface: Handler

Parameters:

`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

Class CookieSessionHandler

sunlabs.brazil.handler
Class CookieSessionHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.CookieSessionHandler
```

All Implemented Interfaces:
Handler

---

public class **CookieSessionHandler**
extends Object
implements Handler

Handler for creating browser sessions using cookies. This handler provides a single cookie-id that may be used by other handlers. The intent is to require only one cookie per server. (See also `SessionFilter`, which manages sessions with or without cookies). The following server properties are used:

prefix, suffix, glob, match
: Specify the URL that triggers this handler (See `MatchString`).

  If `prefix` is specified, it is also used to instruct the client to limit the scope of the browser cookie. to that prefix.

cookie
: the name of the cookie to use (defaults to "cookie").

`map`
: If specified, the `ident` argument to `SessionManager.getSession(java.lang.Object, java.lang.Object, java.lang.Class)` to get the table of valid cookies, used to map the cookie value to a Session ID. By default, the Session ID stored in the request is the cookie value itself.

`exist`
: If specified, this means that the Session ID corresponding to the cookie value must already exist in the `SessionManager`. Normally, if the cookie was not present, a new cookie is automatically created.

persist
: If set, cookies persist across browser sessions

`session`
: The name of the request property that the Session ID will be stored in, to be passed to downstream handler. The default value is "SessionID". If the property already exists, and is not empty, no action will be taken.

If a cookie was returned from the browser, the property: `gotCookie` is set to the cookie name. Otherwise it is left unset.

Field Summary

String	**cookieName**
String	**ident**
boolean	**mustExist**
boolean	**persist**
String	**session**

Constructor Summary

CookieSessionHandler()

Method Summary

boolean	**init**(Server server, String propsPrefix) Initializes the handler.
boolean	**respond**(Request request) Responds to an HTTP request.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

cookieName

```
public String cookieName
```

---

ident

```
public String ident
```

---

mustExist

```
public boolean mustExist
```

---

persist

```
public boolean persist
```

---

session

```
public String session
```

Constructor Detail

CookieSessionHandler

```
public CookieSessionHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String propsPrefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
                throws IOException
```

Description copied from interface: Handler

Responds to an HTTP request.

Specified by:

respond in interface Handler

Tags copied from interface: Handler

Parameters:

`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

Class DefaultFileHandler

sunlabs.brazil.handler
Class DefaultFileHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.DefaultFileHandler
```

All Implemented Interfaces:
Handler

---

public class **DefaultFileHandler**
extends Object
implements Handler

Handler for appending a url ending with '/' into the appropriate url based on a default file in the file system.

The following request properties are used:
defaults
The names of the default files to search for in the directory implied by the URL. The first one that exists will cause its name to be appended to the URL. Defaults to "index.html".
root
The document root to look for files. If none is found with our prefix, then "root" is examined. Defaults to ".".
DirectoryName
This property is set if the URL represents a valid directory in the document root.
fileName
This property is set to the name of the default file, if one was found.
If a url ends with "/", but is a readable plain file, the "/" is removed

---

Constructor Summary

DefaultFileHandler()	

Method Summary

boolean	**init**(Server server, String prefix) Remember our prefix in the properties table.
boolean	**respond**(Request request) If the url ends with a "/" look around in the corrosponding directory to find a suitable default file, and then change the url.

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

DefaultFileHandler

```
public DefaultFileHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Remember our prefix in the properties table.

Specified by:

init in interface Handler

Tags copied from interface: Handler

Parameters:

`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.

`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
               throws IOException
```

If the url ends with a "/" look around in the corrosponding directory to find a suitable default file, and then change the url.

Specified by:

respond in interface Handler

Returns:

Always returns false.

Class DeferredHandler

sunlabs.brazil.handler
Class DeferredHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.DeferredHandler
```

All Implemented Interfaces:
Handler

---

public class **DeferredHandler**
extends Object
implements Handler

Wrap another handler, deferring its initialization until request time. This allows handlers to be configured, but not initialized until a first use is attempted.

Normally, when a handler's class is first resolved, if any of the dependent classes are not available, an error will occur, terminating the server. Using this handler, other handlers can be conditionally configured based on the availability of other specified classes at run time.

NOTE: This functionallity should be integrated into the `ChainHandler`, eliminating the need for this one.

Request Properties
handler
The token representing the handler to conditionally configure. This is used as the handler's prefix
requires
The names of classes required to be resolvable before configuring the handler
[handler].prefix
Used to trigger the configuration
[handler].class
The name of the handler class.

Constructor Summary

`DeferredHandler()`	

Method Summary	
boolean	**init**(Server server, String prefix) Remember the server for deferred initialization.
boolean	**respond**(Request request) Dispatch to the handler, installing it if needed

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

DeferredHandler

```
public DeferredHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Remember the server for deferred initialization.

Specified by:

init in interface Handler

Tags copied from interface: Handler

Parameters:

`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.

`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
                throws IOException
```

Dispatch to the handler, installing it if needed

Specified by:

respond in interface Handler

Tags copied from interface: Handler

Parameters:

`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

Class DialogHandler

sunlabs.brazil.handler
Class DialogHandler

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--java.awt.Panel
                    |
                    +--java.applet.Applet
                          |
                          +--sunlabs.brazil.handler.DialogHandler
```

All Implemented Interfaces:
javax.accessibility.Accessible, Handler, ImageObserver, MenuContainer, Serializable

---

public class **DialogHandler**
extends Applet
implements Handler

Sample handler for popping up a dialog box on the server. This is used for interactive authentication of web pages, allowing an *operator* on the server's computer to allow or deny access to pages on a per request basis. Input parameters examined in the request properties:

prefix, suffix, glob, match
: Specify the URL that triggers this handler. (See `MatchString`).

default
: The message to appear in the dialog box. Defaults to **Request from Client**.

denied
: The message to appear in the "permission denied" spot.

If query data is present, it is used as the message.

Note: This is the only class in the entire system that requires AWT.

See Also:
Serialized Form

Field Summary	
Button	**cancel**
Panel	**frame_1**
Button	**ok**
Label	**title**

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Constructor Summary	
DialogHandler()	

Method Summary	
void	**dismiss**(boolean how)
boolean	**handleEvent**(Event event)
void	**init**() Machine generated code.
boolean	**init**(Server server, String prefix) Do one time initialization.
boolean	**respond**(Request request) Pop up a dialog box on the server machine.

Methods inherited from class java.applet.Applet
destroy, getAccessibleContext, getAppletContext, getAppletInfo, getAudioClip, getAudioClip, getCodeBase, getDocumentBase, getImage, getImage, getLocale, getParameter, getParameterInfo, isActive, newAudioClip, play, play, resize, resize, setStub, showStatus, start, stop

Methods inherited from class java.awt.Panel
addNotify

Methods inherited from class java.awt.Container
add, add, add, add, add, addContainerListener, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getInsets, getLayout, getListeners, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, remove, remove, removeAll, removeContainerListener, removeNotify, setCursor, setFont, setLayout, update, validate

Methods inherited from class java.awt.Component
action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addPropertyChangeListener, addPropertyChangeListener, bounds, checkImage, checkImage, contains, contains, createImage, createImage, disable, dispatchEvent, enable, enable, enableInputMethods, getBackground, getBounds, getBounds, getColorModel, getComponentOrientation, getCursor, getDropTarget, getFont, getFontMetrics, getForeground, getGraphics, getGraphicsConfiguration, getHeight, getInputContext, getInputMethodRequests, getLocation, getLocation, getLocationOnScreen, getName, getParent, getPeer, getSize, getSize, getToolkit, getTreeLock, getWidth, getX, getY, gotFocus, hasFocus, hide, imageUpdate, inside, isDisplayable, isDoubleBuffered, isEnabled, isFocusTraversable, isLightweight, isOpaque, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, printAll, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, reshape, setBackground, setBounds, setBounds, setComponentOrientation, setDropTarget, setEnabled, setForeground, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, show, size, toString, transferFocus

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

frame_1

public Panel **frame_1**

---

ok

```
public Button ok
```

---

title

```
public Label title
```

---

cancel

```
public Button cancel
```

Constructor Detail

DialogHandler

```
public DialogHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Do one time initialization.

Specified by:

init in interface Handler

Tags copied from interface: Handler

Parameters:

`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.

`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
               throws IOException
```

Pop up a dialog box on the server machine. Allow the operator to select **yes** or **no**. If the request is allowed, it is passed on to the next handler.

Specified by:
respond in interface Handler
Tags copied from interface: Handler
Parameters:
`request` - The `Request` object that represents the HTTP request.
Returns:
`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.
Throws:
IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

dismiss

```
public void dismiss(boolean how)
```

---

init

```
public void init()
```

Machine generated code. Everything after here was automatically generated SpecTcl generated class Dialog, version 1.0
Overrides:
init in class Applet

---

handleEvent

```
public boolean handleEvent(Event event)
```

Overrides:
handleEvent in class Component

Class DigestAuthHandler

sunlabs.brazil.handler
Class DigestAuthHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.DigestAuthHandler
```

All Implemented Interfaces:
Handler

---

public class **DigestAuthHandler**
extends Object
implements Handler

Perform digest authentication. This is a minimal implementation of RFC 2617 The "optional" qos parameter is required by IE (only qop="auth" is supported). The "password" file is read at startup time, either as a resource or from the file system, and may contain either plain text or digested passwords (see main() below to digest passwords).

Future enhancements
- Better dynamic operation
- Optional digest parameter handling
- Nonce time-to-live checking

Sample auth request header

```
WWW-Authenticate: Digest
   realm="myrealm",
   qop="auth",                                  [req'd for IE]
   nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
   opaque="5ccc069c403ebaf9f0171e9517f40e41",  [optional]
   domain="/foo"                                        [optional]
```

Sample client return header

```
 Authorization: Digest
   username="name",
   realm="foo@bar",
   nonce="mynonce10",
   uri="/da.html",
   response="d58f3f9fa7554da651d3f1901d22ea04",
   qop=auth,
   nc=00000001,
   cnonce="b6ac242cb324c38a"

response algorithm:

A1 = md5(user:realm:pass)
```

```
A2 = md5(method:uri)
response=md5(A1:nonce:nonceCount:cnonce:qop:A2)
- all MD5's are represented as hex: [0-9a-f]
- all quotes (") are removed before digesting
```

prefix, suffix, glob, match
: Specify which url's this handler applies to.

realm
: The string presented to the user for validation. This must also match any "digested" passwords.

credentials
: A java-properties format file of credentials. The keys are the users, the values are either the "A1" values described above, or the user's password.

isDynamic
: If set (to anything), when authentication for a user is requested that is not in the credentials table and the credentials table has changed since last read, the table is re-read, in case the user has been added since the credentials were loaded.

allowBogusIE
: Internet Explorer does not use the query parameters as part of the "uri" calculation. This is a bug (and a security risk, as it allows replay attacts to other than the url requested). If this variable is set, then it allows IE to work in this case.

username
: If the user was validated, this field is filled out by the handler.

Constructor Summary

`DigestAuthHandler``()`	

Method Summary

static String	**computeA1**(String user, String realm, String pass) Compute the A1 parameter as per the RFC.
static String	**computeA2**(String method, String uri) Compute the A2 parameter as per the RFC.
static String	**computeResponse**(String A1, String A2, String nonce, String nc, String cnonce, String qop) Compute the expected client response attribute value.
static Properties	**extractAuth**(String header) Parse an auth header, placing the results into a Properties object.
static String	**genResponseHeader**(String request, String user, String pass, String method, String uri, String nc, String cnonce) Given the "WWW-Authenticate" header value and additional client info, generate the value of the "Authorization" header.
boolean	**init**(Server server, String propsPrefix) Initializes the handler.
static boolean	**isMd5Digest**(String s) See if a string is a valid md5 digest.
static void	**main**(String[] args) Convert a "plain text" password file into a digested one.
static String	**md5Digest**(String s) Compute the md5 digest of a string, returning the digest as a hex string.
boolean	**respond**(Request request) Responds to an HTTP request.
static boolean	**responseOk**(String A1, String method, Properties h) Check the digest response string.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

DigestAuthHandler

```
public DigestAuthHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String propsPrefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.
Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
                throws IOException
```

Description copied from interface: Handler
Responds to an HTTP request.
Specified by:
respond in interface Handler
Tags copied from interface: Handler
Parameters:
`request` - The `Request` object that represents the HTTP request.
Returns:
`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.
Throws:
IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

responseOk

```
public static boolean responseOk(String A1,
                                 String method,
                                 Properties h)
```

Check the digest response string.

Parameters:

`A1` - The "A1" hash from the RFC

`method` - The http request method.

`h` - Properties containing all the name=value options from the http authentiation header field (see `extractAuth(String)`).

---

computeA1

```
public static String computeA1(String user,
                               String realm,
                               String pass)
```

Compute the A1 parameter as per the RFC.

---

computeA2

```
public static String computeA2(String method,
                               String uri)
```

Compute the A2 parameter as per the RFC.

---

computeResponse

```
public static String computeResponse(String A1,
                                     String A2,
                                     String nonce,
                                     String nc,
                                     String cnonce,
                                     String qop)
```

Compute the expected client response attribute value.

---

genResponseHeader

```
public static String genResponseHeader(String request,
                                       String user,
                                       String pass,
                                       String method,
                                       String uri,
                                       String nc,
                                       String cnonce)
```

Given the "WWW-Authenticate" header value and additional client info, generate the value of the "Authorization" header. The "request" should contain "realm", "nonce", "qop" and optionally "opaque". This is a convenience method for clients to use to athenticate to this server implementation.

Parameters:

`request` - The string value of the "WWW-Authenticate" header from the server
`user` - The userid
`pass` - The password associated with this user
`method` - "GET", "POST", etc.
`uri` - The requested url (e.g. "/index.html")
`nc` - The "nonce count", or number of times the client has used The "nonce" presented by the server (e.g. "0000001").
`cnonce` - An opaque value provided by the client

---

md5Digest

```
public static String md5Digest(String s)
```

Compute the md5 digest of a string, returning the digest as a hex string.

---

isMd5Digest

```
public static boolean isMd5Digest(String s)
```

See if a string is a valid md5 digest.

---

extractAuth

```
public static Properties extractAuth(String header)
```

Parse an auth header, placing the results into a Properties object. Format is: Digest key=value, key=value, ... values may be in "'s.

---

main

```
public static void main(String[] args)
                 throws Exception
```

Convert a "plain text" password file into a digested one. Any existing digests are left alone.

```
Usage: DigestAuthHandler [realm]
```

The stdin, in Properties format, is emitted on stdout with all plain-text passwords digested. If an entry is already digested, it is left alone.

Note, this handler will except either plaintext or digested passwords in the credentials file.

Class DirectoryHandler

sunlabs.brazil.handler
Class DirectoryHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.DirectoryHandler
```

All Implemented Interfaces:
Handler

---

public class **DirectoryHandler**
extends Object
implements Handler

This is a bare-bones handler for providing directory listings for web servers. It is designed to be placed after the `FileHandler`. If no index file is found, Then a simple directory listing will be produced. Only files whose extensions are in the mime properties will be listed.

NOTE: This handler is obsolete, as it provides no control over the format of the directory listing, Use the `DirectoryTemplate` instead.

Configuration properties used:
prefix, suffix, glob, match
: Specify the URL that triggers this handler. (See `MatchString`).

DirectoryName
: This property is set by the `FileHandler` if the URL it was passed resolves to a directory, but no index file (e.g. index.html) was found.

setProps
: If present, no content is returned. Instead, The properties "Directories" and "Files" are set in the request properties, so the format of the output may be generated dynamically. [Note: This feature is deprecated, use the `DirectoryTemplate` instead].

delim
: The delimeter separating the file names. Defaults to a single space.

mime.xxx
: Only documents ending in ".xxx" are considered. more than on mime.xxx parameters may be specified.

Constructor Summary

`DirectoryHandler()`	

Method Summary

boolean	**init**(Server server, String prefix) Get the url prefix for this handler.
boolean	**respond**(Request request) Display files in a directory, after being rejected by the FileHandler.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

DirectoryHandler

```
public DirectoryHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Get the url prefix for this handler.

Specified by:

init in interface Handler

Tags copied from interface: Handler

Parameters:

`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.

`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
                throws IOException
```

Display files in a directory, after being rejected by the FileHandler. The output is very simple.

Specified by:

respond in interface Handler

Tags copied from interface: Handler

Parameters:

`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

Class DynamicConfigHandler

sunlabs.brazil.handler
Class DynamicConfigHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.DynamicConfigHandler
```

All Implemented Interfaces:
Handler

---

public class **DynamicConfigHandler**
extends Object
implements Handler

The `DynamicConfigHandler` allows the user to change the configuration of the server and its handlers on the fly. This handler can therefore be used to swap in and out functionality, for example, by dynamically adding a new `AuthHandler` to add a new form of authentication to a running server.

This handler uses a special set of URLs to allow a new set of configuration properties to be uploaded. The new configuration replaces the old configuration.

The name of another `Handler` is supplied when this `DynamicConfigHandler` is initialized. This `Handler` is the helper or sub-handler for the `DynamicConfigHandler`. When the `DynamicConfigHandler` receives a regular HTTP request (that matches the URL prefix described below), it redirects that request to the `respond` method of the sub-handler.

The uploaded configuration properties are kept in a separate properties object from the server's properties. The server's properties are in fact not accessible from the sub-handler; the sub-handler can only access and/or change the properties owned by the `DynamicConfigHandler`.

This handler uses the following configuration properties:

handler
: The name of the initial sub-handler that this `DynamicConfigHandler` will use to process requests. When new properties are uploaded, the sub-handler will be replaced with whatever is specified in the newly uploaded `handler` property.

prefix
: Only URLs beginning with this string will be redirected to the sub-handler. This property belongs to the `DynamicConfigHandler` and is **not** changed when new properties are uploaded. The default is "/".

config
: URLs beginning with this prefix can be used to upload or download new configuration properties to this handler, which will also dynamically change which sub-handler is installed. This property belongs to the `DynamicConfigHandler` and is **not** changed when new properties are uploaded. The default is "/config/".

Properties may be uploaded by sending them as "name=value" pairs in the body of a POST or in the "?" query parameters. The URL for uploading properties is "*config*/set".

The current set of properties may be retrieved from this handler by sending the URL "*config*/get"
A sample set of configuration parameters illustrating how to use this handler follows:

```
handler=sunlabs.brazil.server.ChainHandler
port=8081
log=5

handlers=dyn cgi

dyn.class=sunlabs.brazil.server.DynamicConfigHandler
dyn.prefix=/sparky/
dyn.config=/config-sparky/
dyn.handler=chain

chain.class=sunlabs.brazil.server.ChainHandler
chain.handlers=foo baz garply

foo.class=sunlabs.brazil.handler.HomeDirHandler
foo.home=/home/users/

baz.class=sunlabs.brazil.handler.FileHandler

garply.class=sunlabs.brazil.handler.NotFoundHandler
garply.root="/errors/"
garply.fileName="nofile.html"

cgi.class = sunlabs.brazil.handler.CgiHandler
.
.
.
```

These parameters set up a normal `Server` on port 8081, running a `ChainHandler` which dispatches to a `DynamicConfigHandler` and a `CgiHandler`.

The `DynamicConfigHandler` will listen for HTTP requests beginning with "/sparky/" and dispatch to its dynamically generated list of handlers, and listen for requests beginning with "/config-sparky/" to dynamically change that set of handlers.

To give this `DynamicConfigHandler` something to do, an initial set of handlers is provided with the same prefix ("dyn") as the `DynamicConfigHandler` itself. The prefix is stripped off those properties and the revised set of properties is passed to the `DynamicConfigHandler` to initialize its dynamically configurable world.

Constructor Summary

`DynamicConfigHandler``()`	

Method Summary

boolean	**init**(Server server, String prefix) Initializes this DynamicConfigHandler by loading the initial handler.
boolean	**respond**(Request request) Responds to an HTTP request by examining the "Host:" request header and dispatching to the main handler of the server that handles that virtual host.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

DynamicConfigHandler

```
public DynamicConfigHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Initializes this DynamicConfigHandler by loading the initial handler. An initial handler does not need to be defined, however, since the handler configuration can be downloaded later.

Specified by:

init in interface Handler

Parameters:

server - The HTTP server that created this handler.

prefix - A prefix to prepend to all of the keys that this handler uses to extract configuration information.

Returns:

false if the initial handler was specified but could not be initialized, true otherwise.

---

respond

```
public boolean respond(Request request)
                throws IOException
```

Responds to an HTTP request by examining the "Host:" request header and dispatching to the main handler of the server that handles that virtual host. If the "Host:" request header was not specified, or named a virtual host that was not initialized in `init` from the list of virtual hosts, this method returns without handling the request.

Specified by:

respond in interface Handler

Parameters:

`request` - The HTTP request to be forwarded to one of the sub-servers.

Returns:

`true` if the sub-server handled the message, `false` if it did not. `false` is also returned if the "Host:" was unspecified or unknown.

Class GenericProxyHandler

sunlabs.brazil.handler
Class GenericProxyHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.GenericProxyHandler
```

All Implemented Interfaces:
Handler
Direct Known Subclasses:
MultiProxyHandler, ProxyPropertiesHandler

---

public class **GenericProxyHandler**
extends Object
implements Handler

Handler for implementing a virtual web site. This causes another web site to "appear" inside our document root. This classes is intended to be sub-classed, so some of the methods in this implementation don't do too much. All of the appropriate links in HTML documents on the virtual site are rewritten, so they appear to be local references. This can be used on a firewall in conjunction with `AclSwitchHandler` to provide authenticated access to selected web sites.

Properties:
prefix
URL prefix must match
host
name of host site to proxy to.
port
Host port to proxy to (defaults to 80).
proxyHost
Which proxy host to use (if any) to contact "host".
proxyPort
The proxy's port (defaults to 80)
headers
A list of white space delimited tokens that refer to additional HTTP headers that are added onto the polled request. For each token the server properties `[token].name` and `[token].value` define a new http header.

Field Summary

static String	**HOST** Handler configuration property **host**.
static String	**NL**
static String	**PORT** Handler configuration property **port**.
static String	**PREFIX** Handler configuration property **prefix**.
static String	**PROXY_HOST** Handler configuration property **proxyHost**.
static String	**PROXY_PORT** Handler configuration property **proxyPort**.

Constructor Summary

GenericProxyHandler()	

Method Summary

void	**addMap**(MapPage mapper) for subclassing
boolean	**init**(Server server, String prefix) Do one-time setup.
boolean	**isMine**(Request request) See if this is one of my requests.
byte[]	**modifyContent**(Request request, byte[] content) Rewrite the links in an html file so they resolve correctly in proxy mode.
boolean	**respond**(Request request) If this is one of "our" url's, fetch the document from the destination server, and return it as if it was local.

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

PREFIX

`public static final String` **`PREFIX`**

Handler configuration property **prefix**. Only URL's that begin with this string are considered by this handler. The default is (/).

HOST

`public static final String` **`HOST`**

Handler configuration property **host**. The actual host site to appear on our site (required)

PORT

`public static final String` **`PORT`**

Handler configuration property **port**. The actual port on the host site (defaults to 80).

PROXY_HOST

`public static final String` **`PROXY_HOST`**

Handler configuration property **proxyHost**. The name of a proxy to use (if any) to get to the **host**.

PROXY_PORT

`public static final String` **`PROXY_PORT`**

Handler configuration property **proxyPort**. The proxy port to use to get to the **host**. defaults to 80.

NL

```
public static final String NL
```

Constructor Detail

GenericProxyHandler

```
public GenericProxyHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Do one-time setup. get and process the handler properties. we can contact the server identified by the **host** parameter.

Specified by:

init in interface Handler

Tags copied from interface: Handler

Parameters:

`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.

`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
                throws IOException
```

If this is one of "our" url's, fetch the document from the destination server, and return it as if it was local.

Specified by:

respond in interface Handler

Tags copied from interface: Handler

Parameters:

`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

isMine

```
public boolean isMine(Request request)
```

See if this is one of my requests. This method can be overridden to do more sophisticated mappings.

Parameters:

`request` - The standard request object

---

modifyContent

```
public byte[] modifyContent(Request request,
                            byte[] content)
```

Rewrite the links in an html file so they resolve correctly in proxy mode.

Parameters:

`request` - The original request to this "proxy"

`headers` - The vector of mime headers for the proxy request

Returns:

true if the headers and content should be sent to the client, false otherwise Modifies "headers" as a side effect

---

addMap

```
public void addMap(MapPage mapper)
```

for subclassing

Class HomeDirHandler

sunlabs.brazil.handler
Class HomeDirHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.HomeDirHandler
```

All Implemented Interfaces:
Handler

---

public class **HomeDirHandler**
extends Object
implements Handler

Handler for converting ~username queries. When invoked upstream of the `FileHandler` This provides Unix user's with individual home pages.

Properties:
subdir
Name of the directory in the user's home directory that represents the user's "doc root"
home
The mount-point for home directories, defaults to "/home/".
root
The name of the root property to set. Defaults to "root".
prefix
The url prefix used to identify home directory queries. Defaults to "/~".
Url's of the form:

```
/~[user]/stuff...
```

are transformed into [home][user]/[subdir]/stuff....

Note: This functionallity has been mostly subsumed by the `UrlMapperHandler`.

Constructor Summary

`HomeDirHandler()`	

Method Summary

boolean	**init**(Server server, String prefix) Get and set the configuration parameters.
boolean	**respond**(Request request) If this is a ~user request, modify the `root` and `url` properties of the request object.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

HomeDirHandler

```
public HomeDirHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Get and set the configuration parameters.

Specified by:

init in interface Handler

Tags copied from interface: Handler

Parameters:

`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.

`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
                throws IOException
```

If this is a ~user request, modify the `root` and `url` properties of the request object.

Specified by:

respond in interface Handler

Tags copied from interface: Handler

Parameters:

`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

Class HtmlRewriter

sunlabs.brazil.handler
Class HtmlRewriter

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.HtmlRewriter
```

Direct Known Subclasses:
RewriteContext

---

public class **HtmlRewriter**
extends Object

This class helps with parsing and rewriting an HTML document. The source document is not changed; a new HTML document is built.

The user can sequentially examine and rewrite each token in the source HTML document. As each token in the document is seen, the user has two choices:

- modify the current token.
- don't modify the current token.

If the user modifies (or replaces, deletes, etc.) the current token, then the resultant HTML document will contain that modification. On the other hand, if the user doesn't do anything with the current token, it will appear, unchanged, in the resultant HTML document.

Parsing is implemented lazily, meaning, for example, that unless the user actually asks for attributes of an HTML tag, this parser does not have to spend the time breaking up the attributes.

This class is used by HTML filters to maintain the state of the document and allow the filters to perform arbitrary rewriting.

Field Summary	
LexHTML	**lex** The parser for the source HTML document.
StringBuffer	**sb** Storage holding the resultant HTML document.

Constructor Summary

HtmlRewriter(LexHTML lex) Creates a new HtmlRewriter from the given HTML parser.	
HtmlRewriter(String str) Creates a new HtmlRewriter that will operate on the given string.	

Method Summary

boolean	**accumulate**(boolean accumulate) Turns on or off the automatic accumulation of each token.
void	**append**(String str) Instead of modifying an existing token, this method allows the user to completely replace the current token with arbitrary new content.
void	**appendToken**() Appends the current token to the resultant HTML document.
String	**get**(String key) Returns the value that the specified case-insensitive key maps to in the attributes for the current tag.
String	**getArgs**() Gets the arguments of the current token as a string.
String	**getBody**() Gets the body of the current token as a string.
StringMap	**getMap**() Return a copy of the StringMap of attributes.
String	**getTag**() Gets the current tag's name.
String	**getToken**() Gets the raw string making up the entire current token, including the angle brackets or comment delimiters, if applicable.
int	**getType**() Gets the type of the current token.
boolean	**isSingleton**() See if the current tag a singleton.
Enumeration	**keys**() Returns an enumeration of the keys in the current tag's attributes.

void	**killToken**() Tells this HtmlRewriter not to append the current token to the resultant HTML document.
boolean	**nextTag**() A convenence method built on top of nextToken.
boolean	**nextToken**() Advances to the next token in the source HTML document.
void	**pushback**() Puts the current token back.
void	**put**(String key, String value) Maps the given case-insensitive key to the specified value in the current tag's attributes.
static String	**quote**(String str) Helper class to quote a attribute's value when the value is being written to the resultant HTML document.
void	**remove**(String key) Removes the given case-insensitive key and its corresponding value from the current tag's attributes.
void	**reset**() Forgets all the tokens that have been appended to the resultant HTML document so far, including the current token.
void	**setSingleton**(boolean singleton) Make the current tag a singleton.
void	**setTag**(String tag) Changes the current tag's name.
void	**setType**(int type) Sets the type of the current token.
int	**tagCount**() Return count of tags seen so far
int	**tokenCount**() Return count of tokens seen so far
String	**toString**() Returns the "new" rewritten HTML document.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

lex

`public LexHTML **lex**`

The parser for the source HTML document.

---

sb

`public StringBuffer **sb**`

Storage holding the resultant HTML document.

Constructor Detail

HtmlRewriter

`public **HtmlRewriter**(LexHTML lex)`

Creates a new `HtmlRewriter` from the given HTML parser.
Parameters:
`lex` - The HTML parser.

HtmlRewriter

`public **HtmlRewriter**(String str)`

Creates a new `HtmlRewriter` that will operate on the given string.
Parameters:
`str` - The HTML document.

Method Detail

toString

`public String **toString**()`

Returns the "new" rewritten HTML document. This is normally called once all of the tokens have been processed, and the user wants to send on this rewritten document.

At any time, this method can be called to return the current state of the HTML document. The return value is the result of processing the source document up to this point in time; the unprocessed remainder of the source document is not considered.

Due to the implementation, calling this method may be expensive. Specifically, calling this method a second (or further) time for a given `HtmlRewriter` may involve copying temporary strings around. The pessimal case would be to call this method every time a new token is appended.

Overrides:

toString in class Object

Returns:

The rewritten HTML document, up to this point in time.

---

nextToken

`public boolean **nextToken**()`

Advances to the next token in the source HTML document.

The other purpose of this function is to "do the right thing", which is to append the token we just processed to the resultant HTML document, unless the user has already appended something else.

A sample program follows. This program changes all `<img>` tags to `<form>` tags, deletes all `<table>` tags, capitalizes and bolds each string token, and passes all other tokens through unchanged, to illustrate how `nextToken` interacts with some of the other methods in this class.

```
HtmlRewriter hr = new HtmlRewriter(str);
while (hr.nextToken()) {
    switch (hr.getType()) {
    case LexHTML.TAG:
        if (hr.getTag().equals("img")) {
            // Change the tag name w/o affecting the attributes.

            hr.setTag("form");
        } else if (hr.getTag().equals("table")) {
            // Eliminate the entire "table" token.

            hr.killToken();
        }
        break;

    case LexHTML.STRING:
        // Append a new sequence in place of the existing token.

        hr.append("<b>" + hr.getToken().toUpperCase() + "</b>");
        break;
    }
    // Any tokens we didn’t modify get copied through unchanged.
}
```

Returns:

`true` if there are tokens left to process, `false` otherwise.

---

nextTag

`public boolean **nextTag**()`

A convenence method built on top of `nextToken`. Advances to the next HTML tag. All intervening strings and comments between the last tag and the new current tag are copied through unchanged. This method can be used when the caller wants to process only HTML tags, without having to manually check the type of each token to see if it is actually a tag.

Returns:

`true` if there are tokens left to process, `false` otherwise.

---

getType

`public int **getType**()`

Gets the type of the current token.

Returns:

The type.

See Also:

`LexML.getType()`

---

setType

`public void **setType**(int type)`

Sets the type of the current token.

---

isSingleton

`public boolean **isSingleton**()`

See if the current tag a singleton. A Singleton tag ends in "/", as in <

.

---

setSingleton

`public void **setSingleton**(boolean singleton)`

Make the current tag a singleton. A Singleton tag ends in "/", as in <

.

---

getToken

`public String **getToken**()`

Gets the raw string making up the entire current token, including the angle brackets or comment delimiters, if applicable.

Returns:

The current token.

See Also:

`LexML.getToken()`

---

getTag

`public String **getTag**()`

Gets the current tag's name. The name returned is converted to lower case.

Returns:

The lower-cased tag name, or `null` if the current token does not have a tag name

See Also:

`LexHTML.getTag()`

---

setTag

`public void **setTag**(String tag)`

Changes the current tag's name. The tag's attributes are not changed.

Parameters:

`tag` - New tag name

---

getBody

`public String **getBody**()`

Gets the body of the current token as a string.

Returns:

The body.

See Also:

`LexML.getBody()`

---

getArgs

`public String **getArgs**()`

Gets the arguments of the current token as a string.

Returns:

The body.

See Also:
`LexML.getArgs()`

---

get

```
public String get(String key)
```

Returns the value that the specified case-insensitive key maps to in the attributes for the current tag. For keys that were present in the tag's attributes without a value, the value returned is the empty string. In other words, for the tag `<table border rows=2>`:

- `get("border")` returns the empty string "".
- `get("rows")` returns *2*.

Surrounding single and double quote marks that occur in the literal tag are removed from the values reported. So, for the tag `<a href="/foo.html" target=_top onclick='alert("hello")'>`:

- `get("href")` returns */foo.html* .
- `get("target")` returns *_top* .
- `get("onclick")` returns *alert("hello")* .

Parameters:
`The` - key to lookup in the current tag's attributes.

Returns:
The value to which the specified key is mapped, or `null` if the key was not in the attributes.

See Also:
`LexML.getAttributes()`

---

put

```
public void put(String key,
                String value)
```

Maps the given case-insensitive key to the specified value in the current tag's attributes.

The value can be retrieved by calling `get` with a key that is case-insensitive equal to the given key.

If the attributes already contained a mapping for the given key, the old value is forgotten and the new specified value is used. The case of the prior key is retained in that case. Otherwise the case of the new key is used and a new mapping is made.

Parameters:
`key` - The new key. May not be `null`.
`value` - The new value. May be not be `null`.

---

remove

`public void **remove**(String key)`

Removes the given case-insensitive key and its corresponding value from the current tag's attributes. This method does nothing if the key is not in the attributes.

Parameters:

`key` - The key that needs to be removed. Must not be `null`.

---

keys

`public Enumeration **keys**()`

Returns an enumeration of the keys in the current tag's attributes. The elements of the enumeration are the string keys. The keys can be passed to `get` to get the values of the attributes.

Returns:

An enumeration of the keys.

---

append

`public void **append**(String str)`

Instead of modifying an existing token, this method allows the user to completely replace the current token with arbitrary new content.

This method may be called multiple times while processing the current token to add more and more data to the resultant HTML document. Before and/or after calling this method, the `appendToken` method may also be called explicitly in order to add the current token to the resultant HTML document.

Following is sample code illustrating how to use this method to put bold tags around all the `<a>` tags.

```
HtmlRewriter hr = new HtmlRewriter(str);
while (hr.nextTag()) {
    if (hr.getTag().equals("a")) {
        hr.append("<b>");
        hr.appendToken();
    } else if (hr.getTag().equals("/a")) {
        hr.appendToken();
        hr.append("</b>");
    }
}
```

The calls to `appendToken` are necessary. Otherwise, the `HtmlRewriter` could not know where and when to append the existing token in addition to the new content provided by the user.

Parameters:

`str` - The new content to append. May be `null`, in which case no new content is appended (the equivalent of appending "").

See Also:

`appendToken()`, `killToken()`

---

appendToken

`public void **appendToken**()`

Appends the current token to the resultant HTML document. If the caller has changed the current token using the `setTag`, `set`, or `remove` methods, those changes will be reflected.

By default, this method is automatically called after each token is processed unless the user has already appended something to the resultant HTML document. Therefore, if the user appends something and also wants to append the current token, or if the user wants to append the current token a number of times, this method must be called.

See Also:

`append(java.lang.String)`, `killToken()`

---

killToken

`public void **killToken**()`

Tells this `HtmlRewriter` not to append the current token to the resultant HTML document. Even if the user hasn't appended anything else, the current token will be ignored rather than appended.

See Also:

`append(java.lang.String)`, `killToken()`

---

accumulate

`public boolean **accumulate**(boolean accumulate)`

Turns on or off the automatic accumulation of each token.

After each token is processed, the current token is appended to to the resultant HTML document unless the user has already appended something else. By setting `accumulate` to `false`, this behavior is turned off. The user must then explicitly call `appendToken` to cause the current token to be appended.

Turning off accumulation takes effect immediately, while turning on accumulation takes effect on the next token. In other words, whether the user turns this setting off or on, the current token will not be added to the resultant HTML document unless the user explicitly calls `appendToken`.

Following is sample code that illustrates how to use this method to extract the contents of the `<head>` of the source HTML document.

```
HtmlRewriter hr = new HtmlRewriter(str);
// Don't accumulate tokens until we see the <head> below.
hr.accumulate(false);
while (hr.nextTag()) {
    if (hr.getTag().equals("head")) {
        // Start remembering the contents of the HTML document,
        // not including the <head> tag itself.
```

```
            hr.accumulate(true);
        } else if (hr.getTag().equals("/head")) {
            // Return everything accumulated so far.

            return hr.toString();
        }
    }
```

This method can be called any number of times while processing the source HTML document.

Parameters:

`accumulate` - `true` to automatically accumulate tokens in the resultant HTML document, `false` to require that the user explicitly accumulate them.

Returns:

The previous accumulate setting

See Also:

`reset()`

---

reset

`public void` **`reset`**`()`

Forgets all the tokens that have been appended to the resultant HTML document so far, including the current token.

---

pushback

`public void` **`pushback`**`()`

Puts the current token back. The next time `nextToken` is called, it will be the current token again, rather than advancing to the next token in the source HTML document.

This is useful when a code fragment needs to read an indefinite number of tokens, but that once some distinguished token is found, needs to push that token back so that normal processing can occur on that token.

---

tokenCount

`public int` **`tokenCount`**`()`

Return count of tokens seen so far

---

tagCount

`public int` **`tagCount`**`()`

Return count of tags seen so far

---

quote

`public static String **quote**(String str)`

Helper class to quote a attribute's value when the value is being written to the resultant HTML document. Values set by the `put` method are automatically quoted as needed. This method is provided in case the user is dynamically constructing a new tag to be appended with `append` and needs to quote some arbitrary values.

The quoting algorithm is as follows:
If the string contains double-quotes, put single quotes around it.
If the string contains any "special" characters, put double-quotes around it.

This algorithm is, of course, insufficient for complicated strings that include both single and double quotes. In that case, it is the user's responsibility to escape the special characters in the string using the HTML special symbols like `"` or `"`

Returns:

The quoted string, or the original string if it did not need to be quoted.

---

getMap

`public StringMap **getMap**()`

Return a copy of the StringMap of attributes.

Class LogHandler

sunlabs.brazil.handler
Class LogHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.LogHandler
```

All Implemented Interfaces:
Handler

---

public class **LogHandler**
extends Object
implements Handler

Handler for logging information about requests. Wraps another handler, and logs information about each HTTP request to a file.

Request properties:
handler
The name of the handler to wrap. This can either be the token for the class, or the class name itself.
logFile
The name of the file to log the output to. If the file already exists, data is appended to it. If the file is removed, a new one is created. If no name is specified, one is invented that contains the name and port of the server. Unless an absolute path is specified, the log file is placed in the current directory.
flush
The number of lines of logging output that may be buffered in memory before being written out to the log file. default to 25.
format
The format of the output string. Embedded strings of the form "%X" are replaced, based on the following values for "X":

- %
  A single "%"
- b
  Bytes written to the client for this request.
- d
  Time to service this request (ms).
- i
  Client ip address.
- m
  Request method (GET, POST, etc)
- M
  Memory utilization (%).
- q
  query string (if any)
- r

Requests used for this connection.

- s
  HTTP result code.
- t
  TimeStamp (ms since epoch).
- T
  Number of active threads.
- u
  URL for this request.
- v
  HTTP protocol version (10 or 11).

Defaults to "%u;%t:%d:%b".

props
: If specified This string is tacked onto the end of the "format" string. Entries in the Request Properties may be included using ${...} substitutions.

headers
: If specified This string is tacked onto the end of the "props" string. Entries in the HTTPrequest headers may be included using ${...} substitutions.

title
: if present, this is output as the first line of the file

See the `ChainSawHandler` for generating standard format log files.

Field Summary	
File	**file**
int	**flush**
String	**format**
Handler	**handler**
String	**headers**
String	**props**
String	**title**

Constructor Summary

LogHandler()	

Method Summary

boolean	**init**(Server server, String prefix) Initializes the handler.
boolean	**respond**(Request request) Dispatch the request to the handler.
static String	**subst**(Request request, String format, long duration) Format a string.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

handler

public Handler **handler**

---

props

public String **props**

---

headers

public String **headers**

---

format

```
public String format
```

---

title

```
public String title
```

---

flush

```
public int flush
```

---

file

```
public File file
```

Constructor Detail

LogHandler

```
public LogHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.
Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
                throws IOException
```

Dispatch the request to the handler. Log information if dispatched handler returns true.

Specified by:

respond in interface Handler

Tags copied from interface: Handler

Parameters:

`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

subst

```
public static String subst(Request request,
                           String format,
                           long duration)
```

Format a string. Replace %X constructs.

Class MapPage

sunlabs.brazil.handler
Class MapPage

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.MapPage
```

---

public class **MapPage**
extends Object

Utility class to rewrite links inside of web pages so they appear to come from a different site.

Note: This only works for "http", not "https".

Field Summary	
int	**count**
static boolean	**log**
Vector	**patternMap**
String	**prefix**
Hashtable	**tagMap** Initialized to all tag/attribute pairs whose attribute values are considered for rewriting.
Hashtable	**urlMap**

Constructor Summary	
MapPage(String prefix) Create a site mapper.	

Method Summary

void	**addMapEntry**(String name, String value) Add or remove an item to the map table
void	**addPatternEntry**(String pattern, String replacement) Add or remove an item to the pattern table
void	**addTag**(String name, String attribute) add a tag/attribute pair to the rewrite list.
String	**convertHtml**(String content) Rewrite all the url's in this document.
String	**convertString**(String fix) Rewrite a url inside a tag parameter.
void	**log**(String message) diagnostic output
int	**mapCount**() How many tags have been mapped?
void	**setMap**(Hashtable map) Set the map table for cross-linked sites.
void	**setPrefix**(String prefix) Change the prefix that will replace the leading "/" in a URL.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

tagMap

public Hashtable **tagMap**

Initialized to all tag/attribute pairs whose attribute values are considered for rewriting.

---

urlMap

```
public Hashtable urlMap
```

---

patternMap

```
public Vector patternMap
```

---

prefix

```
public String prefix
```

---

log

```
public static boolean log
```

---

count

```
public int count
```

Constructor Detail

MapPage

```
public MapPage(String prefix)
```

Create a site mapper. The following table maps all the entity/attribute combinations that are (or could be) URL's to (possibly) rewrite.

Parameters:

`prefix` - Every link starting with "/" has the leading slash replaced by `prefix`. If prefix is *null*, then only fully qualified url's are considered for rewriting.

Method Detail

setPrefix

```
public void setPrefix(String prefix)
```

Change the prefix that will replace the leading "/" in a URL.

---

addTag

```
public void addTag(String name,
                   String attribute)
```

add a tag/attribute pair to the rewrite list. If the attribute is null, remove the tag.

---

setMap

```
public void setMap(Hashtable map)
```

Set the map table for cross-linked sites. Foreach key in the table that matches a URL, replace the key portion of the url with the key's value.

---

addMapEntry

```
public void addMapEntry(String name,
                        String value)
```

Add or remove an item to the map table

Parameters:

`name` - The prefix of the url to match

`value` - The prefix to replace it with. If null, remove the prefix

---

addPatternEntry

```
public void addPatternEntry(String pattern,
                            String replacement)
```

Add or remove an item to the pattern table

Parameters:

`pattern` - The prefix pattern of the url to match Full tcl8.0-style regexps are supported

`replacement` - The prefix to replace it with. If null, remove the prefix. \n's are replaced by the corrosponding sub-matches in the name

Patterns are stored in a vector, with each pattern taking 3 concecutive elements: the pattern, the replacement rule, and the compiled expression. This way they are searched in order. Sample usage: http://www.([^.]*).com/ /site/\1/ will replace the url: http://www.foo.com/a/b.html with /site/foo/a/b.html

---

mapCount

```
public int mapCount()
```

How many tags have been mapped?

---

convertHtml

```
public String convertHtml(String content)
```

Rewrite all the url's in this document. This is accomplished via repeated calls to `convertString(java.lang.String)`.

Parameters:

`content` - The HTML to be processed.

Returns:

The smae HTML, will all URL's rewritten. URL's starting with "/" have the "/" replaced with the prefix. All others are re-written based on the supplied mapping tables.

---

convertString

```
public String convertString(String fix)
```

Rewrite a url inside a tag parameter.

Parameters:

`fix` - The value of the tag to be rewritten (fixed)

Returns:

null of the existing value is OK, otherwise the new value is returned

---

log

```
public void log(String message)
```

diagnostic output

Class MatchString

sunlabs.brazil.handler
Class MatchString

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.MatchString
```

---

public class **MatchString**
extends Object

Utility class for handlers to determine, based on the URL, if the current request should be processed.

Properties:
prefix
: The url prefix the url must match (defaults to "/").

suffix
: The url suffix the url must match (defaults to "").

glob
: The glob pattern the url must match. If defined, this overrides both `prefix` and `suffix`.

match
: The reqular expression pattern the url must match. If defined, this overrides `glob`.

ignoreCase
: If present and `match` is defined, this causes the regular expression match to be case insensitive. By default, case counts.

Constructor Summary

`MatchString``(String propsPrefix)` Create a matcher for per-request URL checking.
`MatchString``(String propsPrefix, Properties props)` Create a matcher for one-time-only checking.

Method Summary

boolean	**match**(String url) See if this is our url.
boolean	**match**(String url, Properties props) See if this is our url.
String	**prefix**() Return our prefix
String	**toString**() print nicely

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

MatchString

```
public MatchString(String propsPrefix)
```

Create a matcher for per-request URL checking. This constructer is used if the properties are to be evaluated on each request.

Parameters:

`propsPrefix` - The prefix to use in the properties object.

MatchString

```
public MatchString(String propsPrefix,
                  Properties props)
```

Create a matcher for one-time-only checking. This constructor is used if the properties are to be computed only once, at "init" time.

Parameters:

`propsPrefix` - The prefix to use in the properties object.

`props` - The table to find the properties in.

Method Detail

match

```
public boolean match(String url)
```

See if this is our url. Use this version for properties evaluated only at init time.

---

match

```
public boolean match(String url,
                     Properties props)
```

See if this is our url. Use this version for properties evaluated at each request.

---

prefix

```
public String prefix()
```

Return our prefix

---

toString

```
public String toString()
```

print nicely

Overrides:

toString in class Object

Class MultiHostHandler

sunlabs.brazil.handler
Class MultiHostHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.MultiHostHandler
```

All Implemented Interfaces:
Handler
Direct Known Subclasses:
RestartingMultiHostHandler

---

public class **MultiHostHandler**
extends Object
implements Handler

The `MultiHostHandler` allows the user to handle a set of host names that are all running on the same IP address. This handler looks at the http "Host" header and redispatches the request to the appropriate sub-server.

Only the main server is actually listening to the port on the specified IP address. The sub-servers are not running in separate threads. Indeed, they are not "running" at all. They exist merely as a convenient bag to hold each of the server-specific configuration parameters.

The `respond` method of the main handler for the appropriate sub-server is called directly from the `respond` method of this handler.

This handler uses the following configuration parameters:

servers
: The list of prefixes for the other servers. Each server will be initialized from the main `server.props` with the specified prefix. In this way, the configuration parameters for all the sub-servers can be stored in the same `Properties` object.

prefix.host
: Each server is started with a given *prefix*. The property *prefix*.host specifies a Glob pattern for a virtual hostname the server will be expected to handle. If this property is not specified, the server's virtual hostname will just be *prefix*. If multiple host patterns could match a given "Host" header, the first match in the "servers" list matches first.

prefix.handler
: The main handler for the server with the given *prefix*. If this property is not specified, it defaults to the `FileHandler`.

prefix.config
: Read in the file specified by "config" to initialize this sub-server's server properties. The file is expected to be in java properties format. If not specified, this sub-server shares a copy of the main server's properties, otherwise, the main server's properties are used as the "default". If this property is specified and no config file is found, then the sub-server isn't started.

The property "root", if included in the "config" file, is treated specially: If it does not represent an absolute path, then it is resolved relative to the main server's root.

prefix.log
The log level for the server with the given *prefix*. If this property is not specified, it defaults to the log level of the parent server.

A sample set of configuration parameters illustrating how to use this handler follows:

```
handler=host
port=8081
log=5

host.class=sunlabs.brazil.server.MultiHostHandler
host.servers=mars jupiter saturn

mars.host=www.mars.com
mars.log=2
mars.handler=mars.file
mars.file.class=sunlabs.brazil.server.FileHandler
mars.file.root=public_html/mars

jupiter.host=jupiter.planet.org
jupiter.handler=sunlabs.brazil.server.FileHandler
jupiter.root=public_html/jupiter

saturn.host=*.saturn.planet.org
saturn.handler=sunlabs.brazil.server.FileHandler
saturn.root=public_html/saturn
```

These parameters set up a normal `Server` on port 8081, running a `MultiHostHandler`. The `MultiHostHandler` will create three additional servers that respond to the virtual hosts "www.mars.com", "jupiter.planet.org", and ".saturn.planet.org". The "mars" server will have a `Server.prefix` of "mars", so that all other configuration parameters that the "mars" server examines can begin with "mars" and be kept distinct from the "jupiter" and "saturn" parameters.

The main server and the three sub-servers will all share the same properties object, but can use their own individual prefixes to keep their data separate (because "inherit" is not set).

Constructor Summary

`MultiHostHandler()`	

Method Summary

boolean	**init**(Server server, String prefix) Initializes the servers for the virtual hosts.
boolean	**respond**(Request request) Responds to an HTTP request by examining the "Host:" request header and dispatching to the main handler of the server that handles that virtual host.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

MultiHostHandler

```
public MultiHostHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Initializes the servers for the virtual hosts. After creating and initializing each sub-server, the `init` method of the main handler for each sub-server is called.

Specified by:

init in interface Handler

Parameters:

`server` - The HTTP server that created this handler.

`prefix` - A prefix to prepend to all of the keys that this handler uses to extract configuration information.

Returns:

`true` if at least one sub-server was found and could be initialized, `false` otherwise. Diagnostic messages are logged for each sub-server started.

---

respond

```
public boolean respond(Request request)
                throws IOException
```

Responds to an HTTP request by examining the "Host:" request header and dispatching to the main handler of the server that handles that virtual host. If the "Host:" request header was not specified, or named a virtual host that was not initialized in `init` from the list of virtual hosts, this method returns without handling the request. Port numbers are not used for host matching.

Specified by:

respond in interface Handler

Parameters:

`request` - The HTTP request to be forwarded to one of the sub-servers.

Returns:

`true` if the sub-server handled the message, `false` if it did not. `false` is also returned if the "Host:" was unspecified or unknown.

Class MultiProxyHandler

sunlabs.brazil.handler
Class MultiProxyHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.GenericProxyHandler
        |
        +--sunlabs.brazil.handler.MultiProxyHandler
```

All Implemented Interfaces:
Handler

---

public class **MultiProxyHandler**
extends GenericProxyHandler

Handler for permitting multiple cross connected virtual web sites. Each handler instance adds its prefix and destination to a static hashtable so the URL rewrite rules rewrite all of them correctly.

The `GenericProxyHandler` rewrites all of the links in each wepage to point back to the local. machine. Using this handler, if multiple virtual websites are configured, then links in one site that point to other virtual web sites are rewritten to point locally as well.

For example, suppose we have 2 MultiProxyhandlers A and B configured into the server as:

```
A.prefix=/foo/
A.host=www.foo.com
B.prefix=/bar/
B.host=www.bar.com.
```

A local request for `/foo/bar/test.html` will fetch `http://www.foo.com/bar/test.html`. If the result is an html page, then any links in that page that resolve to `www.foo.com/...` will be rewritten to the local server as `/foo/...`, and any links that points to `www.bar.com/...` will be rewritten to `/bar/...`.

Field Summary	
static Hashtable	**proxies** Holds all proxy -> prefix mappings for this server.

Fields inherited from class sunlabs.brazil.handler.GenericProxyHandler
HOST, NL, PORT, PREFIX, PROXY_HOST, PROXY_PORT

Constructor Summary

MultiProxyHandler()	

Method Summary

void	**addMap**(MapPage map) this gets called by the parent class.
boolean	**init**(Server server, String prefix) Initialize this handler.

Methods inherited from class sunlabs.brazil.handler.GenericProxyHandler
isMine, modifyContent, respond

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

proxies

public static Hashtable **proxies**

Holds all proxy -> prefix mappings for this server.

Constructor Detail

MultiProxyHandler

```
public MultiProxyHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Initialize this handler. Add rewrite mapping into the global table. If any "virtual" web sites reference other "virtual" web sites, then rewrite the links accordingly.

Overrides:

init in class GenericProxyHandler

Tags copied from interface: Handler

Parameters:

`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.

`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

addMap

```
public void addMap(MapPage map)
```

this gets called by the parent class.

Overrides:

addMap in class GenericProxyHandler

Class NotFoundHandler

sunlabs.brazil.handler
Class NotFoundHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.NotFoundHandler
```

All Implemented Interfaces:
Handler

---

public class **NotFoundHandler**
extends Object
implements Handler

Handler for returning "file not found" errors back to the client. Look for the file "NotFound.html" in the current directory, and return it if it exists. Otherwise, return the "NotFound.html" file in the document root directory. If neither can be found, then punt, and let someone else deal with it.

If more sophisticated processing is desired, then the `UrlMapperHandler` may be used in combination with the `TemplateHandler`.

Configuration parameters understood by this handler
root
: The location of the document root for locating the default "not found" file (also looks using prefix of "").

prefix, suffix, glob, match
: Specify the URL that triggers this handler. (See `MatchString`).

fileName
: The name of the file to send for missing files. Defaults to "notfound.html"

type
: The file type, defaults to text/html

Constructor Summary

`NotFoundHandler()`	

Method Summary

boolean	**init**(Server server, String prefix) Extract the handler properties.
boolean	**respond**(Request request) Look for and deliver the "not found" file Look in the current directory first, then in the doc root.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

NotFoundHandler

```
public NotFoundHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Extract the handler properties. Get the URL prefix and default "missing" file name.

Specified by:

init in interface Handler

Tags copied from interface: Handler

Parameters:

`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.

`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

respond

```
public boolean respond(Request request)
               throws IOException
```

Look for and deliver the "not found" file Look in the current directory first, then in the doc root. Only files whose suffixes have valid mime types are delivered.

Specified by:

respond in interface Handler

Tags copied from interface: Handler

Parameters:

`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

Class PollHandler

sunlabs.brazil.handler
Class PollHandler

```
java.lang.Object
  |
  +--java.lang.Thread
        |
        +--sunlabs.brazil.handler.PollHandler
```

All Implemented Interfaces:
Handler, Runnable
Direct Known Subclasses:
RePollHandler

---

public class **PollHandler**
extends Thread
implements Handler

Handler for periodically polling another web site, whose results are (optionally) added to the server's properties. This also includes the ability to request URL's on a cron-like schedule.

The result of fetching the url is expected to be a text document in java Properties format.

Properties:
url
The fully qualified URL to fetch periodically. any ${...} constructs are evaluated at each poll, with the values in the server properties object.
post
The "post" data, if any. ${...} are evaluates as per `url` above.
headers
A list of white space delimited tokens that refer to additional HTTP headers that are added onto the polled request. For each token the server properties `[token].name` and `[token].value` define a new http header.
interval
The interval (in seconds) to fetch the url. Defaults to 10 seconds. If *match* is specified, this is the interval used to check for a time/date match. At each "interval", the current time format is computed, based on "format", below. If the computed format has not changed since the previous poll, then no poll is done. The interval is recalculated after each poll.
fast
If set, don't wait "interval" before 1st poll.
prepend
The string to prepend to the properties. If not supplied no properties are loaded.
namespace
The namespace to use to store the properties to. If the `sessionTable` (see below)parameter is identical to the `sessionTable` parameter of the `SetTemplate`, then this specifies the `namespace` parameter that may be used with the `SetTemplate` "namespace" parameter to obtain

the extracted data. Defaults to the "prepend" parameter.

match
: If specified, a regular expression that must match the current time for this URL to run. The format to match is specified by the "format" parameter, below. "EEE-dd-HH-mm" (eg: Thu-Dec, 14, 14:12 pm).

format
: a `date format specifier` to use for matching "match" patterns. Defaults to "EE-MM-dd-HH-mm".

proxy
: If specified, connect through a proxy. This should be in the form `host:port`, or `host` it the desired port is `80`.

sessionTable
: The name of the SessionManager table to use for storing values. By default, properties are stored in server.props. The value should match the `sessionTable` used by the `SetTemplate` to allow values obtained by this handler to be accessable from within templates.

  If the `sessionTable` is set, the `namespace` value is used to name the table (e.g. the `namespace` specified by `SetTemplate`. If no namespace parameter is given, then `prepend` is used as the namespace parameter.

If prepend is specified, the following additional properties are created, and added to the properties with the specified prefix.

count.attempts
: The total number of polls attemped.

count.errors
: The total number of poll failures.

error.at
: The poll attempt # for the last failure.

error.msg
: The message describing the last failure.

error.time
: The timestamp of the last failure.

timestamp
: The timestamp for the last successful poll.

Field Summary	
int	**interval**
String	**post**
String	**url**

Fields inherited from class java.lang.Thread
MAX_PRIORITY, MIN_PRIORITY, NORM_PRIORITY

Constructor Summary

PollHandler()	

Method Summary

void	**fillProps**(Properties props, HttpRequest target) Fill the properties from the input stream
boolean	**init**(Server server, String prefix) Set up the initial configuration, and kick off a thread to periodically fetch the url.
boolean	**respond**(Request request) This might allow control over the polling via requests at a later date.
void	**run**() Periodically poll the url, and copy the results into the server properties.

Methods inherited from class java.lang.Thread
activeCount, checkAccess, countStackFrames, currentThread, destroy, dumpStack, enumerate, getContextClassLoader, getName, getPriority, getThreadGroup, interrupt, interrupted, isAlive, isDaemon, isInterrupted, join, join, join, resume, setContextClassLoader, setDaemon, setName, setPriority, sleep, sleep, start, stop, stop, suspend, toString, yield

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

url

```
public String url
```

---

post

```
public String post
```

---

interval

```
public int interval
```

Constructor Detail

PollHandler

```
public PollHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Set up the initial configuration, and kick off a thread to periodically fetch the url.

Specified by:

init in interface Handler

Tags copied from interface: Handler

Parameters:

`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.

`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
```

This might allow control over the polling via requests at a later date. For now, it always returns false.

Specified by:

respond in interface Handler

Tags copied from interface: Handler

Parameters:

`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

run

```
public void run()
```

Periodically poll the url, and copy the results into the server properties.

Overrides:

run in class Thread

---

fillProps

```
public void fillProps(Properties props,
                      HttpRequest target)
               throws IOException
```

Fill the properties from the input stream

Class PropertiesHandler

sunlabs.brazil.handler
Class PropertiesHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.PropertiesHandler
```

All Implemented Interfaces:
Handler

---

public class **PropertiesHandler**
extends Object
implements Handler

Handler for returning selected request properties as a text/plain document in java properties format. A server using this handler may be called by a server using the `ProxyPropertiesHandler` to communicate per-request properties between the two servers.

Properties:
prefix, suffix, glob, match
Specify the URL that triggers this handler. (See `MatchString`).
select
Glob pattern to match properties selected (Defaults to *). This is re-examined at every request.
type
Type of output to generate (defaults to text/plain).
comment
Comment to put on output (defaults to *select*).

Constructor Summary

`PropertiesHandler``()`	

Method Summary

`boolean`	**`init`**`(Server server, String prefix)` Initializes the handler.
`boolean`	**`respond`**`(Request request)` If this is one of our URL's, look through each request property, and selct those that match the *Select* property.

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

PropertiesHandler

```
public PropertiesHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.
Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
                throws IOException
```

If this is one of our URL's, look through each request property, and selct those that match the *Select* property. Then emit them all as text/plain.

Specified by:

respond in interface Handler

Tags copied from interface: Handler

Parameters:

`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

Class ProxyPropertiesHandler

sunlabs.brazil.handler
Class ProxyPropertiesHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.GenericProxyHandler
        |
        +--sunlabs.brazil.handler.ProxyPropertiesHandler
```

All Implemented Interfaces:
Handler

---

public class **ProxyPropertiesHandler**
extends GenericProxyHandler
implements Handler

Obtain properties format content from remote websites, and add it to the current request properties. Many of the handlers are designed to produce side effects, by inserting values into the request properties (see `PropertiesHandler`). If they are instead configured to produce the properties in java properties format, then this handler will read their output, and place the result in the request object on their behalf. This capability allows certain handlers to be run on other web sites, yet behave as if they are in the handler chain. The following request properties are used:

type
: The document type for files to process as java properties (defaults to text/plain)

prepend
: The prefix that should be prepended to each property before it is inserted into the request properties

url
: The url that should be used to fetch the remote content. If not specified, the curent url is used instead. Any ${...} constructs in the url are evaluated at each request.

NOTE: This capability should be generalized.

Fields inherited from class sunlabs.brazil.handler.GenericProxyHandler
`HOST, NL, PORT, PREFIX, PROXY_HOST, PROXY_PORT`

Constructor Summary

`ProxyPropertiesHandler()`	

Method Summary

boolean	**init**(Server server, String prefix) Initializes the handler.
byte[]	**modifyContent**(Request request, byte[] content) Rewrite the links in an html file so they resolve correctly in proxy mode.
boolean	**respond**(Request request) Responds to an HTTP request.

Methods inherited from class sunlabs.brazil.handler.GenericProxyHandler
addMap, isMine

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ProxyPropertiesHandler

```
public ProxyPropertiesHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Overrides:
init in class GenericProxyHandler
Tags copied from interface: Handler

Parameters:

`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.

`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
               throws IOException
```

Description copied from interface: Handler

Responds to an HTTP request.

Specified by:

respond in interface Handler

Overrides:

respond in class GenericProxyHandler

Tags copied from interface: Handler

Parameters:

`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

modifyContent

```
public byte[] modifyContent(Request request,
                            byte[] content)
```

Description copied from class: GenericProxyHandler

Rewrite the links in an html file so they resolve correctly in proxy mode.

Overrides:

modifyContent in class GenericProxyHandler

Tags copied from class: GenericProxyHandler

Parameters:

`request` - The original request to this "proxy"

`headers` - The vector of mime headers for the proxy request

Returns:

true if the headers and content should be sent to the client, false otherwise Modifies "headers" as a side effect

Class PublishHandler

sunlabs.brazil.handler
Class PublishHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.PublishHandler
```

All Implemented Interfaces:
Handler

---

public class **PublishHandler**
extends Object
implements Handler

Handler for supporting publishing from Communicator. Launches an authentication handler to protect the content from malicious users.

Looks for `PUT` requests, and creates or modifies the content as indicated.

The following request properties are used:
prefix, suffix, glob, match
Specify the URL that triggers this handler. (See `MatchString`).
session
The the name of request property holding the session information to provide the credentials for posting. The default is "SessionID".

Field Summary

String	**propsPrefix**
String	**session**

Constructor Summary

PublishHandler()	

Method Summary

boolean	**init**(Server server, String prefix) Start up the authentication handler.
boolean	**respond**(Request request) Make sure this is one of our "PUT" requests.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

session

public String **session**

---

propsPrefix

public String **propsPrefix**

Constructor Detail

PublishHandler

public **PublishHandler**()

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Start up the authentication handler.

Specified by:

init in interface Handler

Tags copied from interface: Handler

Parameters:

`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.

`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
                throws IOException
```

Make sure this is one of our "PUT" requests. Look up the credentials for this request. If no credentials are found, prompt the user for them. IF OK, save file to proper spot.

Specified by:

respond in interface Handler

Tags copied from interface: Handler

Parameters:

`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

Class PushHandler.Split

sunlabs.brazil.handler
Class PushHandler.Split

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.PushHandler.Split
```

Enclosing class:
PushHandler

---

public static class **PushHandler.Split**
extends Object

Split multipart data into its constituent pieces. Use byte[] so we can handle (potentially) large amounts of binary data. This acts as an iterator, stepping through the parts, extracting the appropriate info for eacxh part.

Constructor Summary

PushHandler.Split(byte[] bytes) create a new multipart form thingy	

Method Summary

String	**content**() Get the content as a string
String	**header**() Return the header as a string
int	**length**() Return the content length
String	**name**() get the part name
boolean	**nextPart**() Return true if there is a next part
int	**start**() return the index into the start of the data for this part

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

PushHandler.Split

`public` **`PushHandler.Split`**`(byte[] bytes)`

create a new multipart form thingy

Method Detail

nextPart

`public boolean` **`nextPart`**`()`

Return true if there is a next part

---

content

`public String` **`content`**`()`

Get the content as a string

---

length

`public int` **`length`**`()`

Return the content length

---

start

`public int` **`start`**`()`

return the index into the start of the data for this part

---

header

```
public String header()
```

Return the header as a string

---

name

```
public String name()
```

get the part name

Class PushHandler

sunlabs.brazil.handler
Class PushHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.PushHandler
```

All Implemented Interfaces:
Handler

---

public class **PushHandler**
extends Object
implements Handler

Skeleton Handler for uploading files using multipart/form-data. Application specific functionality is added by overriding `processData(sunlabs.brazil.server.Request)`.
NOTE: Most applications will want to use the `MultipartSetTemplate` to deal with multipart/form data.

Properties:
prefix, suffix, glob, match
Specify the URL that triggers this handler. (See `MatchString`).

Inner Class Summary

static class	**PushHandler.Split** Split multipart data into its constituent pieces.

Constructor Summary

PushHandler()	

Method Summary

static int	**indexOf**(byte[] src, int srcStart, int srcEnd, byte[] dst, int dstStart, int dstEnd) Find the index of dst in src or -1 if not found > This is the byte array equivalent to string.indexOf()
static int	**indexOf**(byte[] src, int srcStart, int srcEnd, String dst)
boolean	**init**(Server server, String prefix) Initializes the handler.
boolean	**processData**(Request request) process the data - this doesn't currently do anything useful.
boolean	**respond**(Request request) Make sure this is one of our requests.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

PushHandler

```
public PushHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler

Parameters:

`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.

`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
                throws IOException
```

Make sure this is one of our requests. IF OK, save file to proper spot.

Specified by:

respond in interface Handler

Tags copied from interface: Handler

Parameters:

`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

processData

```
public boolean processData(Request request)
```

process the data - this doesn't currently do anything useful.

---

indexOf

```
public static int indexOf(byte[] src,
                          int srcStart,
                          int srcEnd,
                          byte[] dst,
                          int dstStart,
                          int dstEnd)
```

Find the index of dst in src or -1 if not found > This is the byte array equivalent to string.indexOf()

---

indexOf

```
public static int indexOf(byte[] src,
                          int srcStart,
                          int srcEnd,
                          String dst)
```

Class RePollHandler.Extract

sunlabs.brazil.handler
Class RePollHandler.Extract

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.RePollHandler.Extract
```

Enclosing class:
RePollHandler

---

public static class **RePollHandler.Extract**
extends Object

A "bag" to store regular expression extraction instructions

Method Summary	
boolean	**extract**()
int	**extract**(Properties props, String data) Do the extraction
boolean	**fail**()
boolean	**replace**() true if the previous replace worked
String	**replace**(String data) Do the substitution
boolean	**reset**()
boolean	**result**()
boolean	**succeed**()
String	**toString**()

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Method Detail

replace

```
public String replace(String data)
```

Do the substitution

---

extract

```
public int extract(Properties props,
                   String data)
```

Do the extraction

Parameters:

`props` - where to put the extracted properties

`data` - the data to extract from

Returns:

The number of extractions performed.

---

replace

```
public boolean replace()
```

true if the previous replace worked

---

extract

```
public boolean extract()
```

---

result

```
public boolean result()
```

---

reset

```
public boolean reset()
```

---

fail

```
public boolean fail()
```

---

succeed

```
public boolean succeed()
```

---

toString

```
public String toString()
```

Overrides:
toString in class Object

Class RePollHandler

sunlabs.brazil.handler
Class RePollHandler

```
java.lang.Object
  |
  +--java.lang.Thread
        |
        +--sunlabs.brazil.handler.PollHandler
              |
              +--sunlabs.brazil.handler.RePollHandler
```

All Implemented Interfaces:
Handler, Runnable
Direct Known Subclasses:
TclRePollHandler

---

public class **RePollHandler**
extends PollHandler

Do regsub processing on content to extract properties.

Properties:
encoding
The character set encoding to use when converting the request results to a string. Defaults to the default encoding.
prepend
The string to prepend to all properties. Extracted properties will contain the the "re" token as an additional prefix.
re
the list of "re" tokens to process in order. Each "re" token has the following attributes:
re.exp
The regular expression to search for.
re.sub
The regular expression substitution pattern. If ’E’ is specified, the substitution is done after the extraction.
re.names
A white-space delimited set of tokens to use instead of numerical indices to name the properties. The first name in the list names the entire match, the remaining names name the sub-expressions. If there are more properties extracted than names provided, the "left over" properties will have numerical indeces. This implies ’E’.

If the name "X" is used, no property will be extracted for that match.
re.key
The index of the sub-match (starting at 1) that will be used to name the row number portion of the property name instead of a counter. This is useful if one of the sub-matches will be unique for each matching pattern. This option is ignored if the "O" flag is specified, as there will be only one match so

no "key" is required.

re.flags

One or more ASCII flags to control how this "re" is processed. Consists of one or more of The following (defaults to "SFE"):. Characters not on this list are ignored.

- **E** Extract current result into server properties. See the rules for naming the properties, below. At least one regular expression Must have an "E" flag.
- **F** Process if previous "RE" failed.
- **I** Ignore case in expression
- **O** only do one substitution or extraction, not all
- **R** Reset content to original before proceeding Otherwise, the result of the previous substitution (if any) is used.
- **S** Process if previous "RE" succeeded

First remote content is obtained. Then each regular expression token is processed in turn for the purpose of extracting portions of that content into server properties. [re].sub is used to transform the content before attempting to extract properties.

Content is extracted into the following properties.

prepend.[re].[m].[n]

The result of the expression associated with token "re". ’n’ is the sub-expression number, and ’m’ is the match number, both starting at ’0’. If the ’O’ flag is specified, there can only be one value for ’m’, so it is not included (e.g. the name of the property will be "prepend.[re].[n]).

prepend.[re].matches

A list of matches, that may be used as an iterator to **foreach**.

prepend.[re].subexpressions

The number of sub-expressions associated with [re].

Inner Class Summary

`static class`	**`RePollHandler.Extract`** A "bag" to store regular expression extraction instructions

Fields inherited from class sunlabs.brazil.handler.PollHandler
`interval, post, url`

Fields inherited from class java.lang.Thread
`MAX_PRIORITY, MIN_PRIORITY, NORM_PRIORITY`

Constructor Summary

RePollHandler()	

Method Summary

void	**fillProps**(Properties props, HttpRequest target) Fill the properties by extracting fields from the response.
boolean	**init**(Server server, String prefix) Set up the initial configuration, and kick off a thread to periodically fetch the url.
void	**processText**(Properties props, String data) Process the contents as a string through the regular expressions.
boolean	**respond**(Request request) Allow The url and post data (if any) to be changed.

Methods inherited from class sunlabs.brazil.handler.PollHandler
run

Methods inherited from class java.lang.Thread
activeCount, checkAccess, countStackFrames, currentThread, destroy, dumpStack, enumerate, getContextClassLoader, getName, getPriority, getThreadGroup, interrupt, interrupted, isAlive, isDaemon, isInterrupted, join, join, join, resume, setContextClassLoader, setDaemon, setName, setPriority, sleep, sleep, start, stop, stop, suspend, toString, yield

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

RePollHandler

```
public RePollHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from class: PollHandler
Set up the initial configuration, and kick off a thread to periodically fetch the url.
Overrides:
init in class PollHandler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.
Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
```

Allow The url and post data (if any) to be changed. A query parameter of the form "url=xxx" replaces the current url. A query parameter of the form "post=xxx" replaces the post data, if any was initially defined.
Overrides:
respond in class PollHandler
Tags copied from interface: Handler
Parameters:
`request` - The `Request` object that represents the HTTP request.
Returns:
`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

fillProps

```
public void fillProps(Properties props,
                      HttpRequest target)
               throws IOException
```

Fill the properties by extracting fields from the response. This overrides fillProps.

Overrides:

fillProps in class PollHandler

---

processText

```
public void processText(Properties props,
                        String data)
```

Process the contents as a string through the regular expressions. This is public, and separate from fillProps to make unit testing easier.

Class ReflectHandler

sunlabs.brazil.handler
Class ReflectHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.ReflectHandler
```

All Implemented Interfaces:
Handler

---

public class **ReflectHandler**
extends Object
implements Handler

Handler for reflecting query data back to the client. This is the example handler to demonstrate how a typical handler is witten. If query data is present, it is formatted into an HTML table, and displayed to the user.

Constructor Summary

ReflectHandler()	

Method Summary

static String	**formatTable**(Dictionary data, String caption) Turn a hash table into html format.
boolean	**init**(Server server, String prefix) Initialize the handler.
boolean	**respond**(Request request) Dispatch and handle the request.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ReflectHandler

```
public ReflectHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Initialize the handler. Handler objects are created by the server using newInstance(). The init method is called first, and exactly one for each instance, and may be used for one-time initializations. This handler doesn't require any.

Specified by:

init in interface Handler

Parameters:

`server` - A reference to the server.

`prefix` - A string identifying this instance of the handler. It is used by the `ChainHandler` to provide the prefix to be prepended onto each property intended for this handler.

Returns:

true Only if the handler is successfully initialized.

---

respond

```
public boolean respond(Request request)
               throws IOException
```

Dispatch and handle the request. This version justs reflects the HTTP header information. It is commonly placed downstream of the `CgiHandler` to allow HTML forms to be tested before the cgi script is written.

Specified by:

respond in interface Handler

Parameters:

`request` - The request object contains all of the information about the request, as well as methods to manipulate it. Although multiple threads may call this method connurrently, each will have its own request object.

---

formatTable

```
public static String formatTable(Dictionary data,
                                 String caption)
```

Turn a hash table into html format. This is a static method so it may be used in other handlers.

Parameters:

`table` - The table to format

Returns:

The html fragment

Class ResourceHandler

sunlabs.brazil.handler
Class ResourceHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.ResourceHandler
```

All Implemented Interfaces:
Handler

---

public class **ResourceHandler**
extends Object
implements Handler

Handler for serving documents out of the jar file. Look for url's as resources, presumably in the same "jar" file as the class files. This allows an entire web site to be included in the jar file. A typical way to use this handler (with java 1.2+) is as follows:

- Add an existing web site to the jar file with:

  ```
  jar uf [jar file] [web site]
  ```

- Create a config file, and add it to the jar file as: `sunlabs/brazil/main/config`. See `Main.java` for more info.
- Create a *startup* file containing:

  ```
  Main-Class: sunlabs.brazil.server.Main
  ```

  , and add it to the manifest with:

  ```
  jar ufm [jar file] [startup file]
  ```

- Start the server with:

  ```
  java -jar [jar file] [optional server options....]
  ```

if no suffix is provided, and the "directory" exists, a redirect is issued, to add-on the trailing slash. The following server properties are used:

root
: The document root path within the jar file

prefix, suffix, glob, match
: Specify the URL that triggers this handler. (See `MatchString`).

default
: The default file name for url references ending in /

mime.xxx
: The mime type for suffix xxx. See `FileHandler` for a description of how to set mime types for url suffixes.

Constructor Summary

ResourceHandler()	

Method Summary

static byte[]	**getResourceBytes**(Properties props, String prefix, String file) Find a file blob as a resource in our jar file (experimental).
static InputStream	**getResourceStream**(Properties props, String prefix, String file) Look for a file in the filesystem.
static String	**getResourceString**(Properties props, String prefix, String file)
boolean	**init**(Server server, String prefix) Initializes the handler.
boolean	**respond**(Request request) Responds to an HTTP request.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ResourceHandler

public **ResourceHandler**()

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.
Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
               throws IOException
```

Description copied from interface: Handler
Responds to an HTTP request.
Specified by:
respond in interface Handler
Tags copied from interface: Handler
Parameters:
`request` - The `Request` object that represents the HTTP request.
Returns:
`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.
Throws:
IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

getResourceStream

```
public static InputStream getResourceStream(Properties props,
                                            String prefix,
                                            String file)
                                      throws IOException
```

Look for a file in the filesystem. If its not there, see if we can find a resource in our jar file. Relative paths are resolved with respect to the document root.

Parameters:

`props` - where to look for server root property
`prefix` - "
`file` - The pseudo file to find as a resource

Returns:

The input stream (or null)

---

getResourceString

```
public static String getResourceString(Properties props,
                                       String prefix,
                                       String file)
                                 throws IOException
```

---

getResourceBytes

```
public static byte[] getResourceBytes(Properties props,
                                      String prefix,
                                      String file)
                                throws IOException
```

Find a file blob as a resource in our jar file (experimental).

Parameters:

`props` - where to look for server root property
`prefix` - "
`file` - The pseudo file to find as a resource

Returns:

The data, if available, or raises an exception.

Class ResourceLimitHandler

sunlabs.brazil.handler
Class ResourceLimitHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.ResourceLimitHandler
```

All Implemented Interfaces:
Handler

---

public class **ResourceLimitHandler**
extends Object
implements Handler

Handler for server resource management. This handler monitors various system load parameters, and rejects each request with a short message if any resource limit is exceeded. The properties are evaluated at init time, to minimize the per-request overhead of this monitor.

Properties:
memory
The minimum # of remaining bytes available to the vm
threads
The Max number of active threads
file
The file name or resource of the html file to return if resources run low. Defaults to "busy.html".
retry
The number of seconds to request the client wait before retrying the request.

Constructor Summary

ResourceLimitHandler()	

Method Summary

boolean	**init**(Server server, String prefix) Initializes the handler.
boolean	**respond**(Request request) Responds to an HTTP request.

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

ResourceLimitHandler

```
public ResourceLimitHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.
Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
                throws IOException
```

Description copied from interface: Handler
Responds to an HTTP request.

Specified by:

respond in interface Handler

Tags copied from interface: Handler

Parameters:

`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

Class RestartingMultiHostHandler

sunlabs.brazil.handler
Class RestartingMultiHostHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.MultiHostHandler
        |
        +--sunlabs.brazil.handler.RestartingMultiHostHandler
```

All Implemented Interfaces:
Handler

---

public class **RestartingMultiHostHandler**
extends MultiHostHandler

Allow the configuration for one virtual host to be restarted.

Constructor Summary

RestartingMultiHostHandler()	

Methods inherited from class sunlabs.brazil.handler.MultiHostHandler
init, respond

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

RestartingMultiHostHandler

```
public RestartingMultiHostHandler()
```

Class RestrictClientHandler

sunlabs.brazil.handler
Class RestrictClientHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.RestrictClientHandler
```

All Implemented Interfaces:
Handler

---

public class **RestrictClientHandler**
extends Object
implements Handler

Simple access control hander based on source ip addresses. Compare the ip address of the client with a regular expression. Only allow access to the specified url prefix if there is a match.

Properties:
prefix, suffix, glob, match
: Specify the URL that triggers this handler. (See `MatchString`).

allow
: The regular expression that matches the ip addresses of clients (in xxx.xxx.xxx.xxx format) that are permitted to access url's starting with `prefix`.

deny
: The regular expression that matches the set of ip **names** that should be denied access. This is to make complying with silly EAR requirements easier. The use of this option implies a reverse DNS lookup, which could be expensive, as DNS names (and not ip addresses) are used for the comparison. Case insensitive matching is used.

redirect
: Name of the url to re-direct to if permission is denied. If not specified, a simple message is sent to the client.

Constructor Summary

`RestrictClientHandler()`	

Method Summary

boolean	**init**(Server server, String prefix) Initializes the handler.
boolean	**respond**(Request request) Responds to an HTTP request.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

RestrictClientHandler

```
public RestrictClientHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.
Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

respond

```
public boolean respond(Request request)
               throws IOException
```

Description copied from interface: Handler
Responds to an HTTP request.
Specified by:
respond in interface Handler
Tags copied from interface: Handler
Parameters:
`request` - The `Request` object that represents the HTTP request.
Returns:
`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.
Throws:
IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

Class RolesHandler

sunlabs.brazil.handler
Class RolesHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.RolesHandler
```

All Implemented Interfaces:
Handler

---

public class **RolesHandler**
extends Object
implements Handler

Handler for associating roles with an id. This is a placeholder until the SunEconomy gets integrated in. It looks for an "id" in the request, looks it up in a property file, then adds the value of the id into the request. It may be used in conjunction with `AclSwitchHandler` to provide role based web access. Properties:

prefix, suffix, glob, match
: Specify the URL that triggers this handler. (See `MatchString`).

SessionID
: The property to use to look up the id. Defaults to "SessionID".

roleName
: The property to place the result of the id lookup into. Defaults to "roleName";

mapFile
: The absolute path to the java properties file containing the it to role mapping.

Field Summary

static String	**ID_KEY** Handler configuration property **SessionID**.
static String	**MAP** Handler configuration property **mapFile**.
static String	**ROLE_KEY** Handler configuration property **roleName**.

Constructor Summary

RolesHandler()	

Method Summary

boolean	**init**(Server server, String prefix) Initializes the handler.
boolean	**respond**(Request request) Dispatch and handle the request.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

ID_KEY

public static final String **ID_KEY**

Handler configuration property **SessionID**. The request property name to find the id string. Defaults to id.

ROLE_KEY

public static final String **ROLE_KEY**

Handler configuration property **roleName**. The request property name to place the roles into. Defaults to roles.

MAP

public static final String **MAP**

Handler configuration property **mapFile**. The path to the java properties file containing the id to roles mapping. The roles are a whitespace delimited list of ascii role names.

Constructor Detail

RolesHandler

```
public RolesHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.
Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
                throws IOException
```

Dispatch and handle the request. This version looks at the request for the id, looks it up in the table, and adds the value, if available
Specified by:
respond in interface Handler
Tags copied from interface: Handler
Parameters:
`request` - The `Request` object that represents the HTTP request.
Returns:
`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.
Throws:
IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

Class SMTPHandler

sunlabs.brazil.handler
Class SMTPHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.handler.SMTPHandler
```

All Implemented Interfaces:
Handler, TemplateInterface

---

public class **SMTPHandler**
extends Template
implements Handler

Handler (or template) for Sending an email message via SMTP. The following server properties are used:
prefix, suffix, glob, match
Specify the URL that triggers this handler. (See `MatchString`).
host
The mail host (e.g. listening on the SMTP port). Defaults to "localhost".

Template to send an email message via SMTP.

The <sendmail> tag looks for the attributes:
to
The message recipient(s) [required] Recipients are be delimited by one or more commas (,), spaces or tabs. This is the actual destination list. If "To:" headers are desired, they should be incorporated using `<param>` tags (see below).
from
The originator [required]
body
The text message [optional]
subject
The subject [optional]
If an error occurred communicating with the smtp server, the property `[prefix].error` will contain the error message.

There are 2 ways of using the `sendmail` template:

1. <sendmail from=... to=... body=... subject=... />
2. <sendmail from=... to=... body=... subject=... >
   <param name=... value=... >
   ...
   <param name=... value=... > </sendmail>

The second method is useful when additonal email headers are required. Each `param` tag adds an additional header to the email message. There may be multiple headers of the same `name`, and their order is preserved.

When a singleton tag is used, the `To:` header is filled in to match the actual recipients. If you need to specify `cc`, `bcc`, or other mail headers use `param` tags.

Note:
The `to` attribute, which is required, specifies the actual recipients. When `to` is specified as part of a `param` tag, it is the recipient list presented to the email recipient, which may have nothing to do wih the actual recipients. In the singleton case, they are made the same.

Send an email message based on the query data

Query parameters:

to
: To address. Multible addresses should be delimited by spaces or commas.

from
: From address

message
: Message

subject
: Subject

Either the message or subject may be null, but not both.

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary	
`SMTPHandler()`	

Method Summary

boolean	**done**(RewriteContext hr) If we run off the end of the page, but there is email pending to be sent, send it anyway.
boolean	**init**(RewriteContext hr) Called before this template processes any tags.
boolean	**init**(Server server, String prefix) Initializes the handler.
boolean	**respond**(Request request) Responds to an HTTP request.
static void	**smtp**(String fromHost, String smtpHost, String from, String to, String body, StringMap headers) Send an email message via smtp - simple version.
static void	**smtp**(String fromHost, String smtpHost, String from, String to, String body, StringMap headers, Server server)
void	**tag_param**(RewriteContext hr) Add an additional email header.
void	**tag_sendmail**(RewriteContext hr) set-up an email message for sending.
void	**tag_slash_sendmail**(RewriteContext hr) If we haven't sent the mail yet - send it.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

SMTPHandler

public **SMTPHandler**()

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.
Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
                throws IOException
```

Description copied from interface: Handler
Responds to an HTTP request.
Specified by:
respond in interface Handler
Tags copied from interface: Handler
Parameters:
`request` - The `Request` object that represents the HTTP request.
Returns:
`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.
Throws:
IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

init

```
public boolean init(RewriteContext hr)
```

Description copied from class: Template
Called before this template processes any tags.
Overrides:
init in class Template

---

tag_sendmail

```
public void tag_sendmail(RewriteContext hr)
```

set-up an email message for sending. If this is a singleton tag, then send the mail.

---

tag_slash_sendmail

```
public void tag_slash_sendmail(RewriteContext hr)
```

If we haven't sent the mail yet - send it.

---

done

```
public boolean done(RewriteContext hr)
```

If we run off the end of the page, but there is email pending to be sent, send it anyway.
Overrides:
done in class Template

---

tag_param

```
public void tag_param(RewriteContext hr)
```

Add an additional email header. Headers are added in the order in which they are processed. The special header "body" changes the body value instead.

look for: `<param name="..." value="...>`. Which will add the email header: `name: value`.

The name "body" is special, and will cause the email body to be replaced.

---

smtp

```
public static void smtp(String fromHost,
                        String smtpHost,
                        String from,
                        String to,
                        String body,
                        StringMap headers)
                 throws IOException
```

Send an email message via smtp - simple version.

Parameters:

`fromHost` - the hostname of the sender (may be null)

`smtpHost` - the SMTP host (whose smtp daemon to contact)

`from` - who the email is from

`to` - a space delimited list of recepients

`body` - The message body

`headers` - message headers (may be null)

Throws:

IOException, - if any errors occured (yuk) Either the headers Or body may be null, but not both.

---

smtp

```
public static void smtp(String fromHost,
                        String smtpHost,
                        String from,
                        String to,
                        String body,
                        StringMap headers,
                        Server server)
                 throws IOException
```

Class SimpleSessionHandler

sunlabs.brazil.handler
Class SimpleSessionHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.SimpleSessionHandler
```

All Implemented Interfaces:
Handler

---

public class **SimpleSessionHandler**
extends Object
implements Handler

Handler for creating browser sessions based on information found in the http request. This handler provides a single session-id that may be used by other handlers.

The following server properties are used:

`prefix, suffix, glob, match`
: Specify the URL that triggers this handler (See `MatchString`).

`session`
: The name of the request property that the Session ID will be stored in, to be passed to downstream handlers. The default value is "SessionID". If the property already exists, and is not empty, no session will be defined (unless force=true).

`extract`
: If specified, a string to use as the session-id. ${...} values will be searched for first in the HTTP header values, and then in the request properties.

  In addition to the actual HTTP headers, the pseudo http headers `ipaddress, url, method, and query` are made available for ${...} substitutions.

`re`
: If specified, a regular expression that the extracted data must match. if it doesn't match, no session id is installed. The default is ".", which matches any non-empty string. If the first character is "!" then the sense of the match is inverted, But only for determining whether a match "succeeded" or not. no sub-matches may be used in computing the key value in this case.

`value`
: The value of the session ID. May contain & or \n (n=0,1,2...) constructs to substitute matched sub-expressions of `re`. The default is "&" , which uses the entire string "extract" as the `session` id. ${...} are substituted (but not \'s) for `value` before looking for '\n' sequences that are part of the regular expression matches.

`digest`
: If set, the "value" is replaced by the base64 encoding of the MD5 checksum of `value`.

`force`
: If set (to anything), a session ID is set even if one already exists.

If no options are provided, the client's IP address is used as the session ID.

Examples:
Pick the session based on the browser

```
[prefix].extract=${user-agent}
[prefix].re=.*(Netscape|Lynx|MSIE).*
[prefix].value=\\1
```

This is similar to the "old" behavior.

```
[prefix].extract=${user-agent}${ipaddress}
[prefix].digest=true
```

Look for a special authorization token, and set a request property to the value

```
[prefix].extract=${Authorization}
[prefix].re=code:([0-9]+)
[prefix].value=id\\1
```

Field Summary

Regexp	**regexp**
String	**value**

Constructor Summary

SimpleSessionHandler()	

Method Summary

boolean	**init**(Server server, String prefix) Initializes the handler.
boolean	**respond**(Request request) Responds to an HTTP request.

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

value

```
public String value
```

---

regexp

```
public Regexp regexp
```

Constructor Detail

SimpleSessionHandler

```
public SimpleSessionHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.
Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
               throws IOException
```

Description copied from interface: Handler
Responds to an HTTP request.
Specified by:
respond in interface Handler
Tags copied from interface: Handler
Parameters:
`request` - The `Request` object that represents the HTTP request.
Returns:
`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.
Throws:
IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

Class SupplyHandler

sunlabs.brazil.handler
Class SupplyHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.SupplyHandler
```

All Implemented Interfaces:
Handler

---

public class **SupplyHandler**
extends Object
implements Handler

Sample Handler for dispatching different users to different url's based on a special http authentication header. This is a re-implementation of the supplier.net content server using the new server apis (e.g. its not used for anything anymore). This handler was originally designed to be a "virtual web site", where credentials are passed in from an upstream proxy. Those credentials are used to provide different views based on the particular credentials supplied. The following configuration properties are used:
mapFile
: properties file
prefix
: url prefix
default
: default map
header
: http header (authentication)
realm
: The authentication realm (basic)

Constructor Summary

SupplyHandler()	

Method Summary

boolean	**init**(Server server, String prefix) Initializes the handler.
boolean	**respond**(Request request) Dispatch and handle the request.

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

SupplyHandler

```
public SupplyHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.
Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
```

Dispatch and handle the request. This version looks at the supplier id, rewrites the url based on that supplier, then lets the default handler do it.
Specified by:
respond in interface Handler

Tags copied from interface: Handler

Parameters:

`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

Class UrlMapperHandler

sunlabs.brazil.handler
Class UrlMapperHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.UrlMapperHandler
```

All Implemented Interfaces:
Handler

---

public class **UrlMapperHandler**
extends Object
implements Handler

Handler for mapping or redirecting URLs. Matches url's against a regexp pattern. If it matches, the url is either rewritten or redirected.

Properties:
match
The regexp to match a url. May contain constructs of the form ${xxx}, which are replaced by the value of `request.props` for the key *xxx*
replace
The url to replace it with. This may contain both regular expression sub-patterns, such as "\1", or variables of the form ${..} which are replaced with the equivalent request properties.
export
If set, use this as a properties prefix, and set request properties for each sub-expression in "match". (E.g. [export]1 [export]2 ...).
redirect
If set, the request is redirected instead of being rewritten
ignoreCase
If set, the case of the expression is ignored.
source
If set, then this string is used instead of the url as the *source* of the match. Variable substitution using ${xxx} is performed on *source*, which, if unset, defaults to "${url}". If set, the request properties "url", "host" and "user-agent" are set first. The *source* property is obtained at init time, but evaluated (for ${...}) at every request.

As an example, the configuration:

```
prefix.source=${user-agent}!${url}
prefix.match=Lynx.*!(.*)
prefix.replace=/text\\1
```

could cause all browsers with "Lynx" in their user agent header to the "text" sub-directory.

Constructor Summary

`UrlMapperHandler``()`	

Method Summary

`boolean`	**`init`**`(Server server, String prefix)` Initializes the handler.
`boolean`	**`respond`**`(Request request)` If this request matches the expression, rewrite it.

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

UrlMapperHandler

```
public UrlMapperHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.

`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
               throws IOException
```

If this request matches the expression, rewrite it.

Specified by:

respond in interface Handler

Tags copied from interface: Handler

Parameters:

`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

Class VirtualHostHandler

sunlabs.brazil.handler
Class VirtualHostHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.VirtualHostHandler
```

All Implemented Interfaces:
Handler

---

public class **VirtualHostHandler**
extends Object
implements Handler

Handler for managing virtual hosts using the same server configuration. This prefixes the host name (from the http "host" header, with the port portion removed) onto the url and passes the request along. If no host is provided, the host "default" is used instead.
If hosts require their own server configurations, use the `MultiHostHandler` instead.

Configuration parameters:
maproot
: If set upon server startup, this handler changes the "root" property instead of the "url" property, by appending the "host" onto the document root, instead of prepending the "host" to the url.

[prefix].[host].
: If the "mapping" property exists that matches the incoming "host" name, then that value is used instead of [host] to rewrite the "url" or "root".

default
: If set, then all hosts for which no mappings are defined are mapped to the value of this property.

levels
: If defined, then for the purpose of host matching, only "levels" of hostnames are considered. If levels=3, then for host: a.b.c.d.e, the host is considered to be "c.d.e". This enables support for wildcard-host matching within a virtual domain.

addlevel=true|false
: If "true", "levels" is specified, and the number of tokens (levels) in the hostname exceeds "levels", then all the extra tokens in the hostname are prepended to the URL as initial directories: If "levels" is 3, and "addlevel=true" then: host `http://a.b.c.d.e/foo.html` will be mapped to `http://c.d.e/b/a/foo.html`, and the file "foo.html" should be at [docroot]/c.d.e/b/a/foo.html].

  If "addlevel=false", then `http://a.b.c.d.e/foo.html` will be mapped to `http://c.d.e/foo.html`, and the file "foo.html" should be at [docroot]/c.d.e/foo.html. In this case, the "a.b" part of the host is available as part of the host property, which retains its original value.

With no configuration options, each virtual host document root is in a subdirectory whose name matches the host (e.g. www.foo.com). The "maproot" property changes how virtual roots are distingished: by URL or by document root. The "mapping" properties are used to choose a name for the subdirectory that differs from the virtual hostname. Finally, if "default" is set, then virtual hosts with no subdirectory are all shunted into

the subdirectory specified.

Constructor Summary

`VirtualHostHandler``()`	

Method Summary

`boolean`	**`init`**`(Server server, String prefix)` Initializes the handler.
`boolean`	**`respond`**`(Request request)` Either look for host header, tack on front of url, or modify the "root" property

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

VirtualHostHandler

```
public VirtualHostHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler

Parameters:

`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.

`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
                throws IOException
```

Either look for host header, tack on front of url, or modify the "root" property

Specified by:

respond in interface Handler

Tags copied from interface: Handler

Parameters:

`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

Package sunlabs.brazil.javascript

Provide ways of integrating the Javascript scripting language into the Brazil project server.

Javascript is typically used to include scripts inside of html or XML documents to do server side processing.

Files in this package depend upon the `js.jar` file, which may be downloaded from the Mozilla Rhino Project We have tested this code with `Rhino 1.5 release 3`

Class JavaScriptTemplate

sunlabs.brazil.javascript
Class JavaScriptTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.javascript.JavaScriptTemplate
```

All Implemented Interfaces:
TemplateInterface

---

public class **JavaScriptTemplate**
extends Template

The `JavaScriptTemplate` looks for each `<server language="javascript">` (or `<javascript>`) tag in an HTML page and treats the following data up to the next `</server>` (or `</javascript>`) tag as a JavaScript script to evaluate.

The reason that JavaScript scripts are included in an HTML page is usually to generate dynamic, server-side content. After running this template, everything between and including the `<server>` and `</server>` (or `<javascript>` and `</javascript>` tags is replaced by all output written to the JavaScript standard output stream (if any).

All JavaScript fragments within a given page are evaluated in the same JavaScript interpreter. The JavaScript interpreter actually lives for the entire duration of this `Template` object, so the user can implement persistence across requests.

The following configuration parameters are used to initialize this template.

script
: The name of the JavaScript script to evaluate when the interpreter is created. This script is only evaluated when the interp is created, not on every request. The variables `prefix` and `server` are set before this file is evaluated, and are references to the parameters passed to a `handler` init method.

root
: The document root, if the script is a relative file name. If the "root" property under the template prefix is not found, the global "root" property is used. If the global "root" property is not found, the current directory is used.

debug
: If this configuration parameter is present, this class replaces the `<server>` and `</server>` tags with comments, so the user can keep track of where the dynamically generated content is coming from by examining the comments in the resultant HTML document. By default, the `<server>` and `</server>` are completely eliminated from the HTML document rather than changed into comments.

Before evaluating each HTML document, this class sets variables in the JavaScript interpreter, which can be used to interact back with Java to do things like set the response headers:

request
: Exposes the `Request` Java object. It is set anew at each request.

prefix
: Exposes the handler prefix String.

server
: Exposes the handler `Server` object.

If the attribute `eval` is present as an attribute, all constructs of the form ${...} are substituted before processing the script.

Here's a simple example of a JavaScript template:

```
<html>
<head>
<title>JavaScript Example</title>
</head>
<body>
<javascript>
var s = "request=" + request;
s += "<br>request.serverUrl is " + request.serverUrl();
s += "<br>server hostName is " + server.hostName;
s += "<br>prefix is " + prefix;
s += "<br> <table>";

// This is an example of using Java in JavaScript with LiveConnect
var e = request.props.propertyNames();
while (e.hasMoreElements()) {
  var prop = e.nextElement();
  s += "<tr><td>" + prop + "</td>";
  s += "<td>" + request.props.getProperty(prop) + "</td></tr>";
}

s += "</table>";

// The last value computed or expressed is returned
s;

</javascript>
</body>
</html>
```

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

JavaScriptTemplate()	

Method Summary

boolean	**init**(RewriteContext hr) Called at the beginning of each HTML document that this JavaScriptTemplate is asked to process.
void	**tag_javascript**(RewriteContext hr) Processes the <javascript> tag.
void	**tag_server**(RewriteContext hr) Processes the <server> tag.

Methods inherited from class sunlabs.brazil.template.Template
done

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

JavaScriptTemplate

public **JavaScriptTemplate**()

Method Detail

init

```
public boolean init(RewriteContext hr)
```

Called at the beginning of each HTML document that this `JavaScriptTemplate` is asked to process.

The first time this method is called, the initialization script is sourced into the interpreter, based on the configuration properties in the `Request`

Overrides:

init in class Template

Parameters:

`hr` - The request and associated HTML document that will be processed.

Returns:

`true`.

---

tag_server

```
public void tag_server(RewriteContext hr)
```

Processes the `<server>` tag. Substitutes the result of evaluating the following JavaScript script into the resultant HTML document.

Note: Currently, there is no mechanism for other language interpreters to share the same `server` tag.

Parameters:

`hr` - The request and associated HTML document that will be processed.

---

tag_javascript

```
public void tag_javascript(RewriteContext hr)
```

Processes the `<javascript>` tag. Substitutes the result of evaluating the following JavaScript script into the resultant HTML document.

Parameters:

`hr` - The request and associated HTML document that will be processed.

Package sunlabs.brazil.ldap

Provide ways of integrating LDAP into the Brazil project server.

LDAP, the Lightweight Directory Access Protocol, is described in (among others) RFCs 1558, 1777 and 1779. All of the classes in this package provide mechanisms for accessing LDAP directory services from within the Brazil project server context. See www.openldap.org for more information about LDAP.

Files in this package were compiled and tested using the `ldap40.jar` file that is part of the Netscape Navigator (version 4) distribution.

Class LDAPTemplate

sunlabs.brazil.ldap
Class LDAPTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.ldap.LDAPTemplate
```

All Implemented Interfaces:
TemplateInterface

---

public class **LDAPTemplate**
extends Template

The `LDAPTemplate` is invoked to process LDAP tags embedded in a document. This version requires the "ldap40.jar" file from the Netscape Navigator distribution.

The LDAPTemplate uses the following special tag:

- `<ldap>`

When an LDAP tag is seen, the LDAP database is searched and the results are used to populate the request properties.

The following configuration parameters are used to perform the search. The parameters may appear either in the request properties (preceded by the prefix of this template as specified in the configuration file) or as named arguments in the LDAP tag.

`prefix`
: The string prefix for the property names that will be stored in the request properties to hold the results. If not specified, defaults to the prefix of this template as specified in the configuration file.

`dn`
: The Distinguished Name (DN) to lookup in the LDAP server. The format of a DN is described in RFC-1779. The "dn" and "search" options are mutually exclusive. When "dn" is specified, only zero or one result will be returned from the LDAP database. The result (if any) will be stored in the request properties as follows:

```
<ldap dn="uid=6105,ou=people,o=WebAuth" prefix=name>
<property name.dn>
<property name.cn>
<property name.sn>
<property name.objectclass>
```

etc. The property *name*`.dn` is the DN that was found. Other properties will be defined as shown, based on the attributes present in the LDAP record.

`search`

The search filter to use when searching the LDAP server. The format of a search filter is described in RFC-1558. The "search" and "dn" options are mutually exclusive. When "search" is specified, zero or more results will be returned from the LDAP database. The results will be stored in the request properties as follows:

```
<ldap search="(givenname=scott)" prefix=name>
<property name.rows>
<property name.rowcount>

<property name.0.dn>
<property name.0.cn>
<property name.0.mail>

<property name.1.dn>
<property name.1.cn>
<property name.1.pager>
```

etc. The property `name.rows` is set to the list of record indices found, and can be used by the BSL tag `<foreach name=x property=name.rows>` to iterate over all records. Other properties will be defined for each of the records found as shown, based on the attributes present in the each of the LDAP records.

`base`

The Distinguished Name of the base record that forms the root of the search tree in the LDAP database. Used only with the "search" option. Defaults to "". This would be a good option to specify in the configuration file rather than in the LDAP tag.

`scope`

The scope of the LDAP search, one of

- "base" Search only in base record (specified by the "base" option).
- "one" Search only records one level below the base record.
- "sub" Search the entire subtree below the base record.

Used only with the "search" option. Defaults to "sub". This would be a good option to specify in the configuration file rather than in the LDAP tag.

`attributes`

The space-delimited list of attribute names to return from the LDAP "dn" or "search" operation. If empty or unspecified, all attributes for the record are returned. Not all records in the LDAP database have the same attributes. Defaults to "".

`host`

The hostname of the LDAP server, of the form `"host"` or `"host:port"` if the server is not running on the standard LDAP port. Defaults to "". This would be a good option to specify in the configuration file rather than in the LDAP tag.

`authenticate`

The Distinguished Name used for authenticating to the LDAP server, if necessary. Defaults to "". This would be a good option to specify in the configuration file rather than in the LDAP tag.

`password`
The password sent when the "authenticate" option is used. Defaults to "".

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

`LDAPTemplate``()`	

Method Summary

`void`	**`tag_ldap`**`(RewriteContext hr)` Process <ldap> tags.

Methods inherited from class sunlabs.brazil.template.Template
`done, init`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

LDAPTemplate

`public` **`LDAPTemplate`**`()`

Method Detail

tag_ldap

```
public void tag_ldap(RewriteContext hr)
```

Process <ldap> tags.

Package sunlabs.brazil.properties

Classes for defining and using "smart" properties with Brazil.

"Smart properties" are sub-classes of ordinary java properties: they have `put()` and `getProperty()` methods. However, rather than saving and returning static string values, the various smart properties compute their values dynamically.

Smart properties may be inserted into the `request.props()` properties chain (see `sunlabs.brazil.server.Request#addSharedProps`) to uniformly add new and dynamic behavior when processing `sunlabs.brazil.server.Request#props`.

Class ExprProps

sunlabs.brazil.properties
Class ExprProps

```
java.lang.Object
  |
  +--java.util.Dictionary
        |
        +--java.util.Hashtable
              |
              +--java.util.Properties
                    |
                    +--sunlabs.brazil.properties.ExprProps
```

All Implemented Interfaces:
Cloneable, Map, Serializable

---

public class **ExprProps**
extends Properties

`ExprProps` is a subclass of `Properties` that is "smart" in the sense that it wraps a `Calculator` object, passing `get` and `getProperty` keys to the `Calculator` for processing.

Keys are first searched for in the `Properties` object. If not found, the key is passed to the `getValue` method of the wrapped `Calculator` object. The `Calculator` will return `null` if the key consists of only a name or Brazil token and that name/token evaluates to 0. In this case, a value associated with the key can not be "found" and the `get` or `getproperty`method returns `null` also.

Only property names that "look" like expressions (e.g. contain at least one of characters in [&|*+/%=!<>-]) are passed to the calculator.

See Also:
`Properties`, `Calculator`, `ExprPropsHandler`, Serialized Form

Constructor Summary

ExprProps() This constructor creates a `Calculator` instance with this instance of `ExprProps` as it's symbol table.
ExprProps(Properties defaults) This constructor creates a `Calculator` instance with this instance of `ExprProps` as it's symbol table and with the `Properties` instance referenced by the parameter `defaults` as it's set of default values.
ExprProps(Request request) This constructor creates a `Calculator` instance with `request.props` as it's symbol table.
ExprProps(Request request, Properties defaults) This constructor creates a `Calculator` instance with `request.props` as it's symbol table and with the `Properties` instance referenced by the parameter `defaults` as it's set of default values.

Method Summary

Object	**get**(Object key) Returns the value to which the specified key is mapped in this `Hashtable`.
String	**getProperty**(String key) Searches for the property with the specified key in this property list.

Methods inherited from class java.util.Properties
getProperty, list, list, load, propertyNames, save, setProperty, store

Methods inherited from class java.util.Hashtable
clear, clone, contains, containsKey, containsValue, elements, entrySet, equals, hashCode, isEmpty, keys, keySet, put, putAll, remove, size, toString, values

Methods inherited from class java.lang.Object
getClass, notify, notifyAll, wait, wait, wait

Constructor Detail

ExprProps

```
public ExprProps()
```

This constructor creates a `Calculator` instance with this instance of `ExprProps` as it's symbol table.

ExprProps

```
public ExprProps(Properties defaults)
```

This constructor creates a `Calculator` instance with this instance of `ExprProps` as it's symbol table and with the `Properties` instance referenced by the parameter `defaults` as it's set of default values.

Parameters:

`defaults` - the defaults

ExprProps

```
public ExprProps(Request request)
```

This constructor creates a `Calculator` instance with `request.props` as it's symbol table.

Parameters:

`request` - the `Request` instance

ExprProps

```
public ExprProps(Request request,
                 Properties defaults)
```

This constructor creates a `Calculator` instance with `request.props` as it's symbol table and with the `Properties` instance referenced by the parameter `defaults` as it's set of default values.

Parameters:

`request` - the `Request` instance
`defaults` - the defaults

Method Detail

get

`public Object **get**(Object key)`

Returns the value to which the specified key is mapped in this `Hashtable`. If the key is not found, then it's value is computed by treating the key as an arithmetic expression or statement.

If, during the computation, an `ArithmeticException` is thrown, the key `compute.error` is set in the base `Properties`. The value associated with the key is an error message.

Overrides:

get in class Hashtable

Parameters:

`key` - the `Hashtable` key

Returns:

the value in this table with the specified key or `null`

---

getProperty

`public String **getProperty**(String key)`

Searches for the property with the specified key in this property list. If the key is not found in this property list, then it's value is computed by treating the key as an arithmetic expression or statement.

If, during the computation, an `ArithmeticException` is thrown, the key `compute.error` is set in the base `Properties`. The value associated with the key is an error message.

If the result of the computation is `null`, the default property list, and its defaults, recursively, are then checked. The method returns `null` if the property is not found.

Overrides:

getProperty in class Properties

Parameters:

`key` - the property key

Returns:

the value in this property list with the specified key or `null`

See Also:

`java.util.Properties#defaults`

Class ExprPropsHandler

sunlabs.brazil.properties
Class ExprPropsHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.properties.ExprPropsHandler
```

All Implemented Interfaces:
Handler

---

public class **ExprPropsHandler**
extends Object
implements Handler

The `ExprPropsHandler` installs an expression evaluator as a "smart properties" into the current request object, enabling arithmetic and logical expression evaluation in property name lookups.

The following configuration parameters are used:
prefix, suffix, glob, match
Only URL's that match are allowed. (See `MatchString`).

Using the expression evaluator can be a bit tricky, as the evaluator works by interpreting a property name as an expression, and using its the expression result as its value. For example, the construct:

```
"${x + 4 == 3}"
```

will evaluate to either "1" or "0", depending upon the value of "x". For use with the `<if>` constuct of the `BSLTemplate`, the following construct:

```
<if name="${x + 4 == 3}"> ... [if expression] ... </if>
```

Will take (or not take) the "if expression" if there is a property named "1" that is set (to anything but 0 or false), but a property named "0" is not set. An entry in a server configuration file:

```
1=true
```

will do the trick.

alternately, the construct:

```
<if name=true value="${x + 4 == 3}"> ... [if expression] ... </if>
```

Will work as expected only if there is a configuration property:

```
true=1
```

The choice of the name "true" is arbitrary, it could be any valiable whose value is "1".

See Also:
`ExprProps`, `Request`

Constructor Summary

`ExprPropsHandler``()`	

Method Summary

`boolean`	**`init`**`(Server server, String prefix)` Initializes the handler.
`boolean`	**`respond`**`(Request request)` Creates an instance of `ExprProps` that uses `request.props` for the wrapped `Calculator`'s symbol table.

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

ExprPropsHandler

```
public ExprPropsHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.

Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.
Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
                throws IOException
```

Creates an instance of `ExprProps` that uses `request.props` for the wrapped `Calculator`'s symbol table.
Specified by:
respond in interface Handler
Returns:
false
See Also:
`Calculator`

Class PropertiesList

sunlabs.brazil.properties
Class PropertiesList

```
java.lang.Object
  |
  +--java.util.Dictionary
        |
        +--java.util.Hashtable
              |
              +--java.util.Properties
                    |
                    +--sunlabs.brazil.properties.PropertiesList
```

All Implemented Interfaces:
Cloneable, Map, Serializable

---

public class **PropertiesList**
extends Properties

A `PropertiesList` instance is intended to be an element of a doubly linked list consisting of other `PropertiesList` instances. Each `PropertiesList` instance "wraps" a `Dictionary` object. A `PropertiesList` is a subclass of `Properties` and therefore provides the same API, including the methods and fields of `Dictionary` and `Hashtable`. The `PropertiesList` class overrides all methods of the `Properties` API and delegates the method evaluation to the wrapped `Properties` object.

The linked list of `PropertiesList` objects is constructed by `Request` for each incoming request. That is, there is a unique `PropertiesList` linked list for each request. The head of the initial list constructed by `request` is `Request.props` and the tail of the two element list is `Request.serverProps`. The former wraps an empty `Properties` object, while the latter wraps `Server.props`. Other `PropertiesList` objects can be added, and removed, from this initial list as required.

Given a reference to a `PropertiesList` object on the linked list (e.g. `request.props`), one typically "looks up" the value associated with a name using the `getProperty` method, which delegates to the wrapped `Properties.getProperty` method. If the result is `null`, meaning the name/value pair is not stored in the wrapped `Properties` object, the request is "forwarded" to the next object on the linked list, and so on until either the name/value pair is found (and the value is returned) or the end of the list is reached (and `null` is returned).

It may be desirable for the name/value lookup to be delayed until after the lookup request has been passed on to subsequent objects on the list. This can be done by using the two parameter constructor and setting the second, boolean, parameter to `true`. Then the `getProperty` request is forwarded to the next object in the list rather than delegated to the wrapped `Properties` object. If the result of the forwarded request is `null`, the request is then passed to the wrapped `Properties` object and it’s result is returned.

See Also:
`Dictionary`, `Hashtable`, `Properties`, Serialized Form

Field Summary

`static boolean`	**`debug`** Set `true` to turn on debug output.

Constructor Summary

`PropertiesList``()` Constructs a new `PropertiesList` object that wraps an empty new `Properties` object.
`PropertiesList``(Dictionary dict)` Constructs a new `PropertiesList` object that wraps the input `Dictionary`.
`PropertiesList``(Dictionary dict, boolean searchNextFirst)` Constructs a new `PropertiesList` object that wraps the input `Dictionary`.

Method Summary

`void`	**`addAfter`**`(PropertiesList cursor)` Adds this `PropertiesList` object into a linked list following the object referenced by the `cursor` parameter.
`void`	**`addBefore`**`(PropertiesList cursor)` Adds this `PropertiesList` object into a linked list preceding the object referenced by the `cursor` parameter.
`void`	**`clear`**`()` Invokes the same method on the wrapped `Hashtable` object.
`Object`	**`clone`**`()` Invokes the same method on the wrapped `Hashtable` object.
`boolean`	**`contains`**`(Object value)` Invokes the same method on the wrapped `Hashtable` object.
`boolean`	**`containsKey`**`(Object key)` Invokes the same method on the wrapped `Hashtable` object.
`void`	**`dump`**`(boolean full, String msg)` Starting with this object, print the contents of this and succeeding objects that are on the same list as this object is.

Enumeration	**elements**() Invokes the same method on the wrapped `Dictionary` object.
Set	**entrySet**() Invokes the same method on the wrapped `Hashtable` object.
boolean	**equals**(Object o) Invokes the same method on the wrapped `Hashtable` object.
Object	**get**(Object key) Invokes the same method on the wrapped `Dictionary` object.
PropertiesList	**getHead**() Returns the `PropertiesList` object that is the first object on the list of which this object is a member.
PropertiesList	**getNext**() Returns the `PropertiesList` object that succedes this object on the list of which this object is a member.
PropertiesList	**getPrior**() Returns the `PropertiesList` object that precedes this object on the list of which this object is a member.
String	**getProperty**(String key) Looks up `key` in the wrapped object.
String	**getProperty**(String key, String defaultValue) Uses `getProperty(String)` to look up the value associated with the key.
Dictionary	**getWrapped**() Returns the `Dictionary` object wrapped by this `PropertiesList`.
int	**hashCode**() Invokes the same method on the wrapped `Hashtable` object.
boolean	**isEmpty**() Invokes the same method on the wrapped `Dictionary` object.
boolean	**isTransient**() Sub-classes of PropertiesList can override this to mark themselves "transient", in which case `addAfter` will skip this list.
Enumeration	**keys**() Invokes the same method on the wrapped `Dictionary` object.
Set	**keySet**() Invokes the same method on the wrapped `Hashtable` object.
void	**list**(PrintStream out) Invokes the same method on the wrapped `Properties` object.
void	**list**(PrintWriter out) Invokes the same method on the wrapped `Properties` object.

void	**load**(InputStream in) Invokes the same method on the wrapped Properties object.
Enumeration	**propertyNames**() Invokes the same method on the wrapped Properties object.
Enumeration	**propertyNames**(String pattern) Returns an Enumeration of property names that match a glob pattern.
Object	**put**(Object key, Object value) Invokes the same method on the wrapped Dictionary object.
void	**putAll**(Map t) Invokes the same method on the wrapped Hashtable object.
boolean	**remove**() Remove this object from the list in which it's a member.
Object	**remove**(Object key) Invokes the same method on the wrapped Dictionary object.
void	**save**(OutputStream out, String header) Invokes the same method on the wrapped Properties object.
Object	**setProperty**(String key, String value) Invokes the same method on the wrapped Properties object if it exists.
int	**size**() Invokes the same method on the wrapped Dictionary object.
void	**store**(OutputStream out, String header) Invokes the same method on the wrapped Properties object.
String	**toString**() Returns a String containing the System.identityHashCodes of this object, the wrapped object, and the preceding and succeding objects on the list of which this object is a member.
Collection	**values**() Invokes the same method on the wrapped Hashtable object.
PropertiesList	**wraps**(Dictionary d) Find the first PropertiesList object on the list of which this object is a member that wraps the Dictionary parameter.

Methods inherited from class java.util.Hashtable
containsValue

Methods inherited from class java.lang.Object
`getClass, notify, notifyAll, wait, wait, wait`

Field Detail

debug

`public static boolean` **`debug`**

Set `true` to turn on debug output. It's alot of output and probably of use only to the author. Note, if `server.props` contains the name `debugProps` this variable will be set `true` by `Server`.

Constructor Detail

PropertiesList

`public` **`PropertiesList`**`()`

Constructs a new `PropertiesList` object that wraps an empty new `Properties` object.

PropertiesList

`public` **`PropertiesList`**`(Dictionary dict)`

Constructs a new `PropertiesList` object that wraps the input `Dictionary`.

Parameters:

`dict` - The `Dictionary` object wrapped by this `PropertiesList`.

PropertiesList

```
public PropertiesList(Dictionary dict,
                      boolean searchNextFirst)
```

Constructs a new `PropertiesList` object that wraps the input `Dictionary`. If the boolean parameter is set `true`, the wrapped `Dictionary` is searched after subsequent `PropertiesList` objects in the linked list are searched, and only if the result of that search was `null`.

Parameters:

`dict` - The `Dictionary` object wrapped by this `PropertiesList`.

`searchNextFirst` - If `true` all the following objects in the list are searched before this one.

Method Detail

getWrapped

`public Dictionary **getWrapped**()`

Returns the `Dictionary` object wrapped by this `PropertiesList`.

---

addAfter

`public void **addAfter**(PropertiesList cursor)`

Adds this `PropertiesList` object into a linked list following the object referenced by the `cursor` parameter. The result is a list that could look like: request.props -> cursor -> this -> serverProps

Any transient properties lists's are skipped over before this one is inserted into the list
Parameters:
`cursor` - The list object that will precede this object.

---

addBefore

`public void **addBefore**(PropertiesList cursor)`

Adds this `PropertiesList` object into a linked list preceding the object referenced by the `cursor` parameter. The result is a list that could look like: request.props -> this -> cursor -> serverProps
Parameters:
`cursor` - The list object that will succede this object.

---

remove

`public boolean **remove**()`

Remove this object from the list in which it's a member.
Returns:
`true`.

---

getNext

`public PropertiesList **getNext**()`

Returns the `PropertiesList` object that succedes this object on the list of which this object is a member.
Returns:
A `PropertiesList` object or `null`.

---

getPrior

```
public PropertiesList getPrior()
```

Returns the `PropertiesList` object that precedes this object on the list of which this object is a member.

Returns:

A `PropertiesList` object or `null`.

---

getHead

```
public PropertiesList getHead()
```

Returns the `PropertiesList` object that is the first object on the list of which this object is a member. Note that the first object may be this object.

Returns:

A `PropertiesList` object.

---

wraps

```
public PropertiesList wraps(Dictionary d)
```

Find the first `PropertiesList` object on the list of which this object is a member that wraps the `Dictionary` parameter.

Parameters:

`d` - The `Dictionary` that is compared with the wrapped `Dictionary`'s for a match.

Returns:

`PropertiesList` object that wraps the input parameter, otherwise `null`.

---

dump

```
public void dump(boolean full,
                 String msg)
```

Starting with this object, print the contents of this and succeeding objects that are on the same list as this object is.

Parameters:

`full` - If `true` also print the contents of the wrapped `Dictionary` object.

`msg` - If not `null`, add this message to the header line.

---

elements

```
public Enumeration elements()
```

Invokes the same method on the wrapped `Dictionary` object.

Overrides:
elements in class Hashtable

---

get

```
public Object get(Object key)
```

Invokes the same method on the wrapped `Dictionary` object.
Overrides:
get in class Hashtable

---

isEmpty

```
public boolean isEmpty()
```

Invokes the same method on the wrapped `Dictionary` object.
Overrides:
isEmpty in class Hashtable

---

keys

```
public Enumeration keys()
```

Invokes the same method on the wrapped `Dictionary` object.
Overrides:
keys in class Hashtable

---

put

```
public Object put(Object key,
                  Object value)
```

Invokes the same method on the wrapped `Dictionary` object.
Overrides:
put in class Hashtable

---

remove

```
public Object remove(Object key)
```

Invokes the same method on the wrapped `Dictionary` object.
Overrides:
remove in class Hashtable

---

size

`public int **size**()`

Invokes the same method on the wrapped `Dictionary` object.
Overrides:
size in class Hashtable

---

clear

`public void **clear**()`

Invokes the same method on the wrapped `Hashtable` object.
Overrides:
clear in class Hashtable

---

clone

`public Object **clone**()`

Invokes the same method on the wrapped `Hashtable` object.
Overrides:
clone in class Hashtable

---

contains

`public boolean **contains**(Object value)`

Invokes the same method on the wrapped `Hashtable` object.
Overrides:
contains in class Hashtable

---

containsKey

`public boolean **containsKey**(Object key)`

Invokes the same method on the wrapped `Hashtable` object.
Overrides:
containsKey in class Hashtable

---

entrySet

`public Set **entrySet**()`

Invokes the same method on the wrapped `Hashtable` object.
Overrides:
entrySet in class Hashtable

---

equals

```
public boolean equals(Object o)
```

Invokes the same method on the wrapped `Hashtable` object.
Overrides:
equals in class Hashtable

---

hashCode

```
public int hashCode()
```

Invokes the same method on the wrapped `Hashtable` object.
Overrides:
hashCode in class Hashtable

---

keySet

```
public Set keySet()
```

Invokes the same method on the wrapped `Hashtable` object.
Overrides:
keySet in class Hashtable

---

putAll

```
public void putAll(Map t)
```

Invokes the same method on the wrapped `Hashtable` object.
Overrides:
putAll in class Hashtable

---

values

```
public Collection values()
```

Invokes the same method on the wrapped `Hashtable` object.
Overrides:
values in class Hashtable

---

toString

```
public String toString()
```

Returns a `String` containing the `System.identityHashCode`s of this object, the wrapped object, and the preceding and succeding objects on the list of which this object is a member.

Additionally, if `debug` is `true`, the result of invoking `toString` on the wrapped `Dictionary` is appended.

Overrides:

toString in class Hashtable

Returns:

`String` representation of this object.

---

getProperty

```
public String getProperty(String key)
```

Looks up `key` in the wrapped object. If the result is `null` the request is forwarded to the succeeding object in the list of which this object is a member. If the search order was changed by constructing this object with the two parameter constructor, the request is first forwarded and then, if the result of the forwarded request is `null`, the `key` is looked up in the wrapped `Properties` object.

Overrides:

getProperty in class Properties

Parameters:

`key` - The key whose value is sought.

Returns:

The value or `null`.

---

getProperty

```
public String getProperty(String key,
                          String defaultValue)
```

Uses `getProperty(String)` to look up the value associated with the key. If the result is `null`, returns the default value.

Overrides:

getProperty in class Properties

Parameters:

`key` - The key whose value is sought.

`defaultValue` - The default value.

Returns:

The value or `null`.

---

list

```
public void list(PrintStream out)
```

Invokes the same method on the wrapped `Properties` object.

Overrides:

list in class Properties

---

list

```
public void list(PrintWriter out)
```

Invokes the same method on the wrapped `Properties` object.
Overrides:
list in class Properties

---

load

```
public void load(InputStream in)
          throws IOException
```

Invokes the same method on the wrapped `Properties` object.
Overrides:
load in class Properties

---

propertyNames

```
public Enumeration propertyNames()
```

Invokes the same method on the wrapped `Properties` object.
Overrides:
propertyNames in class Properties

---

save

```
public void save(OutputStream out,
                 String header)
```

Invokes the same method on the wrapped `Properties` object.
Overrides:
save in class Properties

---

setProperty

```
public Object setProperty(String key,
                          String value)
```

Invokes the same method on the wrapped `Properties` object if it exists. Otherwise invokes `put` on the wrapped `Dictionary` object.
Overrides:
setProperty in class Properties

---

store

```
public void store(OutputStream out,
                  String header)
           throws IOException
```

Invokes the same method on the wrapped `Properties` object.

Overrides:

store in class Properties

---

propertyNames

```
public Enumeration propertyNames(String pattern)
```

Returns an `Enumeration` of property names that match a `glob` pattern.

Parameters:

`pattern` - The `glob` pattern to match.

Returns:

An `Enumeration` containing matching property names, if any.

---

isTransient

```
public boolean isTransient()
```

Sub-classes of PropertiesList can override this to mark themselves "transient", in which case `addAfter` will skip this list.

Interface SubstPropsHandler.Convert

sunlabs.brazil.properties
Interface SubstPropsHandler.Convert
All Known Implementing Classes:
SubstPropsHandler.LowerCase, SubstPropsHandler.Html, SubstPropsHandler.Url, SubstPropsHandler.Resub
Enclosing class:
SubstPropsHandler

---

public static interface **SubstPropsHandler.Convert**

Class that maps strings to strings.

Method Summary

`boolean`	**`init`**`(String prefix, Properties p)` This is called once at creation time to provide this class with configuration information.
`String`	**`map`**`(String value)` Map the value.

Method Detail

init

```
public boolean init(String prefix,
                    Properties p)
```

This is called once at creation time to provide this class with configuration information. Any configuration parameters required in "p" are prefixed with [prefix].

---

map

```
public String map(String value)
```

Map the value.

Class SubstPropsHandler.Html

sunlabs.brazil.properties
Class SubstPropsHandler.Html

```
java.lang.Object
  |
  +--sunlabs.brazil.properties.SubstPropsHandler.Html
```

All Implemented Interfaces:
SubstPropsHandler.Convert
Enclosing class:
SubstPropsHandler

---

public static class **SubstPropsHandler.Html**
extends Object
implements SubstPropsHandler.Convert

HTML escape a value.

Constructor Summary

SubstPropsHandler.Html()	

Method Summary

boolean	**init**(String prefix, Properties p) This is called once at creation time to provide this class with configuration information.
String	**map**(String value) Map the value.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

SubstPropsHandler.Html

```
public SubstPropsHandler.Html()
```

Method Detail

init

```
public boolean init(String prefix,
                    Properties p)
```

Description copied from interface: SubstPropsHandler.Convert
This is called once at creation time to provide this class with configuration information. Any configuration parameters required in "p" are prefixed with [prefix].
Specified by:
init in interface SubstPropsHandler.Convert

---

map

```
public String map(String value)
```

Description copied from interface: SubstPropsHandler.Convert
Map the value.
Specified by:
map in interface SubstPropsHandler.Convert

Class SubstPropsHandler.LowerCase

sunlabs.brazil.properties
Class SubstPropsHandler.LowerCase

```
java.lang.Object
  |
  +--sunlabs.brazil.properties.SubstPropsHandler.LowerCase
```

All Implemented Interfaces:
SubstPropsHandler.Convert
Enclosing class:
SubstPropsHandler

---

public static class **SubstPropsHandler.LowerCase**
extends Object
implements SubstPropsHandler.Convert

Convert a value to lowercase.

Constructor Summary

SubstPropsHandler.LowerCase()	

Method Summary

boolean	**init**(String prefix, Properties p) This is called once at creation time to provide this class with configuration information.
String	**map**(String value) Map the value.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

SubstPropsHandler.LowerCase

```
public SubstPropsHandler.LowerCase()
```

Method Detail

init

```
public boolean init(String prefix,
                    Properties p)
```

Description copied from interface: SubstPropsHandler.Convert
This is called once at creation time to provide this class with configuration information. Any configuration parameters required in "p" are prefixed with [prefix].
Specified by:
init in interface SubstPropsHandler.Convert

---

map

```
public String map(String value)
```

Description copied from interface: SubstPropsHandler.Convert
Map the value.
Specified by:
map in interface SubstPropsHandler.Convert

Class SubstPropsHandler.Resub

sunlabs.brazil.properties
Class SubstPropsHandler.Resub

```
java.lang.Object
  |
  +--sunlabs.brazil.properties.SubstPropsHandler.Resub
```

All Implemented Interfaces:
SubstPropsHandler.Convert
Enclosing class:
SubstPropsHandler

---

public static class **SubstPropsHandler.Resub**
extends Object
implements SubstPropsHandler.Convert

Do a regexp substitution on a value. This takes the following initialization parameters:
match
A Regular expression that matches the string value.
sub
The regular expression substitution to perform. All occurances of "match" are substututed.

Constructor Summary

SubstPropsHandler.Resub()	

Method Summary

boolean	**init**(String prefix, Properties p) This is called once at creation time to provide this class with configuration information.
String	**map**(String value) Map the value.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

SubstPropsHandler.Resub

```
public SubstPropsHandler.Resub()
```

Method Detail

init

```
public boolean init(String prefix,
                    Properties p)
```

Description copied from interface: SubstPropsHandler.Convert
This is called once at creation time to provide this class with configuration information. Any configuration parameters required in "p" are prefixed with [prefix].
Specified by:
init in interface SubstPropsHandler.Convert

---

map

```
public String map(String value)
```

Description copied from interface: SubstPropsHandler.Convert
Map the value.
Specified by:
map in interface SubstPropsHandler.Convert

Class SubstPropsHandler.SubstProps

sunlabs.brazil.properties
Class SubstPropsHandler.SubstProps

```
java.lang.Object
  |
  +--java.util.Dictionary
        |
        +--java.util.Hashtable
              |
              +--java.util.Properties
                    |
                    +--sunlabs.brazil.properties.SubstPropsHandler.SubstProps
```

All Implemented Interfaces:
Cloneable, Map, Serializable
Enclosing class:
SubstPropsHandler

---

public class **SubstPropsHandler.SubstProps**
extends Properties

This class implements a properties object that knows how to extract the "name" and "filter" from a properly constructed name, and to invoke the filter on the value of the encoded name.

See Also:
Serialized Form

Constructor Summary

SubstPropsHandler.SubstProps(Request r)	

Method Summary

Object	**get**(Object key) If the key doesn't exist, but the "derived" key and value do exist, then return the substituted value
String	**getProperty**(String key)

Methods inherited from class java.util.Properties
`getProperty, list, list, load, propertyNames, save, setProperty, store`

Methods inherited from class java.util.Hashtable
`clear, clone, contains, containsKey, containsValue, elements, entrySet, equals, hashCode, isEmpty, keys, keySet, put, putAll, remove, size, toString, values`

Methods inherited from class java.lang.Object
`getClass, notify, notifyAll, wait, wait, wait`

Constructor Detail

SubstPropsHandler.SubstProps

`public` **`SubstPropsHandler.SubstProps`**`(Request r)`

Method Detail

get

`public Object` **`get`**`(Object key)`

If the key doesn't exist, but the "derived" key and value do exist, then return the substituted value

Overrides:

get in class Hashtable

---

getProperty

`public String` **`getProperty`**`(String key)`

Overrides:

getProperty in class Properties

Class SubstPropsHandler.Url

sunlabs.brazil.properties
Class SubstPropsHandler.Url

```
java.lang.Object
  |
  +--sunlabs.brazil.properties.SubstPropsHandler.Url
```

All Implemented Interfaces:
SubstPropsHandler.Convert
Enclosing class:
SubstPropsHandler

---

public static class **SubstPropsHandler.Url**
extends Object
implements SubstPropsHandler.Convert

URL encode a String.

Constructor Summary

SubstPropsHandler.Url()	

Method Summary

boolean	**init**(String prefix, Properties p) This is called once at creation time to provide this class with configuration information.
String	**map**(String value) Map the value.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

SubstPropsHandler.Url

```
public SubstPropsHandler.Url()
```

Method Detail

init

```
public boolean init(String prefix,
                    Properties p)
```

Description copied from interface: SubstPropsHandler.Convert
This is called once at creation time to provide this class with configuration information. Any configuration parameters required in "p" are prefixed with [prefix].
Specified by:
init in interface SubstPropsHandler.Convert

---

map

```
public String map(String value)
```

Description copied from interface: SubstPropsHandler.Convert
Map the value.
Specified by:
map in interface SubstPropsHandler.Convert

Class SubstPropsHandler

sunlabs.brazil.properties
Class SubstPropsHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.properties.SubstPropsHandler
```

All Implemented Interfaces:
Handler

---

public class **SubstPropsHandler**
extends Object
implements Handler

Handler that performs value conversions on ${...} substitutions. For any property whose name matches the supplied regular expression, The source value is "converted" based on a token in the regular expression.

This Handler is a generalization of the `convert` attribute of the `get` tag of the `SetTemplate`. Unlike the implementation in the `SetTemplate` that implements a small, fixed set of conversions of property values in the context of `get`, this handler allows plug-able conversion filters, and performs the conversions any time ${...} substitutions are resolved, not just in the context of the `get` tag.

This requires the addition of new syntax in ${...} substitutions to specify the both the conversion (or filter) to apply, and the value to apply it to. This new syntax is configurable using the `match`, `key`, and `token` attributes, but defaults to: ${filter(value)} where `filter` represents the conversion filter, and `value` represents the property name whose contents is filtered.

Any class that implements the `Convert` interface can be loaded and called to perform filtering. Filters that implement all the options of the `<get ... convert=xxx>` conversion options are included.

See the examples, below for the details.

match
: A regular expression that matches a property name that is a candidate for filtering. This expression should have at least 2 sets of ()'s in order to gather values for "key" and "token" below. The default value is `^([a-z]+)\([^)]+\)$`

key
: The regular expression substitution string used to represent the actual property name to filter. The default is `\\2`

token
: The regular expression substitution string used to represent the filter name or "token". The default is `\\1`

  Using the defaults for "match", "key", and "token", a property named "foo" would be represented as `${xxx(foo)}` where "xxx" is the name of the conversion filter.

tokens
: A witespace separated list of filter names or "token"s that map the conversion filters to conversion classes. For each token (e.g. foo), there should be a property of the form "foo.class" which specifies the name of the class that implements the filter, (and implements the Convert interface described below). Any additional properties (e.g. x, y, z) needed to initialize a filter should be present in the properties file as "foo.x, foo.y...".

[token].code
: The name to match the "token" in the property name. The default is "[token]".

This class contains sample implementations of the `convert` interface. See below for their functions.

See Also:
`Properties`

Inner Class Summary	
`static interface`	**`SubstPropsHandler.Convert`** Class that maps strings to strings.
`static class`	**`SubstPropsHandler.Html`** HTML escape a value.
`static class`	**`SubstPropsHandler.LowerCase`** Convert a value to lowercase.
`static class`	**`SubstPropsHandler.Resub`** Do a regexp substitution on a value.
`class`	**`SubstPropsHandler.SubstProps`** This class implements a properties object that knows how to extract the "name" and "filter" from a properly constructed name, and to invoke the filter on the value of the encoded name.
`static class`	**`SubstPropsHandler.Url`** URL encode a String.

Constructor Summary	
`SubstPropsHandler()`	

Method Summary	
boolean	**init**(Server server, String prefix) Initializes the handler.
boolean	**respond**(Request request) Responds to an HTTP request.
String	**toString**()

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

SubstPropsHandler

```
public SubstPropsHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.
Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

respond

```
public boolean respond(Request request)
               throws IOException
```

Description copied from interface: Handler
Responds to an HTTP request.
Specified by:
respond in interface Handler
Tags copied from interface: Handler
Parameters:
`request` - The `Request` object that represents the HTTP request.
Returns:
`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.
Throws:
IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

toString

```
public String toString()
```

Overrides:
toString in class Object

Package sunlabs.brazil.proxy

Handlers, filters, and utilities for using the Brazil project framework as an HTTP proxy.

Unlike traditional web proxies, a Brazil project server can operate both as a destination server and as a web proxy. As a proxy, the server has access to all of the session and content rewriting capabilities normally associated with a destination server.

The `sunlabs.brazil.proxy.ProxyHandler` uses the HTTP proxy code in the `sunlabs.brazil.util.http` package to provide a handler interface to the proxy.

The other classes in this package provide examples of customizing the proxy, both in terms of content filtering, and name resolution.

Class CookieFilter

sunlabs.brazil.proxy
Class CookieFilter

```
java.lang.Object
  |
  +--sunlabs.brazil.proxy.CookieFilter
```

All Implemented Interfaces:
Filter, Handler

---

public class **CookieFilter**
extends Object
implements Filter

The `CookieFilter` keeps a record of all the browser *cookies* associated with a given session. This can be used to make the user's cookies "mobile" as follows. A user's cookies are normally stored with the browser being used, on the user's machine. If the user runs a different browser or goes to a different machine, the user's cookies will not be there. Instead, the user can access the web via a proxy that keeps all their cookies. No matter which browser the user chooses or machine the user is at, a proxy running with the `CookieFilter` will automatically remember and add in their appropriate cookies. The `CookieFilter` also supports multiple, concurrent users, keeping each user's cookies separate.

- All "Set-Cookie" HTTP response headers are filtered out and saved in the local storage. "Set-Cookie" headers are not transmitted back to the client.
- Requests from the client have the appropriate "Cookie" headers added.
- Users can retrieve, edit, and delete their own cookies.
- JavaScript code that sets cookies is **not** handled by this filter, since the code only runs on the client computer, not on the proxy. For instance: `document.cookie = "userid=778287312"`. Any and all Javascript is passed unchanged by this filter.
- This filter works in both a session-based and a non-session-based fashion. If sessions are used, cookies are kept with respect to the session associated with a user. If sessions are not used, all cookies are kept in one pile for all users. The latter case is valid if, say, only one user is using the proxy running the `CookieFilter`.

Properties:

session
: The request property to find the session id. Defaults to "SessionID"

nosession
: The name of the session to use if no session id is found. defaults to "common".

admin
: A URL prefix that causes status information to be placed in the request properties.

Field Summary

String	**admin**
String	**nosession**
String	**session**

Constructor Summary

CookieFilter()	

Method Summary

byte[]	**filter**(Request request, MimeHeaders headers, byte[] content) Returns the original content, since this filter does not change content; it changes the headers.
boolean	**init**(Server server, String prefix) Initializes the handler.
boolean	**respond**(Request request) Responds to an HTTP request.
boolean	**shouldFilter**(Request request, MimeHeaders headers) Saves all "Set-Cookie" headers from the target in the client's local storage, then removes those headers before allowing the response to go back to the client.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

session

```
public String session
```

---

nosession

```
public String nosession
```

---

admin

```
public String admin
```

Constructor Detail

CookieFilter

```
public CookieFilter()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.
Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
                throws IOException
```

Description copied from interface: Handler
Responds to an HTTP request.
Specified by:
respond in interface Handler
Tags copied from interface: Handler
Parameters:
`request` - The `Request` object that represents the HTTP request.
Returns:
`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.
Throws:
IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

shouldFilter

```
public boolean shouldFilter(Request request,
                            MimeHeaders headers)
```

Saves all "Set-Cookie" headers from the target in the client's local storage, then removes those headers before allowing the response to go back to the client. The client never sees cookies on their local machine.
Specified by:
shouldFilter in interface Filter
Tags copied from interface: Filter
Parameters:
`request` - The in-progress HTTP request.
`headers` - The MIME headers generated by the wrapped `Handler`.
Returns:
`true` if this filter would like to examine and possibly rewrite the content, `false` otherwise.

---

filter

```
public byte[] filter(Request request,
                     MimeHeaders headers,
                     byte[] content)
```

Returns the original content, since this filter does not change content; it changes the headers.

Specified by:

filter in interface Filter

Tags copied from interface: Filter

Parameters:

`request` - The finished HTTP request.

`headers` - The MIME headers generated by the `Handler`.

`content` - The output from the `Handler` that this `Filter` may rewrite.

Returns:

The rewritten content. The `Filter` may return the original `content` unchanged. The `Filter` may return `null` to indicate that the `FilterHandler` should stop processing the request and should not return any content to the client.

Class HistoryFilter

sunlabs.brazil.proxy
Class HistoryFilter

```
java.lang.Object
  |
  +--sunlabs.brazil.proxy.HistoryFilter
```

All Implemented Interfaces:
Filter, Handler

---

public class **HistoryFilter**
extends Object
implements Filter

The `HistoryFilter` is both a `Handler` and a `Filter` that keeps a record of all pages visited by a given session.

The `HistoryFilter` can be used to make a user's session "mobile" as follows: A user's history is normally stored with the browser being used, on the user's machine. If the user runs a different browser or goes to a different machine, the user's history will not be there. Instead, the user can access the web via a proxy that keeps track of their history. No matter which browser the user chooses or machine the user is at, a server running with the `HistoryFilter` will automatically remember and be able to present the user's history.

The history is kept with respect to a Session ID.

This filter uses the following configuration properties:

`prefix`
: This handler will only process URLs beginning with this string. The default value is "", which matches all URLs.

`session`
: The name of the request property that holds the Session ID. The default value is "SessionID".

`nosession`
: The Session ID to use if the Session ID was not specified. The default value is "common".

`admin`
: URLs beginning with this prefix cause the `HistoryFilter` to store the history information for the current Session in the request properties

`filter`
: If specified, then this is a `Regexp` pattern to match against the "Content-Type" of the result. Setting this also implies that the `HistoryFilter` will be invoked as a `Filter` and not a `Handler`. The default value is "", which indicates that the "Content-Type" is **not** examined and that this `HistoryFilter` will be invoked as a `Handler`.

Field Summary

String	**admin**
Regexp	**filter**
String	**nosession**
String	**session**
String	**urlPrefix**

Constructor Summary

HistoryFilter()	

Method Summary

byte[]	**filter**(Request request, MimeHeaders headers, byte[] content) Returns the original content, since this filter does not change content.
boolean	**init**(Server server, String prefix) Initializes this filter by reading all its configuration properties.
boolean	**respond**(Request request) If the admin prefix is seen, store the history information associated with the session in the request properties.
boolean	**shouldFilter**(Request request, MimeHeaders headers) Called when invoked as a Filter.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

urlPrefix

```
public String urlPrefix
```

---

session

```
public String session
```

---

nosession

```
public String nosession
```

---

admin

```
public String admin
```

---

filter

```
public Regexp filter
```

Constructor Detail

HistoryFilter

```
public HistoryFilter()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Initializes this filter by reading all its configuration properties.

It is an error if the `filter` is specified but malformed.

Specified by:

init in interface Handler

Parameters:
`server` - The HTTP server.
`prefix` - The configuration property prefix.
Returns:
`true` if this filter initialized successfully, `false` otherwise.

---

respond

```
public boolean respond(Request request)
```

If the `admin` prefix is seen, store the history information associated with the session in the request properties.

If invoked as a `Handler` and the URL matches the `prefix`, records this page's address in the history.
Specified by:
respond in interface Handler
Parameters:
`request` - The `Request` object that represents the HTTP request.
Returns:
`false`, indicating that this `respond` method ran purely for its side effects.

---

shouldFilter

```
public boolean shouldFilter(Request request,
                            MimeHeaders headers)
```

Called when invoked as a `Filter`. If the URL matches the `prefix` and the returned "Content-Type" matches the `filter`, records this page's address in the history.
Specified by:
shouldFilter in interface Filter
Parameters:
`request` - The in-progress HTTP request.
`headers` - The MIME headers from the result.
Returns:
`false` indicating that this `Filter` does not want to modify the content.

---

filter

```
public byte[] filter(Request request,
                     MimeHeaders headers,
                     byte[] content)
```

Returns the original content, since this filter does not change content. Won't actually be invoked.
Specified by:
filter in interface Filter
Tags copied from interface: Filter

Parameters:

`request` - The finished HTTP request.
`headers` - The MIME headers generated by the `Handler`.
`content` - The output from the `Handler` that this `Filter` may rewrite.

Returns:

The rewritten content. The `Filter` may return the original `content` unchanged. The `Filter` may return `null` to indicate that the `FilterHandler` should stop processing the request and should not return any content to the client.

Class JunkBusterHandler

sunlabs.brazil.proxy
Class JunkBusterHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.proxy.JunkBusterHandler
```

All Implemented Interfaces:
Handler

---

public class **JunkBusterHandler**
extends Object
implements Handler

Remove *junk* images from web pages. This approach is to take all requests for images that look like ads and instead return a dummy bitmap.

Other approaches to removing ads are to filter the HTML returned and (1) remove the ads altogether or (2) change the href in the ads to point to a different bitmap. The advantage of option (2) is that all ads can be changed to point to the same bitmap, increasing the caching performance of the browser.

Properties:
image
The file to contain the replacement image.
host
The regular expression matching url's to reject. If the expression starts with a '@', it interpreted as a file name (minus the @) that contains a new-line separated list of regular exporessions. See `Regexp` for more information on regular expressions.

Constructor Summary

`JunkBusterHandler()`	

Method Summary	
boolean	**init**(Server server, String prefix) Initializes the handler.
Regexp	**loadUrls**(Properties props, String prefix, String file)
boolean	**respond**(Request request) Responds to an HTTP request.
boolean	**sendReplacementImage**(Request request)

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

JunkBusterHandler

```
public JunkBusterHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
                throws IOException
```

Description copied from interface: Handler

Responds to an HTTP request.

Specified by:

respond in interface Handler

Tags copied from interface: Handler

Parameters:

`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

loadUrls

```
public Regexp loadUrls(Properties props,
                       String prefix,
                       String file)
```

---

sendReplacementImage

```
public boolean sendReplacementImage(Request request)
                            throws IOException
```

Class ProxyHandler

sunlabs.brazil.proxy
Class ProxyHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.proxy.ProxyHandler
```

All Implemented Interfaces:
Handler

---

public class **ProxyHandler**
extends Object
implements Handler

Handler for implementing a web proxy. By default, this is a dumb proxy. It can be combined with other handlers to generate side effects, such as content rewriting.

Properties:
useproxy
The name of the SocketFactory class to use for this handler. If additional properties are required to set up the SocketFactory, it should be configured as a handler instead. This is here for convenience only.
auth
The value of the proxy-authenticate header (if any) sent to the upstream proxy
proxyHost
If specified, the name of the upstream proxy
proxyPort
The up stream proxys port, if a proxyHost is specified (defaults to 80)

Field Summary

String	**auth** The string to send as the value for the "Proxy-Authorization" HTTP header (if needed).
static String	**AUTH**
static String	**PROXY_HOST**
static String	**PROXY_PORT**
String	**proxyHost** The proxy server.
int	**proxyPort** The proxy server's port.
static String	**USE_PROXY**

Constructor Summary

ProxyHandler()	

Method Summary

static String	**dumpHeaders**(int count, Request request, MimeHeaders headers, boolean sent) Dump the headers on stderr
boolean	**init**(Server server, String prefix) Do one-time setup.
boolean	**respond**(Request client) Responds to an HTTP request.

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

PROXY_HOST

`public static final String` **`PROXY_HOST`**

PROXY_PORT

`public static final String` **`PROXY_PORT`**

AUTH

`public static final String` **`AUTH`**

USE_PROXY

`public static final String` **`USE_PROXY`**

---

proxyHost

`public String` **`proxyHost`**

The proxy server.

---

proxyPort

`public int` **`proxyPort`**

The proxy server's port. Default is 80.

---

auth

`public String` **`auth`**

The string to send as the value for the "Proxy-Authorization" HTTP header (if needed).

Constructor Detail

ProxyHandler

```
public ProxyHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Do one-time setup. get and process the properties file options, and make sure

Specified by:

init in interface Handler

Tags copied from interface: Handler

Parameters:

`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.

`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request client)
                throws IOException
```

Description copied from interface: Handler

Responds to an HTTP request.

Specified by:

respond in interface Handler

See Also:

`Handler.respond(sunlabs.brazil.server.Request)`

---

dumpHeaders

```
public static String dumpHeaders(int count,
                                 Request request,
                                 MimeHeaders headers,
                                 boolean sent)
```

Dump the headers on stderr

Interface UseProxy

sunlabs.brazil.proxy
Interface UseProxy

---

public interface **UseProxy**

This interface is used by the `ProxyHandler` class to decide whether to issue an HTTP request directly to the specified host, or to issue the request via an HTTP proxy.

Method Summary

`boolean`	**`useProxy`**`(String host, int port)` Determines if the user can issue a direct or proxy request to the specified host and port.

Method Detail

useProxy

```
public boolean useProxy(String host,
                        int port)
```

Determines if the user can issue a direct or proxy request to the specified host and port.

The actual HTTP proxy to use is specified external to this routine by some other mechanism.

Parameters:

`host` - The host name.
`port` - The port number.

Returns:

`true` if the user should send the HTTP request via an HTTP proxy, `false` if the user can send the HTTP request directly to the specified named host.

Package sunlabs.brazil.python

Provide ways of integrating the Python scripting language into the Brazil project server.

Python is an object oriented scripting language, a version of which (jPython, or its successor, Jython) is written in the Java programming language. Various parts of the Brazil project system can be scripted in Python. All of the classes in this package provide mechanisms for accessing the Python language in the Brazil project server context. See www.python.org for more information about Python.

Files in this package depend upon the `jython.jar` file, which may be downloaded from www.jython.org

Class PythonServerTemplate

sunlabs.brazil.python
Class PythonServerTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.python.PythonServerTemplate
```

All Implemented Interfaces:
TemplateInterface

---

public class **PythonServerTemplate**
extends Template

The `PythonServerTemplate` looks for each `<server language="python">` (or `<python>`) tag in an HTML page and treats the following data up to the next `</server>` (or `</python>`) tag as a python script to evaluate.

The reason that python scripts are included in an HTML page is usually to generate dynamic, server-side content. After running this template, everything between and including the `<server>` and `</server>` (or `<python>` and `</python>` tags is replaced by all output written to the Python standard output stream (if any).

All Python fragments within a given page are evaluated in the same Python interpreter. The Python interpreter actually lives for the entire duration of this `Template` object, so the user can implement persistence across requests.

The following configuration parameters are used to initialize this template.

script
: The name of the Python script to evaluate when the interpreter is created. This script only evaluated when the interp is created, not on every request. The variables `prefix` and `server` are set before this file is evaluated, and are references to the parameters passed to a `handler` init method.

root
: The document root, if the script is a relative file name. If the "root" property under the template prefix is not found, the global "root" property is used. If the global "root" property is not found, the current directory is used.

debug
: If this configuration parameter is present, this class replaces the `<server>` and `</server>` tags with comments, so the user can keep track of where the dynamically generated content is coming from by examining the comments in the resultant HTML document. By default, the `<server>` and `</server>` are completely eliminated from the HTML document rather than changed into comments.

Before evaluating each HTML document, this class sets variables in the Python interpreter, which can be used to interact back with Java to do things like set the response headers:

request
: Exposes the `Request` Java object. It is set anew at each request.

prefix
: Exposes the handler prefix String.

server
: Exposes the handler `Server` object.

If the attribute `eval` is present as an attribute, all constructs off the form ${...} are substituted before processing the script.

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

`PythonServerTemplate``()`	

Method Summary

`boolean`	**`init`**`(RewriteContext hr)` Called before this template processes any tags.
`void`	**`tag_python`**`(RewriteContext hr)` Processes the `<python>` tag.
`void`	**`tag_server`**`(RewriteContext hr)` Processes the `<server>` tag.

Methods inherited from class sunlabs.brazil.template.Template
`done`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

PythonServerTemplate

`public PythonServerTemplate()`

Method Detail

init

`public boolean init(RewriteContext hr)`

Description copied from class: Template
Called before this template processes any tags.
Overrides:
init in class Template

---

tag_server

`public void tag_server(RewriteContext hr)`

Processes the `<server>` tag. Substitues the result of evaluating the following Python script into the resultant HTML document.

Note: Currently, there is no mechanism for other language interpreters to share the same `server` tag.
Parameters:
`hr` - The request and associated HTML document that will be processed.

---

tag_python

`public void tag_python(RewriteContext hr)`

Processes the `<python>` tag. Substitues the result of evaluating the following Python script into the resultant HTML document.
Parameters:
`hr` - The request and associated HTML document that will be processed.

Package sunlabs.brazil.server

Generic http protocol stack, essential handlers and drivers.

This package consists of a small server side http protocol stack, designed to be embedded in other applications. The primary classes in this package are `Server`, `Request` and `Main` The remaining classes are in this package either because they are tightly bound to the implementation of the "Server" and "Request" classes, or they are almost always used with the basic server.

The design points are:

Extremely small footprint

Unlike a traditional webserver, designed primarily for serving web documents, this protocol stack is bare-bones, providing only the essentials for manipulating http requests and responses. It supports HTTP/1.0, and the important features of HTTP/1.1.

Small, simple extension API.

The programming interface is designed for simplicity. There are few convenience methods provided, and consequently no bias toward any pre-conceived notions as to what http protocol stacks should be used for.

Classes that implement this API are called **handlers**, to distinguish them from the moral equivalent, but more complex **servlets** used by the Java Web Server.

Delegation based object model

Inheritance is used sparingly in favor of delegation. This alleviates the need for base or abstract classes to *know* what the application has in mind.

To ease the deployment and testing of applications that use the server, two additional capabilities are included: a simple driver (main) program, and a coupler of *handler* classes, so the protocol stack can be used *out of the box* to provide a simple file service.

Class ChainHandler

sunlabs.brazil.server
Class ChainHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.server.ChainHandler
```

All Implemented Interfaces:
Handler
Direct Known Subclasses:
ChainSawHandler

---

public class **ChainHandler**
extends Object
implements Handler

Allows multiple handlers to be invoked sequentially for a single HTTP request. A list of handlers is supplied when this `ChainHandler` is initialized. When an HTTP request is received by this `ChainHandler`, each of the handlers from the list is called in turn until one of them responds and returns `true`.

A useful trick is that some handlers can be run by a `ChainHandler` for their side effects. The handler can modify the `Request` object and then return `false`; the next handler in the list will get a crack at the modified request.

The following configuration parameters eare used to initialize this `Handler`:

`handlers`
: A list of `Handler` names that will be invoked in the given order to handle the request. These are considered the "wrapped" handlers. These handlers will all be initialized at startup by `init(sunlabs.brazil.server.Server, java.lang.String)`. For each name in the list, the property *`name`*`.class` is examined to determine which class to use for this handler. Then `name` is used as the prefix in the handler's init() method.

`report`
: If set, this property will be set to the name of the handler that handled the request (e.g. returned true).

`exitOnError`
: If set, the server's `initFailure` will set any of the handlers fail to initialize. No handler prefix is required.

`prefix`
: If set, All url's must start with this, or the request will be ignored

See Also:
`Handler`

Field Summary

boolean	**exitOnError** A flag to require the successfull initialization of all handlers.
Handler[]	**handlers** The array of handlers that will be invoked to handle the request.
String[]	**names** The names of the above `handlers` as specified by the configuration parameters.
String	**prefix** The prefix used to initialize this `ChainHandler`, used for logging.
String	**report** The name (if any) of the property to receive the name of the handler that handled the request.
String	**urlPrefix** The URL prefix that must match for this handler to run

Constructor Summary

ChainHandler()	

Method Summary

boolean	**init**(Server server, String prefix) Initializes this `ChainHandler` by initializing all the "wrapped" handlers in the list of handlers.
static Handler	**initHandler**(Server server, String prefix, String name) Helper function that allocates and initializes a new `Handler`, given its name.
boolean	**respond**(Request request) Calls each of the `Handler`s in turn until one of them returns `true`.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

handlers

`public Handler[] **handlers**`

The array of handlers that will be invoked to handle the request.

---

names

`public String[] **names**`

The names of the above `handlers` as specified by the configuration parameters. Used for logging the names of each `Handler` as it is invoked.

---

prefix

`public String **prefix**`

The prefix used to initialize this `ChainHandler`, used for logging.

---

urlPrefix

`public String **urlPrefix**`

The URL prefix that must match for this handler to run

---

report

`public String **report**`

The name (if any) of the property to receive the name of the handler that handled the request.

---

exitOnError

`public boolean **exitOnError**`

A flag to require the successfull initialization of all handlers.

Constructor Detail

ChainHandler

```
public ChainHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Initializes this `ChainHandler` by initializing all the "wrapped" handlers in the list of handlers. If a wrapped handler cannot be initialized, this method logs a message and skips it. If no handlers were specified, or no handlers were successfully initialized, then the initialization of this `ChainHandler` is considered to have failed.

Specified by:

init in interface Handler

Parameters:

`server` - The HTTP server that created this `ChainHandler`.

`prefix` - The prefix for this `ChainHandler`'s properties.

Returns:

`true` if at least one of the wrapped handlers was successfully initialized.

---

initHandler

```
public static Handler initHandler(Server server,
                                  String prefix,
                                  String name)
```

Helper function that allocates and initializes a new `Handler`, given its name. In addition to the `ChainHandler`, several other handlers contain embedded `Handler`s -- this method can be used to initialize those embedded `Handler`s.

If there is an error initializing the specified `Handler`, this method will log a dignostic message to the server and return `null`. This happens if the specified class cannot be found or instantiated, if the specified class is not actually a `Handler`, if the `Handler.init` method returns `false`, or if there is any other exception.

Parameters:

`server` - The server that will own the new `Handler`. Mainly used for the server's properties, which contain the configuration parameters for the new handler.

`prefix` - The prefix in the server's properties for the new `Handler`'s configuration parameters. The prefix is prepended to the configuation parameters used by the `Handler`.

`name` - The name of the new `Handler`. The name can be one of two forms:

1. The name of the Java class for the `Handler`. This `Handler` will be initialized using the `prefix` specified above.
2. A symbolic `name`. The configuration parameter *`name`*`.class` is the name of the Java class for the `Handler`. The above `prefix` will be ignored and this `Handler` will be initialized

with the prefix "*name*." (the symbolic name followed by a ".").

Returns:

The newly allocated `Handler`, or `null` if the `Handler` could not be allocated.

---

respond

```
public boolean respond(Request request)
                throws IOException
```

Calls each of the `Handler`s in turn until one of them returns `true`.

Specified by:

respond in interface Handler

Parameters:

`request` - The HTTP request.

Returns:

`true` if one of the `Handler`s returns `true`, `false` otherwise.

Throws:

IOException - if one of the `Handler`s throws an `IOException` while responding.

Class FileHandler

sunlabs.brazil.server
Class FileHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.server.FileHandler
```

All Implemented Interfaces:
Handler

---

public class **FileHandler**
extends Object
implements Handler

Standard handler for fetching static files. This handler does URL to file conversion, file suffix to mime type lookup, delivery of *index* files where providing directory references, and redirection for missing slashes (/) at the end of directory requests.

The following configuration parameters are used:

root
: property for document root (.) Since the document root is common to many handlers, if no root property is found with the supplied prefix, then the root property with the empty prefix ("") is used instead. This allows many handlers to share the common property.

default
: The document to deliver if the URL ends in "/". (defaults to index.html.)

prefix
: Only url's that start with this are allowed. defaults to "". The prefix is removed from the url before looking it up in the file system. So, if prefix is `/foo` then the the file `[root]/foo/bar.html` will be delivered in response to the url `/bar.html`.

mime
: property for mime type For each file suffix .XX, the property mime.XX is used to determine the mime type. If no property exists (or its value is "unknown", the document will not be delivered.

mimePatterns
: List of glob patterns that match file name suffixes for matching mime types. For example:

```
mimePatterns=.x* .a?
mime.x*=text/xml
mime.a?=application/octet-stream
```

The types corrosponding to mime patterns are searched for in mimePattern order, first looking for `prefix.mime.pattern` then `mime.pattern`. If neither property exists, then the type is invalid.

getOnly
: If defined, only "GET" requests will be processed. By default, all request types are handled. (Note: this is the inverse of the previous policy, defined by the undocumented "allow" parameter).

The FileHandler sets the following entries in the request properties as a side-effect:

fileName
: The absolute path of the file that couldn't be found.

DirectoryName
: If the URL specified is a directory name, its absolute path is placed here.

lastModified
: The Time stamp of the last modified time

This handler supports a subset of the http `range` header of the form `range bytes=[start]-[end]`, where *start* and *end* are byte positions in the file, starting at zero. A large or missing *end* value is treated as the end of the file. If a valid `range`header is found, the appropriate `content-range` header is returned, along with the partial contents of the file.

Field Summary

static String	**MIME**
static String	**ROOT**
static String	**UNKNOWN**
String	**urlPrefix**

Constructor Summary

FileHandler()	

Method Summary

static String	**getMimeType**(String name, Properties props, String prefix) Get the mime type based on the suffix of a String.
boolean	**init**(Server server, String prefix) Initialize the file handler.
boolean	**respond**(Request request) Find, read, and deliver via http the requested file.
static void	**sendFile**(Request request, File file, int code, String type) Send a file as a response.
static String	**urlToPath**(String url) Helper function to convert an url into a pathname.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

MIME

public static final String **MIME**

UNKNOWN

public static final String **UNKNOWN**

ROOT

public static final String **ROOT**

---

urlPrefix

public String **urlPrefix**

Constructor Detail

FileHandler

```
public FileHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Initialize the file handler.

Specified by:

init in interface Handler

Returns:

The file handler always returns true.

---

respond

```
public boolean respond(Request request)
               throws IOException
```

Find, read, and deliver via http the requested file. The server property `root` is used as the document root. The document root is recalculated for each request, so an upstream handler may change it for that request. For URL's ending with "/", the server property `default` (normally index.html) is automatically appended. If the file suffix is not found as a server property `mime.`*`suffix`*, the file is not delivered.

Specified by:

respond in interface Handler

Tags copied from interface: Handler

Parameters:

`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

getMimeType

```
public static String getMimeType(String name,
                                 Properties props,
                                 String prefix)
```

Get the mime type based on the suffix of a String. The suffix (e.g. text after the last ".") is used to lookup the entry "prefix.mime.[suffix]" then "mime.[suffix]" in `props`. If neither entry is found, then mime glob pattern are used, if available. If there is no suffix, then the empty string is used.

If the mime type is set to the special string "unknown", then the type is unknown. This allows specific types to be undefined when glob patterns are used.

If the property "prefix.mimePatterns" (or "mimePatterns") exists, then it specifies a white-space delimited set of glob style patterns for matching file suffixes to types. If a match for a specific file suffix fails, then the property "mime.[pattern]" is used for type comparisons.

The entries:

```
mimePatterns=*ml
mime.*ml=text/xml
```

would associate the type "text/xml" with the file foo.html, foo.xml and foo.dhtml. The entries:

```
mimePatterns=*
mime*=application/octet-stream
mime.config=unknown
```

Would set the types for all file types not otherwise defined to be "application/octet-stream", except that files ending in ".config" would have no type (e.g. they would generate a file not found error).

Parameters:

`name` - The string to compute the mime type for

`props` - The properties to look up the mime types in.

Returns:

The type (or null if not found).

---

urlToPath

```
public static String urlToPath(String url)
```

Helper function to convert an url into a pathname.

- Collapse all %XX sequences.
- Ignore missing initial "/".
- Collapse all "/..", "/.", and "//" sequences.

`URL(String)` collapses all "/.." (and "/.") sequences, except for a trailing "/.." (or "/."), which would lead to the possibility of escaping from the document root.

`File.getPath` in jdk-1.1 leaves all the "//" constructs in, but it collapses them in jdk-1.2, so we have to always take it out ourselves, just to be sure.

Parameters:

`url` - The file path from the URL (that is, minus the "http://host" part). May be `null`.

Returns:

The path that corresponds to the URL. The returned value begins with "/". The caller can concatenate this path onto the end of some document root.

---

sendFile

```
public static void sendFile(Request request,
                            File file,
                            int code,
                            String type)
                     throws IOException
```

Send a file as a response.

Parameters:

`request` - The request object

`fileHandle` - The file to output

`type` - The mime type of the file

Interface Handler

sunlabs.brazil.server
Interface Handler
All Known Subinterfaces:
Filter
All Known Implementing Classes:
AclSwitchHandler, BasicAuthHandler, BeanShellHandler, CgiHandler, ChainHandler, ChownHandler, ConfigFileHandler, CookieSessionHandler, DefaultFileHandler, DeferredHandler, DialogHandler, DigestAuthHandler, DirectoryHandler, DynamicConfigHandler, ExprPropsHandler, FilterHandler, GenericProxyHandler, HomeDirHandler, JunkBusterHandler, LogHandler, MultiHostHandler, NotFoundHandler, PollHandler, PropertiesHandler, ProxyHandler, ProxyPropertiesHandler, PublishHandler, PushHandler, ReflectHandler, ResourceHandler, ResourceLimitHandler, RestrictClientHandler, RolesHandler, SimpleSessionHandler, SMTPHandler, SubstPropsHandler, SupplyHandler, UrlMapperHandler, VirtualHostHandler, FileHandler, CacheManager, PropertiesCacheManager, SslHandler, CertHandler, BasicSSLHandler, DelayHandler, RestartHandler, StunnelHandler, TclHandler, SetTemplate, DirectoryTemplate, TemplateHandler, DerbyServer, AsteriskHandler, AsteriskAGIHandler

---

public interface **Handler**

The interface for writing HTTP handlers. Provides basic functionality to accept HTTP requests and dispatch to methods that handle the request.

The `init(Server, String)` method is called before this `Handler` processes the first HTTP request, to allow it to prepare itself, such as by allocating any resources needed for the lifetime of the `server`.

The `respond(Request)` method is called to handle an HTTP request. This method, and all methods it calls must be thread-safe since they may handle HTTP requests from multiple sockets concurrently. However, each concurrent request gets its own individual `Request` object.

Any instance variables should be initialized in the `init(Server, String)`, and only referenced, but not set in the `respond(Request)` method. If any state needs to be retained, it should be done either by associating it with the `Request` object, or using the `session manager`. Class statics should be avoided, as it is possible, and even common to run multiple unrelated Brazil servers in the same JVM. As above, the `session manager` should be used instead.

Method Summary

`boolean`	**`init`**`(Server server, String prefix)` Initializes the handler.
`boolean`	**`respond`**`(Request request)` Responds to an HTTP request.

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Initializes the handler.

Parameters:

`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.

`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
                throws IOException
```

Responds to an HTTP request.

Parameters:

`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

Class Main

sunlabs.brazil.server
Class Main

```
java.lang.Object
  |
  +--sunlabs.brazil.server.Main
```

---

public class **Main**
extends Object

Sample **main** program for starting an http server. A new thread is started for each `Server`, listening on a socket for HTTP connections. As each connection is accepted, a `Request` object is constructed, and the registered `Handler` is called. The configuration properties required by the server and the handler (or handlers), are gathered from command line arguments and configuration files specified on the command line.

The command line arguments are processed in order from left to right, with the results being accumulated in a properties object. The server is then started with `Server.props` set to the final value of the properties. Some of the properties are interpreted directly by the server, such as the port to listen on, or the handler to use (see `Server` for the complete list). The rest are arbitrary name/value pairs that may be used by the handler.

Although any of the options may be specified as name/value pairs, some of them: the ones interpreted by the server, the default handler (`FileHandler`, or `Main`, may be prefixed with a "-". Those options are explained below:

-p(ort)
: The network port number to run the server on (defaults to 8080)

-r(oot)
: The document root directory, used by the FileHandler (defaults to .)

-h(andler)
: The document handler class (defaults to `sunlabs.brazil.handler.FileHandler`)

-c(onfig)
: A java properties file to add to the current properties. There may be several *-config* options. Each file is added to the current properties. If the properties file contains a `root` property, it is treated specially. See below. If the config file is not found in the filesystem, it is read from the jar file, with this class as the virtual current directory if a relative path is provided.

-i(p)
: A space seperated list of hosts allowed to access this server If none are supplied, any host may connect. The ip addresses are resolved once, at startup time.

-l(og)
: The log level (0->none, 5->max) Causes diagnostic output on the standard output.

-s(tart)
: Start a server. Allows multiple servers to be started at once. As soon as a *-s* is processed, as server is started as if all the options had been processed, then the current properties are cleared. Any options that follow are used for the next server.

-D(elay) n
: delay "n" seconds. This is useful in conjuction with "-s" to allow the previous server to initialize before proceeding.

-S(ubstitute)
: Perform ${..} substitutions on the current values.

-x
: Don't read the default resource config (only if first).

Following these options, any additional additional pairs of names and values (no "-"'s allowed) are placed directly in `Server.props`.

If the resource "/sunlabs/brazil/server/config" is found, it is used to initialize the configuration.

If a non absolute `root` property is specified in a *configuration* file, it is modified to resolve relative to the directory containing the *configuration* file, and not the directory in which the server was started. If multiple *configuration* files with root properties (or `-r` options, or "root" properties) are specified, the last one tekes precedence.

The "serverClass" property may be set to the string to use as the server's class, instead of "sunlabs.brazil.server.Server"

Constructor Summary

Main()	

Method Summary

static Object	**initObject**(Server server, String name)
static void	**initProps**(Properties config) Initialize a properties file with some standard mime types The `FileHandler` only delivers files whose suffixes are known to map to mime types.
static void	**main**(String[] args)
static boolean	**startServer**(Properties config) Start a server using the supplied properties.

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

Main

```
public Main()
```

Method Detail

main

```
public static void main(String[] args)
                 throws Exception
```

---

startServer

```
public static boolean startServer(Properties config)
```

Start a server using the supplied properties. The following entries are treated. Specially:

handler
: The name of the handler class (defaults to file handler)

host
: The host name for this server

log
: Diagnostic output level 0-5 (5=most output)

maxRequests
: max number of requests for a single socket (default 25) when using persistent connections.

listenQueue
: max size of the OS’slisten queue for server sockets

maxPost
: max size of a content-length for a Post or Put in bytes. (defaults to 2Meg)

maxThreads
: max number of threads allowed (defaults to 250)

port
: Server port (default 8080)

defaultPrefix
: prefix into the properties file, normally the empty string "".

restrict
: list of hosts allowed to connect (defaults to no restriction)

timeout
: The maximum time to wait for a client to send a complete request. Defaults to 30 seconds.

interfaceHost
: If specified, a host name that represents the network to server. This is for hosts with multiple ip addresses. If no network host is specified, then connections for all interfaces are accepted

Parameters:

`config` - The configuration properties for the server

---

initObject

```
public static Object initObject(Server server,
                                String name)
```

---

initProps

```
public static void initProps(Properties config)
```

Initialize a properties file with some standard mime types The `FileHandler` only delivers files whose suffixes are known to map to mime types. The server is started with the suffixes: .html, .txt, .gif, .jpg, pdf, .png, .css, .class, and .jar predefined. If additional types are required, they should be supplied as command line arguments.

Class Request.HttpOutputStream

sunlabs.brazil.server
Class Request.HttpOutputStream

```
java.lang.Object
  |
  +--java.io.OutputStream
        |
        +--java.io.FilterOutputStream
              |
              +--sunlabs.brazil.server.Request.HttpOutputStream
```

Direct Known Subclasses:
BServletRequest.HttpOutputStream
Enclosing class:
Request

---

public static class **Request.HttpOutputStream**
extends FilterOutputStream

The `HttpOutputStream` provides the convenience method `writeBytes` for writing the byte representation of a string, without bringing in the overhead and the deprecated warnings associated with a `java.io.DataOutputStream`.

The other methods in this class are here to allow the `FilterHandler` and `ChainSawHandler` to alter the behavior in an implementation specific way. This behavior is unfortunate, and might go away when a better strategy comes along.

Field Summary

int	**bytesWritten** Count the number of bytes that are written to this stream

Constructor Summary

Request.HttpOutputStream(OutputStream out)

Method Summary

void	**sendHeaders**(Request request)
void	**write**(byte b)
void	**write**(byte[] buf, int off, int len)
void	**writeBytes**(String s)

Methods inherited from class java.io.FilterOutputStream
close, flush, write, write

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

bytesWritten

public int **bytesWritten**

Count the number of bytes that are written to this stream

Constructor Detail

Request.HttpOutputStream

public **Request.HttpOutputStream**(OutputStream out)

Method Detail

writeBytes

```
public void writeBytes(String s)
               throws IOException
```

---

write

```
public void write(byte b)
           throws IOException
```

---

write

```
public void write(byte[] buf,
                  int off,
                  int len)
           throws IOException
```

Overrides:

write in class FilterOutputStream

---

sendHeaders

```
public void sendHeaders(Request request)
                throws IOException
```

Class Request

sunlabs.brazil.server
Class Request

```
java.lang.Object
  |
  +--sunlabs.brazil.server.Request
```

Direct Known Subclasses:
BServletRequest, TestRequest, VelocityFilter.Vrequest

---

public class **Request**
extends Object

Represents an HTTP transaction. A new instance is created by the server for each connection.

Provides a set of accessor functions to fetch the individual fields of the HTTP request.

Utility methods that are generically useful for manipulating HTTP requests are included here as well. An instance of this class is passed to handlers. There will be exactly one request object per thead at any time.

The fields `headers`, `query`, and `url`, and the method `getQueryData()` are most often used to examine the content of the request. The field `props` contains information about the server, or up-stream handlers.

The methods `sendResponse(String, String, int)` and `sendError(int, String)` are commonly used to return content to the client. The methods `addHeader(String)` and `setStatus(int)` can be used to modify the response headers and return code respectively before the response is sent.

Many of the other methods are used internally, but can be useful to handlers that need finer control over the output that the above methods provide. Note that the order of the methods is important. For instance, the user cannot change the HTTP response headers (by calling the `addHeader` method or by modifying the `responseHeaders` field) after having already sent an HTTP response.

A number of the fields in the `Request` object are public, by design. Many of the methods are convenience methods; the underlying data fields are meant to be accessed for more complicated operations, such as changing the URL or deleting HTTP response headers.

See Also:
`Handler`, `Server`

Inner Class Summary

static class	**Request.HttpOutputStream** The HttpOutputStream provides the convenience method writeBytes for writing the byte representation of a string, without bringing in the overhead and the deprecated warnings associated with a java.io.DataOutputStream.

Field Summary

String	**connectionHeader** The header "Connection" usually controls whether the client connection will be of type "Keep-Alive" or "close".
MimeHeaders	**headers** The HTTP request headers.
boolean	**keepAlive** true if the client requested a persistent connection, false otherwise.
static int	**MAX_BLANKS** Maximum number of blank lines allowed between requests before aborting the connecion.
String	**method** The HTTP request method, such as "GET", "POST", or "PUT".
Request.HttpOutputStream	**out** The HTTP response to the client is written to this stream.
byte[]	**postData** The uploaded content of this request, usually from a POST.
PropertiesList	**props** A set of properties local to this request.
String	**protocol** The HTTP protocol specified in the request, either "HTTP/1.0" or "HTTP/1.1".
String	**query** The query string specified after the URL, or "" if no query string was specified.
MimeHeaders	**responseHeaders** The HTTP response headers.
Server	**server** The server that initiated this request.

PropertiesList	**serverProps** A PropertiesList object that wraps server.props.
String	**serverProtocol** This is the server's protocol.
Socket	**sock** Our connection to the client.
long	**startMillis** Time stamp for start of this request - set, but not used.
String	**url** The URL specified in the request, not including any "?" query string.
int	**version** Derived from protocol, the version of the HTTP protocol used for this request.

Method Summary

void	**addHeader**(String line) Adds a response header to the HTTP response.
void	**addHeader**(String key, String value) Adds a response header to the HTTP response.
boolean	**addSharedProps**(Dictionary d) Adds the given Dictionary to the set of properties that are searched by request.props.getProperty().
Hashtable	**getQueryData**() Retrieves the query data as a hashtable.
Hashtable	**getQueryData**(Hashtable table) Retrieves the query data as a hashtable.
boolean	**getRequest**() Reads an HTTP request from the socket.
String	**getRequestHeader**(String key) Returns the value that the given case-insensitive key maps to in the HTTP request headers.
int	**getReuseCount**() Return uses of this socket

Socket	**getSocket()** The socket from which the HTTP request was received, and to where the HTTP response will be written.
int	**getStatus()** Return the status code.
void	**log**(int level, Object obj, String message) Logs a message by calling Server.log.
void	**log**(int level, String message) Logs a message by calling Server.log.
void	**redirect**(String url, String body) Responds to an HTTP request with a redirection reply, telling the client that the requested url has moved.
boolean	**removeSharedProps**(Dictionary d) Removes a Dictionary added by addSharedProps.
void	**sendError**(int code, String clientMessage) Sends a HTTP error response to the client.
void	**sendError**(int code, String clientMessage, String logMessage) Sends a HTTP error response to the client.
void	**sendHeaders**(int code, String type, int length) Sends the HTTP status line and response headers to the client.
void	**sendResponse**(byte[] body, String type) Sends an HTTP response to the client.
void	**sendResponse**(InputStream in, int length, String type, int code) Sends the contents of the given input stream as the HTTP response.
void	**sendResponse**(String body) Convenience method that sends an HTTP response to the client with a "Content-Type" of "text/html" and the default HTTP status code.
void	**sendResponse**(String body, String type) Convenience method that sends an HTTP response to the client with the default HTTP status code.
void	**sendResponse**(String body, String type, int code) Sends an HTTP response to the client.
String	**serverUrl()** Returns the server’s fully-qualified base URL.
void	**setStatus**(int code) Sets the status code of the HTTP response.

String	**toString()** Returns a string representation of this `Request`.

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Field Detail

MAX_BLANKS

`public static final int` **`MAX_BLANKS`**

Maximum number of blank lines allowed between requests before aborting the connecion. The spec allows 0, but some clients add one or more.

---

server

`public Server` **`server`**

The server that initiated this request. Only under rare circumstances should this be modified.

---

sock

`public Socket` **`sock`**

Our connection to the client. Only under rare circumstances would this need to be modified.

---

props

`public PropertiesList` **`props`**

A set of properties local to this request. The property is wrapped in a `PropertiesList` object and initially is the head of a linked list of properties that are searched in order. This is useful for handlers that wish to communicate via properties to down-stream handlers, such as modifying a server property for a particular request. Some handlers may even add entire new sets of properties onto the front of `request.props` to temporarily modify the properties seen by downstream handlers.

---

serverProps

`public PropertiesList` **`serverProps`**

A `PropertiesList` object that wraps `server.props`. When this `request` is created, a new `PropertiesList` wrapping `server.props` is created and added to a list consisting only of `props` and `serverProps`.

---

out

`public Request.HttpOutputStream` **`out`**

The HTTP response to the client is written to this stream. Normally the convenience methods, such as `sendResponse`, are used to send the response, but this field is available if a handler needs to generate the response specially.

If the user chooses to write the response directly to this stream, the user is still encouraged to use the convenience methods, such as `sendHeaders`, to first send the HTTP response headers. The `FilterHandler` examines the HTTP response headers set by the convenience methods to determine whether to filter the output.

Note that the HTTP response headers will **not** automatically be sent as a side effect if the user writes to this stream. The user would either need to call the convenience method `sendHeaders` or need to generate the HTTP response headers themselves.

This variable is declared as a `Request.HttpOutputStream`, which provides the convenience method `writeBytes` to write the byte representation of a string back to the client. If the user does not need this functionality, this variable may be accessed simply as a normal `OutputStream`.
See Also:
`sendResponse(String, String, int)`, `sendHeaders(int, String, int)`

---

method

`public String` **`method`**

The HTTP request method, such as "GET", "POST", or "PUT".

---

url

`public String` **`url`**

The URL specified in the request, not including any "?" query string.

NOTE: Traditionally handlers modify this as needed. The request property "url.orig" is set to match the url at creation time, and should be considered "Read only", for those cases where the original url is required.

---

query

`public String` **`query`**

The query string specified after the URL, or `""` if no query string was specified.

---

protocol

`public String` **`protocol`**

The HTTP protocol specified in the request, either "HTTP/1.0" or "HTTP/1.1".
See Also:
`version`

---

version

`public int` **`version`**

Derived from `protocol`, the version of the HTTP protocol used for this request. Either `10` for "HTTP/1.0" or `11` for "HTTP/1.1".

---

headers

`public MimeHeaders` **`headers`**

The HTTP request headers. Keys and values in this table correspond the field names and values from each line in the HTTP header; field names are case-insensitive, but the case of the values is preserved. The order of entries in this table corresponds to the order in which the request headers were seen. Multiple header lines with the same key are stored as separate entries in the table.

---

postData

`public byte[]` **`postData`**

The uploaded content of this request, usually from a POST. Set to `null` if the request has no content.

---

keepAlive

`public boolean` **`keepAlive`**

`true` if the client requested a persistent connection, `false` otherwise. Derived from the `protocol` and the `headers`,

When "Keep-Alive" is requested, the client can issue multiple, consecutive requests via a single socket connection. By default:

- HTTP/1.0 requests are not Keep-Alive, unless the "Connection: Keep-Alive" header was present.
- HTTP/1.1 requests are Keep-Alive, unless the "Connection: close" header was present.

The user can change this value from `true` to `false` to forcefully close the connection to the client

after sending the response. The user can change this value from `false` to `true` if the client is using a different header to request a persistent connection. See `connectionHeader`.

Regardless of this value, if an error is detected while receiving or responding to an HTTP request, the connection will be closed.

---

connectionHeader

`public String` **`connectionHeader`**

The header "Connection" usually controls whether the client connection will be of type "Keep-Alive" or "close". The same header is written back to the client in the response headers.

The field `keepAlive` is set based on the value of the "Connection" header. However, not all clients use "Connection" to request that the connection be kept alive. For instance (although it does not appear in the HTTP/1.0 or HTTP/1.1 documentation) both Netscape and IE use the "Proxy-Connection" header when issuing requests via an HTTP proxy. If a `Handler` is written to respond to HTTP proxy requests, it should set `keepAlive` depending on the value of the "Proxy-Connection" header, and set `connectionHeader` to "Proxy-Connection", since the convenience methods like `setResponse()` use these fields when constructing the response. The server does not handle the "Proxy-Connection" header by default, since trying to pre-anticipate all the exceptions to the specification is a "slippery slope".

---

serverProtocol

`public String` **`serverProtocol`**

This is the server's protocol. It is normally null, but may be overriden to change the protocol on a per-request basis. If not set then server.protocol should be used instead.

---

responseHeaders

`public MimeHeaders` **`responseHeaders`**

The HTTP response headers. Keys and values in this table correspond to the HTTP headers that will be written back to the client when the response is sent. The order of entries in this table corresponds to the order in which the HTTP headers will be sent. Multiple header lines with the same key will be stored as separate entries in the table.

See Also:

`addHeader(String, String)`

---

startMillis

```
public long startMillis
```

Time stamp for start of this request - set, but not used.

Method Detail

toString

```
public String toString()
```

Returns a string representation of this `Request`. The string representation is the first line (the method line) of the HTTP request that this `Request` is handling. Useful for debugging.

Overrides:

toString in class Object

Returns:

The string representation of this `Request`.

---

getRequest

```
public boolean getRequest()
                   throws IOException
```

Reads an HTTP request from the socket.

Returns:

`true` if the request was successfully read and parsed, `false` if the request was malformed.

Throws:

IOException - if there was an IOException reading from the socket. See the socket documentation for a description of socket exceptions.

---

getSocket

```
public Socket getSocket()
```

The socket from which the HTTP request was received, and to where the HTTP response will be written. The user should not directly read from or write to this socket. The socket is provided other purposes, for example, imagine a handler that provided different content depending upon the IP address of the client.

Returns:

The client socket that issued this HTTP request.

---

log

```
public void log(int level,
                String message)
```

Logs a message by calling `Server.log`. Typically a message is generated on the console or in a log

file, if the `level` is less than the current server log setting.

Parameters:

`level` - The severity of the message.

`message` - The message that will be logged.

See Also:

`Server.log(int, Object, String)`

---

log

```
public void log(int level,
                Object obj,
                String message)
```

Logs a message by calling `Server.log`. Typically a message is generated on the console or in a log file, if the `level` is less than the current server log setting.

Parameters:

`level` - The severity of the message.

`obj` - The object that the message relates to.

`message` - The message that will be logged.

See Also:

`Server.log(int, Object, String)`

---

getRequestHeader

```
public String getRequestHeader(String key)
```

Returns the value that the given case-insensitive key maps to in the HTTP request headers. In order to do fancier things like changing or deleting an existing request header, the user may directly access the `headers` field.

Parameters:

`key` - The key to look for in the HTTP request headers. May not be `null`.

Returns:

The value to which the given key is mapped, or `null` if the key is not in the headers.

See Also:

`headers`

---

getQueryData

```
public Hashtable getQueryData(Hashtable table)
```

Retrieves the query data as a hashtable. This includes both the query information included as part of the url and any posted "application/x-www-form-urlencoded" data.

Parameters:

`table` - An existing hashtable in which to put the query data as name/value pairs. May be `null`, in which case a new hashtable is allocated.

Returns:
The hashtable in which the query data was stored.

---

getQueryData

`public Hashtable getQueryData()`

Retrieves the query data as a hashtable. This includes both the query information included as part of the url and any posted "application/x-www-form-urlencoded" data.
Returns:
The hashtable in which the query data was stored.

---

setStatus

`public void setStatus(int code)`

Sets the status code of the HTTP response. The default status code for a response is `200` if this method is not called.

An HTTP status phrase will be chosen based on the given status code. For example, the status code `404` will get the status phrase "Not Found".

If this method is called, it must be called before `sendHeaders` is either directly or indirectly called. Otherwise, it will have no effect.
Parameters:
`code` - The HTTP status code, such as `200` or `404`. If < 0, the HTTP status code will not be changed.
See Also:
`sendHeaders(int, String, int)`

---

getStatus

`public int getStatus()`

Return the status code.

---

getReuseCount

`public int getReuseCount()`

Return uses of this socket

---

addHeader

```
public void addHeader(String key,
                      String value)
```

Adds a response header to the HTTP response. In order to do fancier things like appending a value to an existing response header, the user may directly access the `responseHeaders` field.

If this method is called, it must be called before `sendHeaders` is either directly or indirectly called. Otherwise, it will have no effect.

Parameters:

`key` - The header name.

`value` - The value for the request header.

See Also:

`sendHeaders(int, String, int)`, `responseHeaders`

---

addHeader

```
public void addHeader(String line)
```

Adds a response header to the HTTP response. In order to do fancier things like appending a value to an existing response header, the user may directly access the `responseHeaders` field.

If this method is called, it must be called before `sendHeaders` is either directly or indirectly called. Otherwise, it will have no effect.

Parameters:

`line` - The HTTP response header, of the form "`key`: `value`".

See Also:

`sendHeaders(int, String, int)`, `responseHeaders`

---

sendResponse

```
public void sendResponse(byte[] body,
                         String type)
                  throws IOException
```

Sends an HTTP response to the client.

This method first calls `sendHeaders` to send the HTTP response headers, then sends the given byte array as the HTTP response body. If the request method is HEAD, the body is not sent.

The "Content-Length" will be set to the length of the given byte array. The "Content-Type" will be set to the given MIME type.

Parameters:

`body` - The array of bytes to send as the HTTP response body. May not be `null`.

`type` - The MIME type of the response, such as "text/html". May be `null` to use the existing "Content-Type" response header (if any).

Throws:

IOException - if there was an I/O error while sending the response to the client.

See Also:
`sendHeaders(int, String, int)`

---

sendResponse

```
public void sendResponse(String body,
                         String type,
                         int code)
                  throws IOException
```

Sends an HTTP response to the client.

This method first calls `sendHeaders` to send the HTTP response headers. It then writes out the given string to the client as a sequence of bytes. Each character in the string is written out by discarding its high eight bits.

The "Content-Length" will be set to the length of the string. The "Content-Type" will be set to the given MIME type.

Note: to use a different character encoding, use `sendResponse(body.getBytes(encoding)...)` instead.

Parameters:

`body` - The string to send as the HTTP response body. May not be `null`. If the request method is HEAD, the body is not sent.

`type` - The MIME type of the response, such as "text/html". May be `null` to preserve the existing "Content-Type" response header (if any).

`code` - The HTTP status code for the response, such as `200`. May be < 0 to preserve the existing status code.

Throws:

IOException - if there was an I/O error while sending the response to the client.

See Also:
`sendHeaders(int, String, int)`

---

sendResponse

```
public void sendResponse(String body)
                  throws IOException
```

Convenience method that sends an HTTP response to the client with a "Content-Type" of "text/html" and the default HTTP status code.

Parameters:

`body` - The string to send as the HTTP response body.

See Also:
`sendResponse(String, String, int)`

---

sendResponse

```
public void sendResponse(String body,
                         String type)
                  throws IOException
```

Convenience method that sends an HTTP response to the client with the default HTTP status code.

Parameters:

`body` - The string to send as the HTTP response body. If the request method is HEAD, only the headers are sent to the client.

`type` - The MIME type of the response.

See Also:

`sendResponse(String, String, int)`

---

sendResponse

```
public void sendResponse(InputStream in,
                         int length,
                         String type,
                         int code)
                  throws IOException
```

Sends the contents of the given input stream as the HTTP response.

This method first calls `sendHeaders` to send the HTTP response headers. It then transfers a total of `length` bytes of data from the given input stream to the client as the HTTP response body.

This method takes care of setting the "Content-Length" header if the actual content length is known, or the "Transfer-Encoding" header if the content length is not known (for HTTP/1.1 clients only).

This method may set the `keepAlive` to `false` before returning, if fewer than `length` bytes could be read. If the request method is HEAD, only the headers are sent.

Parameters:

`in` - The input stream to read from.

`length` - The content length. The number of bytes to send to the client. May be < 0, in which case this method will read until reaching the end of the input stream.

`type` - The MIME type of the response, such as "text/html". May be `null` to preserve the existing "Content-Type" response header (if any).

`code` - The HTTP status code for the response, such as `200`. May be < 0 to preserve the existing status code.

Throws:

IOException - if there was an I/O error while sending the response to the client.

---

sendError

```
public void sendError(int code,
                      String clientMessage)
```

Sends a HTTP error response to the client.

Parameters:

`code` - The HTTP status code.

`clientMessage` - A short message to be included in the error response and logged to the server.

---

sendError

```
public void sendError(int code,
                      String clientMessage,
                      String logMessage)
```

Sends a HTTP error response to the client.

Parameters:

`code` - The HTTP status code.

`clientMessage` - A short message to be included in the error response.

`logMessage` - A short message to be logged to the server. This message is **not** sent to the client.

---

sendHeaders

```
public void sendHeaders(int code,
                        String type,
                        int length)
                 throws IOException
```

Sends the HTTP status line and response headers to the client. This method is automatically invoked by `sendResponse`, but can be manually invoked if the user needs direct access to the client's output stream. If this method is not called, then the HTTP status and response headers will not automatically be sent to the client; the user would be responsible for forming the entire HTTP response.

The user may call the `addHeader` method or modify the `responseHeaders` field before calling this method. This method then adds a number of HTTP headers, as follows:

- "Date" - the current time, if this header is not already present.
- "Server" - the server's name (from `server.name`), if this header is not already present.
- "Connection" - "Keep-Alive" or "close", depending upon the `keepAlive` field.
- "Content-Length" - set to the given `length`.
- "Content-Type" - set to the given `type`.

The string used for "Connection" header actually comes from the `connectionHeader` field.

Parameters:

`code` - The HTTP status code for the response, such as `200`. May be < 0 to preserve the existing status code.

`type` - The MIME type of the response, such as "text/html". May be `null` to preserve the existing "Content-Type" response header (if any).

`length` - The length of the response body. May be < 0 if the length is unknown and/or to preserve the existing "Content-Length" response header (if any).

Throws:
IOException - if there was an I/O error while sending the headers to the client.
See Also:
`setStatus(int)`, `addHeader(String, String)`, `sendResponse(String, String, int)`, `connectionHeader`

---

redirect

```
public void redirect(String url,
                     String body)
              throws IOException
```

Responds to an HTTP request with a redirection reply, telling the client that the requested url has moved. Generally, this is used if the client did not put a '/' on the end of a directory.
Parameters:
`url` - The URL the client should have requested. This URL may be fully-qualified (in the form "http://....") or host-relative (in the form "/...").
`body` - The body of the redirect response, or `null` to send a hardcoded message.

---

serverUrl

```
public String serverUrl()
```

Returns the server's fully-qualified base URL. This is "http://" followed by the server's hostname and port.

If the HTTP request header "Host" is present, it specifies the hostname and port that will be used instead of the server's internal name for itself. Due bugs in certain browsers, when using the server's internal name, the port number will be elided if it is `80`.
Returns:
The string representation of the server's URL.

---

addSharedProps

```
public boolean addSharedProps(Dictionary d)
```

Adds the given `Dictionary` to the set of properties that are searched by `request.props.getProperty()`. This method is used to optimize the case when the caller has an existing `Dictionary` object that should be added to the search chain.

Assume the caller is constructing a new `Properties` object and wants to chain it onto the front of `request.props`. The following code is appropriate:

```
/* Push a temporary Dictionary onto request.props. */
PropertiesList old = request.props;
(new PropertiesList()).addBefore(request.props);
request.props = request.props.getPrior();
request.props.put("foo", "bar");
```

```
request.props.put("baz", "garply");

/* Do something that accesses new properties. */
    .
    .
    .

/* Restore old Dictionary when done. */
request.props.remove();
request.props = old;
```

However, `addSharedProps` may be called when the caller has an existing set of `Properties` and is faced with copying its contents into `request.props` and/or trying to share the existing `Properties` object among multiple threads concurrently.

```
/* Some properties created at startup. */
static Properties P = new Properties();
    .
    .
    .
/* Share properties at runtime. */
request.addSharedProps(P);
```

is more efficient and esthetically pleasing than:

```
foreach key in P.keys() {
    request.props.put(key, P.get(key));
}
```

The given `Dictionary` object is added to the `Properties.getProperty()` search chain before serverProps; it will be searched after the `request.props` and before `serverProps`. Multiple `Dictionary` objects can be added and they will be searched in the order given. The same `Dictionary` object can be added multiple times safely. However, the search chain for the given `Dictionary` must not refer back to `request.props` itself or a circular chain will be created causing an infinite loop:

```
request.addSharedProps(request.props);             // Bad
request.addSharedProps(request.props.getWrapped());  // Good
Properties d1 = new Properties(request.props);
request.addSharedProps(d1);                          // Bad
Hashtable d2 = new Hashtable();
Properties d3 = new Properties();
request.addSharedProps(d2);                        // Good
request.addSharedProps(d3);                        // Good
```

Subsequent calls to `request.props.getProperty()` may fetch properties from an added `Dictionary`, but `request.put()` will **not** modify those dictionaries.

Parameters:

`d` - A `Dictionary` of `String` key/value pairs that will be added to the chain searched when `request.props.getProperty()` is called. The dictionary `d` is "live", meaning that external changes to the contents of `d` will be seen on subsequent calls to `request.props.getProperty()`.

Returns:

`false` if the dictionary had already been added by a previous call to this method, `true` otherwise.

---

removeSharedProps

```
public boolean removeSharedProps(Dictionary d)
```

Removes a `Dictionary` added by `addSharedProps`. `Dictionary` objects may be removed in any order. `Dictionary` objects do not need to be removed; they will automatically get cleaned up at the end of the request.

Parameters:

`d` - The `Dictionary` object to remove from the `request.props.getProperty()` search chain.

Returns:

`true` if the `Dictionary` was found and removed, `false` if the `Dictionary` was not found (it had already been removed or had never been added).

Class Server

sunlabs.brazil.server
Class Server

```
java.lang.Object
  |
  +--java.lang.Thread
        |
        +--sunlabs.brazil.server.Server
```

All Implemented Interfaces:
Runnable
Direct Known Subclasses:
VelocityFilter.Vserver

---

public class **Server**
extends Thread

Yet another HTTP/1.1 server. This class is the core of a light weight Web Server. This server is started as a Thread listening on the supplied port, and dispatches to an implementation of a `Handler` to service http requests. If no handler is supplied, then the `FileHandler` is used. A `ChainHandler` is provided to allow multiple handlers in one server.

Limitations:
- Starts a new thread for each connection. This may be expensive.

Field Summary	
int	**acceptCount** Count of accepted connections so far.
int	**bufsize** Default buffer size for copies to and from client sockets.
int	**errorCount** Count of errors that occurred so far.
Handler	**handler**
String	**hostName** The hostname that this Server should use to identify itself in an HTTP Redirect.
boolean	**initFailure** If set, the server will terminate with an initialization failure just before creating the listen socket.

ServerSocket	**listen** The listening socket.
static int	**LOG_DIAGNOSTIC**
static int	**LOG_ERROR**
static int	**LOG_INFORMATIONAL**
static int	**LOG_LOG**
static int	**LOG_WARNING**
int	**logLevel** The diagnostic level.
int	**maxPost** Maximum amout of POST data allowed per request (in bytes) (default = 2Meg).
int	**maxRequests** Maximum number of consecutive requests allowed on a single kept-alive socket.
int	**maxThreads** The max number of threads allowed for the entire VM (default is 250).
String	**name** The string to return as the value for the "Server:" line in the HTTP response header.
String	**prefix** The handler is passed a prefix to identify which items in the properties object are relevent.
Properties	**props** Hashtable containing arbitrary information that may be of interest to a Handler.
String	**protocol** The protocol used to access this resource.
int	**requestCount** Count of HTTP requests received so far.
InetAddress[]	**restrict** If non-null, restrict connections to just the specified ip addresses.
int	**timeout** Time in milliseconds before this Server closes an idle socket or in-progress request.

Fields inherited from class java.lang.Thread
`MAX_PRIORITY, MIN_PRIORITY, NORM_PRIORITY`

Constructor Summary

`Server``()` Set up the server.	
`Server``(ServerSocket listen, String handlerName, Properties props)` Create a server using the provided listener socket.	

Method Summary

`void`	**`close`**`()` Stop the server, and kill all pending requests
`boolean`	**`init`**`()`
`void`	**`log`**`(int level, Object obj, String message)` Logs information about the socket to `System.out`.
`boolean`	**`restart`**`(String newHandler)` Restart the server with a new handler.
`void`	**`run`**`()` Loops, accepting socket connections and replying to HTTP requests.
`boolean`	**`setup`**`(ServerSocket listen, String handlerName, Properties props)`

Methods inherited from class java.lang.Thread
`activeCount, checkAccess, countStackFrames, currentThread, destroy, dumpStack, enumerate, getContextClassLoader, getName, getPriority, getThreadGroup, interrupt, interrupted, isAlive, isDaemon, isInterrupted, join, join, join, resume, setContextClassLoader, setDaemon, setName, setPriority, sleep, sleep, start, stop, stop, suspend, toString, yield`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Field Detail

listen

`public ServerSocket listen`

The listening socket. Every time a new socket is accepted, a new thread is created to read the HTTP requests from it.

---

handler

`public Handler handler`

---

props

`public Properties props`

Hashtable containing arbitrary information that may be of interest to a Handler. This table is available to both methods of the `Handler` interface, as `props` in the `Handler.init(Server, String)` method, and as the default properties of `Request.props` in the `Handler.respond(Request)` method.

---

hostName

`public String hostName`

The hostname that this Server should use to identify itself in an HTTP Redirect. If `null`, the hostname is derived by calling `InetAddress.getHostAddress`.

`InetAddress.getHostName` would generally be the wrong thing to return because it returns only the base machine name `xxx` and not the machine name as it needs to appear to the rest of the network, such as `xxx.yyy.com`.

The default value is `null`.

---

protocol

`public String` **`protocol`**

The protocol used to access this resource. Normally `http`, but can be changed for `ssl` to `https`

---

restrict

`public InetAddress[]` **`restrict`**

If non-null, restrict connections to just the specified ip addresses.

The default value is `null`.

---

name

`public String` **`name`**

The string to return as the value for the "Server:" line in the HTTP response header. If `null`, then no "Server:" line is returned.

---

prefix

`public String` **`prefix`**

The handler is passed a prefix to identify which items in the properties object are relevent. By convention, non-empty strings end with ".", allowing nested prefixes to be easily distinguished.

---

timeout

`public int` **`timeout`**

Time in milliseconds before this Server closes an idle socket or in-progress request.

The default value is `30000`.

---

maxRequests

`public int` **`maxRequests`**

Maximum number of consecutive requests allowed on a single kept-alive socket.

The default value is `25`.

---

maxThreads

`public int` **`maxThreads`**

The max number of threads allowed for the entire VM (default is 250).

---

maxPost

`public int` **`maxPost`**

Maximum amout of POST data allowed per request (in bytes) (default = 2Meg).

---

bufsize

`public int` **`bufsize`**

Default buffer size for copies to and from client sockets. (default is 8192)

---

acceptCount

`public int` **`acceptCount`**

Count of accepted connections so far.

---

requestCount

`public int` **`requestCount`**

Count of HTTP requests received so far.

---

errorCount

`public int` **`errorCount`**

Count of errors that occurred so far.

---

logLevel

`public int` **`logLevel`**

The diagnostic level. 0->least, 5->most

---

initFailure

`public boolean` **`initFailure`**

If set, the server will terminate with an initialization failure just before creating the listen socket.

LOG_ERROR

```
public static final int LOG_ERROR
```

LOG_WARNING

```
public static final int LOG_WARNING
```

LOG_LOG

```
public static final int LOG_LOG
```

LOG_INFORMATIONAL

```
public static final int LOG_INFORMATIONAL
```

LOG_DIAGNOSTIC

```
public static final int LOG_DIAGNOSTIC
```

Constructor Detail

Server

```
public Server(ServerSocket listen,
              String handlerName,
              Properties props)
```

Create a server using the provided listener socket.

This server will call the `Handler.respond` method of the specified handler. The specified handler should either respond to the request or perform further dispatches to other handlers.

Parameters:

`listen` - The socket this server should listen to. For ordinary sockets, this is simply: `new ServerSocket(port)`, where `port` is the network port to listen on. Alternate implementations of `ServerSocket`, such as **ssl** versions may be used instead.

`handlerName` - The name of the handler used to process http requests. It must implement the `Handler` interface.

`props` - Arbitrary information made available to the handler. May be `null`.

See Also:

`FileHandler`, `ChainHandler`

Server

```
public Server()
```

Set up the server. this allows a server to be created with newInstance() followed by setup(), instead of using the above initializer, making it easier to start sub-classes of the server.

Method Detail

setup

```
public boolean setup(ServerSocket listen,
                     String handlerName,
                     Properties props)
```

---

init

```
public boolean init()
```

---

restart

```
public boolean restart(String newHandler)
```

Restart the server with a new handler.
Parameters:
`newHandler` - Name of the handler to restart the server with

---

run

```
public void run()
```

Loops, accepting socket connections and replying to HTTP requests. This is called indirectly via Thread.start().

Many things in the server are not initialized until this point, because the user may have set some related configuration options between the time this server was allocated and the time it was started. For instance, the main `Handler` is not initialized until now, because its `Handler.init` method may have wanted to examine server member variables such as `hostName` or `bufsize`.
Overrides:
run in class Thread

---

close

```
public void close()
```

Stop the server, and kill all pending requests

---

log

```
public void log(int level,
                Object obj,
                String message)
```

Logs information about the socket to `System.out`.

Parameters:

`level` - Controls the verbosity (0=least 5=most)

`obj` - The object that the message relates to.

`message` - The message to be logged.

Class TestRequest

sunlabs.brazil.server
Class TestRequest

```
java.lang.Object
  |
  +--sunlabs.brazil.server.Request
        |
        +--sunlabs.brazil.server.TestRequest
```

---

public class **TestRequest**
extends Request

Version of Request for testing purposes. Arranges for all output to be captured in one place so the ouput of a test may be compared with the expected output.

Inner classes inherited from class sunlabs.brazil.server.Request
Request.HttpOutputStream

Fields inherited from class sunlabs.brazil.server.Request
connectionHeader, headers, keepAlive, MAX_BLANKS, method, out, postData, props, protocol, query, responseHeaders, server, serverProps, serverProtocol, sock, startMillis, url, version

Constructor Summary

TestRequest(Server server, String request)	

Method Summary

String	**log**()
void	**log**(int level, Object obj, String message) Logs a message by calling Server.log.
Object	**put**(String key, String value)
String	**result**()

Methods inherited from class sunlabs.brazil.server.Request
addHeader, addHeader, addSharedProps, getQueryData, getQueryData, getRequest, getRequestHeader, getReuseCount, getSocket, getStatus, log, redirect, removeSharedProps, sendError, sendError, sendHeaders, sendResponse, sendResponse, sendResponse, sendResponse, sendResponse, serverUrl, setStatus, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

TestRequest

```
public TestRequest(Server server,
                   String request)
            throws IOException
```

Method Detail

put

```
public Object put(String key,
                  String value)
```

---

log

```
public String log()
```

---

result

```
public String result()
```

---

log

```
public void log(int level,
                Object obj,
                String message)
```

Description copied from class: Request
Logs a message by calling `Server.log`. Typically a message is generated on the console or in a log file, if the `level` is less than the current server log setting.
Overrides:
log in class Request
Tags copied from class: Request
Parameters:
`level` - The severity of the message.
`obj` - The object that the message relates to.
`message` - The message that will be logged.
See Also:
`Server.log(int, Object, String)`

Package sunlabs.brazil.servlet

Run a Brazil application as a Servlet. This is the Brazil Toolkit Servlet Adapter. It allows one to run applications built with the Brazil Toolkit in any web server that provides a `Servlet` container that implements at least the Servlet 2.2 API

When running a Brazil application as a servlet, `sunlabs.brazil.server.Main` is not used to start the application; the servlet container starts the application instead. The deployment descriptor parameters `config` and `root` are used to locate the Brazil server configuration files, and to define the document root, respectively.

To allow identical Brazil configurations to operate unchanged, either standalone or as a servlet, the application context portion of the URL is removed before Brazil handlers access requests. Similarly, any HTML content produced by the Brazil application is re-written to prepend the application context onto any rooted (e.g. starts with '/') URL.

For example, if a brazil application is deployed in the servlet context: `brazil-servlet`, then an http request to the server of the form:

```
http://www.myhost.com/brazil-servlet/foo.html
```

will appear to the Brazil application with a URL of:

```
/foo.html
```

. An HTML response from the Brazil application of the form:

```
<a href="/foo.html"> ...
```

will automatically be rewritten to:

```
<a href="/brazil-servlet/foo.html"> ...
```

The servlet should be installed as a web application as specified in the Java Servlet Specification, v2.2. The servlet deployment descriptor should create a configuration that causes the request path to consist of only the context path and pathInfo elements. There should not be a servlet path element. See section 5 of the specification for more details (note that the definitions of each path element are are more accurately stated in the 2.3 specification). The deployment descriptor should be similar to the following:

```
<web-app>
    <display-name>BrazilServlet servlet container for Brazil</display-name>
    <description>
        This servlet wraps the Brazil Toolkit so it can run within
        any web server environment that supports the 2.2 Servlet API
    </description>

    <servlet>
        <servlet-name>BrazilServlet</servlet-name>
        <servlet-class>sunlabs.brazil.server.BrazilServlet</servlet-class>
    </servlet>

    <servlet-mapping>
```

```
        <servlet-name>BrazilServlet</servlet-name>
        <url-pattern>/*</url-pattern>
    </servlet-mapping>
</web-app>
```

See `sunlabs.brazil.servlet.BrazilServlet` for more instructions.

Class BServletRequest.HttpOutputStream

sunlabs.brazil.servlet
Class BServletRequest.HttpOutputStream

```
java.lang.Object
  |
  +--java.io.OutputStream
        |
        +--java.io.FilterOutputStream
              |
              +--sunlabs.brazil.server.Request.HttpOutputStream
                    |
                    +--sunlabs.brazil.servlet.BServletRequest.HttpOutputStream
```

Enclosing class:
BServletRequest

---

public static class **BServletRequest.HttpOutputStream**
extends Request.HttpOutputStream

An `HttpOutputStream` that is used by the Brazil Servlet adaptor to add the web application context path to the front of all response URI's so subsequent requests will be mapped to the `BrazilServlet`'s context. See comments in `BrazilServlet` that describe the assumptions regarding the context path.

See Also:
`BrazilServlet`

Fields inherited from class sunlabs.brazil.server.Request.HttpOutputStream
`bytesWritten`

Constructor Summary

`BServletRequest.HttpOutputStream``(OutputStream out)`	

Method Summary

void	**close**() Remaps URI's in the buffered HTML content and the writes the result to the response stream.
void	**sendHeaders**(Request request) If the response content type is text/html, this method interposes a buffer before the response stream that gathers all content for subsequent URI rewriting.

Methods inherited from class sunlabs.brazil.server.Request.HttpOutputStream
write, write, writeBytes

Methods inherited from class java.io.FilterOutputStream
flush, write, write

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

BServletRequest.HttpOutputStream

```
public BServletRequest.HttpOutputStream(OutputStream out)
```

Method Detail

sendHeaders

```
public void sendHeaders(Request request)
                 throws IOException
```

If the response content type is text/html, this method interposes a buffer before the response stream that gathers all content for subsequent URI rewriting.

Overrides:
sendHeaders in class Request.HttpOutputStream

---

close

```
public void close()
           throws IOException
```

Remaps URI's in the buffered HTML content and the writes the result to the response stream.
Overrides:
close in class FilterOutputStream

Class BServletRequest

sunlabs.brazil.servlet
Class BServletRequest

```
java.lang.Object
  |
  +--sunlabs.brazil.server.Request
        |
        +--sunlabs.brazil.servlet.BServletRequest
```

---

public class **BServletRequest**
extends Request

This is a subclass of `Request` that is used by `BrazilServlet` to map servlet style requests and responses to and from Brazil style requests.

See Also:
`BServletSocket`, `BServletServerSocket`, `BrazilServlet`, `Request`

Inner Class Summary	
`static class`	**`BServletRequest.HttpOutputStream`** An `HttpOutputStream` that is used by the Brazil Servlet adaptor to add the web application context path to the front of all response URI's so subsequent requests will be mapped to the `BrazilServlet`'s context.

Inner classes inherited from class sunlabs.brazil.server.Request
`Request.HttpOutputStream`

Fields inherited from class sunlabs.brazil.server.Request
`connectionHeader, headers, keepAlive, MAX_BLANKS, method, out, postData, props, protocol, query, responseHeaders, server, serverProps, serverProtocol, sock, startMillis, url, version`

Method Summary

boolean	**getRequest**() Converts an HTTP request from the form encapsulated in a servlet request object to the form expected by Brazil handlers.
void	**redirect**(String url, String body) Responds to an HTTP request with a redirection reply, telling the client that the requested url has moved.
String	**serverUrl**() Returns the server's fully-qualified base URL.
void	**setStatus**(int code) Sets response code in the servlet response object.

Methods inherited from class sunlabs.brazil.server.Request
addHeader, addHeader, addSharedProps, getQueryData, getQueryData, getRequestHeader, getReuseCount, getSocket, getStatus, log, log, removeSharedProps, sendError, sendError, sendHeaders, sendResponse, sendResponse, sendResponse, sendResponse, sendResponse, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Method Detail

getRequest

```
public boolean getRequest()
                  throws IOException
```

Converts an HTTP request from the form encapsulated in a servlet request object to the form expected by Brazil handlers.

Overrides:

getRequest in class Request

Returns:

`true` if the request was successfully read and parsed, `false` if the request was malformed.

Throws:
IOException - if there was an IOException reading from the socket. See the socket documentation for a description of socket exceptions.

---

setStatus

```
public void setStatus(int code)
```

Sets response code in the servlet response object.
Overrides:
setStatus in class Request
Tags copied from class: Request
Parameters:
`code` - The HTTP status code, such as `200` or `404`. If < 0, the HTTP status code will not be changed.
See Also:
`Request.sendHeaders(int, String, int)`

---

serverUrl

```
public String serverUrl()
```

Returns the server's fully-qualified base URL. This is "http://" followed by the server's hostname and port.

If the HTTP request header "Host" is present, it specifies the hostname and port that will be used instead of the server's internal name for itself. Due to bugs in certain browsers, when using the server's internal name, the port number will be elided if it is 80.
Overrides:
serverUrl in class Request
Returns:
The string representation of the server's URL.

---

redirect

```
public void redirect(String url,
                     String body)
              throws IOException
```

Description copied from class: Request
Responds to an HTTP request with a redirection reply, telling the client that the requested url has moved. Generally, this is used if the client did not put a '/' on the end of a directory.
Overrides:
redirect in class Request
Tags copied from class: Request

Parameters:

`url` - The URL the client should have requested. This URL may be fully-qualified (in the form "http://....") or host-relative (in the form "/...").

`body` - The body of the redirect response, or `null` to send a hardcoded message.

Class BServletServerSocket

sunlabs.brazil.servlet
Class BServletServerSocket

```
java.lang.Object
  |
  +--java.net.ServerSocket
        |
        +--sunlabs.brazil.servlet.BServletServerSocket
```

---

public class **BServletServerSocket**
extends ServerSocket

This is a special `ServerSocket` intended for use by the Brazil Servlet Adapter. It provides dummy methods for the `ServerSocket` operations used by the Brazil Toolkit. Do not use this class for anything other than it's intended use as a `ServerSocket` object that is passed to the constructor of a `Server` object by the `BrazilServlet`'s `init` method.

See Also:
`ServerSocket`, `BServletServerSocket`, `Server`, `BrazilServlet`, `BServletRequest`

Constructor Summary

`BServletServerSocket``()` Creates a `BServletServerSocket` object using -1 as a port number.
`BServletServerSocket``(int port)` Creates a `BServletServerSocket` object.

Method Summary	
Socket	**accept**() Dummy method, not implemented.
void	**close**() Closes this socket.
InetAddress	**getInetAddress**() Returns the local address to which this socket is bound.
int	**getLocalPort**() Returns the local port to which this socket is bound.
int	**getSoTimeout**() Dummy method, not implemented.
void	**setLocalPort**(int port) Sets the local port to which this socket is bound.
static void	**setSocketFactory**(SocketImplFactory fac) Dummy method, not implemented.
void	**setSoTimeout**(int timeout) Dummy method, not implemented.
String	**toString**() Converts this socket to a `String`.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

BServletServerSocket

```
public BServletServerSocket()
                    throws IOException
```

Creates a `BServletServerSocket` object using -1 as a port number.

BServletServerSocket

```
public BServletServerSocket(int port)
                     throws IOException
```

Creates a `BServletServerSocket` object.

Parameters:

`port` - the port number this object encapsulates.

See Also:

`setLocalPort(int port)`

Method Detail

getInetAddress

```
public InetAddress getInetAddress()
```

Returns the local address to which this socket is bound.

Overrides:

getInetAddress in class ServerSocket

Returns:

the local address number to which this socket is bound.

---

setLocalPort

```
public void setLocalPort(int port)
```

Sets the local port to which this socket is bound. Not a method in `ServerSocket` because the port is fixed at construction time. It's needed here because a Servlet doesn't know it's local port until it services it's first request.

Returns:

the local port number to which this socket is bound.

---

getLocalPort

```
public int getLocalPort()
```

Returns the local port to which this socket is bound.

Overrides:

getLocalPort in class ServerSocket

Returns:

the local port number to which this socket is bound.

---

accept

```
public Socket accept()
              throws IOException
```

Dummy method, not implemented.

Overrides:

accept in class ServerSocket

Returns:

null

Throws:

IOException - if method called.

---

close

```
public void close()
           throws IOException
```

Closes this socket.

Overrides:

close in class ServerSocket

Throws:

IOException - if an I/O error occurs when closing this socket.

---

setSoTimeout

```
public void setSoTimeout(int timeout)
                  throws SocketException
```

Dummy method, not implemented.

Overrides:

setSoTimeout in class ServerSocket

Parameters:

`timeout` - doesn't do anything.

Throws:

SocketException - if method called.

---

getSoTimeout

```
public int getSoTimeout()
                 throws IOException
```

Dummy method, not implemented.

Overrides:

getSoTimeout in class ServerSocket

Returns:

-1

Throws:
IOException - if method called.

---

toString

```
public String toString()
```

Converts this socket to a `String`.
Overrides:
toString in class ServerSocket
Returns:
a string representation of this socket.

---

setSocketFactory

```
public static void setSocketFactory(SocketImplFactory fac)
                              throws IOException
```

Dummy method, not implemented.
Parameters:
`fac` - doesn't do anything.
Throws:
IOException - if method called.

Class BServletSocket

sunlabs.brazil.servlet
Class BServletSocket

```
java.lang.Object
  |
  +--java.net.Socket
        |
        +--sunlabs.brazil.servlet.BServletSocket
```

---

public class **BServletSocket**
extends Socket

This is a special `Socket` intended for use by the Brazil Servlet Adapter. It provides dummy methods for the `Socket` operations used by the Brazil Toolkit. Do not use this class for anything other than it's intended use as a `Socket` object that is passed to the constructor of a `Request` object by the `BServletRequest`'s constructor.

See Also:
`Socket`, `BServletServerSocket`, `BrazilServlet`, `BServletRequest`, `Request`

Constructor Summary

`BServletSocket``(HttpServletRequest req, HttpServletResponse res)` Creates a `BServletSocket` object.

Method Summary

`void`	**`close`**`()` Closes this socket.
`InetAddress`	**`getInetAddress`**`()` Returns the address to which the socket is connected.
`InputStream`	**`getInputStream`**`()` Returns an input stream for this socket.
`boolean`	**`getKeepAlive`**`()` Dummy method, not implemented.
`InetAddress`	**`getLocalAddress`**`()` Returns the local address to which the socket is bound.
`int`	**`getLocalPort`**`()` Returns the local port to which this socket is bound.

OutputStream	**getOutputStream**() Returns an output stream for this socket.
int	**getPort**() Returns the remote port to which this socket is connected.
int	**getReceiveBufferSize**() Dummy method, not implemented.
int	**getSendBufferSize**() Dummy method, not implemented.
int	**getSoLinger**() Dummy method, not implemented.
int	**getSoTimeout**() Dummy method, not implemented.
boolean	**getTcpNoDelay**() Dummy method, not implemented.
void	**setKeepAlive**(boolean on) Dummy method, not implemented.
void	**setReceiveBufferSize**(int size) Dummy method, not implemented.
void	**setSendBufferSize**(int size) Dummy method, not implemented.
void	**setSoLinger**(boolean on, int linger) Dummy method, not implemented.
void	**setSoTimeout**(int timeout) Dummy method, not implemented.
void	**setTcpNoDelay**(boolean on) Dummy method, not implemented.
void	**shutdownInput**() Dummy method, not implemented.
void	**shutdownOutput**() Dummy method, not implemented.
String	**toString**() Converts this socket to a `String`.

Methods inherited from class java.net.Socket
`setSocketImplFactory`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Constructor Detail

BServletSocket

```
public BServletSocket(HttpServletRequest req,
                      HttpServletResponse res)
               throws UnknownHostException,
                      IOException
```

Creates a `BServletSocket` object.

Parameters:

`req` - the `HttpServletRequest` passed to `BrazilServlet` by the servlet container.

`res` - the `HttpServletResponse` passed to `BrazilServlet` by the servlet container.

See Also:

`BrazilServlet`

Method Detail

getInetAddress

```
public InetAddress getInetAddress()
```

Returns the address to which the socket is connected.

Overrides:

getInetAddress in class Socket

Returns:

the remote IP address to which this socket is connected.

---

getLocalAddress

```
public InetAddress getLocalAddress()
```

Returns the local address to which the socket is bound.

Overrides:

getLocalAddress in class Socket

Returns:

the local address to which the socket is bound.

---

getPort

```
public int getPort()
```

Returns the remote port to which this socket is connected.

Overrides:

getPort in class Socket

Returns:

the remote port number to which this socket is connected.

---

getLocalPort

```
public int getLocalPort()
```

Returns the local port to which this socket is bound.

Overrides:

getLocalPort in class Socket

Returns:

the local port number to which this socket is bound.

---

getInputStream

```
public InputStream getInputStream()
                          throws IOException
```

Returns an input stream for this socket.

Overrides:

getInputStream in class Socket

Returns:

an input stream for reading bytes from this socket.

Throws:

IOException - if an I/O error occurs when creating the input stream.

---

getOutputStream

```
public OutputStream getOutputStream()
                            throws IOException
```

Returns an output stream for this socket.

Overrides:

getOutputStream in class Socket

Returns:

an output stream for writing bytes to this socket.

Throws:

IOException - if an I/O error occurs when creating the output stream.

---

setTcpNoDelay

```
public void setTcpNoDelay(boolean on)
                  throws SocketException
```

Dummy method, not implemented.

Overrides:

setTcpNoDelay in class Socket

Parameters:

`on` - doesn't do anything.

Throws:

SocketException - if method called.

---

getTcpNoDelay

```
public boolean getTcpNoDelay()
                      throws SocketException
```

Dummy method, not implemented.

Overrides:

getTcpNoDelay in class Socket

Returns:

`false`

Throws:

SocketException - if method called.

---

setSoLinger

```
public void setSoLinger(boolean on,
                        int linger)
                 throws SocketException
```

Dummy method, not implemented.

Overrides:

setSoLinger in class Socket

Parameters:

`on` - doesn't do anything.

`linger` - doesn't do anything.

Throws:

SocketException - if method called.

---

getSoLinger

```
public int getSoLinger()
                throws SocketException
```

Dummy method, not implemented.

Overrides:
getSoLinger in class Socket
Returns:
-1
Throws:
SocketException - if method called.

---

setSoTimeout

```
public void setSoTimeout(int timeout)
                  throws SocketException
```

Dummy method, not implemented.
Overrides:
setSoTimeout in class Socket
Parameters:
`timeout` - doesn't do anything.
Throws:
SocketException - if method called.

---

getSoTimeout

```
public int getSoTimeout()
                 throws SocketException
```

Dummy method, not implemented.
Overrides:
getSoTimeout in class Socket
Returns:
-1
Throws:
SocketException - if method called.

---

setSendBufferSize

```
public void setSendBufferSize(int size)
                       throws SocketException
```

Dummy method, not implemented.
Overrides:
setSendBufferSize in class Socket
Parameters:
`size` - doesn't do anything.
Throws:
SocketException - if method called.

---

getSendBufferSize

```
public int getSendBufferSize()
                      throws SocketException
```

Dummy method, not implemented.

Overrides:

getSendBufferSize in class Socket

Returns:

-1

Throws:

SocketException - if method called.

---

setReceiveBufferSize

```
public void setReceiveBufferSize(int size)
                          throws SocketException
```

Dummy method, not implemented.

Overrides:

setReceiveBufferSize in class Socket

Parameters:

`size` - doesn't do anything.

Throws:

SocketException - if method called.

---

getReceiveBufferSize

```
public int getReceiveBufferSize()
                         throws SocketException
```

Dummy method, not implemented.

Overrides:

getReceiveBufferSize in class Socket

Returns:

-1

Throws:

SocketException - if method called.

---

setKeepAlive

```
public void setKeepAlive(boolean on)
                  throws SocketException
```

Dummy method, not implemented.

Overrides:
setKeepAlive in class Socket
Parameters:
`on` - doesn't do anything.
Throws:
SocketException - if method called.

---

getKeepAlive

```
public boolean getKeepAlive()
                     throws SocketException
```

Dummy method, not implemented.
Overrides:
getKeepAlive in class Socket
Returns:
`false`
Throws:
SocketException - if method called.

---

close

```
public void close()
           throws IOException
```

Closes this socket.
Overrides:
close in class Socket
Throws:
IOException - if an I/O error occurs when closing this socket.

---

shutdownInput

```
public void shutdownInput()
                   throws IOException
```

Dummy method, not implemented.
Overrides:
shutdownInput in class Socket
Throws:
IOException - if method called.

---

shutdownOutput

```
public void shutdownOutput()
                    throws IOException
```

Dummy method, not implemented.

Overrides:

shutdownOutput in class Socket

Throws:

IOException - if method called.

---

toString

```
public String toString()
```

Converts this socket to a `String`.

Overrides:

toString in class Socket

Returns:

a string representation of this socket.

Class BrazilServlet

sunlabs.brazil.servlet
Class BrazilServlet

```
java.lang.Object
  |
  +--javax.servlet.GenericServlet
        |
        +--sunlabs.brazil.servlet.BrazilServlet
```

All Implemented Interfaces:
Serializable, Servlet, ServletConfig

---

public final class **BrazilServlet**
extends GenericServlet

This is the Brazil Toolkit Servlet Adapter. It allows one to run applications built with the Brazil Toolkit in any web server that provides a `Servlet` container that implements the Servlet 2.2 API.

The servlet should be installed as a web application as specified in the Java Servlet Specification, v2.2. The servlet deployment descriptor should create a configuration that causes the request path to consist of only the context path and pathInfo elements. There should not be a servlet path element. See section 5 of the specification for more details (note that the definitions of each path element are are more accurately stated in the 2.3 specification). The deployment descriptor should be similar to the following:

```
<web-app>
    <display-name>BrazilServlet servlet container for Brazil</display-name>
    <description>
       This servlet wraps the Brazil Toolkit so it can run within
       any web server environment that supports the 2.2 Servlet API
    </description>

    <servlet>
        <servlet-name>BrazilServlet</servlet-name>
        <servlet-class>sunlabs.brazil.servlet.BrazilServlet</servlet-class>
        <init-param>
            <param-name>config</param-name>
            <param-value>configA|configB</param-value>
        </init-param>
    </servlet>

    <servlet-mapping>
        <servlet-name>BrazilServlet</servlet-name>
        <url-pattern>/*</url-pattern>
    </servlet-mapping>
</web-app>
```

The default configuration file for the Brazil application is loaded from the system resource `/sunlabs/brazil/server/config`. Additional configuration files, which may override properties in the default file, are searched by looking at the "|" separated list of file names associated with `config` servlet parameter. File names that are not "absolute" are located relative to the servlet context root If no

`config` servlet parameter is defined, the file named "config" (if any) in the servlet context root is used.

The Brazil `root` property if obtained from the "root" servlet parameter. If none is provided, the "root" property from the configuration files (last one wins) is used. As before, relative root directories are resolved with respect to the servlet context root.

The following properties are set by this class:

`url.servlet`
: The original URL requested by the client.

`context_path`
: The servlet context path

`servlet_name`
: The servlet's name

`servlet_container`
: The name of the servlet container

See Also:
`BServletSocket`, `BServletServerSocket`, `BServletRequest`, Serialized Form

Constructor Summary

`BrazilServlet()`	

Method Summary

`void`	**`destroy()`** Called by the servlet container to indicate to a servlet that the servlet is being taken out of service.
`void`	**`init(ServletConfig config)`** Called by the servlet container to place the servlet into service.
`void`	**`service(ServletRequest req, ServletResponse res)`** Called by the servlet container to allow the servlet to respond to a request.

Methods inherited from class javax.servlet.GenericServlet
`getInitParameter, getInitParameterNames, getServletConfig, getServletContext, getServletInfo, getServletName, init, log, log`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

BrazilServlet

```
public BrazilServlet()
```

Method Detail

init

```
public void init(ServletConfig config)
          throws ServletException
```

Called by the servlet container to place the servlet into service.

Overrides:

init in class GenericServlet

Parameters:

`config` - the `ServletConfig` object that contains configuration information for this servlet

Throws:

ServletException - if an exception occurs that interrupts the servlet's normal operation

See Also:

`Servlet.init(javax.servlet.ServletConfig)`

---

service

```
public void service(ServletRequest req,
                    ServletResponse res)
             throws ServletException,
                    IOException
```

Called by the servlet container to allow the servlet to respond to a request.

This method implements the abstract method declared in the `GenericServlet` super class.

Overrides:

service in class GenericServlet

Parameters:

`req` - the `ServletRequest` object that contains the client's request

`res` - the `ServletResponse` object that will contain the servlet's response

Throws:

ServletException - if an exception occurs that interferes with the servlet's normal operation occurred

IOException - if an input or output exception occurs

See Also:

`Servlet.service(javax.servlet.ServletRequest, javax.servlet.ServletResponse)`

---

destroy

```
public void destroy()
```

Called by the servlet container to indicate to a servlet that the servlet is being taken out of service.

Overrides:

destroy in class GenericServlet

See Also:

`Servlet.destroy()`

Package sunlabs.brazil.session

A generic, extensible mechanism for managing session state.

The `sunlabs.brazil.session.SessionManager` associates an object with a Session ID to give Handlers the ability to maintain state that lasts for the duration of a session instead of just for the duration of a request.

An example of session-lifetime information is the set of stocks that a particular user is interested in. When the hypothetical `StockHandler` gets a request, it would get determine which stocks to display based on the Session ID of the request.

Operating with the `sunlabs.brazil.session.SessionManager` is the mechanism to discover the Session ID of a request and/or create new Session IDs. For example, the first time a user goes to the server providing the stock market information, they could get a cookie. Subsequently, every time they visit the stock market page the cookie present in the request would be decoded and matched with the existing Session ID for use by the `StockHandler`.

The `sunlabs.brazil.session.SessionManager` operates as a bag of globally accessible resources. Existing subclasses of the `sunlabs.brazil.session.SessionManager` also provide persistence, that is, a way to recover these resources even if the server process is terminated and later restarted, to get back to the state things were in.

Class CacheManager

sunlabs.brazil.session
Class CacheManager

```
java.lang.Object
  |
  +--sunlabs.brazil.session.SessionManager
        |
        +--sunlabs.brazil.session.CacheManager
```

All Implemented Interfaces:
Handler
Direct Known Subclasses:
PropertiesCacheManager

---

public class **CacheManager**
extends SessionManager
implements Handler

This `SessionManager` associates an object with a Session ID to give Handlers the ability to maintain state that lasts for the duration of a session instead of just for the duration of a request. It should be installed as a handler, whoses init method will replace the default session manager.

This version maintains a pool of hashtables. Once they all fill up - one of them gets tossed, causing any session info in it to be lost. It uses a simplified approximate LRU scheme. The default session manager doesn't loose any session information, but grows the heap without bound as the number of sessions increase.

properties:
tables
The number of Hashtables in the pool (defaults to 6)
size
The max number of entries in each table (defaults to 1000).

Constructor Summary

`CacheManager()`	

Method Summary	
boolean	**init**(Server server, String prefix) Install this class as the session manager.
void	**removeObj**(Object session, Object ident) Remove an object from a session table.
boolean	**respond**(Request request) Don't handle any URL requests (yet)

Methods inherited from class sunlabs.brazil.session.SessionManager
get, getSession, put, remove, setSessionManager

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

CacheManager

```
public CacheManager()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Install this class as the session manager. Get the number of tables, and the max size per table.

Specified by:

init in interface Handler

Tags copied from interface: Handler

Parameters:

`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.

`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it

uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
```

Don't handle any URL requests (yet)

Specified by:

respond in interface Handler

Tags copied from interface: Handler

Parameters:

`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

removeObj

```
public void removeObj(Object session,
                      Object ident)
```

Remove an object from a session table. Don't bother to remove the table if its empty

Class PJamaSessionManager

sunlabs.brazil.session
Class PJamaSessionManager

```
java.lang.Object
  |
  +--sunlabs.brazil.session.SessionManager
        |
        +--sunlabs.brazil.session.PJamaSessionManager
```

---

public class **PJamaSessionManager**
extends SessionManager

Use pJama to implement persistant sessions.

Constructor Summary

PJamaSessionManager()

Methods inherited from class sunlabs.brazil.session.SessionManager
get, getSession, put, remove, setSessionManager

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

PJamaSessionManager

public **PJamaSessionManager**()

Interface PropertiesCacheManager.Saveable

sunlabs.brazil.session
Interface PropertiesCacheManager.Saveable
All Known Implementing Classes:
ListTemplate.MyList, XmlTree
Enclosing class:
PropertiesCacheManager

---

public static interface **PropertiesCacheManager.Saveable**

This interface allows for persistence of non-properties session objects. These methods should behave precisely like the corrosponding methods of the Properties class.

Method Summary

boolean	**isEmpty**() The current object state is the "default"; "save" does not need to write out any state.
void	**load**(InputStream in) Recreate the object from the ascii representation stored as a Properties format file.
void	**save**(OutputStream out, String header) Create an ascii representation of this object in a Java Properties format.

Method Detail

load

```
public void load(InputStream in)
          throws IOException
```

Recreate the object from the ascii representation stored as a Properties format file.

---

save

```
public void save(OutputStream out,
                 String header)
          throws IOException
```

Create an ascii representation of this object in a Java Properties format.

---

isEmpty

```
public boolean isEmpty()
```

The current object state is the "default"; "save" does not need to write out any state.

Class PropertiesCacheManager

sunlabs.brazil.session
Class PropertiesCacheManager

```
java.lang.Object
  |
  +--sunlabs.brazil.session.SessionManager
        |
        +--sunlabs.brazil.session.CacheManager
              |
              +--sunlabs.brazil.session.PropertiesCacheManager
```

All Implemented Interfaces:
Handler, Serializable

---

public class **PropertiesCacheManager**
extends CacheManager
implements Handler, Serializable

A version of the CacheManager that saves out any session state that is either a "java properties" object, or implements "Saveable" into a directory in the filesystem, one file per entry, then restores them on server startup. This is a "poor man's" serialization, that saves only ascii state represented in properties files. This permits a wider variety of changes to be made to the server code, yet still have the ability to read in the proper session information.

Classes that are not properties files may implement "Saveable", which has the store() and load() methods from Properties; they are expected to generate and restore the state of the object in Properties format.

This handler/sessionManager can take an ascii-readable "snapshot" of the server state, for all state that is a java properties object (or implements Saveable). It doesn't perturb the existing state.

Properties:
storeDir
: The directory to use to store the state files. It is created as needed when the state is saved. Defalts to "store".

match
: A glob pattern that matches the url (or url?query if a query is used. Defaults to "*\?*save=true".

filePrefix
: A prefix pattern to use for all session files. Defaults to the handler prefix.

defer
: If set, the saved session information is not reconstructed upon startup. Only the list of sessions is read in; the session information is restored only when needed.

See Also:
Serialized Form

Inner Class Summary

static interface	**PropertiesCacheManager.Saveable** This interface allows for persistence of non-properties session objects.

Constructor Summary

PropertiesCacheManager()	

Method Summary

boolean	**init**(Server server, String prefix) Initializes the handler.
boolean	**respond**(Request request) Responds to an HTTP request.

Methods inherited from class sunlabs.brazil.session.CacheManager
removeObj

Methods inherited from class sunlabs.brazil.session.SessionManager
get, getSession, put, remove, setSessionManager

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

PropertiesCacheManager

```
public PropertiesCacheManager()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Overrides:
init in class CacheManager
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.
Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
```

Description copied from interface: Handler
Responds to an HTTP request.
Specified by:
respond in interface Handler
Overrides:
respond in class CacheManager
Tags copied from interface: Handler
Parameters:
`request` - The `Request` object that represents the HTTP request.
Returns:
`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

Class SessionManager

sunlabs.brazil.session
Class SessionManager

```
java.lang.Object
  |
  +--sunlabs.brazil.session.SessionManager
```

Direct Known Subclasses:
CacheManager, PJamaSessionManager

---

public class **SessionManager**
extends Object

The `SessionManager` associates an object with a Session ID to give Handlers the ability to maintain state that lasts for the duration of a session instead of just for the duration of a request.

The `SessionManager` operates as a bag of globally accessible resources. Existing subclasses of the `SessionManager` also provide persistence, that is, a way to recover these resources even if the server process is terminated and later restarted, to get back to the state things were in.

A session manager is like a Dictionary only with fewer guarantees. Enumeration is not possible, and a "get" might return null even if there was a previous "put", so users should be prepared to handle that case.

Unlike a typical dictionary, a session manager uses two keys to identify to identify each resource. The first key, by convention, represents a client session id. The second (or resource) key is chosen to identify the resource within a session.

Care should be used when choosing resource keys, as they are global to a JVM, which may have several unrelated Brazil servers running at once. Sharing session manager resources between servers can cause unexpected behavior if done unintentionally.

Existing session manager implementations arrange for session resources that are `Java Properties` to be saved across restarts of the JVM, and should be used when practical.

Constructor Summary

`SessionManager()`	

Method Summary	
static Object	**get**(Object session, Object ident) get an object from the session manager.
static Object	**getSession**(Object session, Object ident, Class type) Returns the object associated with the given Session ID.
static void	**put**(Object session, Object ident, Object data) put an object into the session manager.
static void	**remove**(Object session, Object ident) Remove an object from the session manager.
static void	**setSessionManager**(SessionManager mgr) Installs the given `SessionManager` object as the default session manager to be invoked when `getSession` is called.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

SessionManager

public **SessionManager**()

Method Detail

setSessionManager

public static void **setSessionManager**(SessionManager mgr)

Installs the given `SessionManager` object as the default session manager to be invoked when `getSession` is called.

Parameters:

`mgr` - The `SessionManager` object.

---

getSession

```
public static Object getSession(Object session,
                                Object ident,
                                Class type)
```

Returns the object associated with the given Session ID. Passing in the same (hash-key equivalent) Session ID will return the same object. This convenience method reflects common usage.

Parameters:

`session` - The Session ID for the persistent session information. If the session does not exist, a new one is created.

`ident` - An arbitray identifier used to determine which object (associated with the given `session`) the caller wants.

`type` - The Class of the object to create. If the given `session` and `ident` did not specify an existing object, a new one is created by calling `newInstance` based on the `type`. If `null`, then this method returns `null` if the object didn't exist, instead of allocating a new object.

Returns:

an object of type `type`, or null if the object doesn't exist and `type` is null.

---

get

```
public static Object get(Object session,
                         Object ident)
```

get an object from the session manager. This static method will dispatch to the currently installed `SessionManager` instance.

---

put

```
public static void put(Object session,
                       Object ident,
                       Object data)
```

put an object into the session manager. This static method will dispatch to the currently installed `SessionManager` instance.

---

remove

```
public static void remove(Object session,
                          Object ident)
```

Remove an object from the session manager. This static method will dispatch to the currently installed `SessionManager` instance.

Package sunlabs.brazil.sql

Provide ways of integrating the SQL database query language into the Brazil project server.

Files in this package depend upon the `jdbc` driver for the database you wish to access. See Sun JDBC Technology Page for more information on JDBC, or the MySQL JDBC Drivers for a sample JDBC implementation we have used successfully in the past.

The Brazil project will also work with the embedded derby server, see: Derby for a copy of `derby.jar. although Brazil will run with versions of the Java JDK that are at least version 1.1, Derby requires at least 1.4 (or 1.3 with additional jar files).`

Class SqlTemplate

sunlabs.brazil.sql
Class SqlTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.sql.SqlTemplate
```

All Implemented Interfaces:
TemplateInterface

---

public class **SqlTemplate**
extends Template

Sample Template class for running SQL queries via jdbc and placing the results into the request properties for further processing.

Foreach session, a connection is made to an sql database via jdbc. Session reconnection is attempted if the server connection breaks. An SQL query is issued, with the results populating the request properties. The following server properties are used:

driver
: The name of the jdbc driver class for the desired database. Currently, only one driver may be specified. (e.g. *prefix*.driver=org.gjt.mm.mysql.Driver).

url
: The jdbc url used to establish a connection with the database. (e.g. *prefix*.url=jdbc:mysql://host/db?user=xxx&password=yyy).

sqlPrefix
: The properties prefix for any additional parameters that are required for this connection. For example:

```
prefix.sqlPrefix=params
params.user=my_name
params.password=xxx
```

All of the parameters are supplied to the jdbc connection at connection time.

The driver and url parameters are required. All of the code between <sql>...</sql> is taken to be an SQL query, and sent to the appropriate database for execution. The result of the query is placed into the request properties for use by other templates, such as the BSLTemplate or PropsTemplate.

For a discussion of how the results map to properties, see below.

Fields inherited from class sunlabs.brazil.template.Template
debug

Constructor Summary

SqlTemplate()	

Method Summary

Statement	**doSQL**(String query, String type, int timeout)
boolean	**init**(RewriteContext hr) Called before this template processes any tags.
Connection	**setupSql**(String url, Properties props) Setup a connection to an SQL server.
void	**tag_slash_sql**(RewriteContext hr)
void	**tag_sql**(RewriteContext hr) Replace the SQL query with the appropriate request properties.

Methods inherited from class sunlabs.brazil.template.Template
done

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

SqlTemplate

```
public SqlTemplate()
```

Method Detail

init

```
public boolean init(RewriteContext hr)
```

Description copied from class: Template
Called before this template processes any tags.
Overrides:
init in class Template

---

tag_sql

```
public void tag_sql(RewriteContext hr)
```

Replace the SQL query with the appropriate request properties. Look for the following parameters:
(NOTE - This interface is preliminary, and subject to change).

debug
: Include diagnostics in html comments

prefix
: prefix to prepend to all results. Defaults to template prefix

max
: The max # of rows returned (default=100)

na
: Value to reurn for NULL. Defaults to "n/a"

type
: The type of SQL command, one of "query", "system", or "update". these values map to the JDBC calls executeQuery(), execute() and executeUpdate() respectively. Defaults to "query".

timeout
: The number of seconds to wait for the query to finish. Defaults to "0": wait forever

eval
: If present, do ${...} to entire query. (see `getProperty`).

zeroIndex
: if true, row counts start at 0, not 1

index
: If present, use column 1 as part of the name. Otherwise, an index name is invented.

For all queries, the following properties (with the prefix prepended) are set:

columns
: The number of columns returned

rowcount
: The number of rows returned

Foreach entry in the resultant table, its property is: `${prefix}.${table_name}.${columname}.${key}`. If the `index` parameter is set, the key is the value of the first column returned. Otherwise the key is the row number, and the additional property `${prefix}.rows` contains a list of all the row numbers returned.

---

doSQL

```
public Statement doSQL(String query,
                       String type,
                       int timeout)
                throws SQLException
```

---

tag_slash_sql

```
public void tag_slash_sql(RewriteContext hr)
```

---

setupSql

```
public Connection setupSql(String url,
                           Properties props)
```

Setup a connection to an SQL server. Assume driver is already registered.

Package sunlabs.brazil.ssl

Provide ways of integrating SSL into the Brazil project server.

This is a description of SSL.

Files in this package are dependent upon the IAIK SSL implementation.

SSL may also be used with the Brazil server via an ssl gateway, where the SSL implementation runs in an external processes. See `StunnelHandler` for more information.

Class BasicSSLHandler

sunlabs.brazil.ssl
Class BasicSSLHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.ssl.BasicSSLHandler
```

All Implemented Interfaces:
Handler

---

public class **BasicSSLHandler**
extends Object
implements Handler

Start an "ssl" server. This requires jdk1.4+.

sslport
: The port to start the server on. If no port is specified, Then the existing server port is reused. **Note**: Do not use a prefix when specifying this property, it is precicely "sslport".

enable
: Must be set to "true" to turn on ssl server.

store
: The path name to the certificate keystore.

password
: The certificate store password. If the password starts with "@", the rest of the password is taken to be the name of the file containing the password. "@-" causes the password to be read from stdin.

auth
: Require a valid client certificate (not useful).

Keystores are created with:

```
keytool -genkey -keystore [store_name] -keyalg RSA
```

Constructor Summary

BasicSSLHandler()	

Method Summary

boolean	**init**(Server server, String prefix) Initializes the handler.
boolean	**respond**(Request request) Responds to an HTTP request.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

BasicSSLHandler

```
public BasicSSLHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.
Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

respond

```
public boolean respond(Request request)
               throws IOException
```

Description copied from interface: Handler
Responds to an HTTP request.
Specified by:
respond in interface Handler
Tags copied from interface: Handler
Parameters:
`request` - The `Request` object that represents the HTTP request.
Returns:
`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.
Throws:
IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

Class CertHandler

sunlabs.brazil.ssl
Class CertHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.ssl.CertHandler
```

All Implemented Interfaces:
Handler

---

public class **CertHandler**
extends Object
implements Handler

Handler for issuing A netscape certificate. Get a certificate request from the user as form data, generate the certificate, and sign it with the CA's certificate. There is currently no certificate management. The serial number is obtained from the form (if available), otherwise it is made up. [This should be converted into a template]. The following fields are required query info, either from a GET or POST:

commonname
: The name of the certificate owner

country
: The country of origin (2 char abbreviation

email
: The email address

expires
: The cert lifetime (in months)

key
: The netscape key information (from KEYGEN)

locality
: The local address of the cert holder

org
: The company

orgunit
: The division

state
: The state or provice

The following fields are optional:

serial
: the cert serial # (this is temporary)

can_sign
: cert may be used for code signing

can_email
: cert may be used for S/mime

can_ssl
: cert may be used for client SSL

The follow server properties are used:

prefix
: The url prefix for this handler to generate a cert

cert
: The file name containing the CA certificate

id
: If set, The the request property "id" MUST be set in order to generate a cert. This allows upstream handlers to do authentication before a cert is issued.

Constructor Summary

`CertHandler()`	

Method Summary

`boolean`	**`init`**`(Server server, String prefix)` Get the CA's certificate chain.
`static void`	**`main`**`(String[] args)` Generate a sample self-signed server certificate to use for signing client certificate requests.
`boolean`	**`respond`**`(Request request)` Responds to an HTTP request.

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

CertHandler

```
public CertHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Get the CA's certificate chain. prompting for the certificate password on the command line.

Specified by:

init in interface Handler

Tags copied from interface: Handler

Parameters:

`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.

`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
                throws IOException
```

Description copied from interface: Handler

Responds to an HTTP request.

Specified by:

respond in interface Handler

Tags copied from interface: Handler

Parameters:

`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

main

```
public static void main(String[] args)
```

Generate a sample self-signed server certificate to use for signing client certificate requests. We'll choose an arbitrary suite of algorithms.

Class SslHandler

sunlabs.brazil.ssl
Class SslHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.ssl.SslHandler
```

All Implemented Interfaces:
Handler, iaik.security.ssl.ServerTrustDecider, iaik.security.ssl.TrustDecider

---

public class **SslHandler**
extends Object
implements Handler, iaik.security.ssl.ServerTrustDecider

Handler for installing SSL into the server. the server's `listen` socket is replaced with a socket that implements SSL, using the implementation from Iaik.

The server requests a certificate from the user, and if provided, incorporates the user's certificate information into `request.props`.

Properties:
certDir
The absolute pathname of the directory containing the server's certificates. The "main" in this class should generate a representitive sample (but doesn't). In the mean time, you can run "java CreateCertificates" with the Iaik jar in your path, to create a set of server test certificates in the `certs` directory.
certRequired
If set, the server will ask the user for a client cert.
issuer.*
Information about the certificate issuer.
owner.*
Information about the certificate owner.
fingerprint.*
The base64 encoded fingerprints of the clients certificates
session
The property to put the certificate serial number into. Defaults to "cert.id".

NOTE: portions of this code were adapted from from the IAIK examples, used with permission.

Constructor Summary

`SslHandler()`	

Method Summary

boolean	**init**(Server server, String prefix) Initializes the handler.
boolean	**isTrustedPeer**(iaik.security.ssl.SSLCertificate cert)
boolean	**respond**(Request request) Extract client and issuer certificate information.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

SslHandler

```
public SslHandler()
```

Method Detail

isTrustedPeer

```
public boolean isTrustedPeer(iaik.security.ssl.SSLCertificate cert)
```

Specified by:
isTrustedPeer in interface iaik.security.ssl.TrustDecider

---

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.

Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.
Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
               throws IOException
```

Extract client and issuer certificate information. Insert intro request properties.
Specified by:
respond in interface Handler
Tags copied from interface: Handler
Parameters:
`request` - The `Request` object that represents the HTTP request.
Returns:
`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.
Throws:
IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

Package sunlabs.brazil.sunlabs

Experimental features that are used in the deployment of internal Brazil applications, but not considered part of the standard release.

They should be used at your own risk, and are subject to change or removal in future releases.

Class DateTemplate

sunlabs.brazil.sunlabs
Class DateTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.sunlabs.DateTemplate
```

All Implemented Interfaces:
TemplateInterface

---

public class **DateTemplate**
extends Template

Template for doing date and time calculations and formatting.

```
<date format="strftime style date string"
         ?time=nnn? ?zone=nnn? ?scan="xxx"? ?set="xxx"? >
```

format
: A "strftime" style date string (defaults to "%D %T").

time
: Number of seconds since the epoc (defaults to now);

ms
: Number of ms since the epoc (if time is not specified)

scan
: A human readable date string to use. *time* is used for relative dates. If not specified, *time* or *ms* are used as the time. The algorithm (and implementation) was borrowed from the TCL "clock scan" command.

zone
: If specified, then the specified timezone is used. GMT is used for unrecognized zones. Oherwise the server's timezone is used. (Example: "GMT");

set
: if specified, set the indicated variable with the result; do not substitute the result inline.

Variable substitutions of the for ${...} are permitted for the attributes.

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

DateTemplate()	

Method Summary

void	**tag_date**(RewriteContext hr)

Methods inherited from class sunlabs.brazil.template.Template
done, init

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

DateTemplate

public **DateTemplate**()

Method Detail

tag_date

public void **tag_date**(RewriteContext hr)

Class DelayHandler

sunlabs.brazil.sunlabs
Class DelayHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.sunlabs.DelayHandler
```

All Implemented Interfaces:
Handler, TemplateInterface

---

public class **DelayHandler**
extends Template
implements Handler

Handler or template for adding a delay into a response.

Properties:
prefix, suffix, match, glob
Specify which url's to process.
delay
The delay, in ms (defaults to 1000).

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary	
`DelayHandler()`	

Method Summary

boolean	**init**(Server server, String prefix) Initializes the handler.
boolean	**respond**(Request request) Delay before servicing this request.
void	**tag_delay**(RewriteContext hr) pause for "delay" ms

Methods inherited from class sunlabs.brazil.template.Template
done, init

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

DelayHandler

```
public DelayHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.

`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
               throws IOException
```

Delay before servicing this request.

```
<delay ms=nnn>
```

Specified by:

respond in interface Handler

Tags copied from interface: Handler

Parameters:

`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

tag_delay

```
public void tag_delay(RewriteContext hr)
```

pause for "delay" ms

Class DigestTemplate

sunlabs.brazil.sunlabs
Class DigestTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.sunlabs.DigestTemplate
```

All Implemented Interfaces:
TemplateInterface

---

public class **DigestTemplate**
extends Template

Compute the Base64 encoded SHA1 digest of a value (so I don't have to store plain text passwords). This should probably be added to the Calculator, but this is easier.
`<digest name=nnn value=vvv>`

Fields inherited from class sunlabs.brazil.template.Template
debug

Constructor Summary

DigestTemplate()	

Method Summary

void	**tag_digest**(RewriteContext hr)

Methods inherited from class sunlabs.brazil.template.Template
done, init

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

DigestTemplate

`public` **`DigestTemplate`**`()`

Method Detail

tag_digest

`public void` **`tag_digest`**`(RewriteContext hr)`

Class ExecTemplate

sunlabs.brazil.sunlabs
Class ExecTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.sunlabs.ExecTemplate
```

All Implemented Interfaces:
TemplateInterface

---

public class **ExecTemplate**
extends Template

template to exec a program, and return its arguments into request properties .

This template processes the `<exec ...>` tag. The following attributes are supported. ${...} substitutions are preformed before the command is run.

command
: The command to run. The environment (and path) are inherited from the server. This is a required parameter.

usesh
: There is a bug in "exec" that prevents passing arguments to a command with embedded whitespace. If this flag is present, then the command "/bin/sh -c [command]" is run. This only works on systems where "/bin/sh" may be executed.

prepend
: The name prepended to the properties produced by this tag

stdin
: The standard input to send to the command (if any)

encoding
: The character set encoding to use when converting the stdout and stderr properties. If no encoding attribute is present, the encoding property is used instead. Defaults to the default encoding.

The following request properties are set as a side effect:

stdout
: The standard output produced by the program, converted to a String using the default encoding.

stderr
: The standard error output produced by the program, converted to a String using the default encoding.

code
: The exit code for the program.

error
: The error message, if something went wrong.

Currently, there is no way to set the environment or current directory for the program.

Fields inherited from class sunlabs.brazil.template.Template
debug

Constructor Summary

ExecTemplate()	

Method Summary

void	**tag_exec**(RewriteContext hr)

Methods inherited from class sunlabs.brazil.template.Template
done, init

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ExecTemplate

public **ExecTemplate**()

Method Detail

tag_exec

```
public void tag_exec(RewriteContext hr)
```

Class ExpContentTemplate

sunlabs.brazil.sunlabs
Class ExpContentTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.template.ContentTemplate
              |
              +--sunlabs.brazil.sunlabs.ExpContentTemplate
```

All Implemented Interfaces:
TemplateInterface

---

public class **ExpContentTemplate**
extends ContentTemplate

Allow extracted content to be filtered through regular expressions. Many sites use comment conventions to demarcate the headers and footer information in their pages. This template allows regular expressions to be used to process the "content" value extracted by the `ContentTemplate`.

Request properties:
extract
A regular expression to match the extracted content
replace
A regular expression substitution string used to replace the content, if the expression matched.
urlPrefix
A prefix the url must match to be considered for rewriting

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

`ExpContentTemplate()`	

Method Summary

boolean	**done**(RewriteContext hr) Run the content through a regexp to do further extraction.
boolean	**init**(RewriteContext hr) Called before this template processes any tags.

Methods inherited from class sunlabs.brazil.template.ContentTemplate
tag_body, tag_content, tag_link, tag_meta, tag_script, tag_slash_body, tag_slash_content, tag_slash_head, tag_slash_title, tag_style, tag_title

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ExpContentTemplate

public **ExpContentTemplate**()

Method Detail

init

public boolean **init**(RewriteContext hr)

Description copied from class: Template
Called before this template processes any tags.
Overrides:
init in class ContentTemplate

---

done

```
public boolean done(RewriteContext hr)
```

Run the content through a regexp to do further extraction. If the regexp didn't match, leave the existing content alone.

Overrides:

done in class ContentTemplate

Class ListTemplate.MyList

sunlabs.brazil.sunlabs
Class ListTemplate.MyList

```
java.lang.Object
  |
  +--java.util.Dictionary
        |
        +--sunlabs.brazil.sunlabs.ListTemplate.MyList
```

All Implemented Interfaces:
PropertiesCacheManager.Saveable
Enclosing class:
ListTemplate

---

public static class **ListTemplate.MyList**
extends Dictionary
implements PropertiesCacheManager.Saveable

Implement a list of strings. This uses a Vector for its internal implementation, and is a Dictionary to allow convenient access to portions of the list. By implementing Saveable, lists can participate in persistence.

Field Summary

int	**chunk**
int	**chunksize**
int	**overlap**

Constructor Summary

ListTemplate.MyList()	
ListTemplate.MyList(String name) Create a named list object.	

Method Summary

void	**append**(String s, String delim) Append a list to the end of the named list
int	**chunks**()
void	**clear**() Clear a list.
void	**delete**(int i) Remove an element by index.
Enumeration	**elements**() Return the actual list items.
Object	**get**(Object k)
void	**insert**(String s, String delim, int n) Insert a list before position n.
boolean	**isEmpty**() The current object state is the "default"; "save" does not need to write out any state.
Enumeration	**keys**() Return an enumeration of the "special" keys for this list.
void	**load**(InputStream in) load a properties representation of the object, then create the object from it.
void	**max**(int n) Set the max list size.
Object	**put**(Object key, Object value) This is never used; It’s required by the interface
Object	**remove**(Object o) We should never call this; it’s required by the interface.
void	**remove**(String s, String delim) Remove items from a list, by name.
void	**save**(OutputStream out, String header) Create an ascii representation of this object in a Java Properties format.
void	**setDelim**(String delim) Set the delimiter for returning ranges.
int	**size**()
void	**sort**()

String	**toString**()
void	**unique**() Remove all non unique elements of the list.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

chunk

public int **chunk**

---

chunksize

public int **chunksize**

---

overlap

public int **overlap**

Constructor Detail

ListTemplate.MyList

public **ListTemplate.MyList**(String name)

Create a named list object.

ListTemplate.MyList

public **ListTemplate.MyList**()

Method Detail

size

```
public int size()
```

Overrides:
size in class Dictionary

---

isEmpty

```
public boolean isEmpty()
```

Description copied from interface: PropertiesCacheManager.Saveable
The current object state is the "default"; "save" does not need to write out any state.
Specified by:
isEmpty in interface PropertiesCacheManager.Saveable
Overrides:
isEmpty in class Dictionary

---

sort

```
public void sort()
```

---

insert

```
public void insert(String s,
                   String delim,
                   int n)
```

Insert a list before position n.
Parameters:
`s` - The list to insert
`delim` - The list delimiter (null for a single item(
`n` - The position to insert before

---

append

```
public void append(String s,
                   String delim)
```

Append a list to the end of the named list

---

remove

```
public void remove(String s,
                   String delim)
```

Remove items from a list, by name.

---

remove

```
public Object remove(Object o)
```

We should never call this; it's required by the interface.

Overrides:

remove in class Dictionary

---

delete

```
public void delete(int i)
```

Remove an element by index.

---

clear

```
public void clear()
```

Clear a list.

---

max

```
public void max(int n)
```

Set the max list size.

---

unique

```
public void unique()
```

Remove all non unique elements of the list. XXX: (cache stupid!)

---

setDelim

```
public void setDelim(String delim)
```

Set the delimiter for returning ranges.

---

keys

```
public Enumeration keys()
```

Return an enumeration of the "special" keys for this list.
Overrides:
keys in class Dictionary

---

elements

```
public Enumeration elements()
```

Return the actual list items.
Overrides:
elements in class Dictionary

---

get

```
public Object get(Object k)
```

Overrides:
get in class Dictionary

---

put

```
public Object put(Object key,
                  Object value)
```

This is never used; It’s required by the interface
Overrides:
put in class Dictionary

---

toString

```
public String toString()
```

Overrides:
toString in class Object

---

chunks

```
public int chunks()
```

---

save

```
public void save(OutputStream out,
                 String header)
          throws IOException
```

Description copied from interface: PropertiesCacheManager.Saveable
Create an ascii representation of this object in a Java Properties format.
Specified by:
save in interface PropertiesCacheManager.Saveable

---

load

```
public void load(InputStream in)
          throws IOException
```

load a properties representation of the object, then create the object from it.
Specified by:
load in interface PropertiesCacheManager.Saveable

Class ListTemplate

sunlabs.brazil.sunlabs
Class ListTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.sunlabs.ListTemplate
```

All Implemented Interfaces:
TemplateInterface

---

public class **ListTemplate**
extends Template

Manage lists of strings in a (hopefully) useful way. A vector is used to store list elements. Attributes to the <list> tag are used to manipulate the list, whereas "smart properties" are used to retrieve list members.

```
<list name=x [namespace=xx clear=true|false delete=xx remove=n insert=xx
                 append=xx unique=true|false sort=true|false delim=d
                 overlap=n chunksize=n chunk=n next=true|false previous=true|false]">
<list name=x ... />
</list>
```

Access to list elements is only valid between <list>... </list>. See below for details.

Inner Class Summary	
static class	**ListTemplate.MyList** Implement a list of strings.

Fields inherited from class sunlabs.brazil.template.Template
debug

Constructor Summary	
ListTemplate()	

Method Summary	
static int	**clamp**(int min, int value, int max) Clamp an integer value.
boolean	**init**(RewriteContext hr) Clear any left-over lists.
void	**tag_list**(RewriteContext hr) Process the list tag.
void	**tag_slash_list**(RewriteContext hr) remove the most recent list from the current scope.

Methods inherited from class sunlabs.brazil.template.Template
done

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ListTemplate

public **ListTemplate**()

Method Detail

init

public boolean **init**(RewriteContext hr)

Clear any left-over lists.

Overrides:

init in class Template

tag_list

```
public void tag_list(RewriteContext hr)
```

Process the list tag.

```
   name=nnn             list name (required)
   namespace=xxxx               namespace for list.  See below
 The following options are processed in order:
   clear=true|false     clear the list if it exists
   delete=xxx  [delim=d]  delete item(s) by value  (items with delimiter d)
   remove  [location=n] remove an item by index (defaults to 0)
   insert=xxx [delim=d] [location=n]
                                insert item (or items with delimiter=d)
                                insert in front of element "n" - defaults to 0
   append=xxx [delim=d] append item(s) to the end of the list
   unique=true|false    eliminate all duplicate items
   max=n                        set the maximum size of the list "n"
   sort=true            do a dictionary sort.
   delim=d              set the delimiter for retrieving values

 Once a "list" tag is seen, the following properties are
  available (for list foo):
   foo.last     - last item on the list
   for.first    - first item on the list
   foo.count    - number of items on the list
   foo.all      - the entire list
   list.gone    - the last single element removed from the list
   foo.n                - the nth element (0 is the first element)
   foo.n:m      - the range from n-m, inclusive, starting from 0
   foo.ismember.xxx - set to "yes" if "xxx" is in the list
   foo.before.xxx       - the range from 0 till the first occurrance of xxx
   foo.after.xxx        - the range after the 1st occurance of xxx til the end
```

The first 4 items, above, always appear (for non empty lists) when the properties are enumerated (as with <foreach>

In the current implementation, "ismember" checks are very fast. However once an "ismember" is accessed, insersion and deletion slows down a bit. Using "clear" will speed up insertion and deletion again.

If no `namespace` parameter is provided, the SessionID is used. This results in per-session lists by default. Specifying a namespace allows lists to be shared across sessions.

An additional set of attributes and properties may be used to manage "chunking", to access a list in pieces for paging. Additional <list> attributes to support chunking:

```
   overlap=nn           - how many items to overlap between each chunk (default=0)
   chunksize=n          - max items per chunk (default=20)
   chunk=n              - Set the current chunk number (starts at 0)
   next=true|false      - it true, advance to the next page (if available)
   previous=true|false  - it true, advance to the previous page (if available)
```

None of the above properties change the contents of the list, only how chunks are extracted using the

properties below:

```
foo.chunk.[n]                - the list of items in chunk "n"
foo.chunk.chunk      - the current list "chunk" (same as foo.chunk,${foo.chunk.current})

foo.chunk.count      - the number of chunks
foo.chunk.chunks     - the list of chunks: "0 1 2 ... n"
foo.chunk.first      - the first chunk number (always 0)
foo.chunk.before     - the list of chunk numbers before current chunk (if any)
foo.chunk.previous   - The previous chunk number (if any)
foo.chunk.current    - The current chunk number
foo.chunk.next               - The next chunk number (if any)
foo.chunk.after      - the list of chunk numbers after current chunk
foo.chunk.last               - the last chunk number
foo.chunk.size               - The max # of items/chunk
foo.chunk.overlap    - The current chunk overlap
```

---

tag_slash_list

```
public void tag_slash_list(RewriteContext hr)
```

remove the most recent list from the current scope.

---

clamp

```
public static int clamp(int min,
                        int value,
                        int max)
```

Clamp an integer value.

Parameters:

`min` - minimum legal value

`value` - the value to clamp

`max` - maximum legal value

Returns:

"value" clamped between min and max

Class MiscTemplate.GlobProperties

sunlabs.brazil.sunlabs
Class MiscTemplate.GlobProperties

```
java.lang.Object
  |
  +--java.util.Dictionary
        |
        +--java.util.Hashtable
              |
              +--java.util.Properties
                    |
                    +--sunlabs.brazil.sunlabs.MiscTemplate.GlobProperties
```

All Implemented Interfaces:
Cloneable, Map, Serializable
Enclosing class:
MiscTemplate

---

public static class **MiscTemplate.GlobProperties**
extends Properties

Special version of a properties that uses the sub expresions of the supplied glob pattern and name to define the keys 1-9. The value "0" takes the supplied default.

See Also:
Serialized Form

Constructor Summary

MiscTemplate.GlobProperties(Properties props, String pattern, String name, String dflt)

Method Summary

String	**getProperty**(String key)

Methods inherited from class java.util.Properties
getProperty, list, list, load, propertyNames, save, setProperty, store

Methods inherited from class java.util.Hashtable
clear, clone, contains, containsKey, containsValue, elements, entrySet, equals, get, hashCode, isEmpty, keys, keySet, put, putAll, remove, size, toString, values

Methods inherited from class java.lang.Object
getClass, notify, notifyAll, wait, wait, wait

Constructor Detail

MiscTemplate.GlobProperties

```
public MiscTemplate.GlobProperties(Properties props,
                                   String pattern,
                                   String name,
                                   String dflt)
```

Method Detail

getProperty

```
public String getProperty(String key)
```

Overrides:

getProperty in class Properties

Class MiscTemplate

sunlabs.brazil.sunlabs
Class MiscTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.sunlabs.MiscTemplate
```

All Implemented Interfaces:
TemplateInterface

---

public class **MiscTemplate**
extends Template

Template for misc string manipulation functions.
<append name="..." value="..." [delim=".." namespace="..."]>
Append a value to an existing variable, using an (optional) delimiter to separate values.
<push name="..." value="..." namespace="...">
Push a variable value on a stack, and replace the value with a new one.
<pop name="..." namespace="...">
Pop a previously pushed value from the stack.
<increment name="..." [incr=n namespace="..."]>
Increment the value of a variable.
<sequence name=".." count=".." [start=".." incr=".." delim=".."]>
Generate a sequence of values.
<expr value="..." [name="..." namespace="..."]>
Do an arithmetic expression evaluation on "value".
<inline name="..." [eval="true|false]> ... </inline>
Place all the text between the "inline" and "/inline" tags into the variable indicated by "name".
<eval markup="..." [atEnd=true|false]>
Evaluate a variable as if the value was in-line markup. If "atEnd" is specified, the markup is interpolated at the end of the file.
<stringop name="..." [newname="..." range="start,end" trim="left|right|both" case="upper|lower|title" convert="url|html">
Perform misc. string functions on the value of a named variable.
<mapnames src=src_pattern dst=dst_pattern [namespace=dest_namespace] [range="start,end", trim="left,right", case="upper|lower|title" convert=url|html]>
Map a set of variables from one namespace to another, and optionally change the names and process the values. The "src" parameter is a glob pattern that is expected to match the names of one of more properties. The "dst" parameter (which defaults to the name actually matched) is the new name of the variable that is copied into the specified "namespace" (which defaults to "local"). ${...} substitutions are performed on the value of "dst" separately for each matched variable. The variables ${1}, ${2} ... ${i} ... ${9} take on the value of the "i"th wildcard match, as in Glob.match(). ${0} refers to the actual name of the matched variable, and ${mapCount} is the index of the current variable being mapped, starting with "1".

All the conversion options provided by <stringop> are also available.

Example:

```
<map src=query.pref_* dst="pref_${1}" namepace=${SessionID}.pref>
```

Will take the query parameters from the current form that refer to user preferences, and save them in a preferences namespace associated with the current session.

<stringlength name="xxx" value="${...}">

Compute the length of the string specified in "value" and place it in the property "name".

<random name=xxx [count=xxx start=xxx type=int|hex|alpha>

Generate a random value. if "type=int", the default, then count (defaults to 2) specifies the range of integers from which the value is taken, and "start" (defaults to 1) specifies the minimum value.

When "type=hex", then "start" is taken to be a hex value (or 0 if start is not a valid hex value).

When "type=alpha", the "start" must contain only a-z, and the result is the alpabetic result, in radix 26.

Inner Class Summary

`static class`	**`MiscTemplate.GlobProperties`** Special version of a properties that uses the sub expresions of the supplied glob pattern and name to define the keys 1-9.

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

`MiscTemplate()`	

Method Summary	
static int	**fromAlpha**(String s) Convert an alpha-only string to an integer.
boolean	**init**(RewriteContext hr) Called before this template processes any tags.
void	**tag_append**(RewriteContext hr) Append a value to a property, with an optional delimeter.
void	**tag_eval**(RewriteContext hr) Evaluate the contents of a variable as a template, as if it was "inline" here.
void	**tag_expr**(RewriteContext hr) Set a variable to the result of an arithmetic expression <expr name=nnn value="expr" [namespace="..."]> sets name to "NaN" if the expression was invalid.
void	**tag_increment**(RewriteContext hr) Increment the value of a variable.
void	**tag_inline**(RewriteContext hr) Set a variable to all the markup ’till the /inline tag.
void	**tag_mapnames**(RewriteContext hr) Map a set of properties from one namespace to another, and (optionally) change their names and values.
void	**tag_pop**(RewriteContext hr) Treat a variable as a stack, and pop a value from it.
void	**tag_push**(RewriteContext hr) Treat a variable as a stack, and push a value onto it.
void	**tag_random**(RewriteContext hr)
void	**tag_sequence**(RewriteContext hr) Generate a (mostly) numeric sequence.
void	**tag_stringlength**(RewriteContext hr) Compute string length from "value" , return in "name"
void	**tag_stringop**(RewriteContext hr) String manipulation functions.
static String	**toAlpha**(int i)

Methods inherited from class sunlabs.brazil.template.Template
done

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

MiscTemplate

```
public MiscTemplate()
```

Method Detail

tag_append

```
public void tag_append(RewriteContext hr)
```

Append a value to a property, with an optional delimeter. <append name="..." value="..." [delim="..." namespace="..."]> If "name" is undefined or empty, it is set to "value". The result is placed in the local namespace, unless the "namespace" is used to override it. Use namespace=${SessionID} to put the result into the current session. Use the ListTemplate for more generic list manipulation.

---

tag_increment

```
public void tag_increment(RewriteContext hr)
```

Increment the value of a variable. <incr name=variable [incr=n namespace="..."]> -If undefined or empty, assume it was ’0’.
-If the value was non numeric, append or modify any numeric suffix.
The result is placed in the local namespace, unless the "namespace" is used to override it. Use namespace=${SessionID} to put the result into the current session.

---

tag_sequence

```
public void tag_sequence(RewriteContext hr)
```

Generate a (mostly) numeric sequence. <sequence name="..." count="..." [start="..." incr="..." delim=".."]>

name
: Where to put the result. Defaults to the local namespace, but may be overridden with "namespace=".

count
: the number of values to generate

start
: The start value. Defaults to "1". If non numeric, any trailing digits are incremented, if present, or appended otherwise.
: The increment value. Defaults to "1" if the value is not an integer.

start
: The starting value

delim
: The delimiter between values. defaults to " ".

---

tag_expr

```
public void tag_expr(RewriteContext hr)
```

Set a variable to the result of an arithmetic expression <expr name=nnn value="expr" [namespace="..."]> sets name to "NaN" if the expression was invalid. The result is placed in the local namespace, unless the "namespace" is used to override it. Use namespace=${SessionID} to put the result into the current session.

If "name" is omitted, the result is placed in the document.

---

tag_eval

```
public void tag_eval(RewriteContext hr)
```

Evaluate the contents of a variable as a template, as if it was "inline" here. <eval markup="..." [atEnd=true|false]>

---

init

```
public boolean init(RewriteContext hr)
```

Description copied from class: Template
Called before this template processes any tags.
Overrides:
: init in class Template

---

tag_push

```
public void tag_push(RewriteContext hr)
```

Treat a variable as a stack, and push a value onto it. <push name="..." value="..." [clear=true|false namespace="..."]> For more generic list manipulation, use the ListTemplate.

The result is placed in the local namespace, unless the "namespace" is used to override it. Use namespace=${SessionID} to put the result into the current session.

---

tag_pop

```
public void tag_pop(RewriteContext hr)
```

Treat a variable as a stack, and pop a value from it. <pop name="..." [clear=true|false namespace="..."]> The result is placed in the local namespace, unless the "namespace" is used to override it. Use namespace=${SessionID} to put the result into the current session.

---

tag_inline

```
public void tag_inline(RewriteContext hr)
```

Set a variable to all the markup ’till the /inline tag. <inline name="..." [eval=true|false]> ... </inline>

If "eval" is true, then ${..} substitutions are performed before assigning the markup the the named variable. If no "name" is specified, the markup is output as-is, after ${..} substitutions (e.g. eval=true is implied).

---

tag_stringop

```
public void tag_stringop(RewriteContext hr)
```

String manipulation functions. Any of the following options are supported. If more than one of "range", "trim", "case", or "convert" is specified, they are performed in the order listed below. The value named by the "name" attribute is modified, and the result is placed in the local namespace, using the same name, unless "newname" is specified, in which case "newname" is used instead.

The "namespace" attribute can be used to alter the namespace to put the result into, in which case the "sessionTable" configuration parameter can be used to alter the namespace class, which defaults to the template handler’s (or filter’s) prefix.
<stringop name=xxx range=x,y [newname=xxx namespace=xxx]>
<stringop name=xxx range=x [newname=xxx namespace=xxx]>
Do a substring from character position ’x’ (starting at 0) up to but not including character position ’y’. Negative values count from the end of the string. If no "y" is specified, then the end of the string is assumed.
<stringop name=xxx trim=left|right|both [newname=xxx namespace=xxx]>
Trim whitespace from the string. "trim=true" is equivalent to "both".

<stringop name=xxx case=lower|upper|title [newname=xxx namespace=xxx]>
Do case conversions. "title" causes the entire string to be lower cased, except for the first character after any whitespace, which is upper-cased.
<stringop name=xxx convert=html|url [newname=xxx namspace=xxx]>
The string is converted exactly the same as the "convert" option of the "get" tag in the SetTemplate().

---

tag_stringlength

```
public void tag_stringlength(RewriteContext hr)
```

Compute string length from "value" , return in "name"

---

tag_mapnames

```
public void tag_mapnames(RewriteContext hr)
```

Map a set of properties from one namespace to another, and (optionally) change their names and values. <mapnames src=glob-pattern namespace=dst_space dst=dst-pattern [stringops options]>

---

tag_random

```
public void tag_random(RewriteContext hr)
```

---

toAlpha

```
public static String toAlpha(int i)
```

---

fromAlpha

```
public static int fromAlpha(String s)
```

Convert an alpha-only string to an integer. no error checking!

Class PlainTemplate

sunlabs.brazil.sunlabs
Class PlainTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.sunlabs.PlainTemplate
```

All Implemented Interfaces:
TemplateInterface

---

public class **PlainTemplate**
extends Template

Template to turn all markup between <plain> and </plain> into ordinary text, by escaping all HTML markup.

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

`PlainTemplate``()`	

Method Summary

`boolean`	**`init`**`(RewriteContext hr)` Called before this template processes any tags.
`void`	**`tag_plain`**`(RewriteContext hr)`

Methods inherited from class sunlabs.brazil.template.Template
`done`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

PlainTemplate

```
public PlainTemplate()
```

Method Detail

init

```
public boolean init(RewriteContext hr)
```

Description copied from class: Template
Called before this template processes any tags.
Overrides:
init in class Template

---

tag_plain

```
public void tag_plain(RewriteContext hr)
```

Class RemoteHostTemplate

sunlabs.brazil.sunlabs
Class RemoteHostTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.sunlabs.RemoteHostTemplate
```

All Implemented Interfaces:
TemplateInterface

---

public class **RemoteHostTemplate**
extends Template

Return the remote host name associated with this request. Note: This involves a reverse dns lookup, which can be expensive.
name
where to put the result, defaults to [prefix].host

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

`RemoteHostTemplate``()`	

Method Summary

`void`	**`tag_remotehost`**`(RewriteContext hr)`

Methods inherited from class sunlabs.brazil.template.Template
`done, init`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

RemoteHostTemplate

```
public RemoteHostTemplate()
```

Method Detail

tag_remotehost

```
public void tag_remotehost(RewriteContext hr)
```

Class RestartHandler

sunlabs.brazil.sunlabs
Class RestartHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.sunlabs.RestartHandler
```

All Implemented Interfaces:
Handler, sun.misc.SignalHandler

---

public class **RestartHandler**
extends Object
implements Handler, sun.misc.SignalHandler

Restart the server when a sigHUP is received. Only The handlers are restarted, by creating new instances of them, and calling the respective init() methods. None of the other server properties (such as the listening port) are effected. Any requests that are currently in-progress complete using the old configuration.

NOTES:
- Supplying an invalid configuration file can render the server inoperable.
- The non-portable `sun.misc.Signal*` classes are used.

Properties:
config
The name of the configuration file to use for this server. Relative paths are resolved relative to the current directory. If no file is specified, the server continues to use its existing configuration [which has presumably been modified in-place]. If a config file is specified and the config file hasn't changed, then no restart is done.
keep
If set, and a config file is specified, the existing configuration (server.props) is not cleared first.

Fields inherited from interface sun.misc.SignalHandler
`SIG_DFL, SIG_IGN`

Constructor Summary

`RestartHandler()`	

Method Summary

void	**handle**(sun.misc.Signal sig) Restart the server after re-reading the config file
boolean	**init**(Server server, String prefix) Initializes the handler.
boolean	**respond**(Request request) Responds to an HTTP request.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

RestartHandler

```
public RestartHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
```

Description copied from interface: Handler
Responds to an HTTP request.
Specified by:
respond in interface Handler
Tags copied from interface: Handler
Parameters:
`request` - The `Request` object that represents the HTTP request.
Returns:
`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.
Throws:
IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

handle

```
public void handle(sun.misc.Signal sig)
```

Restart the server after re-reading the config file
Specified by:
handle in interface sun.misc.SignalHandler

Class SnarfTemplate

sunlabs.brazil.sunlabs
Class SnarfTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.sunlabs.SnarfTemplate
```

All Implemented Interfaces:
TemplateInterface

---

public class **SnarfTemplate**
extends Template

Template class for extracting content out of <snarf property=xxx> ... </snarf> pairs. *xxx* is the name of the property to append the *snarf*ed content to. Defaults to `[prepend].snarf`. All snarf'ed content is deleted.

Properties:
`prepend`
The string to prepend all properties with. Defaults to the handler's prefix.
`debug`
If set, the *snarf* tags will be replaced by comments.

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

`SnarfTemplate()`	

Method Summary	
boolean	**init**(RewriteContext hr) Get the debug flag and reset page.
void	**tag_slash_snarf**(RewriteContext hr) Save the content gathered so far.
void	**tag_snarf**(RewriteContext hr) Mark the current location in the document.

Methods inherited from class sunlabs.brazil.template.Template
done

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

SnarfTemplate

public **SnarfTemplate**()

Method Detail

init

public boolean **init**(RewriteContext hr)

Get the debug flag and reset page.

Overrides:

init in class Template

---

tag_snarf

```
public void tag_snarf(RewriteContext hr)
```

Mark the current location in the document.

---

tag_slash_snarf

```
public void tag_slash_snarf(RewriteContext hr)
```

Save the content gathered so far.

Class SourceTemplate

sunlabs.brazil.sunlabs
Class SourceTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.sunlabs.SourceTemplate
```

All Implemented Interfaces:
TemplateInterface

---

public class **SourceTemplate**
extends Template

Template class for incorporating the content of a local file into the current document.

```
<source src=file
      [encoding=enc name=property eval=true|false reprocess=true|false]
>
```

Attributes:

src
: Where to find the document to source. Unless starting with "/", it is assumed to be relative to the document root.

eval
: If true, all ${...} are evaluated as the file is read in from storage.

encoding
: Specifies the character encoding to use. If not specified or invalid, the default encoding is used.

name
: If set, the content is placed in the named variable instead of being inserted in-line.

reprocess
: If true, and `name` is not set (e.g. the content is inserted in-line, then the content will be run through the normal template processing before being inserted into the current document.

Examples:

The form: `<source src=xml.tmpl eval=true>` can be used to read in XML templates (or template fragments) that contain ${...} constructs that get *filled in* as the template is read.

The form: `<source src=section1.html reprocess=true>` can be used as a convenient way to break a single logical file into separate sections. (Note: this isn't currently implemented in a very efficient way.

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

SourceTemplate()	

Method Summary

void	**tag_source**(RewriteContext hr)

Methods inherited from class sunlabs.brazil.template.Template
done, init

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

SourceTemplate

public **SourceTemplate**()

Method Detail

tag_source

public void **tag_source**(RewriteContext hr)

Class StunnelHandler

sunlabs.brazil.sunlabs
Class StunnelHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.sunlabs.StunnelHandler
```

All Implemented Interfaces:
Handler

---

public class **StunnelHandler**
extends Object
implements Handler

Handler to enable proper interaction with a protocol conversion gateway, by rewriting "redirect" directives properly. For example, this handler may be used with stunnel (see stunnel.org), configured as an SSL gateway. enabling Brazil with an external ssl protocol stack. For example, the stunnel configuration

```
[https]
accept  = 443
connect = 8080
```

Will allow "https" connections on the standard port ssl (443) to access a Brazil server on port 8080. When using Brazil in this configuration without this handler, since Brazil talks to the gateway via "http", it will issue redirects to "http", which is the wrong protocol. This template looks at the origin ip address, and if it matches, changes the server protocol for this request, resulting in the client redirecting back through the gateway properly.

Properties:
ssl
The regexp to match client ip addresses that are coming from ssl gateways (such as stunnel).
protocol
The protocol to replace "http" with when redirection via a gateway (defaults to "https").

Constructor Summary

StunnelHandler()	

Method Summary

boolean	**init**(Server server, String prefix) Initializes the handler.
boolean	**respond**(Request request) If we are coming from the machine which is designated as our ssl gateway, then we need to change the protocol to "https" and remap the default port.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

StunnelHandler

```
public StunnelHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.
Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

respond

```
public boolean respond(Request request)
               throws IOException
```

If we are coming from the machine which is designated as our ssl gateway, then we need to change the protocol to "https" and remap the default port.

Specified by:

respond in interface Handler

Tags copied from interface: Handler

Parameters:

`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

Class SubstAllTemplate

sunlabs.brazil.sunlabs
Class SubstAllTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.sunlabs.SubstAllTemplate
```

All Implemented Interfaces:
TemplateInterface

---

public class **SubstAllTemplate**
extends Template

Template to substitute ${...} for the value of name/value attribute pairs of all html tags that aren't otherwise accounted for by other templates. This enables the elimination of `<tag>...</tag>` constructs in most cases.

See Also:
`MacroTemplate`

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

`SubstAllTemplate``()`	

Method Summary

`void`	**`defaultTag`**`(RewriteContext hr)`

Methods inherited from class sunlabs.brazil.template.Template
done, init

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

SubstAllTemplate

```
public SubstAllTemplate()
```

Method Detail

defaultTag

```
public void defaultTag(RewriteContext hr)
```

Class XmlTree.DefaultNodeName

sunlabs.brazil.sunlabs
Class XmlTree.DefaultNodeName

```
java.lang.Object
  |
  +--sunlabs.brazil.sunlabs.XmlTree.DefaultNodeName
```

All Implemented Interfaces:
XmlTree.NodeName
Enclosing class:
XmlTree

---

public static class **XmlTree.DefaultNodeName**
extends Object
implements XmlTree.NodeName

The node is named by the specified attribute. If no attribute is specified, use the tag name instead, followed by the index in ().

Constructor Summary

XmlTree.DefaultNodeName()
XmlTree.DefaultNodeName(String attribute, String dflt)

Method Summary

String	**getName**(XmlTree.Node n)
String	**toString**()

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

XmlTree.DefaultNodeName

```
public XmlTree.DefaultNodeName()
```

XmlTree.DefaultNodeName

```
public XmlTree.DefaultNodeName(String attribute,
                               String dflt)
```

Method Detail

getName

```
public String getName(XmlTree.Node n)
```

Specified by:
getName in interface XmlTree.NodeName

---

toString

```
public String toString()
```

Overrides:
toString in class Object

Class XmlTree.IllegalXmlException

sunlabs.brazil.sunlabs
Class XmlTree.IllegalXmlException

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.lang.RuntimeException
                    |
                    +--java.lang.IllegalArgumentException
                          |
                          +--sunlabs.brazil.sunlabs.XmlTree.IllegalXmlException
```

All Implemented Interfaces:
Serializable
Enclosing class:
XmlTree

---

public static class **XmlTree.IllegalXmlException**
extends IllegalArgumentException

Do some more reasonable error handling. We'll stuff more info into the exception.

See Also:
Serialized Form

Field Summary

Vector	**errors**

Constructor Summary

XmlTree.IllegalXmlException(String s)	

Method Summary	
int	**addError**(int sl, String st, int cl, String ct)
static XmlTree.IllegalXmlException	**getEx**(XmlTree.IllegalXmlException e, String s, int sl, String st, int cl, String ct)
String	**toString**()

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

errors

public Vector **errors**

Constructor Detail

XmlTree.IllegalXmlException

public **XmlTree.IllegalXmlException**(String s)

Method Detail

addError

```
public int addError(int sl,
                    String st,
                    int cl,
                    String ct)
```

---

getEx

```
public static XmlTree.IllegalXmlException getEx(XmlTree.IllegalXmlException e,
                                                String s,
                                                int sl,
                                                String st,
                                                int cl,
                                                String ct)
```

---

toString

```
public String toString()
```

Overrides:

toString in class Throwable

Class XmlTree.Node

sunlabs.brazil.sunlabs
Class XmlTree.Node

```
java.lang.Object
  |
  +--sunlabs.brazil.sunlabs.XmlTree.Node
```

Enclosing class:
XmlTree

---

public static class **XmlTree.Node**
extends Object

This describes a node of the XML tree

Constructor Summary

XmlTree.Node(String tag, boolean singleton, StringMap attributes, XmlTree.Node parent, int type, int index)

Method Summary	
void	**addChild**(XmlTree.Node child)
void	**appendCdata**(String s)
int	**childCount**()
String	**getAttribute**(String name)
StringMap	**getAttributes**()
String	**getCdata**()
XmlTree.Node	**getChild**(int i)
int	**getIndex**()
XmlTree.Node	**getParent**()
String	**getTag**()
boolean	**isSingle**()
void	**putAttribute**(String key, String value)
void	**setCdata**(String s)
void	**setSingle**(boolean s)
String	**toString**()

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

XmlTree.Node

```
public XmlTree.Node(String tag,
                    boolean singleton,
                    StringMap attributes,
                    XmlTree.Node parent,
                    int type,
                    int index)
```

Method Detail

setCdata

```
public void setCdata(String s)
```

---

putAttribute

```
public void putAttribute(String key,
                         String value)
```

---

appendCdata

```
public void appendCdata(String s)
```

---

addChild

```
public void addChild(XmlTree.Node child)
```

---

getChild

```
public XmlTree.Node getChild(int i)
```

---

getTag

```
public String getTag()
```

---

getIndex

```
public int getIndex()
```

---

isSingle

```
public boolean isSingle()
```

---

setSingle

```
public void setSingle(boolean s)
```

---

getAttributes

```
public StringMap getAttributes()
```

---

getAttribute

```
public String getAttribute(String name)
```

---

getCdata

```
public String getCdata()
```

---

getParent

```
public XmlTree.Node getParent()
```

---

childCount

```
public int childCount()
```

---

toString

```
public String toString()
```

Overrides:
toString in class Object

Interface XmlTree.NodeName

sunlabs.brazil.sunlabs
Interface XmlTree.NodeName
All Known Implementing Classes:
XmlTree.DefaultNodeName
Enclosing class:
XmlTree

---

public static interface **XmlTree.NodeName**

Method Summary

String	**getName**(XmlTree.Node n)

Method Detail

getName

```
public String getName(XmlTree.Node n)
```

Class XmlTree.XmlErrorInfo

sunlabs.brazil.sunlabs
Class XmlTree.XmlErrorInfo

```
java.lang.Object
  |
  +--sunlabs.brazil.sunlabs.XmlTree.XmlErrorInfo
```

Enclosing class:
XmlTree

---

public static class **XmlTree.XmlErrorInfo**
extends Object

Field Summary

int	**currentLine**
String	**currentTag**
int	**startLine**
String	**startTag**

Constructor Summary

XmlTree.XmlErrorInfo(int startLine, String startTag, int currentLine, String currentTag)

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

startLine

```
public int startLine
```

---

startTag

```
public String startTag
```

---

currentLine

```
public int currentLine
```

---

currentTag

```
public String currentTag
```

Constructor Detail

XmlTree.XmlErrorInfo

```
public XmlTree.XmlErrorInfo(int startLine,
                            String startTag,
                            int currentLine,
                            String currentTag)
```

Class XmlTree

sunlabs.brazil.sunlabs
Class XmlTree

```
java.lang.Object
  |
  +--java.util.Dictionary
        |
        +--sunlabs.brazil.sunlabs.XmlTree
```

All Implemented Interfaces:
PropertiesCacheManager.Saveable

---

public class **XmlTree**
extends Dictionary
implements PropertiesCacheManager.Saveable

Create a tree representation of an xml file whose parts may be referenced as a dictionary. This is currently "read only".

Inner Class Summary	
static class	**XmlTree.DefaultNodeName** The node is named by the specified attribute.
static class	**XmlTree.IllegalXmlException** Do some more reasonable error handling.
static class	**XmlTree.Node** This describes a node of the XML tree
static interface	**XmlTree.NodeName**
static class	**XmlTree.XmlErrorInfo**

Constructor Summary	
XmlTree() Make an empty tree.	
XmlTree(String src) Given an XML string, build the tree.	

Method Summary	
Enumeration	**elements**()
static void	**elements**(XmlTree.Node n, Vector v)
Object	**get**(Object k) Given a node description, return the value, if any.
static String	**getFile**(InputStream in)
static String	**getFile**(String s)
String	**getPart**(String s) Given a node descriptor, return the result.
Hashtable	**getTags**()
boolean	**isEmpty**() The current object state is the "default"; "save" does not need to write out any state.
Enumeration	**keys**()
void	**keys**(XmlTree.Node n, String prefix, String delim, Vector v)
void	**load**(InputStream in) Recreate the object from the ascii representation stored as a Properties format file.
static void	**main**(String[] args)
Vector	**match**(String pattern) Find all nodes that match a glob pattern, starting at the root.
void	**match**(XmlTree.Node node, StringTokenizer st, Vector results) Find all nodes that match a glob pattern, starting at any node.
Object	**put**(Object k, Object v)
Object	**remove**(Object o)

void	**replace**(String src) Replace the XmlTree with new markup.
void	**save**(OutputStream out, String header) Create an ascii representation of this object in a Java Properties format.
XmlTree.Node	**search**(String s) Find a node in the tree by name, starting at the root.
XmlTree.Node	**search**(XmlTree.Node node, StringTokenizer st) Find a node in the tree by name, starting under any node.
boolean	**setAttribute**(String name, String key, String value)
boolean	**setCdata**(String name, String data)
void	**setComparator**(XmlTree.NodeName nodeName) Set the class that determines a node's name.
void	**setDelim**(String delim) set the node delimiter.
boolean	**setDflt**(String dflt)
void	**setIdent**(String ident)
void	**setPrefix**(String prefix) set the name of this tree
void	**setTag**(String tag) Add an element to the tag process list.
void	**setTags**(Hashtable tags) Set the list of tags to process
int	**size**()
String	**toString**() Print a tree
void	**toString**(XmlTree.Node node, StringBuffer sb, int level)

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

XmlTree

```
public XmlTree()
```

Make an empty tree.

XmlTree

```
public XmlTree(String src)
        throws IllegalArgumentException
```

Given an XML string, build the tree.

Method Detail

setIdent

```
public void setIdent(String ident)
```

---

setTag

```
public void setTag(String tag)
```

Add an element to the tag process list. Once a Process tag is defined, only tags defined are processed. All other tags are treated as singletons

---

setTags

```
public void setTags(Hashtable tags)
```

Set the list of tags to process

---

getTags

```
public Hashtable getTags()
```

---

setPrefix

```
public void setPrefix(String prefix)
```

set the name of this tree

---

setDelim

```
public void setDelim(String delim)
```

set the node delimiter.

---

setDflt

```
public boolean setDflt(String dflt)
```

---

setComparator

```
public void setComparator(XmlTree.NodeName nodeName)
```

Set the class that determines a node's name. This may be used to change the way nodes are named in an arbitrary fashion.

---

toString

```
public String toString()
```

Print a tree

Overrides:

toString in class Object

Parameters:

`node:` - The starting node

`sb:` - where to append the results to

`level:` - the nesting level

---

toString

```
public void toString(XmlTree.Node node,
                     StringBuffer sb,
                     int level)
```

---

replace

```
public void replace(String src)
             throws XmlTree.IllegalXmlException
```

Replace the XmlTree with new markup.

Parameters:
`src` : - the xml data

---

setAttribute

```
public boolean setAttribute(String name,
                            String key,
                            String value)
```

---

setCdata

```
public boolean setCdata(String name,
                        String data)
```

---

search

```
public XmlTree.Node search(String s)
```

Find a node in the tree by name, starting at the root.
Parameters:
`s` - The node pathname
Returns:
The node, if found, or null

---

search

```
public XmlTree.Node search(XmlTree.Node node,
                           StringTokenizer st)
```

Find a node in the tree by name, starting under any node.

---

match

```
public Vector match(String pattern)
```

Find all nodes that match a glob pattern, starting at the root.

---

match

```
public void match(XmlTree.Node node,
                  StringTokenizer st,
                  Vector results)
```

Find all nodes that match a glob pattern, starting at any node.

---

elements

```
public Enumeration elements()
```

Overrides:
elements in class Dictionary

---

elements

```
public static void elements(XmlTree.Node n,
                            Vector v)
```

---

keys

```
public Enumeration keys()
```

Overrides:
keys in class Dictionary

---

keys

```
public void keys(XmlTree.Node n,
                 String prefix,
                 String delim,
                 Vector v)
```

---

get

```
public Object get(Object k)
```

Given a node description, return the value, if any. Descriptions are of the form: [prefix].name.[suffix] where : [prefix] is the name of the tree "." is the current delimiter, name is the path name of a node in the tree [suffix] specifies which part of the node to return as a string. See getpart() for the list of valid suffixes.

Overrides:
get in class Dictionary

---

getPart

```
public String getPart(String s)
```

Given a node descriptor, return the result. XXX not done modifiers: cdata: return cdata tag: the name index: which tag within whis parent attributes: the list of attribute names children: the list of children childCount: the number of children .value the value for attribute glob nodes matching the glob pattern all all nodes under this one

---

put

```
public Object put(Object k,
                  Object v)
```

Overrides:
put in class Dictionary

---

remove

```
public Object remove(Object o)
```

Overrides:
remove in class Dictionary

---

size

```
public int size()
```

Overrides:
size in class Dictionary

---

isEmpty

```
public boolean isEmpty()
```

Description copied from interface: PropertiesCacheManager.Saveable
The current object state is the "default"; "save" does not need to write out any state.
Specified by:
isEmpty in interface PropertiesCacheManager.Saveable
Overrides:
isEmpty in class Dictionary

---

load

```
public void load(InputStream in)
          throws IOException
```

Description copied from interface: PropertiesCacheManager.Saveable
Recreate the object from the ascii representation stored as a Properties format file.
Specified by:
load in interface PropertiesCacheManager.Saveable

---

save

```
public void save(OutputStream out,
                 String header)
          throws IOException
```

Description copied from interface: PropertiesCacheManager.Saveable
Create an ascii representation of this object in a Java Properties format.
Specified by:
save in interface PropertiesCacheManager.Saveable

---

main

```
public static void main(String[] args)
```

---

getFile

```
public static String getFile(String s)
                     throws IOException
```

---

getFile

```
public static String getFile(InputStream in)
                     throws IOException
```

Class XmlTreeTemplate

sunlabs.brazil.sunlabs
Class XmlTreeTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.sunlabs.XmlTreeTemplate
```

All Implemented Interfaces:
TemplateInterface

---

public class **XmlTreeTemplate**
extends Template

Manage a tree representation of an XML document. The xml document is manipulated via the "xmltree" tag. the "content" of the tree is retrieved by "flattening" the tree and accessing its contents as name/value pairs.

```
Attributes:
  src: the source file for this template, relative to the doc root
  eval:        process source file through subst()
  root:        where to lookup the template file
  xml:       The source xml, if "src" is not specified
               This may be used if the "xml" resides in a property
  name:        the name (and prefix) of the tree thingy
  delim:     which delimeter to use (defaults to ".")
  default:     what to use as the default "suffix" (defaults to "cdata")
  attribute: If not "", it specifies an attribute to use to name a
             particular node.  If an attribute is specified, and multiple
             nodes have the same name, then all nodes in the tree may not
               be uniquly identifiable.
  dflt:      The value to use if attribute is specified, but the node doesn't
               have one (defaults to "unknown").
  namespace: which namespace to use (defaults to session id)
  tags:        The list of xml tags to be processed.  All tags not on this
               list are considered to be "singletons".  tags="" clears the
               list, and all tags will be processed.

Portions of the tree are accessed using ordinary BSL variable, whos names
map to elements of the tree.  The variables have the form:
name.path.suffix, where:
- name is the name of the tree
- path is the path name of a particular node
- suffix indicates which aspect of the node is returned.  See below
  for the list of suffixes.

The "." which separates the name, path, and suffix, as well as
the elements within the path may be changed to another character
with the "delim" attribute.

Nodes are identified by their path name, which is the "delim" separated
list of node names, starting from the root node.  The node names, which
identify a particular node in the tree, are configurable using
"attribute" attribute.  By default, a node is named by:
 tag(index), where "tag" is the xml tag, and "index" is the index of
 that tag witin its parent, starting from 0.

The suffix may be one of:
- cdata                Returns the plain text between this tag and the next
                       one.  Since closing tags (e.g. ) are not
                       considered nodes in the tree, any "cdata" after
```

```
                        a closing tag is associated with the previous tag.
 - tag          The name of the current tag
 - index                The index of this tag within its parent.
 - attributes           A space delimited list of attribute names associated
                        with this node
 - children             A space delimited list of direct child node names
                        for this node.
 - childCount           The number of direct child nodes
 - [attribute].value    The value of [attribute], if it exists
 - all          A space delimited list of all decendants of this
                        node (including this node)
 - glob         A space delimited list of all nodes whos names match
                        the supplied glob pattern [broken]
 Examples:
 Consider the following XML snippet in the file called sample.xml:

  <sample name="main" a="1" b="2">
    <part name="shoe" price="17">hard soles</part>
    <part name="hat" price="11">Green, with feathers</part>
  </sample>
  <order name="joe">
    <description>
      This is a description of the order
    </description>
    <payment name="ante up" method="cash" amount="28" />
  </order>

 The following tags should cause the indicated variables to "exist" with
 the values shown:
  <xmltree name=tree src=sample.xml>
  tree.all= tree.sample(0) tree.sample(0).part(0) tree.sample(0).part(1) tree.order(0) tree.order(0).description(0)
tree.order(0).payment(0)
  tree.order(0).description(0).cdata = This is a description of the order
  tree.order(0).description(0) = This is a description of the order
    [the default suffix is "cdata"]
  tree.sample(0).children = part(0) part(1)
  sample(0).all = tree.sample(0) tree.sample(0).part(0) tree.sample(0).part(1)
  sample(0).part(0) = hard soles
  sample(0).part(0).attributes = name price
  sample(0).part(0).price.value = "17"
  sample(0).childCount = 2

  <xmltree name=tree delim=: attribute=name>
   [ since src=sample.xml is already defined, it isn't required]
  all = tree:"main" tree:"main":"shoe" tree:"main":"hat" tree:"joe" tree:"joe":unknown tree:"joe":"ante up"
  "main":"shoe" = hard soles
  "main":"shoe":price:value) = "17"

  <xmltree name=tree delim=. attribute=name default=children>
  sample(0) = part(0) part(1)

  When combined with the BSLTemplate, the "foreach" tag may be used
  to iterate through all the node names, in which case the default
  value (e.g. cdata) is used.

  If the XML is invalid, the property "name".error is set to a hopefully
  useful error message
```

Fields inherited from class sunlabs.brazil.template.Template
debug

Constructor Summary

XmlTreeTemplate()	

Method Summary

void	**tag_xmltree**(RewriteContext hr)

Methods inherited from class sunlabs.brazil.template.Template
done, init

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

XmlTreeTemplate

public **XmlTreeTemplate**()

Method Detail

tag_xmltree

public void **tag_xmltree**(RewriteContext hr)

Package sunlabs.brazil.tcl

Provide ways of integrating the TCL scripting language into the Brazil project server.

TCL is an interpreted scripting language written in the Java programming language. Various parts of the Brazil project system can be scripted in TCL. All of the classes in this package provide mechanisms for accessing the TCL language in the Brazil project server context. See *Tcl + Java = A match made for scripting* for an introduction to scripting in the Java language using TCL.

Files in this package depend upon the `tcljava.jar` and `jacl.jar` files, which may be downloaded from tcljava.sourceforge.net

The tests included in the distribution require Jacl to run. They don't yet work using jacl version 1.3, so a version jacl1.2+ is included in the distribution to run the tests. The included version (called tcl.jar) may be used with all classes in this package.

Class TclFilter

sunlabs.brazil.tcl
Class TclFilter

```
java.lang.Object
  |
  +--sunlabs.brazil.tcl.TclFilter
```

All Implemented Interfaces:
Filter, Handler

---

public class **TclFilter**
extends Object
implements Filter

Wrapper for writing FilterHandler filters in TCL. Runs a tcl startup script when the handler is initialized. Any time a request is made, the tcl **filter** proc (which should be defined in the init script) is called, provided with the `Request` object as an argument.

This provides a bare-bones tcl interface. The startup script should provide a friendlier interface.

This handler requires the `tcl.jar` jar file, a version of jacl included in the release.

The following server properties are used:
script
The name of the TCL file sourced on startup. The `init(sunlabs.brazil.server.Server, java.lang.String)` parameters are made available as the global variables `prefix` and `server`.

Constructor Summary	
`TclFilter()`	

Method Summary

byte[]	**filter**(Request request, MimeHeaders headers, byte[] content) Filters the content generated by the wrapped Handler.
boolean	**init**(Server server, String prefix) extract the filter properties.
boolean	**respond**(Request request) We don't need to look at the request.
boolean	**shouldFilter**(Request request, MimeHeaders headers) For now, only filter text/html.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

TclFilter

```
public TclFilter()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

extract the filter properties.

Specified by:

init in interface Handler

Tags copied from interface: Handler

Parameters:

server - The HTTP server that created this Handler. Typical Handlers will use Server.props to obtain run-time configuration information.

prefix - The handlers *name*. The string this Handler may prepend to all of the keys that it uses to extract configuration information from Server.props. This is set (by the Server and ChainHandler) to help avoid configuration parameter namespace collisions.

Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
```

We don't need to look at the request.
Specified by:
respond in interface Handler
Tags copied from interface: Handler
Parameters:
`request` - The `Request` object that represents the HTTP request.
Returns:
`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.
Throws:
IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

shouldFilter

```
public boolean shouldFilter(Request request,
                            MimeHeaders headers)
```

For now, only filter text/html. This restriction will be lifted once I figure out how
Specified by:
shouldFilter in interface Filter
Tags copied from interface: Filter
Parameters:
`request` - The in-progress HTTP request.
`headers` - The MIME headers generated by the wrapped `Handler`.
Returns:
`true` if this filter would like to examine and possibly rewrite the content, `false` otherwise.

---

filter

```
public byte[] filter(Request request,
                     MimeHeaders headers,
                     byte[] content)
```

Description copied from interface: Filter
Filters the content generated by the wrapped `Handler`. The content may be arbitrarily rewritten by this method.

The MIME headers may also be modified by this `Filter`, for instance, to change the "Content-Type" of a web page. The "Content-Length" will automatically be computed by the `FilterHandler`.

Specified by:
filter in interface Filter

Tags copied from interface: Filter

Parameters:
`request` - The finished HTTP request.
`headers` - The MIME headers generated by the `Handler`.
`content` - The output from the `Handler` that this `Filter` may rewrite.

Returns:
The rewritten content. The `Filter` may return the original `content` unchanged. The `Filter` may return `null` to indicate that the `FilterHandler` should stop processing the request and should not return any content to the client.

Class TclHandler

sunlabs.brazil.tcl
Class TclHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.tcl.TclHandler
```

All Implemented Interfaces:
Handler

---

public class **TclHandler**
extends Object
implements Handler

Handler for writing handlers in tcl. Anytime a request is made, call the **respond** callback in the tcl code, which is provided with the `Request` object as an argument.

This provides a bare-bones tcl interface. The startup script, which is sourced once upon startup, should provide a friendlier interface.

One Tcl interpreter is started for each session. The SessionID property of the request is used to choose the session. if no Session ID is available, a single interpreter is used for each request. The interpreter is Initialized the first time it is referenced for a session, or if the "SessionID" variable is NOT set.

This handler requires `tcl.jar`, a version of jacl, included in the release.

The following server properties are used:
callback
: The name of the TCL script to call at each request. Defaults to `respond`.

prefix, suffix, glob, match
: Specify the URL that triggers this handler. (See `MatchString`).

script
: The name of the TCL file sourced on startup. The `init(sunlabs.brazil.server.Server, java.lang.String)` parameters a make available as the global variables `prefix` and `server`.

Constructor Summary

`TclHandler()`	

Method Summary

boolean	**init**(Server server, String prefix) Create a tcl interp, extract the properties, and run the init script
boolean	**respond**(Request request) Responds to an HTTP request.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

TclHandler

```
public TclHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Create a tcl interp, extract the properties, and run the init script

Specified by:

init in interface Handler

Tags copied from interface: Handler

Parameters:

`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.

`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
                throws IOException
```

Description copied from interface: Handler
Responds to an HTTP request.
Specified by:
respond in interface Handler
Tags copied from interface: Handler
Parameters:
`request` - The `Request` object that represents the HTTP request.
Returns:
`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.
Throws:
IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

Class TclRePollHandler

sunlabs.brazil.tcl
Class TclRePollHandler

```
java.lang.Object
  |
  +--java.lang.Thread
        |
        +--sunlabs.brazil.handler.PollHandler
              |
              +--sunlabs.brazil.handler.RePollHandler
                    |
                    +--sunlabs.brazil.tcl.TclRePollHandler
```

All Implemented Interfaces:
Handler, Runnable

---

public class **TclRePollHandler**
extends RePollHandler

Post-process all "polled" properties with tcl code. The following server properties are used:
script
The name of the TCL script to call at each request. The interpreter is created with the following global variables. The script is "evaluted" once on startup.
- **prefix** The handler prefix
- **server** The current server instance
- **logLevel** The current server log level setting
- **argv0** The name of the running script

The tcl procedure "process" is called with the java properties object. Any modifications to that object are done here.
debug
If set, the "script" is sourced each time.

Inner classes inherited from class sunlabs.brazil.handler.RePollHandler
`RePollHandler.Extract`

Fields inherited from class sunlabs.brazil.handler.PollHandler
`interval, post, url`

Fields inherited from class java.lang.Thread
MAX_PRIORITY, MIN_PRIORITY, NORM_PRIORITY

Constructor Summary

TclRePollHandler()	

Method Summary

void	**fillProps**(Properties props, HttpRequest target) Fill the properties from the input stream
boolean	**init**(Server server, String prefix) Create a tcl interp, extract the properties, and run the init script

Methods inherited from class sunlabs.brazil.handler.RePollHandler
processText, respond

Methods inherited from class sunlabs.brazil.handler.PollHandler
run

Methods inherited from class java.lang.Thread
activeCount, checkAccess, countStackFrames, currentThread, destroy, dumpStack, enumerate, getContextClassLoader, getName, getPriority, getThreadGroup, interrupt, interrupted, isAlive, isDaemon, isInterrupted, join, join, join, resume, setContextClassLoader, setDaemon, setName, setPriority, sleep, sleep, start, stop, stop, suspend, toString, yield

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Constructor Detail

TclRePollHandler

```
public TclRePollHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Create a tcl interp, extract the properties, and run the init script

Overrides:

init in class RePollHandler

Tags copied from interface: Handler

Parameters:

`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.

`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

fillProps

```
public void fillProps(Properties props,
                      HttpRequest target)
                throws IOException
```

Fill the properties from the input stream

Overrides:

fillProps in class RePollHandler

Class TclServerTemplate

sunlabs.brazil.tcl
Class TclServerTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.tcl.TclServerTemplate
```

All Implemented Interfaces:
TemplateInterface

---

public class **TclServerTemplate**
extends Template

The `TclServerTemplate` looks for each `<server language="tcl">` (or `<"tcl">`) tag in an HTML page and treats the following data up to the next `</server>` tag as a Tcl script to evaluate. If the optional attribute `eval` is present, the all ${...} constructs are replaced using `Format.subst(java.util.Dictionary, java.lang.String)` before being passed to the Tcl interpreter.

The reason that Tcl scripts are included in an HTML page is usually to generate dynamic, server-side content. After running this template, everything between and including the `<server>` and `</server>` tags is replaced with the result of evaluating the Tcl script as follows:

- Anything printed to the standard output of the Tcl interpreter is added to the HTML document (for instance, `puts "hello"`).
- Additionally, if the Tcl script returns a value, that value is added to the HTML document (for instance, `return "bob"`).

Multiple `puts` and a final `return` can both be used within a single Tcl fragment.

All Tcl fragments within a given page are evaluated in the same Tcl interpreter. The Tcl interpreter actually lives for the entire duration of this `Template` object, so the user can implement persistence across requests.

The following configuration parameters are used to initialize this template.

script
: The name of the Tcl script to evaluate when the interpreter is created. This script only evaluated when the interp is created, not on every request. The variables `prefix` and `server` are set before this file is evaluated, and are references to the parameters passed to a `handler` init method.

root
: The document root, if the script is a relative file name. If the "root" property under the template prefix is not found, the global "root" property is used. If the global "root" property is not found, the current directory is used.

debug
: If this configuration parameter is present, this class replaces the `<server>` and `</server>` tags with comments, so the user can keep track of where the dynamically generated content is coming from

by examining the comments in the resultant HTML document. By default, the `<server>` and `</server>` are completely eliminated from the HTML document rather than changed into comments.

Before evaluating each HTML document, this class variables in the Tcl interpreter, which can be used to interact back with Java to do things like set the response headers:

request
: Exposes the `Request` Java object. It is set anew at each request.

prefix
: Exposes the handler prefix String.

server
: Exposes the handler `Server` object.

SessionId
: Exposes the session id for this interp, or "none".

If a serialized version of this object is reconstituted, the init method must be called again.

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

`TclServerTemplate``()`	

Method Summary

`boolean`	**`done`**`(RewriteContext hr)` Called after the HTML document has been processed.
`boolean`	**`init`**`(RewriteContext hr)` Defer setting up the interpreter until its first use.
`boolean`	**`setup`**`(RewriteContext hr)` Called at the first **tcl** code in the document `TclServerTemplate` is asked to process.
`void`	**`tag_server`**`(RewriteContext hr)` Processes the `<server>` tag.
`void`	**`tag_tcl`**`(RewriteContext hr)` Processes the `<tcl>` tag.

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

TclServerTemplate

`public TclServerTemplate()`

Method Detail

init

`public boolean init(RewriteContext hr)`

Defer setting up the interpreter until its first use. This way we don't initialize a tcl interp (a heavyweight operation) until its actually needed.

Overrides:

init in class Template

---

setup

`public boolean setup(RewriteContext hr)`

Called at the first **tcl** code in the document `TclServerTemplate` is asked to process.

Redirects the standard output of the Tcl interpreter to the resultant HTML document and exposes the `Request` object as a Tcl variable.

The first time this method is called, the initialization script is sourced into the interpreter, based on the configuration properties in the `Request`

Parameters:

`hr` - The request and associated HTML document that will be processed.

Returns:

`true` interpreter was successfully initialized `false` otherwise. About the only way that the initialization could fail would be

1. there was an error sourcing the initialization script.
2. if this class previously evaluated a Tcl fragment that redefined the scalar Tcl variables `request` or `server` as array variables.

If `false` is returned, an error message is logged.

---

done

```
public boolean done(RewriteContext hr)
```

Called after the HTML document has been processed.

Releases the resources allocated by `init`. This method should be called to ensure that the `HtmlRewriter` and all its attendant data structures are not preserved indefinitely (until the next request).

Overrides:

done in class Template

Parameters:

`hr` - The request and associated HTML document that was processed.

Returns:

`true` always, indicating that this document was successfully processed to completion.

---

tag_server

```
public void tag_server(RewriteContext hr)
```

Processes the `<server>` tag. Substitues the result of evaluating the following Tcl script into the resultant HTML document.

Note: Currently, there is no mechanism for other language interpreters to share the same `server` tag. Use the `<tcl>` tag instead.

Parameters:

`hr` - The request and associated HTML document that will be processed.

---

tag_tcl

```
public void tag_tcl(RewriteContext hr)
```

Processes the `<tcl>` tag. Substitues the result of evaluating the following Tcl script into the resultant HTML document.

Parameters:

`hr` - The request and associated HTML document that will be processed.

Package sunlabs.brazil.template

Template classes for use with `sunlabs.brazil.template.TemplateHandler` or `sunlabs.brazil.filter.TemplateFilter` for filtering HTML and XML content. Each class in this package defines a set of methods for converting *new* html tags into standard html. The `sunlabs.brazil.template.Template` interface used to define templates doesn't define the methods used for processing. Instead, the `TemplateHandler` introspects all of the methods in a template class to determine which ones should be called when HTML tags are seen in the input document.

This is an **XML** like capability that is backward compatible with existing HTML practice.

See Summary of Template tags

Class AddHeaderTemplate

sunlabs.brazil.template
Class AddHeaderTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.template.AddHeaderTemplate
```

All Implemented Interfaces:
TemplateInterface

---

public class **AddHeaderTemplate**
extends Template

Template class for adding arbitrary mime headers to a reply. Add a header onto the response, removing the tag from the HTML. `<addheader name1=value1 name2=value2 ...>` where `name` is the name of an HTTP header, and `value` is its value. If no value is provided, then the header is removed. If multiple name/value pairs are provided, they are processed in arbitrary order.

Special headers.
If a "location" header is added, the status code is automatically set to "302". If the value doesn't start with "http://" or "https://", then the value is turned into an absolute URL by prepending the the hostname (and the path from the document root if the value doesn't start with '/'.
Removing location headers is ill-advised.

The special header **status** is used to set the status code.

If the attribute `nocontent` is present, the http header status is set to **204 no content**. This causes the browser to ignore the contents of the page.

The values are subject to ${...} substitutions.
Note: Setting invalid headers will lead to unpredictable results,

---

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

`AddHeaderTemplate()`	

Method Summary

void	**tag_addheader**(RewriteContext hr) Process the special addheader tag.

Methods inherited from class sunlabs.brazil.template.Template
done, init

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

AddHeaderTemplate

public **AddHeaderTemplate**()

Method Detail

tag_addheader

public void **tag_addheader**(RewriteContext hr)

Process the special addheader tag.

Class AllowGtTemplate

sunlabs.brazil.template
Class AllowGtTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.template.AllowGtTemplate
```

All Implemented Interfaces:
TemplateInterface

---

public class **AllowGtTemplate**
extends Template

Template that changes the behavior of the HTML/XML parser to allow unescaped >'s inside of entity attribute values. On the down side, it doesn't deal with unmatched "'s gracefully

By default, the LexML parser does not allow a > inside of an entity body. The parser allows applications to provide ways of allowing embedded >'s, based on whatever syntax they like, This example will ignore a > (as the end of the entity) if inside a quoted attribute value.

No new markup is supported; the behavior of the HTML/XML parsing is altered for the duration of the page. This template is useful primarily to demonstrate how to change the parser token processing.

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

`AllowGtTemplate``()`	

Method Summary

`boolean`	**`init`**`(RewriteContext hr)` Replace the "default" parser with our modified one.

Methods inherited from class sunlabs.brazil.template.Template
done

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

AllowGtTemplate

```
public AllowGtTemplate()
```

Method Detail

init

```
public boolean init(RewriteContext hr)
```

Replace the "default" parser with our modified one.

Overrides:

init in class Template

Class BSLTemplate

sunlabs.brazil.template
Class BSLTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.template.BSLTemplate
```

All Implemented Interfaces:
Serializable, TemplateInterface

---

public class **BSLTemplate**
extends Template
implements Serializable

The `BSLTemplate` takes an HTML document with embedded "BSL" markup tags in it and evaluates those special tags to produce a standard HTML document.

BSL stands for Brazil Scripting Language. BSL can be used to substitute data from the request properties into the resultant document. However, rather than simple property substitution as is provided by the `SetTemplate`, this class provides the ability to iterate over and choose amongst the values substituted with a set of simple flow-control constructs.

BSL uses the following special tags as its language constructs:

- `<if>`
- `<foreach>`
- `<abort>`
- `<break>`
- `<continue>`
- `<extract>`

This template recursively evalutes the bodies/clauses of the BSL commands, meaning that they may contain nested BSL and/or other tags defined by other templates.

The following configuration parameter is used to initialize this template.

`debug`
: If this configuration parameter is present, this template replaces the BSL tags with comments, so the user can keep track of where the dynamically generated content is coming from by examining the comments in the resultant HTML document. By default, the BSL tags are completely eliminated from the HTML document rather than changed into comments.

<if> TAG

The `<if>` tag evaluates one of its clauses dependant upon the value of the provided conditions. The other clauses are not evaluated and do not appear in the resultant HTML document. The general format of the

`<if>` tag is as follows:

```
<if [not] condition>
    clause
<elseif [not] condition>
    clause
<else>
    clause
</if>
```

The `<elseif>` and `<else>` tags are optional, and multiple `<elseif>` tags may be present. `<elseif>` may also be spelled `<elif>` or `<else if>`. The optional parameter `not` reverses the sense of the specified condition.

Following are the formats of the *condition*:

`<if name=var>`
: Test if the value of property *var* is set and is not "", "false", "no", "off", or the number 0.

`<if name=var value=string>`
: Test if the value of property *var* is equal to the given *string*.

`<if name=pattern any>`
: Test if any property exists that matches matches the given `glob` *pattern*. Note: This may be expensive if there are large numbers of properties.

`<if name=var glob=pattern>`
: Test if the value of property *var* matches the given `glob` *pattern*.

`<if name=var match=pattern>`
: Test if the value of property *var* matches the given `regular expression` *pattern*.

  if the attribute `nocase` is present, then a case insensitive match is performed.

`<if expr=numeric expression>`
: The numeric expression is evaluated. If the result is "1", then the condition is satisfied. This uses `Calculator` to evaluate the expression. Any vaiable that is defined, but not "0"m "off" or "no" is considered to have a value of "1" for the purposes of the expression evaluation. This allows (as an example) the expression `"x && y && ! z"` to evaluate to "true" only if the variables "x" and "y", but not "z" are defined *(and not "0", "no" or "false".

Normally, the variable is look-up in the Properties Stack, starting with the Reqest Properties. The attribute "namespace" may be used to look-up the variable in the specified namespace.

<foreach> TAG

The `<foreach>` tag repeatedly evaluates its body a selected number of times. Each time the body is evaluated, the provided named property is set to the next word in the provided list of words. The body is terminated by the `</foreach>` tag. This tag is especially useful for dynamically producing lists and tables.

-

```
<foreach name=var list="value1 value2 ..." [delim="chars"]>
    <get var>
    body
</foreach>
```

Iterate over the set of values "*value1 value2* ...". The named property *var* is assigned each value in turn.

If the optional parameter `delim` specifies, the delimiter for splitting the list into elements.

- If `delim` is a single character, then that character is used as the delimiter.
- If `delim` is not specified or is the empty string "", the delimiter is whitespace.
- Otherwise, the delimeter is taken to be a regular expression. If the regular expression is invalid, the entire list is taken as a single element.
- If the delimiter is a regular expression, and no sorting is requested (I was lazy), in addition to the property *var*, the properties *var.delime*, *var.delim.1* ... are made available that represent the value of the previous delimiter and all of its sub-matches (if any).

.

○

```
<foreach name=var property=property [delim="chars"]>
    <get var>
    body
</foreach>
```

Iterate over the values in the other *property*. The value of the other property is broken into elements and each element is assigned to the named property *var* in turn. This form is equivalent to `<foreach name=var list=${property}>`.

If the optional parameter `delim` is specified, the characters are delimiters for splitting the list into elements using the `StringTokenizer` rules. If `delim` is not specified or is the empty string "", the delimiter is whitespace.

○

```
<foreach name=var glob=pattern>
    <get var.name>
    <get var.value>

    <get var.name.1>
    <get var.name.2>
    body
</foreach>
```

Iterate over all the properties whose name matches the `glob` *pattern*. In turn, the following properties are set:
`var.name` is the name of the property.
`var.value` is the value of the property.
`var.name.1`, `var.name.2`, ... are the substrings matching the wildcard characters in the pattern, if any.

-

```
<foreach name=var match=pattern>
    <get var.name>
    <get var.value>

    <get var.name.0>
    <get var.name.1>
    <get var.name.2>
    body
</foreach>
```

Iterate over all the properties whose name matches the `regular expression` *pattern*. In turn, the following properties are set:
var`.name` is the name of the property.
var`.value` is the value of the property.
var`.name.0` is the substring that matched the whole pattern.
var`.name.1`, *var*`.name.2`, ... are the substrings matching the parenthesized subexpressions, if any.

NOTE: In the current implementation, when there are large numbers of property values, using `glob` is (potentially) much more efficient than `match` for locating names.

All the temporary properties that this tag creates are visible only within the *body* for the duration of the `<foreach>`.

if the attribute `nocase` is present, then a case insensitive match is performed.

Sorting using `foreach`

The <foreach> tag contains a feature to change the order of iteration. This facility is intended for common sorting operations. For general purpose manipulation of the iteration order, the order should be defined either in another handler, or by using the `<server>` directive.

The four additional parameters used to control sorting are:

`reverse`
: The list of items is iterated in the reverse order.

`sort[=`*key*`]`
: The items to be iterated over are sorted. If no *key* is supplied, the items are sorted by the property name. If a *key* is supplied, its value is used as the sort key for the iteration. For this to be meaningful, the key should contain variable substitutions (e.g. ${...}, see `getProperty`). A sample use of the sort key would be:

```
<foreach name=id property=employee.ids sort="${employee.${id}.last}, ${employee.${id}.first}">
```

This option can be tricky to use correctly. The following example will not sort employees by last name:

```
1. <foreach name=id property=employee.ids sort="employee.${id}.last">
```

Why? Because another level of ${...} needs to be inserted in the sort key:

```
2. <foreach name=id property=employee.ids sort="${employee.${id}.last}">
```

Example (1) will just sort the literal strings "employee.1234.last", "employee.5678.last", etc. while example (2) will do the correct thing and sort the values "Stevens" and "Johnson". Remember that BSL sorts based on exactly what you pass it and does not know that the provided string should be treated as another variable itself.

`numeric`

When used in conjunction with the `sort` parameter, it causes the items to be interpreted as numbers (or zero if the item doesn't look like a number).

`nocase`

When used in conjunction with the `sort` parameter, it causes the items to be sorted in a case-insensitive fashion.

Note that when used with `"glob=..."` or `"match=..."`, it does **NOT** cause the glob or regular expression to be case insensitive. It causes the results of the glob or regexp to be sorted in a case-insensitive fashion. There is currently no way to specify a case-insensitive glob or regular expression.

<abort> TAG

The `<abort>` tag terminates processing of the current HTML page at the point it is evaluated. All HTML on the page after the `<abort>` tag is discarded, and the HTML processed up to that point is returned. This tag can be placed anywhere on a page, including within a `<foreach>` or `<if>` construct.

The following is an example usage of this tag:

```
<foreach name=x list="0 1 2 3">
  <if name=x value=3>
    <abort>
  </if>
  <get name=x>
</foreach>
Testing
```

This example produces the output:

```
0 1 2
```

<break> TAG

The `<break>` tag terminates processing within a `<foreach>` construct. The processing of HTML continues immediately after the `</foreach>`. This tag can only be used inside of a `<foreach>` tag.

The following is an example usage of this tag:

```
<foreach name=x list="0 1 2 3">
  <if name=x value=3>
    <break>
  </if>
  <get name=x>
</foreach>
Testing
```

This example produces the output:

```
0 1 2 Testing
```

<continue> TAG

The `<continue>` tag continues processing at the top of a `<foreach>` construct. This skips any HTML after the `<continue>` tag and before the `</foreach>`. This tag can only be used inside of a `<foreach>` tag.

The following is an example usage of this tag:

```
<foreach name=x list="0 1 2 3">
  <if name=x value=2>
    <continue>
  </if>
  <get name=x>
</foreach>
Testing
```

This example produces the output:

```
0 1 3 Testing
```

<extract> TAG

The `<extract>` tag permits portions of a property's value to be extracted into additional properties, based on either glob or regular expression patterns. The extract tag takes the following tag parameters:

`name=`*`var`*

The name of the property whose value will be split up and extracted.

`prepend=`*`base`*

Optional parameter that specifies the *base* string to prepend to the extracted properties. The default value for *base* is the specified property name *var*.

`glob=`*`pattern`*

The `glob` *pattern* to match against. In turn, the following properties are set:
`base``.1`, *`base`*`.2`, ... are the substrings that matched the wildcard characters in the *pattern*.

`match=`*`pattern`*

The `regular expression` *pattern* to match against. In turn, the following properties are set:
`base``.0` is the substring that matched the whole *pattern*.
`base``.1`, *`base`*`.2`, ... are the substrings that matched the parenthesized subexpressions in *pattern*, if any.

If the attribute `all` is present, then all matches and submatches are extracted into properties. The properties *`base`*`.0`, *`base`*`.1`, etc., are set to the 1st matched expression, the 2nd matched expression, etc. The properties *`base`*`.0.0`, *`base`*`.0.1`... are set to the sub-matches of the first full match, and so forth. If any matches are found the following additional properties are set:

`base``matches`
: The number of times the regular expression was matched.

`base``submatches`.
: The number of sub-expressions for this regular expression.

`base``matchelist`.
: The list of matches (e.g. "1 2 3 ...").

`replace=`*`substitution`*
: If specified (with match), then no portions of the value are extracted. Instead, a regular expression substitution is performed (see `substitution`). The resultant substituted value is placed in the property: *prefix*.replace.

`map`
: A white space separated list of names that will be used to name sub-matches, instead of .1, .2, ... etc. See `tag_extract(sunlabs.brazil.template.RewriteContext)` for more detail.

One of `glob` or `match` must be specified.

In addition, the property *`base`*`.matches` is set to a value indicating the number of matches and submatches stored. This property can be examined and compared with `0` to determine if the `<extract>` tag matched at all. If there was no match, the numbered properties *`base`*`.N` are not set or changed from their previous value.

Anytime an argument is specified to one of the BSL tags, variable substitution as described in `getProperty` may be used.

Any time a boolean parameter (XXX) is allowed (nocase, not, numeric, or reverse) it is considered false if it takes any of the forms: XXX=0, XXX=no, XXX=false XXX="". If it takes the forms XXX or XXX="anything else", the value is true.

see a sample HTML page that contains some BSL markup.

See Also:
`SetTemplate`, Serialized Form

Field Summary

static int	**ABORT**
static int	**BREAK**
static int	**CONTINUE**

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

`BSLTemplate``()`	

Method Summary

`boolean`	**`init`**`(RewriteContext hr)` Called before this template processes any tags.
`void`	**`tag_abort`**`(RewriteContext hr)` Handles the "abort" tag.
`void`	**`tag_break`**`(RewriteContext hr)` Handles the "break" tag.
`void`	**`tag_continue`**`(RewriteContext hr)` Handles the "continue" tag.
`void`	**`tag_extract`**`(RewriteContext hr)` Handle the [experimental] "extract" tag.
`void`	**`tag_foreach`**`(RewriteContext hr)` Handles the "foreach" tag.
`void`	**`tag_if`**`(RewriteContext hr)` Handles the "if" tag.

Methods inherited from class sunlabs.brazil.template.Template
`done`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

ABORT

`public static final int` **`ABORT`**

BREAK

`public static final int` **`BREAK`**

CONTINUE

`public static final int` **`CONTINUE`**

Constructor Detail

BSLTemplate

`public` **`BSLTemplate`**`()`

Method Detail

init

`public boolean` **`init`**`(RewriteContext hr)`

Description copied from class: Template
Called before this template processes any tags.
Overrides:
init in class Template

---

tag_abort

`public void` **`tag_abort`**`(RewriteContext hr)`

Handles the "abort" tag.

---

tag_break

`public void **tag_break**(RewriteContext hr)`

Handles the "break" tag.

---

tag_continue

`public void **tag_continue**(RewriteContext hr)`

Handles the "continue" tag.

---

tag_foreach

`public void **tag_foreach**(RewriteContext hr)`

Handles the "foreach" tag.

---

tag_if

`public void **tag_if**(RewriteContext hr)`

Handles the "if" tag.

---

tag_extract

`public void **tag_extract**(RewriteContext hr)`

Handle the [experimental] "extract" tag. This permits parts of a property's value to be extracted into additional properties, based on either glob or regular expression patterns.
<extract name= prepend= glob= match=>

name
: The name of the property to extract

prepend
: The base name for all extracted properties (defaults to "name"). If it doesn't end with a ".", one is added.

glob
: The glob pattern to use for extraction. The text matching each wildcard in the pattern is extracted.

match
: The regular expression pattern to use for extraction. The text matching each sub-expression is extracted. If "glob" is specified, then "match" is ignored.

null
: the value to return if there was no match [or sub-match] (defaults to "").

map
: a white space separated list of names that will be used to name sub-matches, instead of .1, .2, ... etc. If there are more sub expressions than names, then the indeces are used after the names run

out. The example:

```
<set name=entry value="joe:211A:x3321">
<extract name=entry glob=*:*:* map="name room phone">
```

Will return the values:

```
entry.name=joe
entry.room=211A
entry.phone=x3321
```

namespace
Normally, results are extracted into the current request namespace. If *namespace* is specified, then the results are placed into the named namespace. The names "server" and "local" are special (see SetTemplate).

NOTE: The namespace will be accessable by any other templates associated with the same TemplateRunner, using the default sessionTable (see SetTemplate).

Class ChangedTemplate

sunlabs.brazil.template
Class ChangedTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.template.ChangedTemplate
```

All Implemented Interfaces:
Serializable, TemplateInterface

---

public class **ChangedTemplate**
extends Template
implements Serializable

This `Template` adds an icon to HREFs to indicate when the file being referred to is new, changed, or unchanged with respect to the user's session.

In order for the `ChangedTemplate` to work, the following must happen.

- All files whose HREFs should be rewritten must pass through the `ChangedTemplate`. All HREFs seen between `<changed>` and `</changed>` tags will be rewritten so that an appropriate icon appears next to the HREF.
- All files whose last-accessed time is being tracked must also pass through this `ChangedTemplate`. Whenever the `ChangedTemplate` sees a file that was named in some previously seen `<changed>` section, that file's last-accessed time will be updated. Only the files named in a `<changed>` section are tracked.

Warning: The `ChangedTemplate` may have to keep track of a lot of data per session, specifically, the names of all the files being tracked and the last time the user accessed them.

The `ChangedTemplate` examines the property "fileName", set (for example) by the `FileHandler`, in order to update the last-accessed time of a file as it passes by. If the "fileName" property is not set, the last-accessed time will not be updated.

The `ChangedTemplate` also assumes that all local HREFs it sees can be directly translated into the corresponding file name based on the "root" property and the URL of the current file. Getting that file name is necessary so its last-modified time (on disk) can be compared to its last-accessed time (per session).

The `ChangedTemplate` uses the following properties:

fileName
: A request property containing the full path name of the current file, used to keep track of the last time that file was accessed by the current user. A `Handler` or other code may set this property if it wishes the file to be tracked.

root
: The root of the document hierarchy. An HREF must resolve to a file in this hierarchy so its last-modified time can be checked. If the file does not exist, the HREF will not be rewritten.

always
: If this property is present, the `ChangedTemplate` always rewrites the HREFs, instead of just when they appear within the `<changed>` and `</changed>` tags.

new
: The HTML to substitute into the document if the HREF refers to a file that has never been accessed by the user. If absent, the HREF for new files will not be rewritten.

changed
: The HTML to substitute into the document if the HREF refers to a file that has changed since the last time it was accessed by the user. If absent, the HREF for changed files will not be rewritten.

unchanged
: The HTML to substitute into the document if the HREF refers to a file that has not changed since the last time it was accessed by the user. If absent, the HREF for unchanged files will not be rewritten.

See Also:
Serialized Form

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

`ChangedTemplate``()`	

Method Summary

`boolean`	**`done`**`(RewriteContext hr)` Called after all tags have been processed, one final chance.
`boolean`	**`init`**`(RewriteContext hr)` Records that this file has just been accessed.
`void`	**`tag_a`**`(RewriteContext hr)`
`void`	**`tag_changed`**`(RewriteContext hr)`
`void`	**`tag_slash_changed`**`(RewriteContext hr)`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

ChangedTemplate

`public ChangedTemplate()`

Method Detail

init

`public boolean init(RewriteContext hr)`

Records that this file has just been accessed.
Overrides:
init in class Template

---

done

`public boolean done(RewriteContext hr)`

Description copied from class: Template
Called after all tags have been processed, one final chance.
Overrides:
done in class Template

---

tag_changed

`public void tag_changed(RewriteContext hr)`

---

tag_slash_changed

`public void tag_slash_changed(RewriteContext hr)`

---

tag_a

```
public void tag_a(RewriteContext hr)
```

Class ContentTemplate

sunlabs.brazil.template
Class ContentTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.template.ContentTemplate
```

All Implemented Interfaces:
TemplateInterface
Direct Known Subclasses:
ExpContentTemplate

---

public class **ContentTemplate**
extends Template

Template class for extracting content out of remote html pages. This class is used by the TemplateHandler, for extracting the "content" out of html documents for later integration with a look-and-feel template using one or more of: `SetTemplate`, `BSLTemplate`, or `ReplaceFilter`, The plan is to snag the title and the content, and put them into request properties. The resultant processed output will be discarded. The following properties are gathered:

title
: The document title

all
: The entire content

bodyArgs
: The attributes to the body tag, if any

content
: The body, delimited by content.../content>. The text inside multiple <content> ... </content> pairs are concatenated together.

script
: All "<script>"..."</script>" tags found in the document head

style
: All "<style">..."</style"> tags found in the document head

meta-[name]
: Every meta tag "name" and "content"

link-[rel]
: Every link tag "rel" and "href"

user-agent
: The origin user agent

referer
: The user agent referrer (if any)

last-modified
: The document last modified time (if any) in std format

content-length
The document content length, as fetched from the origin server
Properties:
prepend
Prepend this string to the property names define above, that are populated by this template. (defaults to "").

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

`ContentTemplate()`	

Method Summary

boolean	**done**(RewriteContext hr) Extract useful properties out of the http mime headers.
boolean	**init**(RewriteContext hr) Called before this template processes any tags.
void	**tag_body**(RewriteContext hr) Grab the "body" attributes, and toss all output to this point.
void	**tag_content**(RewriteContext hr) toss everything up to and including here, but turn on content accumulation.
void	**tag_link**(RewriteContext hr) Extract data out of link tags into the properties.
void	**tag_meta**(RewriteContext hr) Extract data out of meta tags into the properties.
void	**tag_script**(RewriteContext hr) Append all "script" code while in the head section.
void	**tag_slash_body**(RewriteContext hr) If no content tags are present, use the entire "body" instead.
void	**tag_slash_content**(RewriteContext hr) Save the content gathered so far, and turn off content accumulation.
void	**tag_slash_head**(RewriteContext hr) Mark end of head section.
void	**tag_slash_title**(RewriteContext hr) Gather up the title - no tags allowed between title ....
void	**tag_style**(RewriteContext hr) Append all "style" code while in the head section.
void	**tag_title**(RewriteContext hr) Toss everything up to and including this entity.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ContentTemplate

`public ContentTemplate()`

Method Detail

init

`public boolean init(RewriteContext hr)`

Description copied from class: Template
Called before this template processes any tags.
Overrides:
init in class Template

---

tag_title

`public void tag_title(RewriteContext hr)`

Toss everything up to and including this entity.

---

tag_slash_title

`public void tag_slash_title(RewriteContext hr)`

Gather up the title - no tags allowed between title .... /title.

---

tag_script

`public void tag_script(RewriteContext hr)`

Append all "script" code while in the head section.

---

tag_style

`public void tag_style(RewriteContext hr)`

Append all "style" code while in the head section.

---

tag_slash_head

`public void tag_slash_head(RewriteContext hr)`

Mark end of head section. All "script" content in the "body" is left alone.

---

tag_content

`public void **tag_content**(RewriteContext hr)`

toss everything up to and including here, but turn on content accumulation.

---

tag_body

`public void **tag_body**(RewriteContext hr)`

Grab the "body" attributes, and toss all output to this point.

---

tag_slash_content

`public void **tag_slash_content**(RewriteContext hr)`

Save the content gathered so far, and turn off content accumulation.

---

tag_slash_body

`public void **tag_slash_body**(RewriteContext hr)`

If no content tags are present, use the entire "body" instead.

---

tag_meta

`public void **tag_meta**(RewriteContext hr)`

Extract data out of meta tags into the properties. For "http-equiv" tags, set the corrosponding http respones header.

---

tag_link

`public void **tag_link**(RewriteContext hr)`

Extract data out of link tags into the properties. Prefix the "rel" attribute with "link-" to use as the property name.

---

done

`public boolean **done**(RewriteContext hr)`

Extract useful properties out of the http mime headers.
Overrides:
done in class Template

Class DeCommentTemplate

sunlabs.brazil.template
Class DeCommentTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.template.DeCommentTemplate
```

All Implemented Interfaces:
TemplateInterface

---

public class **DeCommentTemplate**
extends Template

Template class for removing comments from html pages. Properties:
disable
if true, disable comment removal. This is checked at each request.

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

`DeCommentTemplate``()`	

Method Summary

`void`	**`comment`**`(RewriteContext hr)`
`boolean`	**`init`**`(RewriteContext hr)` Called before this template processes any tags.

Methods inherited from class sunlabs.brazil.template.Template
`done`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

DeCommentTemplate

`public DeCommentTemplate()`

Method Detail

init

`public boolean init(RewriteContext hr)`

Description copied from class: Template
Called before this template processes any tags.
Overrides:
init in class Template

---

comment

`public void comment(RewriteContext hr)`

Class DebugTemplate

sunlabs.brazil.template
Class DebugTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.template.DebugTemplate
```

All Implemented Interfaces:
Serializable, TemplateInterface

---

public class **DebugTemplate**
extends Template
implements Serializable

Template class for printing stuff to stderr (for template debugging). This class is used by the TemplateHandler.

A new HTML tag, `<debug>` is defined. Any text between the `<debug` and `>` is printed on stderr, along with the session id and the url. Variable substitutions of the form ${...} are performed on the text.

The property `debug` must be present for this template to function. Otherwise, all `debug` tags are removed.
`debug`
If this configuration parameter is true, debugging is enabled.

See Also:
Serialized Form

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

`DebugTemplate()`	

Method Summary

boolean	**init**(RewriteContext hr) Called before this template processes any tags.
void	**tag_debug**(RewriteContext hr)

Methods inherited from class sunlabs.brazil.template.Template
done

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

DebugTemplate

public **DebugTemplate**()

Method Detail

init

public boolean **init**(RewriteContext hr)

Description copied from class: Template
Called before this template processes any tags.
Overrides:
init in class Template

---

tag_debug

public void **tag_debug**(RewriteContext hr)

Class DirectoryTemplate

sunlabs.brazil.template
Class DirectoryTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.template.DirectoryTemplate
```

All Implemented Interfaces:
Handler, TemplateInterface

---

public class **DirectoryTemplate**
extends Template
implements Handler

Put current directory information (based on the URL) into the request properties. The `<filelist>` tag, if present in the document, triggers the generation of a directory and file listing, based on the current URL.

Template Properties:

prepend
: String to prepend to the properties "Directories" and "Files" That contain the directory and file lists respectively. Defaults to the Templates properties prefix.

delimiter
: Delimiter character to separate entries, defaults to " ".

directory
: The directory to use instead of the one implied by the URL. If it starts with "/", then it is resolved relative to the document root, otherwise it is resolved relative to the directory implied by the URL. The document root is found in the porperty `"[prefix].root"`, or in `"root"`, or (if neither exists), the current directory of the server.

DirectoryName
: If set (usually by an upstream handler, such as the FileHandler, or TemplateHandler), this is used as the directory name instead of deriving it from the URL.

debug
: if set, a comment is emitted indicating where the file-list entitiy was encountered.

[prepend].Directories
: List of sub-directories in current directory is set by this template.

[prepend].Files
: List of files with valid suffixes in current directory. is set by this template.

mime.xxx
: An indication that suffix "xxx" is valid. Only valid file names are returned. See `FileHandler` for a description of how to set mime types for url suffixes.

select
: Specifies a "glob" pattern to restrict the names of files and directories returned. The form "![pattern]" selects the inverse of the glob pattern.

stats
: If specified, then for each file, the properties: **[prepend].[file].mod** and **[prepend].[file].size** are set, containing the file last modified time (seconds since epoch) and size (bytes) respectively.

This class may also be used as a handler, in which case the property `prefix` is used to match the leading portion of a URL.

The tag takes the optional parameters: `prepend` and `stats` that override the corrosponding request properties (above). In addition, the attribute `dir` may be used to select an alternate directory for files.

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

`DirectoryTemplate``()`	

Method Summary

`static void`	**`getFiles`**`(String dir, String glob, Request request, String prefix)` Generate properties containing the files and directories in the "current" directory.
`boolean`	**`init`**`(RewriteContext hr)` Reset at each page
`boolean`	**`init`**`(Server server, String prefix)` Initializes the handler.
`boolean`	**`respond`**`(Request request)` Compute the directory info, and add it to the request properties.
`void`	**`tag_filelist`**`(RewriteContext hr)` Turn on the directory calculator.

Methods inherited from class sunlabs.brazil.template.Template
`done`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

DirectoryTemplate

```
public DirectoryTemplate()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.
Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
```

Compute the directory info, and add it to the request properties.
Specified by:
respond in interface Handler
Tags copied from interface: Handler

Parameters:

`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

init

```
public boolean init(RewriteContext hr)
```

Reset at each page

Overrides:

init in class Template

---

tag_filelist

```
public void tag_filelist(RewriteContext hr)
```

Turn on the directory calculator. The presense of this tag causes the files and subdirectories in the current directory to be added to the request properties.

The attribute "stats" may be specified to enable additional statistics, overriding the request properties.

---

getFiles

```
public static void getFiles(String dir,
                            String glob,
                            Request request,
                            String prefix)
```

Generate properties containing the files and directories in the "current" directory. The current directory is taken from the "DirectoryName" request property, or derived from the URL. This functionality was culled from the FileHandler and the Directory Handler.

If "directory" is specified (e.g. not null), then it is used as the directory instead. If "directory" starts with "/" then the directory is resolved relative to the document root, otherwise it is resolved relative to the current directory. If "select" is specified, then only files or directories matching the supplied **glob** pattern are selected. If the first character of "select" is "!", then the sense of the glob pattern is inverted.

Class FormClientTemplate

sunlabs.brazil.template
Class FormClientTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.template.FormClientTemplate
```

All Implemented Interfaces:
TemplateInterface

---

public class **FormClientTemplate**
extends Template

[*Deprecated, use the FormTemplate and SetTemplate instead*]
SAMPLE Template class for substituting Default values into html forms The data is retrieved from the client, and sent back to the client later on. This will be used for e-business cards stored on java rings/cards. This template also incorporates the functionallity of the PropsTemplate, as the current scheme doesn't allow composition of template handler classes (at least for now). If a URL contains query data, and the value of the server property "uploadContains" occurs in the URL, then all of the query data is saved in the server on behalf of the client. IF no "uploadContains" string is set, all query data is saved on the server. The following Html entities processed by this class:

input
: if the "name" attribute has saved data, the value attribute is replaced by the caved value. This allows default form values to be replaced by previous data submitted to the server by the client.

property
: This tag is replaced by the value of a server property.

subst
: This tag is replaced by the value previously uploaded to the the server from the client for this value.

tag, /tag
: Inserts a "<" or ">" respectively, allowing parameters to be substituted inside of other entities. For example, suppose the client uploaded the value **HOME** as `http://my.home.com/`, and the server withes to create a link to that page. The template fragment: "<a href=<subst HOME>gt; won't work, as nested entities are not allowed. Instead, try: "<tag>a href=<subst HOME>lt;tag>

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

FormClientTemplate()	

Method Summary

boolean	**init**(RewriteContext hr) Save a reference to our request properties.
void	**tag_input**(RewriteContext hr) Look for , and replace the value with the entry in the previously saved client data.
void	**tag_property**(RewriteContext hr) Convert the html tag "property" in to the request's property.
void	**tag_slash_tag**(RewriteContext hr)
void	**tag_subst**(RewriteContext hr) Tag to do substitution of previously uploaded data.
void	**tag_tag**(RewriteContext hr) Using the current scheme, there is no easy way to substitute into a tag parameter.

Methods inherited from class sunlabs.brazil.template.Template
done

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

FormClientTemplate

```
public FormClientTemplate()
```

Method Detail

init

```
public boolean init(RewriteContext hr)
```

Save a reference to our request properties. If the URL contains the upload string, save all of the query parameters on behalf of the user.

Overrides:

init in class Template

---

tag_input

```
public void tag_input(RewriteContext hr)
```

Look for , and replace the value with the entry in the previously saved client data.

Parameters:

`h` - The attribute/value pairs for this entity.

---

tag_property

```
public void tag_property(RewriteContext hr)
```

Convert the html tag "property" in to the request's property.

Parameters:

`h` - Hashtable containing tag parameters "name" The property name "default" a default value, if no name available

---

tag_subst

```
public void tag_subst(RewriteContext hr)
```

Tag to do substitution of previously uploaded data.

Parameters:

`name` - The name of the token to replace with client data.

---

tag_tag

```
public void tag_tag(RewriteContext hr)
```

Using the current scheme, there is no easy way to substitute into a tag parameter. So we'll invent a "magic" tag (called tag) that will allow us to create entities dynamically

---

tag_slash_tag

```
public void tag_slash_tag(RewriteContext hr)
```

Class FormTemplate

sunlabs.brazil.template
Class FormTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.template.FormTemplate
```

All Implemented Interfaces:
Serializable, TemplateInterface

---

public class **FormTemplate**
extends Template
implements Serializable

Template class for substituting default values into html forms. This class is used by the TemplateHandler. The default values in form elements are replaced by the request property that matches the field name. The following field elements are processed:

- <input name=*x* value=*y*>
- <input type=input name=*x* value=*y*>
- <input type=radio name=*x* value=*y*>
- <option value=*x*>

In all cases, the `value` attribute must be present. additional information is provided below.

If the enclosing <form> tag has the attribute "prepend", then "prepend" is tacked on the front of each variable name before its value is looked-up. The "prepend" attribute is then removed from the form tag.

See Also:
Serialized Form

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

`FormTemplate()`	

Method Summary	
boolean	**done**(RewriteContext hr) This is for debugging only !!
boolean	**init**(RewriteContext hr) Save a reference to our request properties.
void	**tag_form**(RewriteContext hr) Look for a "prepend" attrubute, remember its value, then remove it from the tag.
void	**tag_input**(RewriteContext hr) Look for <input name=[x] value=[v]> and replace the value with the entry in the request properties.
void	**tag_option**(RewriteContext hr) Look at the option tag, set the "selected" attribute as needed.
void	**tag_select**(RewriteContext hr) Remember the variable name for the next group of option tags.
void	**tag_slash_form**(RewriteContext hr) Forget about the "prepend" value
void	**tag_slash_select**(RewriteContext hr) Forget the variable name for the next group of option tags

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

FormTemplate

public **FormTemplate**()

Method Detail

init

```
public boolean init(RewriteContext hr)
```

Save a reference to our request properties.
Overrides:
init in class Template

---

tag_form

```
public void tag_form(RewriteContext hr)
```

Look for a "prepend" attrubute, remember its value, then remove it from the tag.

---

tag_slash_form

```
public void tag_slash_form(RewriteContext hr)
```

Forget about the "prepend" value

---

tag_input

```
public void tag_input(RewriteContext hr)
```

Look for <input name=[x] value=[v]> and replace the value with the entry in the request properties. If no value is supplied, no substitution is done. If `value` contains any ${..} constructs, the substituted value is used instead of the value in the corrosponding request property.

---

tag_select

```
public void tag_select(RewriteContext hr)
```

Remember the variable name for the next group of option tags.

---

tag_slash_select

```
public void tag_slash_select(RewriteContext hr)
```

Forget the variable name for the next group of option tags

---

tag_option

```
public void tag_option(RewriteContext hr)
```

Look at the option tag, set the "selected" attribute as needed. In order for this to work, the VALUE tag *must* be used Do ${...} substitutions on the value.

---

done

```
public boolean done(RewriteContext hr)
```

This is for debugging only !!

Overrides:

done in class Template

Class HighlightTemplate

sunlabs.brazil.template
Class HighlightTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.template.HighlightTemplate
```

All Implemented Interfaces:
Serializable, TemplateInterface

---

public class **HighlightTemplate**
extends Template
implements Serializable

Template class for highlighting text that matches a regular expression. All text between html/xml entities is matched against a regular expression. For each portion of text that matches the expression, a pair of html/xml tags is added on either side of all matched text. Highlighting is automatically turned off inside of head, script, style, and server tags.

Properties. These are recomputed for every page that `highlight` changes.

highlight
: A regular expression that with match any text between entities.

tag
: the html/xml tag pair that will be added before and after all text maching "highlight", above. The default is " .....

options
: the set of name=value options that will be added to the starting tag of the tag pair, above. The default is "color=red".

matchCase
: If specifies, matches are case sensitive. The default is to ignore case when matching.

substitute
: The string to substitute for the matched text. This is for advanced uses. If specified, the values for `tag` and `options` are ignored. The default is: <${tag} ${options}>&</${tag}> The format of the string is a regular expression substitution string, which supports ${} style variable substitutions from the request properties.

mustHighlight
: If not set, the entire document is surrounded by implicit `highlight` tags. If set no highlighting will take place until an actual `highlight` tag is present.

exit
: If set, the template "init" method will return false, and no further processing will take place. This is useful if this template is used by itself.

The following html tags are processed:
highlight
/highlight
Only text between these tags is considered for highlighting.
nohighlight
/nohighlight
Temporarily undoes the effect of a `highlight` tag. In the current implementation, `highlight` and `nohighlight` don't nest.

See Also:
Serialized Form

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

`HighlightTemplate()`	

Method Summary	
boolean	**init**(RewriteContext hr) This gets called at every page, at the beginning.
void	**string**(RewriteContext hr) Gets all text between tags - highlighting it appropriately.
void	**tag_head**(RewriteContext hr) Don't do highlight inside the following sections
void	**tag_highlight**(RewriteContext hr) The special entities `highlight` and `nohighlight` may be used to turn highlighting on or off in certain areas.
void	**tag_nohighlight**(RewriteContext hr)
void	**tag_script**(RewriteContext hr)
void	**tag_server**(RewriteContext hr)
void	**tag_slash_head**(RewriteContext hr)
void	**tag_slash_highlight**(RewriteContext hr)
void	**tag_slash_nohighlight**(RewriteContext hr)
void	**tag_slash_script**(RewriteContext hr)
void	**tag_slash_server**(RewriteContext hr)
void	**tag_slash_style**(RewriteContext hr)
void	**tag_style**(RewriteContext hr)

Methods inherited from class sunlabs.brazil.template.Template
done

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

HighlightTemplate

`public HighlightTemplate()`

Method Detail

init

`public boolean init(RewriteContext hr)`

This gets called at every page, at the beginning. If this is our first time, get the config stuff out of the request properties.

Overrides:

init in class Template

---

string

`public void string(RewriteContext hr)`

Gets all text between tags - highlighting it appropriately. To restrict the tag set, define the entities and set the shouldHighlight flag appropriately.

---

tag_head

`public void tag_head(RewriteContext hr)`

Don’t do highlight inside the following sections

---

tag_slash_head

`public void tag_slash_head(RewriteContext hr)`

---

tag_script

```
public void tag_script(RewriteContext hr)
```

---

tag_slash_script

```
public void tag_slash_script(RewriteContext hr)
```

---

tag_style

```
public void tag_style(RewriteContext hr)
```

---

tag_slash_style

```
public void tag_slash_style(RewriteContext hr)
```

---

tag_server

```
public void tag_server(RewriteContext hr)
```

---

tag_slash_server

```
public void tag_slash_server(RewriteContext hr)
```

---

tag_highlight

```
public void tag_highlight(RewriteContext hr)
```

The special entities `highlight` and `nohighlight` may be used to turn highlighting on or off in certain areas.

---

tag_slash_highlight

```
public void tag_slash_highlight(RewriteContext hr)
```

---

tag_nohighlight

```
public void tag_nohighlight(RewriteContext hr)
```

---

tag_slash_nohighlight

```
public void tag_slash_nohighlight(RewriteContext hr)
```

Class IncludeTemplate

sunlabs.brazil.template
Class IncludeTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.template.IncludeTemplate
```

All Implemented Interfaces:
TemplateInterface

---

public class **IncludeTemplate**
extends Template

Convert the html tag "include" in to text for an included html page. Attributes processed

href
: Absolute url to fetch, and insert here

post
: Post data if any. If set, an http POST is issued. The post data is expected to be in www-url-encoded format.

alt
: Text to insert if URL can't be obtained.

name
: The name of the variable to put the result in. If this is specified, the content is not included in place. The template prefix is automatically prepended to the name.

proxy
: The proxy:port to use as a proxy (if any). If specified, it overrides the `proxy` property, in `request.props`.

addheaders
: A white space delimited set of token names that represent additional http headers to add to the target request. For each token, the values *[token].name* and *[token].value* in the `request.props` are used for the header name and value respectively.

encoding
: The character encoding to use if it can't be automatically determined. Defaults to the default platform encoding.

getheaders
: The name of the variable prefix to use to extract the http response headers. If this not specified, no response headers are retrieved. The result will be properties of the form: `[prefix].[getheaders].[header_name]=[header_value]`. If multiple entries exist for a particular header name, the values are combined as per HTTP conventions (e.g. v1, v2, ... vn). The pseudo headers `status` and `encoding` will contain the http status line and charset encoding, respectively.

error
: If *name* is specified and the result could not be obtained, the error message is placed here.

queryList
A list of property names whose names and values will be added to the url as query parameters. The values are properly url-encoded.

Example: `<include href=http://www.foo.com/index.html>`

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

`IncludeTemplate``()`	

Method Summary

`void`	**`comment`**`(RewriteContext hr)` Treat comments of the form: `<!#include ...>` as if they were include tags.
`void`	**`tag_include`**`(RewriteContext hr)`

Methods inherited from class sunlabs.brazil.template.Template
`done, init`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

IncludeTemplate

```
public IncludeTemplate()
```

Method Detail

tag_include

```
public void tag_include(RewriteContext hr)
```

---

comment

```
public void comment(RewriteContext hr)
```

Treat comments of the form: `<!#include ...>` as if they were include tags.

Class MacroTemplate

sunlabs.brazil.template
Class MacroTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.template.MacroTemplate
```

All Implemented Interfaces:
TemplateInterface

---

public class **MacroTemplate**
extends Template

Template class for defining macros. Macros are defined by:

```
<definemacro name=macro-name [global=true|false]> .... </definemacro>
```

The text (.....) has any leading and trailing whitespace removed.

To expand a macro: <macro-name name1=value1 ...>. All ${...} constructs in The previously saved macro body are processed, and the result replaces the `macro-name` tag. Attributes provided in the macro name override any variables that exist in the request properties.

Several special variables are supplied, as if they were specified as attributes, but only if otherwise not already defined either as an attribute or as a property,

`isSingleton`
is set to "true" or "false" to indicate the tag has been specified as a singleton.

`args`
is the string of un-parsed arguments to this macro.

Templates are processed by reading the input "document" a tag at a time to generate the output document. By default, the macro body is pushed onto the not-yet-processed input stream; any tags contained in the macro body will be processed. If the `defer` attribute is present, the macro body is placed onto the output stream instead. This is more efficient, but requires an additional filter pass if the body of the macro contains tags that need to be processed in the current context.

init
The name of the file (or resource) to read a default set of macro definitions from. If an absolute path isn't specified, the file is taken relative to the document root. The default macros are kept in the SessionManager on a per-server basis. All macros defined in the "init" file are global. All markup in this file outside of a macro definition is ignored. If "subst" is present as an attribute of a macro definition in this file, then all ${...} are evaluated relative to "server.props" before the macro is defined.

subst
: If specified, then any tags that are not processed by any templates will have all ${..} contructs in attribute values substituted. This subsumes the function of the "SubstAllTemplate".

This is an experiment. The current implementation is flawed, although it is less flawed than the previous one.

NOTE:
The init files in previous releases used java properties format files to define macros; This version uses xml templates. See "main" below for a utility to convert the old properties format files to the new format.

Field Summary

Properties	**macroTable**

Fields inherited from class sunlabs.brazil.template.Template
debug

Constructor Summary

MacroTemplate()	

Method Summary

void	**defaultTag**(RewriteContext hr) Run the macro, push formal parameters on the stack first.
boolean	**init**(RewriteContext hr) Read in the inital macros, if needed.
static void	**main**(String[] args) Convert stdin properties format macro definition files to the new template style.
static String	**snarfTillClose**(LexML lex, String tag) Grab all the markup starting from the current tag until the matching closing tag, and return as a string.
void	**tag_definemacro**(RewriteContext hr) Define a new macro.

Methods inherited from class sunlabs.brazil.template.Template
done

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

macroTable

public Properties **macroTable**

Constructor Detail

MacroTemplate

public **MacroTemplate**()

Method Detail

init

```
public boolean init(RewriteContext hr)
```

Read in the inital macros, if needed.
Overrides:
init in class Template

---

snarfTillClose

```
public static String snarfTillClose(LexML lex,
                                    String tag)
```

Grab all the markup starting from the current tag until the matching closing tag, and return as a string.

---

tag_definemacro

```
public void tag_definemacro(RewriteContext hr)
```

Define a new macro. use "name" as the macro name. Once defined, the tag `<name ...>` will be replaced by the contents of the macro named "name".

```
<definemacro name=nnn [global=true|false]>...</definemacro>
```

- If "global" is true, then the macro is defined for the entire server, otherwise it is for the session only.
- if the body of the macro is an empty string, the macro definition is removed
- Macros defined for the session override the global ones.
- `<definemacro nnn>` is a shortcut for `<definemacro name=nnn>`

---

defaultTag

```
public void defaultTag(RewriteContext hr)
```

Run the macro, push formal parameters on the stack first. This examimes all non-processed tags to see if they are macros, and processes those that are.

If the parameter `defer=true` is present, the text of the macro is output directly, with only ${..} substitutions performed. Otherwise, the markup in the macro body is rescanned and processed.

---

main

```
public static void main(String[] args)
                 throws IOException
```

Convert stdin properties format macro definition files to the new template style. Use this to update to the new initial macro template format.

Class ModifiedTemplate

sunlabs.brazil.template
Class ModifiedTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.template.ModifiedTemplate
```

All Implemented Interfaces:
Serializable, TemplateInterface

---

public class **ModifiedTemplate**
extends Template
implements Serializable

Template class for computing `last-modified` times for content that is processed through templates.

For traditional web content that is stored in a file, it is easy to keep track of the last time the content changed, simply by looking at the modify-time attribute of the file. Many browsers (and caches) use this information, obtained from the `last-modified` http header to determine whether to use an existing copy of the document, or a cached copy.

When the content is dynamically transformed, however, the last modified time is more complex: a combination of the original file modification time combined with the last change made to the transformation parameters.

A new HTML tag, `<modified>` is defined, and is intended to be used in conjunction with the BSLTemplate. When present in a page, the `last-modified` time of the transformation is set to the server's current time. When the content is delivered to the client, the `last-modified` header is set to the more recent of the origin last-modified time and the transformation modified time.

Properties:
debug
If this configuration parameter is present, `modified` tag is replaced by a comment. Otherwise it is removed from the document.

See Also:
Serialized Form

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

ModifiedTemplate()	

Method Summary

boolean	**done**(RewriteContext hr) Compute the http last modified value by comparing the origin last-modified value (if any) with the transform value
void	**tag_modified**(RewriteContext hr) Set the content transformation modifiy time to NOW

Methods inherited from class sunlabs.brazil.template.Template
init

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ModifiedTemplate

public **ModifiedTemplate**()

Method Detail

tag_modified

```
public void tag_modified(RewriteContext hr)
```

Set the content transformation modifiy time to NOW

---

done

```
public boolean done(RewriteContext hr)
```

Compute the http last modified value by comparing the origin last-modified value (if any) with the transform value

Overrides:

done in class Template

Class MultipartSetTemplate

sunlabs.brazil.template
Class MultipartSetTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.template.SetTemplate
              |
              +--sunlabs.brazil.template.MultipartSetTemplate
```

All Implemented Interfaces:
Handler, Serializable, TemplateInterface

---

public class **MultipartSetTemplate**
extends SetTemplate

Version of the SetTemplate that reflects form/multipart data in Request.props. For ordinary forms, the values placed into `request.props` are the same as for forms of type `www-url-encoded`, or `method=get`, as long as the `query` option is set.

For file input fields (e.g. `<input type=file...>`), the file content is associated with the field *name*, and the properties *name*.`filename`, *name*.`type`, and *name*.`encoding` are set to to the name of the file uploaded, its type, and (unless `noEncode` is set), the encoding, which is either none (for text files), or Base64.

The file contents are automatically Base64 encoded for binary files.

Properties:
query
If present, The form data is translated from form/multipart and placed into the request properties, prefixed by the value of `query`.
noEncode
If present, no encoding is performed on file uploads.
savePattern=[glob pattern]
If set, then the form is scanned for field names that match *glob pattern*. If a match is found, then the *next* form element of type *file* is saved to a file in the document root instead of being loaded as a property. The name of the file is specified by the value of the `saveName` entry.
saveName=`name`
The name to use to save the file. May contain ${...} substitutions. The variables ${fileName}, ${fieldName) and ${prefix} may be used here as "special" variables to make creating a file name easier. `saveName`defaults to: `${prefix}-${fieldName}-${fileName}`
[This has only been tested with Netscape Navigator and Mozilla.]

See Also:
Serialized Form

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

`MultipartSetTemplate``()`	

Method Summary

`boolean`	**`init`**`(RewriteContext hr)` Chain the session-id properties into the request chain, if there are any.
`void`	**`tag_get`**`(RewriteContext hr)` Replace the tag "get" with the value of the variable specified by the "name" attribute.
`void`	**`tag_import`**`(RewriteContext hr)` Import all the data from the named namespace.
`void`	**`tag_property`**`(RewriteContext hr)` Convert the html tag "property" in to the request's property DEPRECATED - use "get"
`void`	**`tag_set`**`(RewriteContext hr)` Set the value of a variable.
`void`	**`tag_slash_tag`**`(RewriteContext hr)` Insert a literal ">"
`void`	**`tag_tag`**`(RewriteContext hr)` Insert a literal "<".

Methods inherited from class sunlabs.brazil.template.SetTemplate
`file2path, init, respond, tag_namespace, tag_slash_namespace`

Methods inherited from class sunlabs.brazil.template.Template
done

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

MultipartSetTemplate

```
public MultipartSetTemplate()
```

Method Detail

init

```
public boolean init(RewriteContext hr)
```

Description copied from class: SetTemplate
Chain the session-id properties into the request chain, if there are any. If "query" or "headers" are requested for "get", then add in those properties to request.props.
Overrides:
init in class SetTemplate

---

tag_set

```
public void tag_set(RewriteContext hr)
```

Description copied from class: SetTemplate
Set the value of a variable. Allow variable substitutions in the name and value. Don't create property tables needlessly.

Attributes:
name
The name of the variable to set
value
The value to set if to.

namespace
The namespace to look in. By default, the variable is set in the namespace associated with the current "SessionId".
local
A (deprecated) alias for "namespace=local", or the current request.props.
Overrides:
tag_set in class SetTemplate

---

tag_property

```
public void tag_property(RewriteContext hr)
```

Description copied from class: SetTemplate
Convert the html tag "property" in to the request's property DEPRECATED - use "get"
Overrides:
tag_property in class SetTemplate

---

tag_get

```
public void tag_get(RewriteContext hr)
```

Description copied from class: SetTemplate
Replace the tag "get" with the value of the variable specified by the "name" attribute.

Attributes:
name
The name of the variable to get
namespace
The namespace to look in. By default, the variable is searched for in "request.props". The namespace "server" is used to look in the server's namespace. The namespace "local" is a synonym for the default namespace.
default
The value to use if no value matches "name".
convert
The conversion to perform on the value of the data before substitution: "html", "url", or "none" (the default). For "html", any special html syntax is escaped. For "url", the data will be suitable for transmission as an http URL.
max
Truncate the String to at most `max` characters. Max must be at least one, and truncation occurs after any conversions.
set
The resultant value is placed into the request property named by the `set` attribute, and not inserted into the HTML stream. If none of "namespace", "convert", or "match" is used, then this simply copies the property from one name to another.
If a single attribute is specified, with no "=", then is is taken to be the "name" parameter. Thus: `<get foo>` is equivalent to: `<get name="foo">`.

Overrides:
tag_get in class SetTemplate

---

tag_import

```
public void tag_import(RewriteContext hr)
```

Description copied from class: SetTemplate
Import all the data from the named namespace. The namespace associated with the session ID is imported automatically for backward compatibility. If the namespace doesn't exist, don't create it now, but remember it needs to be "Chained" if it is created on this page.
Overrides:
tag_import in class SetTemplate

---

tag_tag

```
public void tag_tag(RewriteContext hr)
```

Description copied from class: SetTemplate
Insert a literal "<". Using the current scheme, there is no easy way to substitute into a tag parameter. So we'll invent a "magic" tag (called tag) that will allow us to create entities dynamically. Thus values can be substituted into entities by escaping the entity as in:

```
<tag>a href=<property href></tag>
```

The [optional] attribute "name" may be used to specify the name of the tag, which will be emmitted just after the "<".
Overrides:
tag_tag in class SetTemplate

---

tag_slash_tag

```
public void tag_slash_tag(RewriteContext hr)
```

Description copied from class: SetTemplate
Insert a literal ">"
Overrides:
tag_slash_tag in class SetTemplate

Class NoImageTemplate

sunlabs.brazil.template
Class NoImageTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.template.NoImageTemplate
```

All Implemented Interfaces:
TemplateInterface

---

public class **NoImageTemplate**
extends Template

Sample template class for removing all images from a web page, and replacing them with their alt strings. This class is used by the TemplateHandler Each image is replaced by a text string defined by the server property "template", which the first "%" replaced by the contents of the alt attribute. Defaults to "[<b>%<b>]".

`template`
The text used to replace the image. The first "%" will contain the image "alt" string, if any.

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

`NoImageTemplate``()`	

Method Summary

`boolean`	**`init`**`(RewriteContext hr)` Save a reference to our request properties We'll use it some day to tailor the image display
`void`	**`tag_img`**`(RewriteContext hr)` Convert the html tag img into text using the alt string

Methods inherited from class sunlabs.brazil.template.Template
done

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

NoImageTemplate

```
public NoImageTemplate()
```

Method Detail

init

```
public boolean init(RewriteContext hr)
```

Save a reference to our request properties We'll use it some day to tailor the image display

Overrides:

init in class Template

---

tag_img

```
public void tag_img(RewriteContext hr)
```

Convert the html tag img into text using the alt string

Parameters:

h - The name/value pairs src= alt=<...>

Class PropsTemplate

sunlabs.brazil.template
Class PropsTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.template.PropsTemplate
```

All Implemented Interfaces:
Serializable, TemplateInterface

---

public class **PropsTemplate**
extends Template
implements Serializable

[*Deprecated, use the the SetTemplate*.]
Template class for substituting request properties into an HTML page. This class is used by the TemplateHandler The following request properties are used:

query
: The query parameters are placed into the request object, prefixed by the value assigned to "query".

headers
: The mime headers are placed into the request object, prefixed by the value assigned to "headers". The values: url, query, method, and version are copied from the request object into the properties. The clients IP address is saved in the "address" property.

url.orig
: If set and "headers" are requested, this value is used as the `url` instead of the one in request.url.

A new HTML tag, `<property>` is defined. It takes the following tag attributes:

name
: The name of the property in `props` to replace the `property` tag with.

default
: The value to use if the property is not defined. If no `default` is specified, the empty string is used instead.

convert
: The value is converted before substitution into the content. Convert accepts either:
  - **convert=html**Protect plain text by escaping the characters " < > and &.
  - **convert=url**Make sure the string is suitable for use in a url or query string.

See Also:
Serialized Form

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

PropsTemplate()	

Method Summary

boolean	**init**(RewriteContext hr) This gets called at every page, at the beginning.
void	**tag_property**(RewriteContext hr) Convert the html tag "property" in to the request's property
void	**tag_slash_tag**(RewriteContext hr) Insert a literal ">"
void	**tag_tag**(RewriteContext hr) Insert a literal "<".

Methods inherited from class sunlabs.brazil.template.Template
done

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

PropsTemplate

public **PropsTemplate**()

Method Detail

init

```
public boolean init(RewriteContext hr)
```

This gets called at every page, at the beginning. See if we should add the mime headers and query parameters into the request object

Overrides:

init in class Template

---

tag_property

```
public void tag_property(RewriteContext hr)
```

Convert the html tag "property" in to the request's property

Parameters:

`key` - The name of the property to substitute. Variable substitution using the style described in `Format.getProperty(java.util.Properties, java.lang.String, java.lang.String)` is permitted, e.g.: `employee.${id}.last`

---

tag_tag

```
public void tag_tag(RewriteContext hr)
```

Insert a literal "<". Using the current scheme, there is no easy way to substitute into a tag parameter. So we'll invent a "magic" tag (called tag) that will allow us to create entities dynamically. Thus values can be substituted into entities by escaping the entity as in:

```
<tag>a href=<property href></tag>
```

---

tag_slash_tag

```
public void tag_slash_tag(RewriteContext hr)
```

Insert a literal ">"

Class QueueTemplate.Queue

sunlabs.brazil.template
Class QueueTemplate.Queue

```
java.lang.Object
  |
  +--sunlabs.brazil.template.QueueTemplate.Queue
```

Enclosing class:
QueueTemplate

---

public static class **QueueTemplate.Queue**
extends Object

Create an object queue. "Getters" wait ’till something appears in the queue.

Field Summary

static int	**max**

Constructor Summary

QueueTemplate.Queue() Create a new Q of a maximum possible size

Method Summary	
void	**clear**() Clear the queue.
long	**count**() Return the total number of items Q'd.
Object	**get**(int timeout) Return the next item on the queue, waiting for up to "timeout" seconds or for an interrupt.
boolean	**isClosed**() Get the closed state.
void	**kick**() Send a notify: for debugging
long	**lastIn**() Return the last time a Q insertion was attempted.
long	**lastOut**() Return the last time a Q removal was attempted.
boolean	**put**(Object item) Put an item on the queue if it's open and not full.
boolean	**put**(Object item, boolean force) Put an item on the queue if it's not full.
boolean	**setClosed**(boolean closed) Set the closed state.
int	**size**() How many items are queue'd.
String	**toString**()

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

max

```
public static int max
```

Constructor Detail

QueueTemplate.Queue

```
public QueueTemplate.Queue()
```

Create a new Q of a maximum possible size

Method Detail

get

```
public Object get(int timeout)
```

Return the next item on the queue, waiting for up to "timeout" seconds or for an interrupt.

Returns:

the top of the Q, or null.

---

put

```
public boolean put(Object item)
```

Put an item on the queue if it’s open and not full.

---

put

```
public boolean put(Object item,
                   boolean force)
```

Put an item on the queue if it’s not full. If "force" is true, override the "closed" flag.

---

size

```
public int size()
```

How many items are queue’d.

---

kick

```
public void kick()
```

Send a notify: for debugging

---

lastIn

```
public long lastIn()
```

Return the last time a Q insertion was attempted.
Returns:
-1 if no attempts were made.

---

lastOut

```
public long lastOut()
```

Return the last time a Q removal was attempted.
Returns:
-1 if no attempts were made.

---

count

```
public long count()
```

Return the total number of items Q'd.
Returns:
The # of Q'd items.

---

clear

```
public void clear()
```

Clear the queue.

---

setClosed

```
public boolean setClosed(boolean closed)
```

Set the closed state.

---

isClosed

```
public boolean isClosed()
```

Get the closed state.

---

toString

```
public String toString()
```

Overrides:

toString in class Object

Class QueueTemplate.QueueItem

sunlabs.brazil.template
Class QueueTemplate.QueueItem

```
java.lang.Object
  |
  +--sunlabs.brazil.template.QueueTemplate.QueueItem
```

Enclosing class:
QueueTemplate

---

public static class **QueueTemplate.QueueItem**
extends Object

A bag of items to keep on the Q. We could add other stuff later.

Field Summary

Dictionary	**data**
String	**from**
String	**message**
String	**meta**
long	**timestamp**

Constructor Summary

QueueTemplate.QueueItem(String from, Dictionary data)
QueueTemplate.QueueItem(String from, String message, String meta) **Deprecated.** *Use the other constructor instead*

Method Summary

String	**toString**()

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

timestamp

public long **timestamp**

---

from

public String **from**

---

data

public Dictionary **data**

---

message

public String **message**

---

meta

public String **meta**

Constructor Detail

QueueTemplate.QueueItem

public **QueueTemplate.QueueItem**(String from,
Dictionary data)

QueueTemplate.QueueItem

```
public QueueTemplate.QueueItem(String from,
                               String message,
                               String meta)
```

Deprecated. *Use the other constructor instead*

Add an item to the Q. "meta" is application specific "meta" data associated with this Q item.

Method Detail

toString

```
public String toString()
```

Overrides:
toString in class Object

Class QueueTemplate

sunlabs.brazil.template
Class QueueTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.template.QueueTemplate
```

All Implemented Interfaces:
TemplateInterface

---

public class **QueueTemplate**
extends Template

Template class for Managing simple Queues, allowing text communication among sessions. The "name" attribute names the Q. When enqueueing messages, "name" is a white space separated list of queue recipients. When Dequeueing Messages, "name" is the recipient Queue.

The following tags are recognized.

- `<enqueue name="recipients ..." data=var1 var2 ... varn" from="sender" [nocreate="true|false"] >`
- `<dequeue name="name" prepend="props prefix" timelimit="sec">` If "timelimit" has a suffix of "ms", then the time is taken in ms.
- `<queueinfo name="q_name" prepend="props prefix" clear>`

This format is supported for backwar compatibility with the old behavior. (Just incase the compatibility isn't quite right, see QueueTemplateOld).

- `<enqueue name="recipients ..." meta="anything" message="msg" from="sender" [nocreate="true|false"] >`

Inner Class Summary	
static class	**QueueTemplate.Queue** Create an object queue.
static class	**QueueTemplate.QueueItem** A bag of items to keep on the Q.

Field Summary

static String	**Q_ID**

Fields inherited from class sunlabs.brazil.template.Template
debug

Constructor Summary

QueueTemplate()	

Method Summary

static QueueTemplate.QueueItem	**dequeue**(String name, int timelimit) Program access to the Q.
static boolean	**enqueue**(String to, String from, Dictionary data, boolean noCreate, boolean force) Allow a message to be enqueued from java code.
static boolean	**enqueue**(String to, String from, String message, String meta, boolean noCreate, boolean force) **Deprecated.**
static QueueTemplate.Queue	**getQ**(String name) Return a Q
void	**tag_dequeue**(RewriteContext hr) Remove an item from the queue, and generate the appropriate properties.
void	**tag_enqueue**(RewriteContext hr) Add a text message onto a named queue.
void	**tag_queueinfo**(RewriteContext hr) Return info about the Q, and optionally clear it.

Methods inherited from class sunlabs.brazil.template.Template
done, init

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

Q_ID

```
public static final String Q_ID
```

Constructor Detail

QueueTemplate

```
public QueueTemplate()
```

Method Detail

tag_enqueue

```
public void tag_enqueue(RewriteContext hr)
```

Add a text message onto a named queue. Attributes:
- name: the name of the queue (required)
- from: the sender name
- data: a set of property names to send as name/value pairs. If Names provided of the form nnn#vvv, then "vvv" is taken as the value to use if it is not already set, otherwise the empty string is used as the value.
- nocreate: If set, The item will not be Queued if the Q for that recipient does not already exist.
- force: put to the queue even if its closed

The property "count" contains the number of recipients for which the message was successfully Queued. The property "error.name" will contain an error message if Queueing failed. In both cases, the template prefix will be prepended. It is not considered an error condition for a message not to be

delivered to a non existent Queue if "nocreate" is set.

---

enqueue

```
public static boolean enqueue(String to,
                              String from,
                              String message,
                              String meta,
                              boolean noCreate,
                              boolean force)
```

Deprecated.

Allow a message to be enqueued from java code. Use the `enqueue(String to, String from, Dictionary data, boolean noCreate, boolean force)` method instead.

Parameters:

`to:` - The queue name
`from:` - The sender of the data
`message:` - The message to enqueue
`meta:` - The meta data, if any
`noCreate:` - If true, don't create a Q if it doesn'a already exist
`force:` - Force item onto Q even if it is closed

Returns:

True, if the data was enqueued

---

enqueue

```
public static boolean enqueue(String to,
                              String from,
                              Dictionary data,
                              boolean noCreate,
                              boolean force)
```

Allow a message to be enqueued from java code.

Parameters:

`to:` - The queue name (only a single q)
`from:` - The sender of the data
`data:` - a dictionary f name/value pairs
`noCreate:` - If true, don't create a Q if it doesn'a already exist
`force:` - Force item onto Q even if it is closed

Returns:

True, if the data was enqueued

---

tag_dequeue

```
public void tag_dequeue(RewriteContext hr)
```

Remove an item from the queue, and generate the appropriate properties. Attributes:

name
: The name of the queue to examine

prepend
: The prefix in the properties table to use to set the results. Defaults to our prefix.

timelimit
: how long to wait for an item to appear, in sec. Defaults to 30.

If an item is retrieved, the following request properties are set (preceded by prepend):

age
: Set how long the message has been q's (in seconds)

sent
: Set the timestamp (in sec) of when the item was q'd.

items
: Set the number of Q'd items.

from
: Set the (unauthenticated) sender.

error
: Something went wrong. Set an error message.

In addition, all name/value pairs send as part of the message will be set as well. (In prewvious versions of this template, that consists of "message" and (optionally) "meta".)

Note: this tag blocks until an item is received in the Queue (or a timelimit expires). As template processing is synchronized based on sessions, care should be taken to avoid blocking other (unrelated) session based requests while waiting on the queue.

---

dequeue

```
public static QueueTemplate.QueueItem dequeue(String name,
                                              int timelimit)
```

Program access to the Q.

Parameters:

`name;` - The name of the Q. A new Q will be created if it doesn't already exist.

`timelimit:` - how long (in ms) to wait before returning

Returns:

The Queue item, or Null if no item is available.

---

tag_queueinfo

```
public void tag_queueinfo(RewriteContext hr)
```

Return info about the Q, and optionally clear it. Attributes:

name
: The name of the queue to examine

prepend
: The prefix in the properties table to use to set the results. Defaults to our prefix.

clear
: If set, then clear the queue

remove
: If set, then remove the queue

closed=[true|false]
: set the closed state of the Q

The following request properties are set (preceded by prepend):

lastIn
: The timestamp (ms) of the last Q insert attempt.

lastOut
: The timestamp (ms) of the last Q retrieval attempt.

size
: The number of items in the Q.

count
: The number of items inserted into the Q

---

getQ

```
public static QueueTemplate.Queue getQ(String name)
```

Return a Q

Class RedirectTemplate

sunlabs.brazil.template
Class RedirectTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.template.RedirectTemplate
```

All Implemented Interfaces:
TemplateInterface

---

public class **RedirectTemplate**
extends Template

[*Deprecated, use the AddHeaderTemplate instead.*]
Template class for redirecting an html page This class is used by the TemplateHandler

---

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

`RedirectTemplate``()`	

Method Summary

`boolean`	**`done`**`(RewriteContext hr)` adjust the response headers to reflect redirection, if supplied.
`void`	**`redirect`**`(RewriteContext hr)` Look for a redirect tag, change it to an HREF, and remember where to redirect to (e.g.

Methods inherited from class sunlabs.brazil.template.Template
`init`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

RedirectTemplate

`public` **`RedirectTemplate`**`()`

Method Detail

redirect

`public void` **`redirect`**`(RewriteContext hr)`

Look for a redirect tag, change it to an HREF, and remember where to redirect to (e.g. `<redirect http://some.where.com>`)

---

done

`public boolean` **`done`**`(RewriteContext hr)`

adjust the response headers to reflect redirection, if supplied. Otherwise, ignore this request.
Overrides:
done in class Template

Class RewriteContext

sunlabs.brazil.template
Class RewriteContext

```
java.lang.Object
  |
  +--sunlabs.brazil.handler.HtmlRewriter
        |
        +--sunlabs.brazil.template.RewriteContext
```

---

public class **RewriteContext**
extends HtmlRewriter

A variant containing instance variables that may be referenced by rewriting filters. Every implementation of the template class may define methods of the form: `tag_`*xxx* or `tag_slash_`*xxx* which will get called when the corrosponding HTML entity `<xxx ...>` or `</xxx ...>` is found in the content being filtered.

An instance of this class is passed to each `tag` method, permitting introspection of the current filtering context.

Field Summary	
String	**prefix** The prefix to use for locating keys in the server or request properties objects.
Request	**request** The Request object, as passed to a handler's respond method.
Server	**server** The server object, as passed to a handler's init method.
String	**sessionId** A unique **session id**, if available.
String	**templatePrefix** This is the prefix defined by the invoker of a set of templates.

Fields inherited from class sunlabs.brazil.handler.HtmlRewriter
lex, sb

Constructor Summary

RewriteContext(Server server, String prefix, Request request, String content, String sessionId, TemplateRunner runner, Vector templates)

Method Summary	
void	**abort**() Cause this RewriteContext to abort its processing.
void	**addClosingTag**(String tag) Add a closing tag to the list of tags that mark un-interpreted text Deal with the tag prefix, if any
boolean	**checkRewriteState**(int state) Check whether this RewriteContext is in the specified state.
void	**decrNestingLevel**() Decrement the nesting level counter.
String	**get**(String name) overwrite "get" to automatically do ${...} substitutions The default is "true".
String	**get**(String name, boolean subst) Get an attribute value, and optionally perform ${...} substitutions.
String	**get**(String name, String dflt) Get a tag attribute, with a default value.
Properties	**getNamespaceProperties**() Get the proper properties table based on the "namespace" attribute.
int	**getNestingLevel**() Return the current nesting level counter.
void	**incrNestingLevel**() Increment the nesting level counter.
boolean	**isClosingFor**(String tag) See if the current token is the closing tag for the given string.
boolean	**isClosingFor**(String tag, boolean close) See if the current token is the closing tag for the given string.
boolean	**isTrue**(String name) Determine the value of a boolean attribute in the current tag.
void	**process**() Invoke a template on this token, if any template is interested in this token.
boolean	**setRewriteState**(int state) Set the given state of this RewriteContext.
int	**tagsSeen**() Return the number of HTML tags seen to this point.
void	**unsetRewriteState**(int state) Unset the given state of this RewriteContext.

Methods inherited from class sunlabs.brazil.handler.HtmlRewriter
`accumulate, append, appendToken, getArgs, getBody, getMap, getTag, getToken, getType, isSingleton, keys, killToken, nextTag, nextToken, pushback, put, quote, remove, reset, setSingleton, setTag, setType, tagCount, tokenCount, toString`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Field Detail

server

`public Server` **`server`**

The server object, as passed to a handler's init method.

---

prefix

`public String` **`prefix`**

The prefix to use for locating keys in the server or request properties objects. This is a dynamic value that changes during template processing. Before each template is processed, it is set to the prefix associated with the template class that the current tag belongs to.

---

templatePrefix

`public String` **`templatePrefix`**

This is the prefix defined by the invoker of a set of templates. Its value corrosponds to the class that invokes a set of templates.

---

request

`public Request` **`request`**

The Request object, as passed to a handler's respond method.

---

sessionId

```
public String sessionId
```

A unique **session id**, if available.

Constructor Detail

RewriteContext

```
public RewriteContext(Server server,
                      String prefix,
                      Request request,
                      String content,
                      String sessionId,
                      TemplateRunner runner,
                      Vector templates)
```

Method Detail

abort

```
public void abort()
```

Cause this RewriteContext to abort its processing.

---

setRewriteState

```
public boolean setRewriteState(int state)
```

Set the given state of this RewriteContext.

---

unsetRewriteState

```
public void unsetRewriteState(int state)
```

Unset the given state of this RewriteContext.

---

checkRewriteState

```
public boolean checkRewriteState(int state)
```

Check whether this RewriteContext is in the specified state.

---

incrNestingLevel

`public void **incrNestingLevel**()`

Increment the nesting level counter.

---

decrNestingLevel

`public void **decrNestingLevel**()`

Decrement the nesting level counter.

---

getNestingLevel

`public int **getNestingLevel**()`

Return the current nesting level counter.

---

process

`public void **process**()`

Invoke a template on this token, if any template is interested in this token. The template may consume more than just this token, if it wants.

---

tagsSeen

`public int **tagsSeen**()`

Return the number of HTML tags seen to this point.

---

get

`public String **get**(String name)`

overwrite "get" to automatically do ${...} substitutions The default is "true". Setting the default to "false" and recompiling is more backward compatible (but less useful).

Overrides:

get in class HtmlRewriter

Tags copied from class: HtmlRewriter

Parameters:

`The` - key to lookup in the current tag's attributes.

Returns:

The value to which the specified key is mapped, or `null` if the key was not in the attributes.

See Also:

`LexML.getAttributes()`

---

get

```
public String get(String name,
                  boolean subst)
```

Get an attribute value, and optionally perform ${...} substitutions.

---

get

```
public String get(String name,
                  String dflt)
```

Get a tag attribute, with a default value. The name is looked-up as an attribute in the current tag. If it is not found, the configuration property by the same name is used. If that is not found, `dflt` is used instead.

Parameters:

`name` - The name of the attribute to look up

`dflt` - The default attribute value (may be null)

Returns:

The value of "name"

---

isTrue

```
public boolean isTrue(String name)
```

Determine the value of a boolean attribute in the current tag. ${...} substitution is performed on the value.

Parameters:

`name` - The name of the boolean attribute

Returns:

false if the value is: null, "", "0", "no", "off", or "false" true otherwise. "attribute=" is false, but "attribute" with no value is true;

---

isClosingFor

```
public boolean isClosingFor(String tag)
```

See if the current token is the closing tag for the given string. Account for tag prefix, if any.

Parameters:

`tag` - tag whose match (e.g. /[prefix]tag is to be found

Returns:

true if the current tag closes "tag"

---

isClosingFor

```
public boolean isClosingFor(String tag,
                            boolean close)
```

See if the current token is the closing tag for the given string. Account for tag prefix, if any.

Parameters:

`tag` - tag whose match (e.g. /[prefix]tag is to be found

`close` - if set, "tag" matches "/[prefix]tag", otherwise tag matches "[prefix]tag".

Returns:

true if the current tag closes "tag"

---

addClosingTag

```
public void addClosingTag(String tag)
```

Add a closing tag to the list of tags that mark un-interpreted text Deal with the tag prefix, if any

---

getNamespaceProperties

```
public Properties getNamespaceProperties()
```

Get the proper properties table based on the "namespace" attribute. The namespaces "local" (the default) and "server" are special. The "sessionTable" cinfiguration property can be used to override the default session-table, which defaults to the template handler/filter prefix.

Class ScriptEvalTemplate

sunlabs.brazil.template
Class ScriptEvalTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.template.ScriptEvalTemplate
```

All Implemented Interfaces:
TemplateInterface

---

public class **ScriptEvalTemplate**
extends Template

Template class for performing ${...} substitutions inside javascript and style tags. This class is used by the TemplateHandler

A new attribute `eval` is defined for the `script` and `style` tags. If `eval` is present, any ${...} constructs are evaluated in the body of the "script" or "style".

If the attribute `esc` is true, then strings of the form "\X" are replaced as per `Format`. Otherwise "\X" is treated specially only for X = $, to escape variable substitution.

Both "eval" and "esc" attributes are removed from the "script" or "style" tags.

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

`ScriptEvalTemplate()`	

Method Summary	
void	**tag_script**(RewriteContext hr)
void	**tag_style**(RewriteContext hr)

Methods inherited from class sunlabs.brazil.template.Template
done, init

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ScriptEvalTemplate

public **ScriptEvalTemplate**()

Method Detail

tag_script

public void **tag_script**(RewriteContext hr)

---

tag_style

public void **tag_style**(RewriteContext hr)

Class SetTemplate

sunlabs.brazil.template
Class SetTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.template.SetTemplate
```

All Implemented Interfaces:
Handler, Serializable, TemplateInterface
Direct Known Subclasses:
MultipartSetTemplate

---

public class **SetTemplate**
extends Template
implements Serializable, Handler

Template (and handler) class for setting and getting values to and from the current (or other) request context. This class is used by the TemplateHandler. The following tags are processed by this class:

- <set name=.. [value=..] [namespace=..]>
- <get name=.. [default=..] [namespace=..] [max=nnn] [convert=html|url|none] [set=..]>
- <namespace name=.. [clear] [remove] [store=xxx] [load=xxx]>
- <import namespace=.. >
- <tag [name=..]>
- </tag>

The tag`<set>` is processed with the following attributes:

name=*value*
: The name of the entry to set in the request properties.

value=*value*
: The value to store in the property. If this option is not specified, then the existing value for this property will be removed.

default=*other*
: If `default` is specified and the specified *value* was the empty string "", then store this *other* value in the property instead. *other* is only evaluated in this case.

local
: By default, variables are set (or removed from) a namespace that is persistent across all requests for the same session. If `local` is specified, the values only remain for the duration of the current request. [Deprecated - use namespace=local]

namespace
: The namespace to use to set or delete the property. The default is `namespace=${SessionID}`, unless `set` is inside the scope of an enclosing `namespace` tag. Namespaces are used to define variable scopes that may be managed explicitly in template pages.

Two namespaces are treated specially:

- **local**
  may be used to set (or clear) values from `request.props`.
- **server**
  may be used to set (or clear) values from `server.props`, unless "noserver" is defined in which case hte "Server" namespace is not treated specially.

The `<namespace>..</namespace>` tag pairs are used to specify a default namespace other than the `SessionId` for the **set** and **import** tags or manipulate a namespace. These tags don't nest.

attributes for `namespace`:

name=*value*
: The default namespace to use for the **set** and `import` default namespace name.

clear
: If set, namespace `name` is cleared.

remove
: If set, namespace `name` is removed.

load=*file_name*
: If specified, and the property `saveOk` is set, then the specified namespace will be saved loaded from a file in java properties format. Relative path names are resolved with respect to the document root.

store=*file_name*
: If specified, and the property `saveOk` is set, then the specified namespace will be stored to a file in java properties format. Relative path names are resolved with respect to the document root.

When using `load` or `store`, the namespace names "local" and "server" are treated specially, referring to "request.props" and "server.props" respectively.

The tag`<import>` is processed with the following attributes:

namespace=*value*
: The name/value pairs assiciated with the namespace are made available for the remainder of the page.

The "get" (formerly property) tag has the following attributes:

name
: The name of the variable to get

namespace
: The namespace to look in. By default, the variable is searched for in "request.props"

default
: The value to use if no value matches "name".

convert
: The conversion to perform on the value of the data before substitution: "html", "url", or "none" (the default). For "html", any special html syntax is escaped. For "url", the data will be suitable for transmission as an http URL.

max
: The output is truncated to at most `max` characters.

If a single attribute is specified, with no "=", then is is taken to be the "name" parameter. Thus: `<get foo>` is equivalent to: `<get name="foo">`.

Request Properties:

sessionTable
: The name of the SessionManager table to use for storing values. Defaults to the template handler's prefix. When configured in one server both as a handler and a template, the sessionTable should be set to the same value in both cases to get a useful result.

debug
: If set, the original tag is included in a comment, otherwise it is removed entirely.

mustMatch
: Set to a glob pattern that all names must match in order to be set. This may be used to prevent malicious html pages (what a concept) from changing inappropriate values.

noserver
: The "server" namespace will no longer be mapped to `server.props`

noSet
: If set, then the "set" tag will be disabled.

querySet
: If set, then properties may be set in query parameters, to the "handler" portion, but only if they match the glob pattern.

autoImport
: If set to "1", the namespace for the session is automatically imported. (defaults to "1");

imports
: Defines a set of (white space delimited) namespaces that will automatically be imported at the beginning of each page. Each namespace name will be processed for ${...} substitutions before an import is attempted. If the namespace doesn't already exist, the import is ignored.

query
: The query parameters are inserted into the request object, prefixed by the value assigned to "query".

headers
: The mime headers are inserted into the request object, prefixed by the value assigned to "headers".

  In addition, the following properties (prefixed with "headers") are also set:

  - **address** The ip address of the client
  - **counter** A monotonically increasing counter (# of client requests accepted since the server started).
  - **method** The request method (typically GET or POST).
  - **protocol** The client HTTP protocol level (for 1.0 or 1.1)
  - **query** The current query data, if any.
  - **timestamp** A timestamp (in ms since epoch) from when this request was first accepted.
  - **url** The current url.
  - **hostname, hostport** The name and port parts of the "host" header, if set.

url.orig
: If set and "headers" are requested, this value is used as the `url` instead of the one in request.url.

Normally, any persistent properties held by the SetTemplate are chained onto the request.props when the init method is found. If this template is installed as an up-stream handler, then the persistent properties associated with the session are made available at that time.

When used as a handler, the following property is used:

session=*value*

The request property to find the session information in. Normally this should be the same as the session property used by the container calling this as a template.

saveOk

This must be specified in order for the "namespace store" or "namespace load" functions to operate.

See Also:

Serialized Form

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

`SetTemplate()`	

Method Summary	
File	**file2path**(Properties p, String prefix, String name) Convert a file name into a file object.
boolean	**init**(RewriteContext hr) Chain the session-id properties into the request chain, if there are any.
boolean	**init**(Server server, String prefix) Get the 2 SessionManager keys, "sessionTable" and "session" (SessionID).
boolean	**respond**(Request request) Chain a SessionManager entries onto the request properties, and optionally allow setting of request props from query parameters.
void	**tag_get**(RewriteContext hr) Replace the tag "get" with the value of the variable specified by the "name" attribute.
void	**tag_import**(RewriteContext hr) Import all the data from the named namespace.
void	**tag_namespace**(RewriteContext hr) Set the default namespace for "set" and "import".
void	**tag_property**(RewriteContext hr) Convert the html tag "property" in to the request's property DEPRECATED - use "get"
void	**tag_set**(RewriteContext hr) Set the value of a variable.
void	**tag_slash_namespace**(RewriteContext hr) Clear the default namespace for "set" and "import".
void	**tag_slash_tag**(RewriteContext hr) Insert a literal ">"
void	**tag_tag**(RewriteContext hr) Insert a literal "<".

Methods inherited from class sunlabs.brazil.template.Template
done

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

SetTemplate

```
public SetTemplate()
```

Method Detail

init

```
public boolean init(RewriteContext hr)
```

Chain the session-id properties into the request chain, if there are any. If "query" or "headers" are requested for "get", then add in those properties to request.props.

Overrides:

init in class Template

---

tag_set

```
public void tag_set(RewriteContext hr)
```

Set the value of a variable. Allow variable substitutions in the name and value. Don't create property tables needlessly.

Attributes:

name
: The name of the variable to set

value
: The value to set if to.

namespace
: The namespace to look in. By default, the variable is set in the namespace associated with the current "SessionId".

local
: A (deprecated) alias for "namespace=local", or the current request.props.

---

tag_property

```
public void tag_property(RewriteContext hr)
```

Convert the html tag "property" in to the request's property DEPRECATED - use "get"

---

tag_get

```
public void tag_get(RewriteContext hr)
```

Replace the tag "get" with the value of the variable specified by the "name" attribute.

Attributes:

name
: The name of the variable to get

namespace
: The namespace to look in. By default, the variable is searched for in "request.props". The namespace "server" is used to look in the server's namespace. The namespace "local" is a synonym for the default namespace.

default
: The value to use if no value matches "name".

convert
: The conversion to perform on the value of the data before substitution: "html", "url", or "none" (the default). For "html", any special html syntax is escaped. For "url", the data will be suitable for transmission as an http URL.

max
: Truncate the String to at most `max` characters. Max must be at least one, and truncation occurs after any conversions.

set
: The resultant value is placed into the request property named by the `set` attribute, and not inserted into the HTML stream. If none of "namespace", "convert", or "match" is used, then this simply copies the property from one name to another.

If a single attribute is specified, with no "=", then is is taken to be the "name" parameter. Thus: `<get foo>` is equivalent to: `<get name="foo">`.

---

tag_import

```
public void tag_import(RewriteContext hr)
```

Import all the data from the named namespace. The namespace associated with the session ID is imported automatically for backward compatibility. If the namespace doesn't exist, don't create it now, but remember it needs to be "Chained" if it is created on this page.

---

tag_namespace

```
public void tag_namespace(RewriteContext hr)
```

Set the default namespace for "set" and "import".

---

file2path

```
public File file2path(Properties p,
                      String prefix,
                      String name)
```

Convert a file name into a file object. relative paths use are resolved relative to the document root

Parameters:

`p` - properties file to find the doc root in
`prefix` - the properties prefix for root
`name` - the name of the file

---

tag_slash_namespace

```
public void tag_slash_namespace(RewriteContext hr)
```

Clear the default namespace for "set" and "import".

---

tag_tag

```
public void tag_tag(RewriteContext hr)
```

Insert a literal "<". Using the current scheme, there is no easy way to substitute into a tag parameter. So we'll invent a "magic" tag (called tag) that will allow us to create entities dynamically. Thus values can be substituted into entities by escaping the entity as in:

```
<tag>a href=<property href></tag>
```

The [optional] attribute "name" may be used to specify the name of the tag, which will be emmitted just after the "<".

---

tag_slash_tag

```
public void tag_slash_tag(RewriteContext hr)
```

Insert a literal ">"

---

init

```
public boolean init(Server server,
                    String prefix)
```

Get the 2 SessionManager keys, "sessionTable" and "session" (SessionID).

Specified by:

init in interface Handler

Tags copied from interface: Handler

Parameters:

`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.

Returns:

`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
```

Chain a SessionManager entries onto the request properties, and optionally allow setting of request props from query parameters. Only the session id table can be chained, and values (if any) are set in request.props (i.e. namespace=local)

Specified by:

respond in interface Handler

Tags copied from interface: Handler

Parameters:

`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

Class TOCTemplate

sunlabs.brazil.template
Class TOCTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.template.TOCTemplate
```

All Implemented Interfaces:
TemplateInterface

---

public class **TOCTemplate**
extends Template

Template class for extracting table of contents information out of an html page by examining the "H1" tags, and setting request properties that can be used to build a table of contents. This class is used by the TemplateHandler.

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

`TOCTemplate``()`	

Method Summary

`boolean`	**`init`**`(RewriteContext hr)` Called before this template processes any tags.
`void`	**`tag_h1`**`(RewriteContext hr)` Add a name anchor to the H1 tag, so we can go there, and set the request properties:`TOC.[anchor]` to the text of the `H1` tag.

Methods inherited from class sunlabs.brazil.template.Template
`done`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

TOCTemplate

`public TOCTemplate()`

Method Detail

init

`public boolean init(RewriteContext hr)`

Description copied from class: Template
Called before this template processes any tags.
Overrides:
init in class Template

---

tag_h1

`public void tag_h1(RewriteContext hr)`

Add a name anchor to the H1 tag, so we can go there, and set the request properties:`TOC.[anchor]` to the text of the `H1` tag.

Class Template

sunlabs.brazil.template
Class Template

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
```

All Implemented Interfaces:
TemplateInterface
Direct Known Subclasses:
AddHeaderTemplate, AllowGtTemplate, AsteriskAGIHandler, AsteriskHandler, BeanShellServerTemplate, BSLTemplate, ChangedTemplate, ContentTemplate, DateTemplate, DebugTemplate, DeCommentTemplate, DelayHandler, DigestTemplate, DirectoryTemplate, EmailTemplate, ExecTemplate, FormClientTemplate, FormTemplate, GroovyServerTemplate, HighlightTemplate, IncludeTemplate, JavaScriptTemplate, LDAPTemplate, ListTemplate, MacroTemplate, MiscTemplate, ModifiedTemplate, NoImageTemplate, PlainTemplate, PropsTemplate, PythonServerTemplate, QueueTemplate, RedirectTemplate, RemoteHostTemplate, ScriptEvalTemplate, SetTemplate, SMTPHandler, SnarfTemplate, SourceTemplate, SqlTemplate, SubstAllTemplate, TclServerTemplate, TemplateTemplate, TOCTemplate, UrlNavBarTemplate, XmlTreeTemplate

---

public class **Template**
extends Object
implements TemplateInterface

Field Summary	
boolean	**debug**

Constructor Summary	
Template()	

Method Summary	
boolean	**done**(RewriteContext hr) Called after all tags have been processed, one final chance.
boolean	**init**(RewriteContext hr) Called before this template processes any tags.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

debug

```
public transient boolean debug
```

Constructor Detail

Template

```
public Template()
```

Method Detail

init

```
public boolean init(RewriteContext hr)
```

Called before this template processes any tags.

Specified by:

init in interface TemplateInterface

---

done

```
public boolean done(RewriteContext hr)
```

Called after all tags have been processed, one final chance.

Specified by:

done in interface TemplateInterface

Class TemplateHandler

sunlabs.brazil.template
Class TemplateHandler

```
java.lang.Object
  |
  +--sunlabs.brazil.template.TemplateHandler
```

All Implemented Interfaces:
Handler

---

public class **TemplateHandler**
extends Object
implements Handler

The `TemplateHandler` reads a template file from the document root, based on the URL, and passes the content through one or more template filters.

The following configuration parameters are used to initialize this `Handler`:

prefix, suffix, glob, match
: Specify the URL that triggers this handler. By default, all URL's are considered. Only documents that are mime sub-types of `text` are processed. See `FileHandler` for a description of how to set mime types for url suffixes. (See `MatchString`).

`templates`
: A list of template names. For each name in the list, the property *`name`*`.class` is examined to determine which class to use for each template. Then `name` is used as the prefix for other template specific properties if any. If *`name`*`.class` does not exist, then `name` is assumed to be the class name, and there are no template specific properties for the template. Methods in the template classes will be invoked to process the XML/HTML tags present in the content.

`session`
: The name of the request property that the Session ID will be found in, used to identify the proper template instance. The default value is "SessionID". Typically, a sessionHandler, such as `CookieSessionHandler` is used upstream to create the sessionID. If no id is found, then the session named "common" is used instead. Exactly one instance of each template class is created for each session.

`default`
: The default file in the directory to use as a template if a directory name is specified. Defaults to index[suffix], or "index.html" if no suffix is provided.

`encoding`
: The character encoding to use to interpret the template. If no encoding is specified, the default encoding is used. The template is read from the filesystem, and converted into a String using this encoding. All template processing is done using the String representation.

`outputEncoding`
: The character encoding to use to interpret the template results. If no "outputEncoding" is specified, then "encoding" is used. Once template processing is complete, the results are converted into a byte stream for transmission to the client using the "outputEncoding", if specified. If not specified then the

HTTP default (8-bit ASCII) encoding is used.

`modified`
: if present (e.g. set to any value) an HTTP `last-modified` header is added to the response with the current time.

`debug` if set to "true", template debugging is enabled: templates will emit their pre-processed markup as an HTML comment. This parameter only takes effect if the `debug` option is not specified for an individual template.

`tagPrefix`
: If specified, all tag names defined for each template class are prefixed with *tagPrefix*. This parameter only takes effect if the `tagPrefix` option is not specified for an individual template.

The request properties `DirectoryName` and `fileName` may be set as a convenience for downstream handlers.

This handler duplicates some of the functionality of the `template filter`, so that it may be used by itself in simple configurations. As such, if issues re-directs if directories are given without a trailing "/", and uses an "index" file (see `default` above) if a directory name is specified.

To filter content other than from the file system, use the `template filter` instead.

Constructor Summary

`TemplateHandler``()`	

Method Summary

`String`	**`getContent`**`(Request request, File file, String encoding)` get the content associated with this template.
`boolean`	**`init`**`(Server server, String propsPrefix)` Initializes the handler.
`boolean`	**`respond`**`(Request request)` Process an html template file, using the supplied template processing classes.

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

TemplateHandler

```
public TemplateHandler()
```

Method Detail

init

```
public boolean init(Server server,
                    String propsPrefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.
Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
                throws IOException
```

Process an html template file, using the supplied template processing classes.
Specified by:
respond in interface Handler
Tags copied from interface: Handler
Parameters:
`request` - The `Request` object that represents the HTTP request.
Returns:
`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

getContent

```
public String getContent(Request request,
                         File file,
                         String encoding)
                  throws IOException
```

get the content associated with this template. This version reads it from a file.

Parameters:

`request` - The standard request object

`file` - The file object to get the template from

Returns:

The content of the template to be processed

Interface TemplateInterface

sunlabs.brazil.template
Interface TemplateInterface
All Known Implementing Classes:
Template

---

public interface **TemplateInterface**

Interface for templates. Most templates are expected to extend the Template class. This interface is for those cases where a template must extend something else.

Method Summary

boolean	**done**(RewriteContext hr) Called after all tags have been processed, one final chance.
boolean	**init**(RewriteContext hr) Called before this template processes any tags.

Method Detail

init

```
public boolean init(RewriteContext hr)
```

Called before this template processes any tags.

---

done

```
public boolean done(RewriteContext hr)
```

Called after all tags have been processed, one final chance.

Class TemplateRunner

sunlabs.brazil.template
Class TemplateRunner

```
java.lang.Object
  |
  +--sunlabs.brazil.template.TemplateRunner
```

---

public class **TemplateRunner**
extends Object

Class for processing html templates. An html template is an ordinary html string, with additional application specific tags added (sort of like XML). Each tag is mapped to a java method of a template class, that rewrites its tag into normal html.

The mechanism used to map templates into sessions is inadaquate, and should be fixed in a future version. In the current implementation, Each session maintains its own set of template instances. Instance variables in template classes may be used to hold session specific state. Calls to a template are synchronized on the session id; only one request per session is dealt with simultaneously.

Constructor Summary

TemplateRunner(Server server, String prefix, String names) Process an HTML template with a template processing class.	

Method Summary

String	**getError**() Return the last error message generated, or null of no errors have occurred since the last call to "process".
String	**process**(Request request, String content, String sessionId) Process an html template file, using the supplied template processing Return the content of the template just processed, or null if there was no template processed.
void	**process**(RewriteContext hr) Processes the next token in the HTML document represented by the given RewriteContext.
int	**tagsProcessed**() Return the # of tags replaced in the previous call to "process".
int	**tagsSeen**() Return the # of HTML tags seen in the previous call to "process".

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

TemplateRunner

```
public TemplateRunner(Server server,
                      String prefix,
                      String names)
               throws ClassNotFoundException,
                      ClassCastException
```

Process an HTML template with a template processing class. We peruse the template class, looking at all of its methods. When when we process a template, we match the html tags against the declared methods in the template class. Each method name of the form `tag_`*xxx* or `tag_slash_`*xxx* is invoked when the corrosponding <*xxx*> or <*/xxx*> tag is found.

Each instance of `_x`*nn* in the method name is replaced by the corrosponding hex code for the character. This allows non-printable tags to to be processed with templates.

The methods `init` and `done` are each called once, at the beginning and end of the page respectively. These methods are called for all templates, in the order they are specified in the *templates* parameter.

There are three methods taht may be defined that don't follow the naming convension described above. They are:

- `comment`
  is called for each html/XML comment.
- `string`
  is called for all text between any tags.
- `defaultTag`
  is called for every tag that does not specifically have a tag method. If more than one template defines one of these methods, only the first template's method will be called.

If the server property "tagPrefix" associated with each template's properties prefix exists, it is used to prefix each tag name (this feature is for experimental support of XML namespaces, and probably doesn't belong here).

Parameters:

`server` - The HTTP server that created the `Handler` or `Filter` that invoked this `TemplateRunner`
`prefix` - The prefix associated with the parent `Handler` or `Filter`

`names` - The names of the Template classes or property prefixes (i.e. tokens) that, when concatenated with ".class" define a property that names a Template class. This TemplateRunner will dispatch to the methods described by the union of all the tag methods in the given Template classes.

The `init` and `done` methods for each template specified will be called in order. If any of the calls returns `false`, this handler terminates and no output is generated.

The names "comment", "string", and "defaultTag" are handled specially.

Method Detail

process

```
public String process(Request request,
                      String content,
                      String sessionId)
```

Process an html template file, using the supplied template processing Return the content of the template just processed, or null if there was no template processed.

Parameters:

`content` - The template.

`sessionId` - An arbitrary identifier used to locate the correct instance of the template class for processing this template. The first time an identifier is used, a new instance is created.

`args` - The arguments passed to the templates init method.

Returns:

content or null

---

process

```
public void process(RewriteContext hr)
```

Processes the next token in the HTML document represented by the given `RewriteContext`. Processing a token involves either dispatching to a tag-handling method in one of the `Template` objects, or just advancing to the next token if no `Template` was interested.

Parameters:

`hr` - The RewriteContext holding the state of the HTML document.

---

getError

```
public String getError()
```

Return the last error message generated, or null of no errors have occurred since the last call to "process". XXX not thread safe between calls to process() and getError().

---

tagsSeen

```
public int tagsSeen()
```

Return the # of HTML tags seen in the previous call to "process".

---

tagsProcessed

```
public int tagsProcessed()
```

Return the # of tags replaced in the previous call to "process".

Class TemplateTemplate

sunlabs.brazil.template
Class TemplateTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.template.TemplateTemplate
```

All Implemented Interfaces:
TemplateInterface

---

public class **TemplateTemplate**
extends Template

Template class for processing markup through a sequence of template filters. This class processes:

```
<template data=... [token=...] [prefix=...] [ignore=true|false]>
```

data
: contains the markup to process.

token
: names the request property that contains the class names or tokens that will define the templates that will process the data (see the `TemplateFilter` discussion regarding the *templates* property). It defaults to "templates" with this class's prefix.

prefix
: Which prefix to assign the templates, if the "token" list contains class names. Defaults to our prefix.

ignore
: If true, the result of the template processing is ignored. by default it is inserted into the resultant markup.

The templates for processing each token are setup only once, the first time the token is referenced.

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

`TemplateTemplate()`	

Method Summary

void	**tag_template**(RewriteContext hr)

Methods inherited from class sunlabs.brazil.template.Template
done, init

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

TemplateTemplate

public **TemplateTemplate**()

Method Detail

tag_template

public void **tag_template**(RewriteContext hr)

Class UrlNavBarTemplate

sunlabs.brazil.template
Class UrlNavBarTemplate

```
java.lang.Object
  |
  +--sunlabs.brazil.template.Template
        |
        +--sunlabs.brazil.template.UrlNavBarTemplate
```

All Implemented Interfaces:
TemplateInterface

---

public class **UrlNavBarTemplate**
extends Template

Template class for dynamically generating a navigation bar by looking at portions of the url. Given url: `/main/next/last/foo.html` generate the request properties for the directories: `main`, `next`, and `last`. The properties will be:

```
NAV.main=/main/
NAV.next=/main/next/ ....
NAV.=main/next/....
```

These properties may be incorporated into web pages using the BSLTemplate's <foreach> tag, using a delimeter of "/" to iterate over the listings.

The follow request properties are consulted:
prepend
Use as a prefix on the property name, instead of "NAV.".
includeDir
Normally, if the URL refers to the directory (e.g. it ends with a /), no nav bar entry is generated. If this property is set, the entry is generated.

Fields inherited from class sunlabs.brazil.template.Template
`debug`

Constructor Summary

`UrlNavBarTemplate()`	

Method Summary

boolean	**init**(RewriteContext hr) Compute a set of properties based on the URL

Methods inherited from class sunlabs.brazil.template.Template
done

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

UrlNavBarTemplate

public **UrlNavBarTemplate**()

Method Detail

init

public boolean **init**(RewriteContext hr)

Compute a set of properties based on the URL

Overrides:

init in class Template

Package sunlabs.brazil.util

Utility classes that are generically useful in Java language programs. They have no dependencies out side of the JDK, and none require AWT.

Class Base64

sunlabs.brazil.util
Class Base64

```
java.lang.Object
  |
  +--sunlabs.brazil.util.Base64
```

---

public class **Base64**
extends Object

Utility to base64 encode and decode a string.

Method Summary	
static byte[]	**decode**(String s) A Base64 decoder.
static String	**encode**(byte[] src) base-64 encode a byte array
static String	**encode**(byte[] src, int start, int length) base-64 encode a byte array
static String	**encode**(String s) base-64 encode a string
static void	**main**(String[] args) Test the decoder and encoder.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Method Detail

encode

public static String **encode**(String s)

base-64 encode a string

Parameters:
`s` - The ascii string to encode
Returns:
The base64 encoded result

---

encode

```
public static String encode(byte[] src)
```

base-64 encode a byte array
Parameters:
`src` - The byte array to encode
Returns:
The base64 encoded result

---

encode

```
public static String encode(byte[] src,
                            int start,
                            int length)
```

base-64 encode a byte array
Parameters:
`src` - The byte array to encode
`start` - The starting index
`len` - The number of bytes
Returns:
The base64 encoded result

---

decode

```
public static byte[] decode(String s)
```

A Base64 decoder. This implementation is slow, and doesn't handle wrapped lines. The output is undefined if there are errors in the input.
Parameters:
`s` - a Base64 encoded string
Returns:
The byte array eith the decoded result

---

main

```
public static void main(String[] args)
```

Test the decoder and encoder. Call as `Base64 [string]`.

Class Calculator

sunlabs.brazil.util
Class Calculator

```
java.lang.Object
  |
  +--sunlabs.brazil.util.Calculator
```

---

public class **Calculator**
extends Object

Calculator implements a simple arithmetic expression evaluator. It can evaluate typical expressions with the "normal" operators and precedence. Formally, the BNF for the supported grammar is:

```
<stmt>    ::= <var> = <expr> | <expr>
<expr>    ::= <rexpr> | <expr> <bool op> <rexpr>
<bool op> ::= && | <or>
<or>      ::= ||
<rexpr>   ::= <aexpr> | <rexpr> <rel op> <aexpr>
<rel op>  ::= < | <= | > | >= | == | !=
<aexpr>   ::= <term> | <aexpr> <add op> <term>
<add op>  ::= + | -
<term>    ::= <factor> | <term> <mult op> <factor>
<mult op> ::= * | / | %
<factor>  ::= <var> | <num> | ! <factor> | ( <expr> )
<var>     ::= <letter> | <var> <var2>
<var2>    ::= <letterordigit> | . | _
<num>     ::= <unum> | + <unum> | - <unum>
<unum>    ::= <int> | <int> . | <int> . <int> | . <int>
<int>     ::= <digit> | <int> <digit>
```

A <letter> is defined as a Java `char` for which `Char.isLetter(char)` is `true`. A <letterordigit> is defined as a Java `char` for which `Char.isLetterOrDigit(char)` is `true`. A digit is defined as a Java `char` for which `Char.isDigit(char)` is `true`.

Values for `<var>` are looked up in the supplied `Dictionary`. If `<var>` can not be found, it is assumed to have the value zero. If the value found is "true" or "yes" (case insensitive), it is assumed to be one. Similarly, if the value found is "false" or "no", it is assumed to be zero. Assignment to `<var>` stores the computed value in the same `Dictionary`.

The period in `<unum>`, if there is one, must be immediately adjacent to surrounding `<int>`s.

Field Summary

`boolean`	**`debugging`** Set `true` for debug output.

Constructor Summary

Calculator() The no argument constructor will create an internal `Hashtable` in which it looks up and stores values associated with variables.
Calculator(Dictionary d) This constructor will use the `Dictionary` parameter to lookup and store values associated with variables.

Method Summary

String	**getValue**(String stmt) Computes the value of the statement passed in the parameter string and returns a string representation of the result.
String	**getValue**(String stmt, Dictionary d) Computes the value of the statement passed in the parameter string and returns a string representation of the result.
static void	**main**(String[] args) A test driver for the calculator.
void	**stringsValid**(boolean allStringsValid) Normally, variables whose values are "on", "yes", or "true" and converted to "1.0", while the values "off", "no", and "false" are converted to "0.0".

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

debugging

public boolean **debugging**

Set `true` for debug output. The output probably won't make sense to anyone other than the author.

Constructor Detail

Calculator

```
public Calculator()
```

The no argument constructor will create an internal `Hashtable` in which it looks up and stores values associated with variables.

See Also:

`Hashtable`

Calculator

```
public Calculator(Dictionary d)
```

This constructor will use the `Dictionary` parameter to lookup and store values associated with variables.

Parameters:

`d` - the `Dictionary` object that serves as a symbol table

See Also:

`Dictionary`

Method Detail

stringsValid

```
public void stringsValid(boolean allStringsValid)
```

Normally, variables whose values are "on", "yes", or "true" and converted to "1.0", while the values "off", "no", and "false" are converted to "0.0". All other values are considered an error. By passing "true", all normally invalid strings are given a value of "1.0".

---

getValue

```
public String getValue(String stmt)
                throws ArithmeticException
```

Computes the value of the statement passed in the parameter string and returns a string representation of the result. If the input statement consists only of a variable name and the result of the computation is zero, `null` is returned.

Parameters:

`stmt` - a string representation of an arithmetic expression or assignment

Returns:

a string representation of the computed result or `null`

Throws:

ArithmeticException - occurs when a result is improper (e.g. infinity) or when the input statement can not be parsed

---

getValue

```
public String getValue(String stmt,
                       Dictionary d)
                throws ArithmeticException
```

Computes the value of the statement passed in the parameter string and returns a string representation of the result. If the input statement consists only of a variable name and the result of the computation is zero, `null` is returned. The second parameter is used as a symbol table for the duration of this method call. Note this method is not thread safe!

Parameters:

`stmt` - a string representation of an arithmetic expression or assignment

`d` - the temporary symbol table

Returns:

a string representation of the computed result or `null`

Throws:

ArithmeticException - occurs when a result is improper (e.g. infinity) or when the input statement can not be parsed

---

main

```
public static void main(String[] args)
```

A test driver for the calculator. Type in arithmetic expressions or assignments and see the results. Use "dump" to see contents of all assigned variables.

Parameters:

`args` - required signature for `main` method, not used

Class ClockFormat

sunlabs.brazil.util
Class ClockFormat

```
java.lang.Object
  |
  +--sunlabs.brazil.util.ClockFormat
```

---

public class **ClockFormat**
extends Object

This class implements the "strftime" style clock format command. It decodes the following %X format strings:

'%%'
: Insert a %.

'%A'
: Full weekday name (Monday, Tuesday, etc.).

'%a'
: Abbreviated weekday name (Mon, Tue, etc.).

'%B'
: Full month name.

'%C'
: Century (00 - 99).

'%c'
: Locale specific date and time.

'%D'
: Date as %m/%d/%y.

'%d'
: Day of month (01 - 31).

'%e'
: Day of month (1 - 31), no leading zeros.

'%H'
: Hour in 24-hour format (00 - 23).

'%h'
: Abbreviated month name (Jan,Feb,etc.).

'%I'
: Hour in 12-hour format (01 - 12).

'%j'
: Day of year (001 - 366).

'%k'
: Hour in 24-hour format (0 - 23), no leading zeros.

'%l'
: Hour in 12-hour format (1 - 12), no leading zeros.

'%M'
: Minute (00 - 59).

’%m’
Month number (01 - 12).
’%n’
Insert a newline.
’%p’
AM/PM indicator.
’%R’
Time as %H:%M.
’%r’
Time as %I:%M:%S %p.
’%S’
Seconds (00 - 59).
’%s’
seconds since epoch.
’%T’
Time as %H:%M:%S.
’%t’
Insert a tab.
’%U’
Week of year (01-52), Sunday is first day.
’%u’
Weekday number (1 - 7) Sunday = 7.
’%V’
ISO 8601 Week Of Year (01 - 53).
’%W’
Week of year (01-52), Monday is first day.
’%w’
Weekday number (0 - 6) Sunday = 0.
’%X’
Locale specific time format.
’%x’
Locale specific date format.
’%Y’
Year with century (e.g. 1990)
’%y’
Year without century (00 - 99).
’%Z’
Time zone name.

Constructor Summary

`ClockFormat()`	

Method Summary

static String	**format**(int clockVal, String format, String zone) Formats a time value based on seconds into a human readable string.
static boolean	**haveZone**(String zone) See if a particular timezone is valid
static void	**main**(String[] args) Test main: ClockFormat format ?time? ?zone?

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ClockFormat

```
public ClockFormat()
```

Method Detail

format

```
public static String format(int clockVal,
                            String format,
                            String zone)
```

Formats a time value based on seconds into a human readable string.

Parameters:

`clockVal` - Seconds since the epoch

`format` - The strftime style format string. If format is null, then "%a %b %d %H:%M:%S %Z %Y" is used.

`zone` - The time zone abbreviation (e.g. GMT, or PST)

Returns:

The formatted string.

---

haveZone

```
public static boolean haveZone(String zone)
```

See if a particular timezone is valid

---

main

```
public static void main(String[] args)
                 throws Exception
```

Test main: ClockFormat format ?time? ?zone?

Class ClockScan

sunlabs.brazil.util
Class ClockScan

```
java.lang.Object
  |
  +--sunlabs.brazil.util.ClockScan
```

public class **ClockScan**
extends Object

This class implements time and date scanning. It was adapted from the TCL implementation found in JACL.

Field Summary

static boolean	**debug**

Constructor Summary

ClockScan()

Method Summary

static Date	**GetDate**(String dateString, Date baseDate, String zone) Scan a human readable date string and construct a Date.
static void	**main**(String[] args)

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

debug

```
public static boolean debug
```

Constructor Detail

ClockScan

```
public ClockScan()
```

Method Detail

GetDate

```
public static Date GetDate(String dateString,
                           Date baseDate,
                           String zone)
```

Scan a human readable date string and construct a Date.

Results: The scanned date (or null, if an error occured).

---

main

```
public static void main(String[] args)
                 throws Exception
```

Class Format

sunlabs.brazil.util
Class Format

```
java.lang.Object
  |
  +--sunlabs.brazil.util.Format
```

public class **Format**
extends Object

Format a string by substituting values into it from a properties object.

Constructor Summary

Format()	

Method Summary

static String	**deQuote**(String str) Remove surrounding quotes (" or ’) from a string.
static String	**getProperty**(Properties props, String expr, String defaultValue) Allow a property name to contain the value of another property, permitting nested variable substitution in attribute values.
static boolean	**isFalse**(String s) See if a String represents a "false" boolean value, which consists of: "no", "false", "off", or "0", in any case.
static boolean	**isTrue**(String s) See if a String represents a "true" boolean value, which consists of: "yes", "true", "on", or "1", in any case.
static String	**subst**(Dictionary props, String str) Allow a tag attribute value to contain the value of another property, permitting nested variable substitution in attribute values.
static String	**subst**(Dictionary props, String str, boolean noEsc) Allow a tag attribute value to contain the value of another property, permitting nested variable substitution in attribute values.

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

Format

```
public Format()
```

Method Detail

getProperty

```
public static String getProperty(Properties props,
                                 String expr,
                                 String defaultValue)
```

Allow a property name to contain the value of another property, permitting nested variable substitution in attribute values. The name of the embedded property to be substituted is bracketted by "${" and "}". See `subst(java.util.Dictionary, java.lang.String)`.

"ghi" => "foo"
"deffoojkl" => "baz"
"abcbazmno" => "garply"
getProperty("ghi") => "foo"
getProperty("def${ghi}jkl") => "baz"
getProperty("abc${def${ghi}jkl}mno") => "garply"

Parameters:

`props` - The table of variables to use when substituting.
`expr` - The property name, prossibly containing substitutions.
`defaultValue` - The value to use if the top-level substitution does not exist. May be `null`.

---

subst

```
public static String subst(Dictionary props,
                           String str)
```

Allow a tag attribute value to contain the value of another property, permitting nested variable substitution in attribute values. To escape ${XXX}, use \${XXX}.

The sequence "\X" is identical to "X", except when "X" is one of:

$
: A literal "$", that will not introduce a variable substitution if it is followed by "{".

n
: Insert a NL (newline) character

r
: Insert a CR (Carriage return) character

"
: Insert a single quote (").

l
: Insert a (<).

g
: Insert a (>).

a
: Insert a (&).

end of value
: Insert a "\".

```
"ghi" = "foo"
"deffoojkl" = "baz"
"abcbazmno" = "garply"
subst("ghi") = "ghi"
subst("def${ghi}jkl") = "deffoojkl"
subst("def\${ghi}jkl") = "def${ghi}jkl"
subst("abc${def${ghi}jkl}mno") = "abcbazmno"
subst("${abc${def${ghi}jkl}mno}") = "garply"
```

Parameters:

`props` - The table of variables to substitute. If this is a Properties object, then the getProperty() method is used instead of the Dictionary class get() method.

`str` - The expression containing the substitutions. Embedded property names, bracketted by "${" and "}" are looked up in the props table and replaced with their value. Nested substitutions are allowed.

Returns:

The substituted string. If a variable is not found in the table, the empty string is used.

---

subst

```
public static String subst(Dictionary props,
                           String str,
                           boolean noEsc)
```

Allow a tag attribute value to contain the value of another property, permitting nested variable substitution in attribute values. To escape ${XXX}, use \${XXX}. See `above`.

if `noEsc` is true, then The sequence "\X" is identical to "\X" for all X except X=$.

---

isTrue

```
public static boolean isTrue(String s)
```

See if a String represents a "true" boolean value, which consists of: "yes", "true", "on", or "1", in any case.

---

isFalse

```
public static boolean isFalse(String s)
```

See if a String represents a "false" boolean value, which consists of: "no", "false", "off", or "0", in any case.

---

deQuote

```
public static String deQuote(String str)
```

Remove surrounding quotes (" or ’) from a string.

Class Glob

sunlabs.brazil.util
Class Glob

```
java.lang.Object
  |
  +--sunlabs.brazil.util.Glob
```

---

public class **Glob**
extends Object

Glob-style string matching and substring extraction. Glob was implemented by translating the glob package for tcl8.0.

- "*" matches 0 or more characters
- "?" matches a single character
- "[...]" matches a set and/or range of characters
- "\" following character is not special

Each of the substrings matching (?, *, or [..]) are returned.

Method Summary	
static boolean	**match**(String pattern, String string) Match a string against a pattern.
static boolean	**match**(String pattern, String string, String[] substr) Match a string against a pattern, and return sub-matches.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Method Detail

match

```
public static boolean match(String pattern,
                            String string)
```

Match a string against a pattern.

Parameters:

`pattern` - Glob pattern. Nothing matches if pattern==null.

`string` - String to match against pattern.

Returns:

`true` if the string matched the pattern, `false` otherwise.

---

match

```
public static boolean match(String pattern,
                            String string,
                            String[] substr)
```

Match a string against a pattern, and return sub-matches.

The caller can provide an array of strings that will be filled in with the substrings of `string` that matched the glob meta-characters in `pattern`. The array of strings may be partially modified even if the string did not match the glob pattern. The array may contain more elements than glob meta-characters, in which case those extra elements will not be modified; the array may also contain fewer elements or even be `null`, to ignore some or all of the glob meta-characters. In other words, the user can pass pretty much anything and this method defines errors out of existence.

Parameters:

`pattern` - Glob pattern.

`string` - String to match against pattern.

`substr` - Array of strings provided by the caller, to be filled in with the substrings that matched the glob meta-characters. May be `null`.

Returns:

`true` if the string matched the pattern, `false` otherwise.

Class Guid

sunlabs.brazil.util
Class Guid

```
java.lang.Object
  |
  +--sunlabs.brazil.util.Guid
```

---

public class **Guid**
extends Object

Utility to generate GUID's (Globally Unique Identifiers). We'll fill in more methods as needed.

Constructor Summary

Guid()	

Method Summary

static String	**getString**() Return a GUID as a string.
static void	**setGuidImpl**(Guid factory) Allow global replacement of the GUID generator.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Guid

public **Guid**()

Method Detail

setGuidImpl

```
public static void setGuidImpl(Guid factory)
```

Allow global replacement of the GUID generator. Applications wishing to install their own GUID generators should sub-class Guid, override the getGuid() method, and use this method to install their generator.

---

getString

```
public static String getString()
```

Return a GUID as a string.

Class LexHTML

sunlabs.brazil.util
Class LexHTML

```
java.lang.Object
  |
  +--sunlabs.brazil.util.LexML
        |
        +--sunlabs.brazil.util.LexHTML
```

---

public class **LexHTML**
extends LexML

This class breaks up HTML into tokens.

This class differs slightly from LexML as follows: after certain tags, like the `<script>` tag, the body that follows is uninterpreted data and ends only at the next, in this case, `</script>` tag, not at the just the next "<" or ">" character. This is one way that HTML is not fully compliant with XML.

The default set of tags that have this special processing is `<script>`, `<style>`, and `<xmp>`. The user can change this by retrieving the Vector of special tags via `getClosingTags`, and modifying it as needed.

Fields inherited from class sunlabs.brazil.util.LexML
`COMMENT, STRING, TAG`

Constructor Summary

`LexHTML``(String str)` Creates a new HTML parser, which can be used to iterate over the tokens in the given string.	

Method Summary

Vector	**getClosingTags()** Get the set of HTML tags that have the special body-processing behavior mentioned above [p ??] .
String	**getTag()** Gets the tag name at the begining of the current tag.
boolean	**nextToken()** Advances to the next token, correctly handling HTML tags that have the special body-processing behavior mentioned above [p ??] .
void	**replace(String str)** Changes the string that this LexHTML is parsing.

Methods inherited from class sunlabs.brazil.util.LexML
getArgs, getAttributes, getBody, getLocation, getString, getToken, getType, isSingleton, rest

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

LexHTML

public **LexHTML**(String str)

Creates a new HTML parser, which can be used to iterate over the tokens in the given string.
Parameters:
str - The HTML to parse.

Method Detail

getClosingTags

`public Vector **getClosingTags**()`

Get the set of HTML tags that have the special body-processing behavior mentioned above [p ??] . The Vector is returned; the caller may modify it after calling this method, which will affect this parser's settings.

Parameters:

`tags` - The array of case-insensitive tag names that are only closed by seeing their "slashed" version.

---

nextToken

`public boolean **nextToken**()`

Advances to the next token, correctly handling HTML tags that have the special body-processing behavior mentioned above [p ??] . The user can then call the other methods in this class to get information about the new current token.

This method returns the uninterpreted data making up the body of a special HTML tag as a token of type `LexML.STRING`, even if the body was actually a comment or another tag.

Overrides:

nextToken in class LexML

Returns:

`true` if a token was found, `false` if there were no more tokens left.

---

getTag

`public String **getTag**()`

Gets the tag name at the begining of the current tag. In HTML, tag names are defined as case-insensitive, so the name returned is converted to lower case for the convenience of the user.

Overrides:

getTag in class LexML

Returns:

The lower-cased tag name, or `null` if the current token does not have a tag name.

See Also:

`LexML.getTag()`

---

replace

`public void **replace**(String str)`

Changes the string that this LexHTML is parsing.

Overrides:

replace in class LexML

Parameters:

`str` - The string that this LexHTML should now parse.

Class LexML

sunlabs.brazil.util
Class LexML

```
java.lang.Object
  |
  +--sunlabs.brazil.util.LexML
```

Direct Known Subclasses:
LexHTML

---

public class **LexML**
extends Object

This class breaks angle-bracket-separated markup languages like SGML, XML, and HTML into tokens. It understands three types of tokens:

tags
: Formally known as "entities", tags are delimited by "<" and ">". The first word in the tag is the tag name and the rest of the tag consists of the attributes, a set of "name=value" or "name" data. Spaces in tags are not significant except for quoted values in the attributes.

string
: Plain strings that are not in angle-brackets. Spaces are significant and preserved.

comments
: Delimited by "<!--" and "-->". All text between the delimiters is part of the comment. However, by convention, some comments actually contain data and so the methods that extract the fields from tags can be used to attempt to extract the fields from comments, too. Spaces are significant and preserved in a comment, unless the comment is treated as a tag, in which the tag rules apply.

This class is intended to parse markup languages, not to validate them. "Malformed" data is interpreted as graciously as possible, in order to extract as much information as possible. For instance: spaces are allowed between the "<" and the tag name, values in tags do not need to be quoted, and unbalanced quotes are accepted.

One type of "malformed" data specifically not handled is a quoted ">" character occurring within the body of a tag. Even if it is quoted, a ">" in the attributes of a tag will be interpreted as the end of the tag. For example, the single tag `<img src='foo.jpg' alt='xyz > abc'>` will be erroneously broken by this parser into two tokens:

- the tag `<img src='foo.jpg' alt='xyz >`
- the string "abc'>" (and possibly whatever text follows after).

Unfortunately, this type of "malformed" data is known to occur regularly.

This class also may not properly parse all well-formed XML tags, such as tags with extended paired delimiters `<&` and `&>`, `<?` and `?>`, or `<![CDATA[` and `]]>`. Additionally, XML tags that have embedded comments containing the ">" character will not be parsed correctly (for example: `<!DOCTYPE foo SYSTEM -- a > b -- foo.dtd>`), since the ">" in the comment will be interpreted as the end of declaration tag, for the same reason mentioned above [p ??] .
Note: this behavior may be changed on a per-application basis by overriding the `findClose` method in a

subclass.

Field Summary

`static int`	**`COMMENT`** The value returned by `getType` for comment tokens
`static int`	**`STRING`** The value returned by `getType` for string tokens
`static int`	**`TAG`** The value returned by `getType` for tag tokens

Constructor Summary

`LexML``(String str)` Create a new ML parser, which can be used to iterate over the tokens in the given string.

Method Summary

String	**getArgs()** Gets the name/value pairs in the body of the current tag as a string.
StringMap	**getAttributes()** Gets the name/value pairs in the body of the current tag as a table.
String	**getBody()** Gets the string making up the current token, not including the angle brackets or comment delimiters, if appropriate.
int	**getLocation()** Return the current processing location.
String	**getString()** Return the string we are currently processing
String	**getTag()** Gets the tag name at the beginning of the current tag.
String	**getToken()** Gets the string making up the whole current token, including the brackets or comment delimiters, if appropriate.
int	**getType()** Gets the type of the current token.
boolean	**isSingleton()** A tag is a "singleton" if the closing ">" is preceded by a slash (/).
boolean	**nextToken()** Advances to the next token.
void	**replace**(String str) Changes the string that this LexML is parsing.
String	**rest()** Gets the rest of the string that has not yet been parsed.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

COMMENT

`public static final int **COMMENT**`

The value returned by `getType` for comment tokens

TAG

`public static final int **TAG**`

The value returned by `getType` for tag tokens

STRING

`public static final int **STRING**`

The value returned by `getType` for string tokens

Constructor Detail

LexML

`public **LexML**(String str)`

Create a new ML parser, which can be used to iterate over the tokens in the given string.
Parameters:
`str` - The ML to parse.

Method Detail

nextToken

`public boolean **nextToken**()`

Advances to the next token. The user can then call the other methods in this class to get information about the new current token.
Returns:
`true` if a token was found, `false` if there were no more tokens left.

---

getType

`public int **getType**()`

Gets the type of the current token.

Returns:
The type.
See Also:
`COMMENT`, `TAG`, `STRING`

---

isSingleton

`public boolean **isSingleton**()`

A tag is a "singleton" if the closing ">" is preceded by a slash (/). (e.g. `<br/>`

---

getToken

`public String **getToken**()`

Gets the string making up the whole current token, including the brackets or comment delimiters, if appropriate.
Returns:
The current token.

---

getBody

`public String **getBody**()`

Gets the string making up the current token, not including the angle brackets or comment delimiters, if appropriate.
Returns:
The body of the token.

---

getString

`public String **getString**()`

Return the string we are currently processing

---

getLocation

`public int **getLocation**()`

Return the current processing location.
Returns:
The character index of the current tag.

---

getTag

`public String **getTag**()`

Gets the tag name at the beginning of the current tag. In other words, the tag name for `<table border=3>` is "table". Any surrounding space characters are removed, but the case of the tag is preserved.

For comments, the "tag" is the first word in the comment. This can be used to help parse comments that are structured similar to regular tags, such as server-side include comments like `<!--#include virtual="file.inc">`. The tag in this case would be "!--#include".

Returns:

The tag name, or `null` if the current token was a string.

---

getArgs

`public String **getArgs**()`

Gets the name/value pairs in the body of the current tag as a string.

Returns:

The name/value pairs, or `null` if the current token was a string.

---

getAttributes

`public StringMap **getAttributes**()`

Gets the name/value pairs in the body of the current tag as a table.

Any quote marks in the body, either single or double quotes, are left on the values, so that the values can be easily re-emitted and still form a valid body.

For names that have no associated value in the tag, the value is stored as the empty string "". Therefore, the two tags `<table border>` and `<table border="">` cannot be distinguished based on the result of calling `getAttributes`.

Returns:

The table of name/value pairs, or `null` if the current token was a string.

---

rest

`public String **rest**()`

Gets the rest of the string that has not yet been parsed.

Example use: to help the parser in circumstances such as the HTML "<script>" tag where the script body doesn't the obey the rules because it might contain lone "<" or ">" characters, which this parser would interpret as the start or end of funny-looking tags.

Returns:

The unparsed remainder of the string.

See Also:
`replace(java.lang.String)`

---

replace

```
public void replace(String str)
```

Changes the string that this LexML is parsing.

Example use: the caller decided to parse part of the body, and now wants this LexML to pick up and parse the rest of it.

Parameters:

`str` - The string that this LexML should now parse. Whatever string this LexML was parsing is forgotten, and it now starts parsing at the beginning of the new string.

See Also:
`rest()`

Interface SocketFactory

sunlabs.brazil.util
Interface SocketFactory

---

public interface **SocketFactory**

This interface is used as a heap to control the allocation of sockets.

An instance of this interface can be passed to methods that allocate sockets. In this way, the actual, underlying type of socket allocated can be replaced (for instance, with an SSL socket or an firewall-tunnelling socket), without the user of the socket having to explicitly be aware of the underlying implementation.

In some ways, this class is a replacement for the `SocketImplFactory` class. This class addresses the following issues.

- A `SocketImplFactory` may be installed only once for the entire process, so different policies cannot be used concurrently and/or consectively. For instance, imagine a situation where the user wants one part of the program talking via SSL to some port on machine A and via standard sockets to some port on machine B. It is not possible to install separate `SocketImplFactory` objects to allow both.
- The standard `Socket` class presumes a highly-connected network with the ability to resolve hostnames to IP addresses. The standard `Socket` class always converts the hostname to an IP address before calling `SocketImplFactory`. If the hostname does not have an IP address, then the `SocketImplFactory` never gets a chance to intercept the host name and perform alternate routing based on the name. For instance, imagine that the user has implemented a firewall-tunnelling socket; the raw hostname must be passed to the firewall machine, which allows the socket to be established once some out-of-band credentials are supplied. But we could never get this far because the standard `Socket` class would have already rejected the request since the IP address of the target machine was unknown.

Field Summary

`static SocketFactory`	**`defaultFactory`** The default socket factory.

Method Summary

`Socket`	**`newSocket`**`(String host, int port)` Creates a new `Socket` that talks to the specified port on the named host.

Field Detail

defaultFactory

```
public static final SocketFactory defaultFactory
```

The default socket factory. It just creates a standard `Socket` to the specified host and port, and is exactly equivalent to calling `new Socket(host, port)`.

Method Detail

newSocket

```
public Socket newSocket(String host,
                        int port)
                 throws IOException
```

Creates a new `Socket` that talks to the specified port on the named host.

The implementation may choose any way it wants to provide a socket-like object (essentially any mechanism that supports bidirectional communication). The returned `Socket` (or subclass of `Socket`) might not be based on TCP/IP, or it might involve running a TCP/IP stack over some other protocol, or it might actually redirect all connections via some other proxy machine, etc.

Parameters:

`host` - The host name.

`port` - The port number.

Returns:

An object that provides socket-like communication.

Throws:

IOException - If there is some problem establishing the socket to the specified port on the named host.

Interface Sort.Compare

sunlabs.brazil.util
Interface Sort.Compare
Enclosing class:
Sort

---

public static interface **Sort.Compare**

This interface is used by the `Sort` class to compare elements when an array is being sorted.

Method Summary

`int`	**`compare`**`(Object array, int index1, int index2)` Compare two elements in the given array.

Method Detail

compare

```
public int compare(Object array,
                   int index1,
                   int index2)
```

Compare two elements in the given array. The implementor must know what the actual type of the array is and cast it to that type in order to do the comparison.

Parameters:

`array` - Array being sorted.
`index1` - The index in the given array of the first element to compare.
`index2` - The index in the given array of the second element to compare.

Returns:

The implementation must return a number less than, equal to, or greater than zero to indicate whether the array element at `index1` should be considered less than, equal to, or greater than the array element at `index2`.

Class Sort

sunlabs.brazil.util
Class Sort

```
java.lang.Object
  |
  +--sunlabs.brazil.util.Sort
```

---

public class **Sort**
extends Object

Placeholder for useful sorting utilities. Currently, sorting arrays and Vectors using the qsort algorithm are preovided.

Inner Class Summary

static interface	**Sort.Compare** This interface is used by the `Sort` class to compare elements when an array is being sorted.

Method Summary

static void	**qsort**(Object array) Sorts an array of the basic types (ints, floats, bytes, etc.) or Strings.
static void	**qsort**(Object array, Sort.Compare compare) Sorts an array.
static void	**qsort**(Vector strings) Sort a vector of strings using the Qsort algorithm.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Method Detail

qsort

```
public static void qsort(Vector strings)
```

Sort a vector of strings using the Qsort algorithm. The compareTo method of the String class is used for comparison.

---

qsort

```
public static void qsort(Object array)
                  throws IllegalArgumentException
```

Sorts an array of the basic types (ints, floats, bytes, etc.) or Strings. The sort is in increasing order, and is case-sensitive for strings. Sorting an array of booleans or an array of objects other than Strings is not supported.

Parameters:

`array` - The array to sort in place.

Throws:

IllegalArgumentException - if `array` is not an array of the types listed above.

---

qsort

```
public static void qsort(Object array,
                         Sort.Compare compare)
                  throws IllegalArgumentException
```

Sorts an array. The specified comparator is used to control the sorting order. Arrays of any type may be sorted, depending upon what the comparator accepts.

Parameters:

`array` - The array to sort in place.

`compare` - The comparator for sort order.

Throws:

IllegalArgumentException - if `array` is not an array.

Class StringMap

sunlabs.brazil.util
Class StringMap

```
java.lang.Object
  |
  +--java.util.Dictionary
        |
        +--sunlabs.brazil.util.StringMap
```

Direct Known Subclasses:
AsteriskHandler.AmiStringMap, MimeHeaders

---

public class **StringMap**
extends Dictionary

The `StringMap` class is a substitute for the Hashtable. The StringMap has the following properties:

- Maps case-insensitive string keys to string values.
- The case of the keys is preserved.
- Values may be `null`.
- Preserves the relative order of the data.
- The same key may appear multiple times in a single map.
- This map is implemented via a Vector, and as such, as the number of keys increases, the time required to search will go up.

Constructor Summary

`StringMap()` Creates an empty StringMap.	

Method Summary

`void`	**`add`**`(String key, String value)` Maps the given case-insensitive key to the specified value in this StringMap.
`void`	**`append`**`(StringMap other, boolean noReplace)` Append another Stringmap onto this one.
`void`	**`clear`**`()` Removes all the keys and values from this StringMap.
`Enumeration`	**`elements`**`()` Returns an enumeration of the values in this StringMap.

String	**get**(int index) Returns the value at the specified index.
Object	**get**(Object key) Performs the same job as `get(String)`.
String	**get**(String key) Returns the value that the specified case-insensitive key maps to in this StringMap.
String	**getKey**(int index) Returns the key at the specified index.
boolean	**isEmpty**() Tests if there are any elements in this StringMap.
Enumeration	**keys**() Returns an enumeration of the keys in this StringMap.
void	**put**(int index, String value) Maps the key at the given index to the specified value in this StringMap.
Object	**put**(Object key, Object value) Performs the same job as `put(String, String)`.
void	**put**(String key, String value) Maps the given case-insensitive key to the specified value in this StringMap.
void	**remove**(int i)
Object	**remove**(Object key) Performs the same job as `remove(String)`.
void	**remove**(String key) Removes the given case-insensitive key and its corresponding value from this StringMap.
int	**size**() Returns the number of elements in this StringMap.
String	**toString**() Returns a string representation of this `StringMap` in the form of a set of entries, enclosed in braces and separated by the characters ", ".

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

StringMap

`public StringMap()`

Creates an empty StringMap.

Method Detail

size

`public int size()`

Returns the number of elements in this StringMap. Every occurrence of keys that appear multiple times is counted.

Overrides:

size in class Dictionary

Returns:

The number of elements in this StringMap.

See Also:

`keys()`

---

isEmpty

`public boolean isEmpty()`

Tests if there are any elements in this StringMap.

Overrides:

isEmpty in class Dictionary

Returns:

Returns `true` if there are no elements, `false` otherwise.

---

keys

`public Enumeration keys()`

Returns an enumeration of the keys in this StringMap. The elements of the enumeration are strings.

The same key may appear multiple times in the enumeration, not necessarily consecutively. Since `get` always returns the value associated with the first occurrence of a given key, a StringMap cannot be enumerated in the same fashion as a Hashtable. Instead, the caller should use:

```
Enumeration keys = map.keys();
Enumeration values = map.elements();
while (keys.hasMoreElements()) {
    String key = (String) keys.nextElement();
    String value = (String) values.nextElement();
}
```

or:

```
for (int i = 0; i < map.size(); i++) {
    String key = map.getKey(i);
    String value = map.get(i);
}
```

Overrides:
keys in class Dictionary
Returns:
An enumeration of the keys.
See Also:
`elements()`, `size()`, `getKey(int)`, `get(int)`

---

elements

```
public Enumeration elements()
```

Returns an enumeration of the values in this StringMap. The elements of the enumeration are strings.
Overrides:
elements in class Dictionary
Returns:
An enumeration of the values.
See Also:
`keys()`

---

getKey

```
public String getKey(int index)
              throws IndexOutOfBoundsException
```

Returns the key at the specified index. The index ranges from `0` to `size() - 1`.

This method can be used to iterate over all the keys in this StringMap in the order in which they were inserted, subject to any intervening deletions.
Parameters:
`index` - The index of the key.
Returns:
The key at the specified index.
Throws:
IndexOutOfBoundsException - if the index is out of the allowed range.

---

get

```
public String get(int index)
           throws IndexOutOfBoundsException
```

Returns the value at the specified index. The index ranges from `0` to `size() - 1`.

This method can be used to iterate over all the values in this StringMap in the order in which they were inserted, subject to any intervening deletions.

Parameters:

`index` - The index of the key.

Returns:

The value at the specified index.

Throws:

IndexOutOfBoundsException - if the index is out of the allowed range.

---

get

```
public String get(String key)
```

Returns the value that the specified case-insensitive key maps to in this StringMap.

The same key may appear multiple times in the enumeration; this method always returns the value associated with the first occurrence of the specified key. In order to get all the values, it is necessary to iterate over the entire StringMap to retrieve all the values associated with a given key.

Parameters:

`key` - A key in this StringMap. May not be `null`.

Returns:

The value to which the specified key is mapped, or `null` if the key is not in the StringMap.

See Also:

`keys()`

---

get

```
public Object get(Object key)
```

Performs the same job as `get(String)`. It exists so this class can extend the `Dictionary` class.

Overrides:

get in class Dictionary

Parameters:

`key` - Must be a String.

Returns:

A String value.

Throws:

ClassCastException - if the `key` is not a String.

See Also:
`get(String)`

---

put

```
public void put(int index,
                String value)
```

Maps the key at the given index to the specified value in this StringMap. The index ranges from `0` to `size() - 1`.

Parameters:
`index` - The index of the key.

Returns:
The value at the specified index.

Throws:
IndexOutOfBoundsException - if the index is out of the allowed range.

---

put

```
public void put(String key,
                String value)
```

Maps the given case-insensitive key to the specified value in this StringMap.

The value can be retrieved by calling `get` with a key that is case-insensitive equal to the given key.

If this StringMap already contained a mapping for the given key, the old value is forgotten and the new specified value is used. The case of the prior key is retained in that case. Otherwise the case of the new key is used.

Parameters:
`key` - The new key. May not be `null`.
`value` - The new value. May be `null`.

Returns:
The previous value to which `key` was mapped, or `null` if the the key did not map to any value.

---

put

```
public Object put(Object key,
                  Object value)
```

Performs the same job as `put(String, String)`. It exists so this class can extend the `Dictionary` class.

Overrides:
put in class Dictionary

Parameters:
`key` - Must be a String.
`value` - Must be a String.

Returns:
The previous value to which `key` was mapped, or `null` if the the key did not map to any value.
Throws:
ClassCastException - if the `key` or `value` is not a String.
See Also:
`put(String, String)`

---

add

```
public void add(String key,
                String value)
```

Maps the given case-insensitive key to the specified value in this StringMap.

The new mapping is added to this StringMap even if the given key already has a mapping. In this way it is possible to create a key that maps to two or more values.

Since the same key may appear multiple times in this StringMap, it is necessary to iterate over the entire StringMap to retrieve all values associated with a given key.
Parameters:
`key` - The new key. May not be `null`.
`value` - The new value. May be `null`.
See Also:
`put(String, String)`, `keys()`

---

remove

```
public void remove(String key)
```

Removes the given case-insensitive key and its corresponding value from this StringMap. This method does nothing if the key is not in this StringMap.

The same key may appear in multiple times in this StringMap; this method only removes the first occurrence of the key.
Parameters:
`key` - The key that needs to be removed. Must not be `null`.

---

remove

```
public void remove(int i)
```

---

remove

```
public Object remove(Object key)
```

Performs the same job as `remove(String)`. It exists so this class can extend the `Dictionary` class.

Overrides:

remove in class Dictionary

Parameters:

`key` - Must be a String.

Returns:

The string value to which the key had been mapped, or `null` if the key did not have a mapping.

Throws:

ClassCastException - if the `key` is not a String.

---

clear

```
public void clear()
```

Removes all the keys and values from this StringMap.

---

append

```
public void append(StringMap other,
                   boolean noReplace)
```

Append another Stringmap onto this one.

Parameters:

`other` - the map to append to this one

`noReplace` - should existing values be replaced?

---

toString

```
public String toString()
```

Returns a string representation of this `StringMap` in the form of a set of entries, enclosed in braces and separated by the characters ", ". Each entry is rendered as the key, an equals sign "=", and the associated value.

Overrides:

toString in class Object

Returns:

The string representation of this `StringMap`.

Package sunlabs.brazil.util.http

Utility classes for dealing with the HTTP protocol.

This package provides utility classes for dealing with the HTTP protocol. It is used internally by the `server` package, These utilities are made publicly available for use in those circumstances where direct manipulation of the http protocol is desired.

Class HttpInputStream

sunlabs.brazil.util.http
Class HttpInputStream

```
java.lang.Object
  |
  +--java.io.InputStream
        |
        +--java.io.FilterInputStream
              |
              +--sunlabs.brazil.util.http.HttpInputStream
```

---

public class **HttpInputStream**
extends FilterInputStream

This class is an input stream that provides added methods that are of help when reading the result of an HTTP request. By setting up this input stream, the user can conveniently read lines of text and copy the contents of an input stream to an output stream.

The underlying assumption of this class is that when reading the result of an HTTP request, each byte in the input stream represents an 8-bit ASCII character, and as such, it is perfectly valid to treat each byte as a character. Locale-based conversion is not appropriate in this circumstance, so the `java.io.BufferedReader.readLine` method should not be used.

Field Summary

`static int`	**`defaultBufsize`** The default size of the temporary buffer used when copying from an input stream to an output stream.

Constructor Summary

`HttpInputStream``(InputStream in)` Creates a new HttpInputStream that reads its input from the specified input stream.

Method Summary	
int	**copyTo**(OutputStream out) Copies bytes from this input stream to the specified output stream until end of the input stream is reached.
int	**copyTo**(OutputStream out, int len) Copies bytes from this input stream to the specified output stream until the specified number of bytes are copied or the end of the input stream is reached.
int	**copyTo**(OutputStream out, int len, byte[] buf) Copies bytes from this input stream to the specified output stream until the specified number of bytes are copied or the end of the input stream is reached.
int	**readFully**(byte[] buf) Reads buf.length bytes from the input stream.
int	**readFully**(byte[] buf, int off, int len) Reads the specified number of bytes from the input stream.
String	**readLine**() Reads the next line of text from the input stream.
String	**readLine**(int limit) Reads the next line of text from the input stream, up to the limit specified.

Methods inherited from class java.io.FilterInputStream
available, close, mark, markSupported, read, read, read, reset, skip

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

defaultBufsize

public static int **defaultBufsize**

The default size of the temporary buffer used when copying from an input stream to an output stream.

See Also:
`copyTo(OutputStream, int, byte[])`

Constructor Detail

HttpInputStream

```
public HttpInputStream(InputStream in)
```

Creates a new HttpInputStream that reads its input from the specified input stream.
Parameters:
`in` - The underlying input stream.

Method Detail

readLine

```
public String readLine()
                throws IOException
```

Reads the next line of text from the input stream.

A line is terminated by "\r", "\n", "\r\n", or the end of the input stream. The line-terminating characters are discarded.
Returns:
The next line from the input stream, or `null` if the end of the input stream is reached and no bytes were found.
Throws:
IOException - if the underlying input stream throws an IOException while being read.

---

readLine

```
public String readLine(int limit)
                throws IOException
```

Reads the next line of text from the input stream, up to the limit specified.

A line is terminated by "\r", "\n", "\r\n", the end of the input stream, or when the specified number of characters have been read. The line-terminating characters are discarded. It is not possible to distinguish, based on the result, between a line that was exactly `limit` characters long and a line that was terminated because `limit` characters were read.
Returns:
The next line from the input stream, or `null` if the end of the input stream is reached and no bytes were found.

Throws:
IOException - if the underlying input stream throws an IOException while being read.

---

readFully

```
public int readFully(byte[] buf)
              throws IOException
```

Reads `buf.length` bytes from the input stream. This method reads repeatedly from the input stream until the specified number of bytes have been read or the end of the input stream is reached.

The standard `InputStream.read` method will generally return less than the specified number of bytes if the underlying input stream is "bursty", such as from a network source. Sometimes it is important to read the exact number of bytes desired.

Parameters:
`buf` - Buffer in which the data is stored. If buffer is of length 0, this method will return immediately.

Returns:
The number of bytes read. This will be less than `buf.length` if the end of the input stream was reached.

Throws:
IOException - if the underlying input stream throws an IOException while being read.

---

readFully

```
public int readFully(byte[] buf,
                     int off,
                     int len)
              throws IOException
```

Reads the specified number of bytes from the input stream. This method reads repeatedly from the input stream until the specified number of bytes have been read or the end of the input stream is reached.

The standard `InputStream.read` method will generally return less than the specified number of bytes if the underlying input stream is "bursty", such as from a network source. Sometimes it is important to read the exact number of bytes desired.

Parameters:
`buf` - Buffer in which the data is stored.
`off` - The starting offset into the buffer.
`len` - The number of bytes to read.

Returns:
The number of bytes read. This will be less than `len` if the end of the input stream was reached.

Throws:
IOException - if the underlying input stream throws an IOException while being read.

---

copyTo

```
public int copyTo(OutputStream out)
           throws IOException
```

Copies bytes from this input stream to the specified output stream until end of the input stream is reached.

Parameters:

`out` - The output stream to copy the data to.

Returns:

The number of bytes copied to the output stream.

Throws:

IOException - if the underlying input stream throws an IOException while being read or if the output stream throws an IOException while being written. It may not be possible to distinguish amongst the two conditions.

---

copyTo

```
public int copyTo(OutputStream out,
                  int len)
           throws IOException
```

Copies bytes from this input stream to the specified output stream until the specified number of bytes are copied or the end of the input stream is reached.

Parameters:

`out` - The output stream to copy the data to.

`len` - The number of bytes to copy, or < 0 to copy until the end of this stream.

Returns:

The number of bytes copied to the output stream.

Throws:

IOException - if the underlying input stream throws an IOException while being read or if the output stream throws an IOException while being written. It may not be possible to distinguish amongst the two conditions.

---

copyTo

```
public int copyTo(OutputStream out,
                  int len,
                  byte[] buf)
           throws IOException
```

Copies bytes from this input stream to the specified output stream until the specified number of bytes are copied or the end of the input stream is reached.

Parameters:

`out` - The output stream to copy the data to.

`len` - The number of bytes to copy, or < 0 to copy until the end of this stream.

`buf` - The buffer used to for holding the temporary results while copying data from this input stream to the output stream. May be `null` to allow this method copy in chunks of length

`defaultBufsize`.

Returns:

The number of bytes read from the input stream, and copied to the output stream.

Throws:

IOException - if the underlying input stream throws an IOException while being read.

Class HttpRequest

sunlabs.brazil.util.http
Class HttpRequest

```
java.lang.Object
  |
  +--sunlabs.brazil.util.http.HttpRequest
```

---

public class **HttpRequest**
extends Object

Sends an HTTP request to some target host and gets the answer back. Similar to the `URLConnection` class.

Caches connections to hosts, and reuses them if possible. Talks HTTP/1.1 to the hosts, in order to keep alive connections as much as possible.

The sequence of events for using an `HttpRequest` is similar to how `URLConnection` is used:

1. A new `HttpRequest` object is constructed.
2. The setup parameters are modified:
   - `setMethod`
   - `setRequestHeader`
   - `getOutputStream`
3. The host (or proxy) is contacted and the HTTP request is issued:
   - `connect`
   - `getInputStream`
4. The response headers and body are examined:
   - `getResponseCode`
   - `getResponseHeader`
   - `getContentLength`
5. The connection is closed:
   - `close`

In the common case, all the setup parameters are initialized to sensible values and won't need to be modified. Most users will only need to construct a new `HttpRequest` object and then call `getInputStream` to read the contents. The rest of the member variables and methods are only needed for advanced behavior.

The `HttpRequest` class is intended to be a replacement for the `URLConnection` class. It operates at a lower level and makes fewer decisions on behavior. Some differences between the `HttpRequest` class and the `URLConnection` class follow:

- there are no undocumented global variables (specified in `System.getProperties`) that modify the behavior of `HttpRequest`.
- `HttpRequest` does not automatically follow redirects.
- `HttpRequest` does not turn HTTP responses with a status code other than "200 OK" into

`IOExceptions`. Sometimes it may be necessary and even quite useful to examine the results of an "unsuccessful" HTTP request.

- `HttpRequest` issues HTTP/1.1 requests and handles HTTP/0.9, HTTP/1.0, and HTTP/1.1 responses.
- the `URLConnection` class leaks open sockets if there is an error reading the response or if the target does not use Keep-Alive, and depends upon the garabge collector to close and release the open socket in these cases, which is unreliable because it may lead to intermittently running out of sockets if the garbage collector doesn't run often enough.
- If the user doesn't read all the data from an `URLConnection`, there are bugs in its implementation (as of JDK1.2) that may cause the program to block forever and/or read an insufficient amount of data before trying to reuse the underlying socket.

A number of the fields in the `HttpRequest` object are public, by design. Most of the methods mentioned above are convenience methods; the underlying data fields are meant to be accessed for more complicated operations, such as changing the socket factory or accessing the raw HTTP response line. Note however, that the order of the methods described above is important. For instance, the user cannot examine the response headers (by calling `getResponseHeader` or by examining the variable `responseHeaders`) without first having connected to the host.

However, if the user wants to modify the default behavior, the `HttpRequest` uses the value of a number of variables and automatically sets some HTTP headers when sending the request. The user can change these settings up until the time `connect` is called, as follows:

variable `version`

By default, the `HttpRequest` issues HTTP/1.1 requests. The user can set `version` to change this to HTTP/1.0.

variable `method`

If `method` is `null` (the default), the `HttpRequest` decides what the HTTP request method should be as follows: If the user has called `getOutputStream`, then the method will be "POST", otherwise the method will be "GET".

variable `proxyHost`

If the proxy host is specified, the HTTP request will be sent via the specified proxy:

- `connect` opens a connection to the proxy.
- uses the "Proxy-Connection" header to keep alive the connection.
- sends a fully qualified URL in the request line, for example "http://www.foo.com/index.html". The fully qualified URL tells the proxy to forward the request to the specified host.

Otherwise, the HTTP request will go directly to the host:

- `connect` opens a connection to the remote host.
- uses the "Connection" header to keep alive the connection.
- sends a host-relative URL in the request line, for example "/index.html". The relative URL is derived from the fully qualified URL used to construct this `HttpRequest`.

header "Connection" or "Proxy-Connection"

The `HttpRequest` sets the appropriate connection header to "Keep-Alive" to keep alive the connection to the host or proxy (respectively). By setting the appropriate connection header, the user can control whether the `HttpRequest` tries to use Keep-Alives.

header "Host"
: The HTTP/1.1 protocol requires that the "Host" header be set to the name of the machine being contacted. By default, this is derived from the URL used to construct the `HttpRequest`, and is set automatically if the user does not set it.

header "Content-Length"
: If the user calls `getOutputStream` and writes some data to it, the "Content-Length" header will be set to the amount of data that has been written at the time that `connect` is called.

Once all data has been read from the remote host, the underlying socket may be automatically recycled and used again for subsequent requests to the same remote host. If the user is not planning on reading all the data from the remote host, the user should call `close` to release the socket. Although it happens under the covers, the user should be aware that if an IOException occurs or once data has been read normally from the remote host, `close` is called automatically. This is to ensure that the minimal number of sockets are left open at any time.

The input stream that `getInputStream` provides automatically hides whether the remote host is providing HTTP/1.1 "chunked" encoding or regular streaming data. The user can simply read until reaching the end of the input stream, which signifies that all the available data from this request has been read. If reading from a "chunked" source, the data is automatically de-chunked as it is presented to the user. Currently, no access is provided to the underlying raw input stream.

Field Summary

`static String`	**`defaultHTTPVersion`** The default HTTP version string to send to the remote host when issuing requests.
`static String`	**`defaultProxyHost`** The default proxy host for HTTP requests.
`static int`	**`defaultProxyPort`** The default proxy port for HTTP requests.
`static boolean`	**`displayAllHeaders`** setting this to "true" causing all http headers to be printed on the standard error stream; useful for debugging client/server interactions.
`static int`	**`DRAIN_TIMEOUT`** Timeout (in msec) to drain an input stream that has been closed before the entire HTTP response has been read.
`String`	**`host`** The host extracted from the URL used to construct this `HttpRequest`.
`static int`	**`LINE_LIMIT`** Maximum length of a line in the HTTP response headers (sanity check).

String	**method** The HTTP method, such as "GET", "POST", or "HEAD".
static HttpSocketPool	**pool** The cache of idle sockets.
int	**port** The port extracted from the URL used to construct this `HttpRequest`.
String	**proxyHost** If non-`null`, sends this HTTP request via the specified proxy host and port.
int	**proxyPort** The proxy port.
MimeHeaders	**requestHeaders** The headers for the HTTP request.
MimeHeaders	**responseHeaders** The headers that were present in the HTTP response.
MimeHeaders	**responseTrailers** An artifact of HTTP/1.1 chunked encoding.
static SocketFactory	**socketFactory** The factory for constructing new Sockets objects used to connect to remote hosts when issuing HTTP requests.
String	**status** The status line from the HTTP response.
URL	**url** The URL used to construct this `HttpRequest`.
String	**version** The HTTP version string.

Constructor Summary

HttpRequest(String url) Creates a new `HttpRequest` object that will send an HTTP request to fetch the resource represented by the URL.
HttpRequest(URL url) Creates a new `HttpRequest` object that will send an HTTP request to fetch the resource represented by the URL.

Method Summary	
int	**addHeaders**(String tokens, Properties props) Convenience method for adding request headers by looking them up in a properties object.
void	**close**() Gracefully closes this HTTP request when user is done with it.
void	**connect**() Connect to the target host (or proxy), send the request, and read the response headers.
void	**disconnect**() Interrupts this HTTP request.
String	**getContent**() Return the content as a string.
String	**getContent**(String encoding) Get the content as a string.
int	**getContentLength**() Convenience method to get the "Content-Length" header from the HTTP response.
String	**getEncoding**()
HttpInputStream	**getInputStream**() Gets an input stream that can be used to read the body of the HTTP response.
OutputStream	**getOutputStream**() Gets an output stream that can be used for uploading data to the host.
int	**getResponseCode**() Gets the HTTP response status code.
String	**getResponseHeader**(String key) Gets the value associated with the given case-insensitive header name from the HTTP response.
static void	**removePointToPointHeaders**(MimeHeaders headers, boolean response) Removes all the point-to-point (hop-by-hop) headers from the given mime headers.
void	**setMethod**(String method) Sets the HTTP method to the specified value.

void	**setProxy**(String proxyHost, int proxyPort) Sets the proxy for this request.
void	**setRequestHeader**(String key, String value) Sets a request header in the HTTP request that will be issued.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

DRAIN_TIMEOUT

public static int **DRAIN_TIMEOUT**

Timeout (in msec) to drain an input stream that has been closed before the entire HTTP response has been read.

If the user closes the `HttpRequest` before reading all of the data, but the remote host has agreed to keep this socket alive, we need to read and discard the rest of the response before issuing a new request. If it takes longer than `DRAIN_TIMEOUT` to read and discard the data, we will just forcefully close the connection to the remote host rather than waiting to read any more.

Default value is 10000.

LINE_LIMIT

public static int **LINE_LIMIT**

Maximum length of a line in the HTTP response headers (sanity check).

If an HTTP response line is longer than this, the response is considered to be malformed.

Default value is 1000.

---

defaultHTTPVersion

public static String **defaultHTTPVersion**

The default HTTP version string to send to the remote host when issuing requests.

The default value can be overridden on a per-request basis by setting the `version` instance variable.

Default value is "HTTP/1.1".
See Also:
`version`

---

defaultProxyHost

`public static String` **`defaultProxyHost`**

The default proxy host for HTTP requests. If non-`null`, then all new HTTP requests will be sent via this proxy. If `null`, then all new HTTP requests are sent directly to the host specified when the `HttpRequest` object was constructed.

The default value can be overridden on a per-request basis by calling the `setProxy` method or setting the `proxyHost` instance variables.

Default value is `null`.
See Also:
`defaultProxyPort`, `proxyHost`, `setProxy(java.lang.String, int)`

---

defaultProxyPort

`public static int` **`defaultProxyPort`**

The default proxy port for HTTP requests.

Default value is `80`.
See Also:
`defaultProxyHost`, `proxyPort`

---

socketFactory

`public static SocketFactory` **`socketFactory`**

The factory for constructing new Sockets objects used to connect to remote hosts when issuing HTTP requests. The user can set this to provide a new type of socket, such as SSL sockets.

Default value is `null`, which signifies plain sockets.

---

pool

`public static HttpSocketPool` **`pool`**

The cache of idle sockets. Once a request has been handled, the now-idle socket can be remembered and reused later if another HTTP request is made to the same remote host.

---

url

`public URL **url**`

The URL used to construct this `HttpRequest`.

---

host

`public String **host**`

The host extracted from the URL used to construct this `HttpRequest`.
See Also:
`url`

---

port

`public int **port**`

The port extracted from the URL used to construct this `HttpRequest`.
See Also:
`url`

---

proxyHost

`public String **proxyHost**`

If non-`null`, sends this HTTP request via the specified proxy host and port.

Initialized from `defaultProxyHost`, but may be changed by the user at any time up until the HTTP request is actually sent.
See Also:
`defaultProxyHost`, `proxyPort`, `setProxy(java.lang.String, int)`, `connect()`

---

proxyPort

`public int **proxyPort**`

The proxy port.
See Also:
`proxyHost`

---

method

`public String` **`method`**

The HTTP method, such as "GET", "POST", or "HEAD".

May be set by the user at any time up until the HTTP request is actually sent.

---

version

`public String` **`version`**

The HTTP version string.

Initialized from `defaultHTTPVersion`, but may be changed by the user at any time up until the HTTP request is actually sent.

---

requestHeaders

`public MimeHeaders` **`requestHeaders`**

The headers for the HTTP request. All of these headers will be sent when the connection is actually made.

---

displayAllHeaders

`public static boolean` **`displayAllHeaders`**

setting this to "true" causing all http headers to be printed on the standard error stream; useful for debugging client/server interactions.

---

status

`public String` **`status`**

The status line from the HTTP response. This field is not valid until after `connect` has been called and the HTTP response has been read.

---

responseHeaders

`public MimeHeaders` **`responseHeaders`**

The headers that were present in the HTTP response. This field is not valid until after `connect` has been called and the HTTP response has been read.

---

responseTrailers

`public MimeHeaders **responseTrailers**`

An artifact of HTTP/1.1 chunked encoding. At the end of an HTTP/1.1 chunked response, there may be more MimeHeaders. It is only possible to access these MimeHeaders after all the data from the input stream returned by `getInputStream` has been read. At that point, this field will automatically be initialized to the set of any headers that were found. If not reading from an HTTP/1.1 chunked source, then this field is irrelevant and will remain `null`.

Constructor Detail

HttpRequest

`public **HttpRequest**(URL url)`

Creates a new `HttpRequest` object that will send an HTTP request to fetch the resource represented by the URL.

The host specified by the URL is **not** contacted at this time.
Parameters:
`url` - A fully qualified "http:" URL.
Throws:
IllegalArgumentException - if `url` is not an "http:" URL.

HttpRequest

`public **HttpRequest**(String url)`

Creates a new `HttpRequest` object that will send an HTTP request to fetch the resource represented by the URL.

The host specified by the URL is **not** contacted at this time.
Parameters:
`url` - A string representing a fully qualified "http:" URL.
Throws:
IllegalArgumentException - if `url` is not a well-formed "http:" URL.

Method Detail

setMethod

`public void **setMethod**(String method)`

Sets the HTTP method to the specified value. Some of the normal HTTP methods are "GET", "POST", "HEAD", "PUT", "DELETE", but the user can set the method to any value desired.

If this method is called, it must be called before `connect` is called. Otherwise it will have no effect.

Parameters:

`method` - The string for the HTTP method, or `null` to allow this `HttpRequest` to pick the method for itself.

---

setProxy

```
public void setProxy(String proxyHost,
                     int proxyPort)
```

Sets the proxy for this request. The HTTP proxy request will be sent to the specified proxy host.

If this method is called, it must be called before `connect` is called. Otherwise it will have no effect.

Parameters:

`proxyHost` - The proxy that will handle the request, or `null` to not use a proxy.

`proxyPort` - The port on the proxy, for the proxy request. Ignored if `proxyHost` is `null`.

---

setRequestHeader

```
public void setRequestHeader(String key,
                             String value)
```

Sets a request header in the HTTP request that will be issued. In order to do fancier things like appending a value to an existing request header, the user may directly access the `requestHeaders` variable.

If this method is called, it must be called before `connect` is called. Otherwise it will have no effect.

Parameters:

`key` - The header name.

`value` - The value for the request header.

See Also:

`requestHeaders`

---

getOutputStream

```
public OutputStream getOutputStream()
                             throws IOException
```

Gets an output stream that can be used for uploading data to the host.

If this method is called, it must be called before `connect` is called. Otherwise it will have no effect.

Currently the implementation is not as good as it could be. The user should avoid uploading huge amounts of data, for some definition of huge.

---

connect

```
public void connect()
             throws UnknownHostException,
                    IOException
```

Connect to the target host (or proxy), send the request, and read the response headers. Any setup routines must be called before the call to this method, and routines to examine the result must be called after this method.

Throws:

UnknownHostException - if the target host (or proxy) could not be contacted.
IOException - if there is a problem writing the HTTP request or reading the HTTP response headers.

---

getInputStream

```
public HttpInputStream getInputStream()
                              throws IOException
```

Gets an input stream that can be used to read the body of the HTTP response. Unlike the other convenience methods for accessing the HTTP response, this one automatically connects to the target host if not already connected.

The input stream that `getInputStream` provides automatically hides the differences between "Content-Length", no "Content-Length", and "chunked" for HTTP/1.0 and HTTP/1.1 responses. In all cases, the user can simply read until reaching the end of the input stream, which signifies that all the available data from this request has been read. (If reading from a "chunked" source, the data is automatically de-chunked as it is presented to the user. There is no way to access the raw underlying stream that contains the HTTP/1.1 chunking packets.)

Throws:

IOException - if there is problem connecting to the target.

See Also:

`connect()`

---

close

```
public void close()
```

Gracefully closes this HTTP request when user is done with it.

The user can either call this method or `close` on the input stream obtained from the `getInputStream` method -- the results are the same.

When all the response data is read from the input stream, the input stream is automatically closed (recycled). If the user is not going to read all the response data from input stream, the user must call `close` to release the resources associated with the open request. Otherwise the program may consume all available sockets, waiting forever for the user to finish reading.

Note that the input stream is automatically closed if the input stream throws an exception while reading.

In order to interrupt a pending I/O operation in another thread (for example, to stop a request that is taking too long), the user should call `disconnect` or interrupt the blocked thread. The user should not call `close` in this case because `close` will not interrupt the pending I/O operation.

Closing the request multiple times is allowed.

In order to make sure that open sockets are not left lying around the user should use code similar to the following:

```
OutputStream out = ...
HttpRequest http = new HttpRequest("http://bob.com/index.html");
try {
    HttpInputStream in = http.getInputStream();
    in.copyTo(out);
} finally {
    // Copying to "out" could have failed.  Close "http" in case
    // not all the data has been read from it yet.
    http.close();
}
```

---

disconnect

`public void` **`disconnect`**`()`

Interrupts this HTTP request. Can be used to halt an in-progress HTTP request from another thread, by causing it to throw an `InterruptedIOException` during the connect or while reading from the input stream, depending upon what state this HTTP request is in when it is disconnected.
See Also:
`close()`

---

getResponseCode

`public int` **`getResponseCode`**`()`

Gets the HTTP response status code. From responses like:

```
HTTP/1.0 200 OK
HTTP/1.0 401 Unauthorized
```

this method extracts the integers `200` and `401` respectively. Returns `-1` if the response status code was malformed.

If this method is called, it must be called after `connect` has been called. Otherwise the information is not yet available and this method will return `-1`.

For advanced features, the user can directly access the `status` variable.

Returns:

The integer status code from the HTTP response.

See Also:

`connect()`, `status`

---

getResponseHeader

```
public String getResponseHeader(String key)
```

Gets the value associated with the given case-insensitive header name from the HTTP response.

If this method is called, it must be called after `connect` has been called. Otherwise the information is not available and this method will return `null`.

For advanced features, such as enumerating over all response headers, the user should directly access the `responseHeaders` variable.

Parameters:

`key` - The case-insensitive name of the response header.

Returns:

The value associated with the given name, or `null` if there is no such header in the response.

See Also:

`connect()`, `responseHeaders`

---

getContentLength

```
public int getContentLength()
```

Convenience method to get the "Content-Length" header from the HTTP response.

If this method is called, it must be called after `connect` has been called. Otherwise the information is not available and this method will return `-1`.

Returns:

The content length specified in the response headers, or `-1` if the length was not specified or malformed (not a number).

See Also:

`connect()`, `getResponseHeader(java.lang.String)`

---

removePointToPointHeaders

```
public static void removePointToPointHeaders(MimeHeaders headers,
                                             boolean response)
```

Removes all the point-to-point (hop-by-hop) headers from the given mime headers.

Parameters:

`headers` - The mime headers to be modified.

`response` - `true` to remove the point-to-point **response** headers, `false` to remove the

point-to-point **request** headers.

See Also:

RFC 2068

---

addHeaders

```
public int addHeaders(String tokens,
                      Properties props)
```

Convenience method for adding request headers by looking them up in a properties object.

Parameters:

`tokens` - a white space delimited set of tokens that refer to headers that will be added to the HTTP request.

`props` - Keys of the form `[token].name` and `[token].value` are used to lookup additional HTTP headers to be added to the request.

Returns:

The number of headers added to the request

See Also:

`setRequestHeader(java.lang.String, java.lang.String)`

---

getContent

```
public String getContent(String encoding)
                  throws IOException,
                         UnsupportedEncodingException
```

Get the content as a string. Uses the character encoding specified in the HTTP headers if available. Otherwise the supplied encoding is used, or (if encoding is null), the platform default encoding.

Parameters:

`encoding` - The ISO character encoding to use, if the encoding can't be determined by context.

Returns:

The content as a string.

---

getContent

```
public String getContent()
                  throws IOException,
                         UnsupportedEncodingException
```

Return the content as a string.

---

getEncoding

```
public String getEncoding()
```

Class HttpSocket

sunlabs.brazil.util.http
Class HttpSocket

```
java.lang.Object
  |
  +--sunlabs.brazil.util.http.HttpSocket
```

---

public class **HttpSocket**
extends Object

This class is used as the bag of information kept about a open, idle socket. It is not meant to be used externally by anyone except someone writing a new implementation of an `HttpSocketPool` for the `HttpRequest` object.

This class should not be visible at this scope. It is only here until a better place for it is found.

Field Summary	
boolean	**firstTime**
String	**host**
InputStream	**in**
long	**lastUsed**
OutputStream	**out**
int	**port**
Socket	**sock**
int	**timesUsed**

Constructor Summary

HttpSocket(String host, int port)	

Method Summary

String	**toString**()

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

host

public String **host**

---

port

public int **port**

---

firstTime

public boolean **firstTime**

---

lastUsed

public long **lastUsed**

---

timesUsed

public int **timesUsed**

---

sock

```
public Socket sock
```

---

in

```
public InputStream in
```

---

out

```
public OutputStream out
```

Constructor Detail

HttpSocket

```
public HttpSocket(String host,
                  int port)
           throws IOException,
                  UnknownHostException
```

Method Detail

toString

```
public String toString()
```

Overrides:
toString in class Object

Interface HttpSocketPool

sunlabs.brazil.util.http
Interface HttpSocketPool

---

public interface **HttpSocketPool**

This interface represents a cache of idle sockets. Once a request has been handled, the now-idle socket can be remembered and reused later in case another HTTP request is made to the same remote host. Currently, the only instance of this interface is used by the `HttpRequest` class.

Method Summary

`void`	**`close`**`(HttpSocket hs, boolean reuse)` Releases an `HttpSocket` to this pool when it is not in use any more.
`HttpSocket`	**`get`**`(String host, int port, boolean reuse)` Returns an `HttpSocket` that can be used to communicate with the specified port on the named host.

Method Detail

get

```
public HttpSocket get(String host,
                      int port,
                      boolean reuse)
               throws IOException
```

Returns an `HttpSocket` that can be used to communicate with the specified port on the named host.

It is this method's responsibility to to fill in all the public member variables of the `HttpSocket` before returning.

For each call to this method, there should eventually be a call to `close` when the `HttpSocket` isn't needed anymore.

Parameters:

`host` - The host name.
`port` - The port number.
`reuse` - `true` to request that this pool attempt to find and reuse an existing idle connection, `false` to request that this pool establish a new connection to the named host.

Returns:

The `HttpSocket`.

Throws:

IOException - if there is a problem connecting to the specified port on the named host. The `IOException`s (and subclasses) that might be thrown depend upon how the socket connection is established. See the socket documentation for further details. Some subclasses that might be thrown are as follows:
java.io.UnknownHostException - if the host name cannot be resolved.
java.io.ConnectionException - if the named host is not listening on the specified port.
InterruptedIOException - if the connection times out or this thread is interrupted by `Thread.interrupt`.

---

close

```
public void close(HttpSocket hs,
                  boolean reuse)
```

Releases an `HttpSocket` to this pool when it is not in use any more.

It is this method's responsibility to release resources used by the `HttpSocket`, such as closing the underlying socket.

After calling this method, the user should not refer to the specified `HttpSocket` any more.

Parameters:

`hs` - The `HttpSocket` to release.
`reuse` - `true` if the specified `HttpSocket` should be put back into the idle pool, `false` if it should be released immediately.

Class HttpUtil

sunlabs.brazil.util.http
Class HttpUtil

```
java.lang.Object
  |
  +--sunlabs.brazil.util.http.HttpUtil
```

---

public class **HttpUtil**
extends Object

The `HttpUtil` class contains methods for performing simple HTTP operations.

Method Summary

static void	**extractQuery**(String query, Dictionary table) Turns x-www-form-urlencoded form data into a dictionary.
static String	**extractUrlHost**(String url) Get the host portion of a Url String.
static String	**extractUrlPath**(String url) Get the path portion of a Url String.
static String	**extractUrlPort**(String url) Get the port portion of a Url String as a string.
static String	**extractUrlProtocol**(String url) Get the protocol portion of a Url String.
static String	**formatTime**() Returns a string containing the current time as an HTTP-formatted date.
static String	**formatTime**(long time) Returns a string containing an HTTP-formatted date.
static String	**getStatusPhrase**(int code) Returns the HTTP error string associated with the integer error code.
static String	**htmlEncode**(String src) Converts a string into a valid HTML fragment.
static long	**parseTime**(String time) Convert a last-modified date in "standard" format into a time stamp.
static String	**urlDecode**(String src) Decodes a URL-encoded string by replacing all the "%XX" escape sequences in the string with the corresponding character.
static String	**urlEncode**(String src) Maps a string to be used in a query or post into a form that is acceptable in an URL.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Method Detail

htmlEncode

```
public static String htmlEncode(String src)
```

Converts a string into a valid HTML fragment. Escapes the characters ", &, <, >, and all non-printables into the form `&#xx;` (their "decimal reference" form).

Parameters:

`src` - The string to convert.

Returns:

The string with all the special characters converted to decimal reference form.

---

urlEncode

```
public static String urlEncode(String src)
```

Maps a string to be used in a query or post into a form that is acceptable in an URL. Typically used when the caller wants to safely generate an HREF containing an arbitrary string that may have special characters.

URL strings may not contain non-alphanumeric characters. All non-alphanumeric characters are converted to the escape sequence "%XX", where XX is the hexadecimal value of that character's code.

Note that the space character " " is NOT converted to "+". That is a common misconception. "+" represents a space only in query strings, not in the URL. "%20" is how an actual space character must be passed in an URL, and is also an acceptable way of passing a space in a query string.

Parameters:

`string` - The string to convert.

Returns:

The URL-encoded version of the given string.

---

urlDecode

```
public static String urlDecode(String src)
```

Decodes a URL-encoded string by replacing all the "%XX" escape sequences in the string with the corresponding character.

Malformed "%XX" sequences are silently ignored.

Parameters:

`string` - The URL-encoded string.

Returns:

The decoded version of the given string.

---

formatTime

```
public static String formatTime()
```

Returns a string containing the current time as an HTTP-formatted date.
Returns:
HTTP date string representing the current time.

---

formatTime

```
public static String formatTime(long time)
```

Returns a string containing an HTTP-formatted date.
Parameters:
`time` - The date to format (current time in msec).
Returns:
HTTP date string representing the given time.

---

parseTime

```
public static long parseTime(String time)
```

Convert a last-modified date in "standard" format into a time stamp. This "inverses" formatTime.
Parameters:
`time` - A correctly formatted HTTP date string.
Returns:
milliseconds since the epoch, or 0 if the conversion failed.

---

extractQuery

```
public static void extractQuery(String query,
                                Dictionary table)
```

Turns x-www-form-urlencoded form data into a dictionary.
Parameters:
`query` - The x-www-form-urlencoded string. May be `null`
`table` - The dictionary to insert the form data into.

---

extractUrlProtocol

```
public static String extractUrlProtocol(String url)
```

Get the protocol portion of a Url String.
Returns:
null if the string is an invalid URL.

---

extractUrlHost

```
public static String extractUrlHost(String url)
```

Get the host portion of a Url String.
Returns:
null if the string is an invalid URL.

---

extractUrlPort

```
public static String extractUrlPort(String url)
```

Get the port portion of a Url String as a string.
Returns:
null if the string is an invalid URL, the empy string if no port was specified.

---

extractUrlPath

```
public static String extractUrlPath(String url)
```

Get the path portion of a Url String.
Returns:
null if the string is an invalid URL.

---

getStatusPhrase

```
public static String getStatusPhrase(int code)
```

Returns the HTTP error string associated with the integer error code. This error string can be used in HTTP responses. Unknown codes return the string "Error"
Parameters:
`code` - The code to look up.
Returns:
The associated error string.

Class MimeHeaders

sunlabs.brazil.util.http
Class MimeHeaders

```
java.lang.Object
  |
  +--java.util.Dictionary
        |
        +--sunlabs.brazil.util.StringMap
              |
              +--sunlabs.brazil.util.http.MimeHeaders
```

---

public class **MimeHeaders**
extends StringMap

This class is build on top of the `StringMap` class and provides added methods that are of help when manipulating MIME headers. By creating an instance of this class, the user can conveniently read, write, and modify MIME headers.

Field Summary

static int	**MAX_LINE**
static int	**MAX_LINES**

Constructor Summary

MimeHeaders() Creates a new, empty `MimeHeaders` object.
MimeHeaders(HttpInputStream in) Creates a new `MimeHeaders` object and then initializes it by reading MIME headers from the specified input stream.

Method Summary

void	**add**(String key, int value) Adds a mapping for the given case-insensitive key to the specified value in this MimeHeaders object.
void	**copyTo**(MimeHeaders other) Copies the contents of this MimeHeaders object, adding all the other's keys and values to the other.
void	**print**(OutputStream out) Writes this MimeHeaders object to the given output stream.
void	**print**(PrintStream out) Writes this MimeHeaders object to the given output stream.
void	**put**(String key, int value) Maps the given case-insensitive key to the specified value in this MimeHeaders object, replacing the old value.
void	**putIfNotPresent**(String key, String value) Maps the given case-insensitive key to the specified value if the key does not already exist in this MimeHeaders object.
void	**read**(HttpInputStream in) Reads MIME headers from the specified input stream.
void	**read**(HttpInputStream in, boolean shouldReplace) Reads MIME headers from the specified input stream.

Methods inherited from class sunlabs.brazil.util.StringMap
add, append, clear, elements, get, get, get, getKey, isEmpty, keys, put, put, put, remove, remove, remove, size, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

MAX_LINE

```
public static final int MAX_LINE
```

MAX_LINES

```
public static final int MAX_LINES
```

Constructor Detail

MimeHeaders

```
public MimeHeaders()
```

Creates a new, empty `MimeHeaders` object.

MimeHeaders

```
public MimeHeaders(HttpInputStream in)
            throws IOException
```

Creates a new `MimeHeaders` object and then initializes it by reading MIME headers from the specified input stream.
Parameters:
`in` - The input stream to read.

Method Detail

read

```
public void read(HttpInputStream in)
          throws IOException
```

Reads MIME headers from the specified input stream. This method reads up to and consumes the blank line that marks the end of the MIME headers. It also stops reading if it reaches the end of the input stream.

The MIME headers read from the input stream are stored in this `MimeHeaders` object. All headers read are **added** to the existing headers; the new headers do not replace the existing ones. The order of the headers in this object will reflect the order of the headers from the input stream, but space characters surrounding the keys and values are not preserved.

In a set of MIME headers, the given key may appear multiple times (that is, on multiple lines, not necessarily consecutively). In that case, that key will appear multiple times in this `MimeHeaders` object also. The HTTP spec says that if a given key appears multiple times in a set of MIME headers, the values can be concatenated together with commas between them. However, in practice, it appears that some browsers and HTTP servers get confused when encountering such collapsed MIME headers, for instance, the Yahoo mail reader program.

MIME headers also support the idea of continuation lines, where a key (and optionally its value) is followed on subsequent line(s) by another value without a key. The HTTP spec says that in this case the values can be concatenated together with space characters between them. In practice, joining continuation lines together does not seem to confuse any browsers or HTTP servers. This method joins continuation lines together by putting the space-equivalent characters "\r\n\t" between the values so it can be easily parsed with `StringTokenizer` and also easily written to an output stream in a format that actually preserves its formatting as a continuation line.

Parameters:

`in` - The input stream to read from.

Throws:

IOException - if the input stream throws an IOException while being read.

---

read

```
public void read(HttpInputStream in,
                 boolean shouldReplace)
          throws IOException
```

Reads MIME headers from the specified input stream. Same as (@link #read(HttpInputStream in)), only existing keys may be replaced or augmented.

Parameters:

`in` - The input stream to read from.

`shouldReplace` - If true, existing keys are replaced instead of augmented.

Throws:

IOException - if the input stream throws an IOException while being read.

See Also:

`read(HttpInputStream in)`

---

print

```
public void print(OutputStream out)
```

Writes this `MimeHeaders` object to the given output stream. This method does `not` write a blank line after the headers are written.

Parameters:

`out` - The output stream.

---

print

```
public void print(PrintStream out)
```

Writes this `MimeHeaders` object to the given output stream. This method does `not` write a blank line after the headers are written.

Parameters:

`out` - The output stream.

---

putIfNotPresent

```
public void putIfNotPresent(String key,
                            String value)
```

Maps the given case-insensitive key to the specified value if the key does not already exist in this `MimeHeaders` object.

Often, when dealing with MIME headers, the user will want to set a header only if that header is not already set.

Parameters:

`key` - The new key. May not be `null`.
`value` - The new value. May be `null`.

---

put

```
public void put(String key,
                int value)
```

Maps the given case-insensitive key to the specified value in this `MimeHeaders` object, replacing the old value.

This is convenience method that automatically converts the integer value to a string before calling the underlying `put` method.

Parameters:

`key` - The new key. May not be `null`.
`value` - The new value.

---

add

```
public void add(String key,
                int value)
```

Adds a mapping for the given case-insensitive key to the specified value in this `MimeHeaders` object. It leaves any existing key-value mapping alone.

This is convenience method that automatically converts the integer value to a string before calling the underlying `add` method.

Parameters:

`key` - The new key. May not be `null`.
`value` - The new value.

---

copyTo

```
public void copyTo(MimeHeaders other)
```

Copies the contents of this `MimeHeaders` object, `adding` all the other's keys and values to the

other.

Parameters:

`other` - The `MimeHeaders` object to copy to.

Package sunlabs.brazil.util.regexp

This package contains a converted-to-Java-language version of Henry Spencer's regular expression package contained in TCL version 8.0.

Interface Regexp.Filter

sunlabs.brazil.util.regexp
Interface Regexp.Filter
Enclosing class:
Regexp

---

public static interface **Regexp.Filter**

This interface is used by the `Regexp` class to generate the replacement string for each pattern match found in the source string.

Method Summary

boolean	**filter**(Regsub rs, StringBuffer sb) Given the current state of the match, generate the replacement string.

Method Detail

filter

```
public boolean filter(Regsub rs,
                      StringBuffer sb)
```

Given the current state of the match, generate the replacement string. This method will be called for each match found in the source string, unless this filter decides not to handle any more matches.

The implementation can use whatever rules it chooses to generate the replacement string. For example, here is an example of a filter that replaces the first **5** occurrences of "%XX" in a string with the ASCII character represented by the hex digits "XX":

```
String str = ...;

Regexp re = new Regexp("%[a-fA-F0-9][a-fA-F0-9]");

Regexp.Filter rf = new Regexp.Filter() {
    int count = 5;
    public boolean filter(Regsub rs, StringBuffer sb) {
        String match = rs.matched();
        int hi = Character.digit(match.charAt(1), 16);
        int lo = Character.digit(match.charAt(2), 16);
        sb.append((char) ((hi << 4) | lo));
        return (--count > 0);
```

```
    }
}

String result = re.sub(str, rf);
```

Parameters:

`rs` - `Regsub` containing the state of the current match.

`sb` - The string buffer that this filter should append the generated string to. This string buffer actually contains the results the calling `Regexp` has generated up to this point.

Returns:

`false` if no further matches should be considered in this string, `true` to allow `Regexp` to continue looking for further matches.

Class Regexp

sunlabs.brazil.util.regexp
Class Regexp

```
java.lang.Object
  |
  +--sunlabs.brazil.util.regexp.Regexp
```

All Implemented Interfaces:
Serializable

---

public class **Regexp**
extends Object
implements Serializable

The `Regexp` class can be used to match a pattern against a string and optionally replace the matched parts with new strings.

Regular expressions were implemented by translating Henry Spencer's regular expression package for tcl8.0. Much of the description below is copied verbatim from the tcl8.0 regsub manual entry.

REGULAR EXPRESSIONS

A regular expression is zero or more `branches`, separated by "|". It matches anything that matches one of the branches.

A branch is zero or more `pieces`, concatenated. It matches a match for the first piece, followed by a match for the second piece, etc.

A piece is an `atom`, possibly followed by "*", "+", or "?".
- An atom followed by "*" matches a sequence of 0 or more matches of the atom.
- An atom followed by "+" matches a sequence of 1 or more matches of the atom.
- An atom followed by "?" matches either 0 or 1 matches of the atom.

An atom is
- a regular expression in parentheses (matching a match for the regular expression)
- a `range` (see below)
- "." (matching any single character)
- "^" (matching the null string at the beginning of the input string)
- "$" (matching the null string at the end of the input string)
- a "\" followed by a single character (matching that character)
- a single character with no other significance (matching that character).

A `range` is a sequence of characters enclosed in "[]". The range normally matches any single character from the sequence. If the sequence begins with "^", the range matches any single character **not** from the rest of the sequence. If two characters in the sequence are separated by "-", this is shorthand for the full list of characters between them (e.g. "[0-9]" matches any decimal digit). To include a literal "]" in the sequence, make it the first character (following a possible "^"). To include a literal "-", make it the first or last character.

In general there may be more than one way to match a regular expression to an input string. For example, consider the command

```
String[] match = new String[2];
Regexp.match("(a*)b*", "aabaaabb", match);
```

Considering only the rules given so far, `match[0]` and `match[1]` could end up with the values

- "aabb" and "aa"
- "aaab" and "aaa"
- "ab" and "a"

or any of several other combinations. To resolve this potential ambiguity, Regexp chooses among alternatives using the rule "first then longest". In other words, it considers the possible matches in order working from left to right across the input string and the pattern, and it attempts to match longer pieces of the input string before shorter ones. More specifically, the following rules apply in decreasing order of priority:

1. If a regular expression could match two different parts of an input string then it will match the one that begins earliest.
2. If a regular expression contains "|" operators then the leftmost matching sub-expression is chosen.
3. In "*", "+", and "?" constructs, longer matches are chosen in preference to shorter ones.
4. In sequences of expression components the components are considered from left to right.

In the example from above, "(a*)b*" therefore matches exactly "aab"; the "(a*)" portion of the pattern is matched first and it consumes the leading "aa", then the "b*" portion of the pattern consumes the next "b". Or, consider the following example:

```
String match = new String[3];
Regexp.match("(ab|a)(b*)c", "abc", match);
```

After this command, `match[0]` will be "abc", `match[1]` will be "ab", and `match[2]` will be an empty string. Rule 4 specifies that the "(ab|a)" component gets first shot at the input string and Rule 2 specifies that the "ab" sub-expression is checked before the "a" sub-expression. Thus the "b" has already been claimed before the "(b*)" component is checked and therefore "(b*)" must match an empty string.

REGULAR EXPRESSION SUBSTITUTION

Regular expression substitution matches a string against a regular expression, transforming the string by replacing the matched region(s) with new substring(s).

What gets substituted into the result is controlled by a `subspec`. The subspec is a formatting string that specifies what portions of the matched region should be substituted into the result.

- "&" or "\0" is replaced with a copy of the entire matched region.
- "\`n`", where `n` is a digit from 1 to 9, is replaced with a copy of the `n`*th* subexpression.

- "\&" or "\\" are replaced with just "&" or "\" to escape their special meaning.
- any other character is passed through.

In the above, strings like "\2" represents the two characters `backslash` and "2", not the Unicode character 0002.

Here is an example of how to use Regexp

```
public static void
main(String[] args)
    throws Exception
{
    Regexp re;
    String[] matches;
    String s;

    /*
     * A regular expression to match the first line of a HTTP request.
     *
     * 1. ^               - starting at the beginning of the line
     * 2. ([A-Z]+)        - match and remember some upper case characters
     * 3. [ \t]+          - skip blank space
     * 4. ([^ \t]*)       - match and remember up to the next blank space
     * 5. [ \t]+          - skip more blank space
     * 6. (HTTP/1\\.[01]) - match and remember HTTP/1.0 or HTTP/1.1
     * 7. $               - end of string - no chars left.
     */

    s = "GET http://a.b.com:1234/index.html HTTP/1.1";

    re = new Regexp("^([A-Z]+)[ \t]+([^ \t]+)[ \t]+(HTTP/1\\.[01])$");
    matches = new String[4];
    if (re.match(s, matches)) {
        System.out.println("METHOD  " + matches[1]);
        System.out.println("URL     " + matches[2]);
        System.out.println("VERSION " + matches[3]);
    }

    /*
     * A regular expression to extract some simple comma-separated data,
     * reorder some of the columns, and discard column 2.
     */

    s = "abc,def,ghi,klm,nop,pqr";

    re = new Regexp("^([^,]+),([^,]+),([^,]+),(.*)");
    System.out.println(re.sub(s, "\\3,\\1,\\4"));
}
```

See Also:

`Regsub`, Serialized Form

Inner Class Summary

`static interface`	**`Regexp.Filter`** This interface is used by the `Regexp` class to generate the replacement string for each pattern match found in the source string.

Constructor Summary

`Regexp``(String pat)` Compiles a new Regexp object from the given regular expression pattern.	
`Regexp``(String pat, boolean ignoreCase)` Compiles a new Regexp object from the given regular expression pattern.	

Method Summary	
static void	**applySubspec**(Regsub rs, String subspec, StringBuffer sb) Utility method to give access to the standard substitution algorithm used by sub and subAll.
static void	**main**(String[] args)
String	**match**(String str) Matches the given string against this regular expression.
boolean	**match**(String str, int[] indices) Matches the given string against this regular expression, and computes the set of substrings that matched the parenthesized subexpressions.
boolean	**match**(String str, String[] substrs) Matches the given string against this regular expression, and computes the set of substrings that matched the parenthesized subexpressions.
String	**sub**(String str, Regexp.Filter rf)
String	**sub**(String str, String subspec) Matches a string against a regular expression and replaces the first match with the string generated from the substitution parameter.
String	**subAll**(String str, String subspec) Matches a string against a regular expression and replaces all matches with the string generated from the substitution parameter.
int	**subspecs**() Returns the number of parenthesized subexpressions in this regular expression, plus one more for this expression itself.
String	**toString**() Returns a string representation of this compiled regular expression.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

Regexp

```
public Regexp(String pat)
       throws IllegalArgumentException
```

Compiles a new Regexp object from the given regular expression pattern.

It takes a certain amount of time to parse and validate a regular expression pattern before it can be used to perform matches or substitutions. If the caller caches the new Regexp object, that parsing time will be saved because the same Regexp can be used with respect to many different strings.

Parameters:

`pat` - The string holding the regular expression pattern.

Throws:

IllegalArgumentException - if the pattern is malformed. The detail message for the exception will be set to a string indicating how the pattern was malformed.

Regexp

```
public Regexp(String pat,
              boolean ignoreCase)
       throws IllegalArgumentException
```

Compiles a new Regexp object from the given regular expression pattern.

Parameters:

`pat` - The string holding the regular expression pattern.

`ignoreCase` - If `true` then this regular expression will do case-insensitive matching. If `false`, then the matches are case-sensitive. Regular expressions generated by `Regexp(String)` are case-sensitive.

Throws:

IllegalArgumentException - if the pattern is malformed. The detail message for the exception will be set to a string indicating how the pattern was malformed.

Method Detail

main

```
public static void main(String[] args)
                 throws Exception
```

---

subspecs

```
public int subspecs()
```

Returns the number of parenthesized subexpressions in this regular expression, plus one more for this expression itself.

Returns:

The number.

---

match

```
public String match(String str)
```

Matches the given string against this regular expression.

Parameters:

`str` - The string to match.

Returns:

The substring of `str` that matched the entire regular expression, or `null` if the string did not match this regular expression.

---

match

```
public boolean match(String str,
                     String[] substrs)
```

Matches the given string against this regular expression, and computes the set of substrings that matched the parenthesized subexpressions.

`substrs[0]` is set to the range of `str` that matched the entire regular expression.

`substrs[1]` is set to the range of `str` that matched the first (leftmost) parenthesized subexpression. `substrs[n]` is set to the range that matched the `n`*th* subexpression, and so on.

If subexpression `n` did not match, then `substrs[n]` is set to `null`. Not to be confused with "", which is a valid value for a subexpression that matched 0 characters.

The length that the caller should use when allocating the `substr` array is the return value of `Regexp.subspecs`. The array can be shorter (in which case not all the information will be returned), or longer (in which case the remainder of the elements are initialized to `null`), or `null` (to ignore the subexpressions).

Parameters:

`str` - The string to match.

`substrs` - An array of strings allocated by the caller, and filled in with information about the portions of `str` that matched the regular expression. May be `null`.

Returns:

`true` if `str` that matched this regular expression, `false` otherwise. If `false` is returned, then the contents of `substrs` are unchanged.

See Also:

`subspecs()`

---

match

```
public boolean match(String str,
                     int[] indices)
```

Matches the given string against this regular expression, and computes the set of substrings that matched the parenthesized subexpressions.

For the indices specified below, the range extends from the character at the starting index up to, but not including, the character at the ending index.

`indices[0]` and `indices[1]` are set to starting and ending indices of the range of `str` that matched the entire regular expression.

`indices[2]` and `indices[3]` are set to the starting and ending indices of the range of `str` that matched the first (leftmost) parenthesized subexpression. `indices[n * 2]` and `indices[n * 2 + 1]` are set to the range that matched the `n`*th* subexpression, and so on.

If subexpression `n` did not match, then `indices[n * 2]` and `indices[n * 2 + 1]` are both set to `-1`.

The length that the caller should use when allocating the `indices` array is twice the return value of `Regexp.subspecs`. The array can be shorter (in which case not all the information will be returned), or longer (in which case the remainder of the elements are initialized to `-1`), or `null` (to ignore the subexpressions).

Parameters:

`str` - The string to match.

`indices` - An array of integers allocated by the caller, and filled in with information about the portions of `str` that matched all the parts of the regular expression. May be `null`.

Returns:

`true` if the string matched the regular expression, `false` otherwise. If `false` is returned, then the contents of `indices` are unchanged.

See Also:

`subspecs()`

---

sub

```
public String sub(String str,
                  String subspec)
```

Matches a string against a regular expression and replaces the first match with the string generated from the substitution parameter.

Parameters:

`str` - The string to match against this regular expression.

`subspec` - The substitution parameter, described in REGULAR EXPRESSION SUBSTITUTION [p ??] .

Returns:

The string formed by replacing the first match in `str` with the string generated from `subspec`. If no matches were found, then the return value is `null`.

---

subAll

```
public String subAll(String str,
                     String subspec)
```

Matches a string against a regular expression and replaces all matches with the string generated from the substitution parameter. After each substutition is done, the portions of the string already examined, including the newly substituted region, are **not** checked again for new matches -- only the rest of the string is examined.

Parameters:

`str` - The string to match against this regular expression.

`subspec` - The substitution parameter, described in REGULAR EXPRESSION SUBSTITUTION [p ??] .

Returns:

The string formed by replacing all the matches in `str` with the strings generated from `subspec`. If no matches were found, then the return value is a copy of `str`.

---

applySubspec

```
public static void applySubspec(Regsub rs,
                                String subspec,
                                StringBuffer sb)
```

Utility method to give access to the standard substitution algorithm used by `sub` and `subAll`. Appends to the string buffer the string generated by applying the substitution parameter to the matched region.

Parameters:

`rs` - Information about the matched region.

`subspec` - The substitution parameter.

`sb` - StringBuffer to which the generated string is appended.

---

sub

```
public String sub(String str,
                  Regexp.Filter rf)
```

---

toString

```
public String toString()
```

Returns a string representation of this compiled regular expression. The format of the string representation is a symbolic dump of the bytecodes.

Overrides:

toString in class Object

Returns:

A string representation of this regular expression.

Class Regsub

sunlabs.brazil.util.regexp
Class Regsub

```
java.lang.Object
  |
  +--sunlabs.brazil.util.regexp.Regsub
```

---

public class **Regsub**
extends Object

The `Regsub` class provides an iterator-like object to extract the matched and unmatched portions of a string with respect to a given regular expression.

After each match is found, the portions of the string already checked are not searched again -- searching for the next match will begin at the character just after where the last match ended.

Here is an example of using Regsub to replace all "%XX" sequences in a string with the ASCII character represented by the hex digits "XX":

```
public static void
main(String[] args)
    throws Exception
{
    Regexp re = new Regexp("%[a-fA-F0-9][a-fA-F0-9]");
    Regsub rs = new Regsub(re, args[0]);

    StringBuffer sb = new StringBuffer();

    while (rs.nextMatch()) {
        sb.append(rs.skipped());

        String match = rs.matched();

        int hi = Character.digit(match.charAt(1), 16);
        int lo = Character.digit(match.charAt(2), 16);
        sb.append((char) ((hi << 4) | lo));
    }
    sb.append(rs.rest());

    System.out.println(sb);
}
```

See Also:
`Regexp`

Constructor Summary

Regsub(Regexp r, String str) Construct a new Regsub that can be used to step through the given string, finding each substring that matches the given regular expression.

Method Summary

Regexp	**getRegexp**() Return the regexp used by this regsub.
String	**matched**() Returns a substring consisting of the characters that matched the entire regular expression during the last call to nextMatch.
boolean	**nextMatch**() Searches for the next substring that matches the regular expression.
String	**rest**() Returns a substring consisting of all the characters that come after the last match.
String	**skipped**() Returns a substring consisting of all the characters skipped between the end of the last match (or the start of the original search string) and the start of this match.
String	**submatch**(int i) Returns a substring consisting of the characters that matched the given parenthesized subexpression during the last call to nextMatch.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Regsub

```
public Regsub(Regexp r,
              String str)
```

Construct a new Regsub that can be used to step through the given string, finding each substring that

matches the given regular expression.

`Regexp` contains two substitution methods, `sub` and `subAll`, that can be used instead of `Regsub` if just simple substitutions are being done.

Parameters:

`r` - The compiled regular expression.
`str` - The string to search.

See Also:

`Regexp.sub(java.lang.String, java.lang.String)`, `Regexp.subAll(java.lang.String, java.lang.String)`

Method Detail

nextMatch

```
public boolean nextMatch()
```

Searches for the next substring that matches the regular expression. After calling this method, the caller would call methods like `skipped`, `matched`, etc. to query attributes of the matched region.

Calling this function again will search for the next match, beginning at the character just after where the last match ended.

Returns:

`true` if a match was found, `false` if there are no more matches.

---

skipped

```
public String skipped()
```

Returns a substring consisting of all the characters skipped between the end of the last match (or the start of the original search string) and the start of this match.

This method can be used extract all the portions of string that **didn't** match the regular expression.

Returns:

The characters that didn't match.

---

matched

```
public String matched()
```

Returns a substring consisting of the characters that matched the entire regular expression during the last call to `nextMatch`.

Returns:

The characters that did match.

See Also:
`submatch(int)`

---

submatch

```
public String submatch(int i)
```

Returns a substring consisting of the characters that matched the given parenthesized subexpression during the last call to `nextMatch`.

Parameters:

`i` - The index of the parenthesized subexpression.

Returns:

The characters that matched the subexpression, or `null` if the given subexpression did not exist or did not match.

---

rest

```
public String rest()
```

Returns a substring consisting of all the characters that come after the last match. As the matches progress, the `rest` gets shorter. When `nextMatch` returns `false`, then this method will return the rest of the string that can't be matched.

Returns:

The rest of the characters after the last match.

---

getRegexp

```
public Regexp getRegexp()
```

Return the regexp used by this regsub.

Package sunlabs.brazil.velocity

Integrates the Velocity template engine with the Brazil project server.

Velocity is a Java-based template engine. It permits anyone to use the simple yet powerful template language to reference objects defined by Java code. See The Velocity home page for more information about Velocity.

Files in this package depend upon the velocity.jar and commons-collections.jar files.

Class VelocityFilter.Vrequest

sunlabs.brazil.velocity
Class VelocityFilter.Vrequest

```
java.lang.Object
  |
  +--sunlabs.brazil.server.Request
        |
        +--sunlabs.brazil.velocity.VelocityFilter.Vrequest
```

Enclosing class:
VelocityFilter

---

public class **VelocityFilter.Vrequest**
extends Request

A helper class for Velocity that provides read only access to the public fields of the `Request` object. This class and it's methods should only be used by the Velocity engine.

Inner classes inherited from class sunlabs.brazil.server.Request
`Request.HttpOutputStream`

Fields inherited from class sunlabs.brazil.server.Request
`connectionHeader, headers, keepAlive, MAX_BLANKS, method, out, postData, props, protocol, query, responseHeaders, server, serverProps, serverProtocol, sock, startMillis, url, version`

Constructor Summary

`VelocityFilter.Vrequest``(Request request)`	

Method Summary

`Object`	**`get`**`(String fieldName)`

Methods inherited from class sunlabs.brazil.server.Request
addHeader, addHeader, addSharedProps, getQueryData, getQueryData, getRequest, getRequestHeader, getReuseCount, getSocket, getStatus, log, log, redirect, removeSharedProps, sendError, sendError, sendHeaders, sendResponse, sendResponse, sendResponse, sendResponse, sendResponse, serverUrl, setStatus, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

VelocityFilter.Vrequest

```
public VelocityFilter.Vrequest(Request request)
                        throws Exception
```

Method Detail

get

```
public Object get(String fieldName)
```

Class VelocityFilter.Vserver

sunlabs.brazil.velocity
Class VelocityFilter.Vserver

```
java.lang.Object
  |
  +--java.lang.Thread
        |
        +--sunlabs.brazil.server.Server
              |
              +--sunlabs.brazil.velocity.VelocityFilter.Vserver
```

All Implemented Interfaces:
Runnable
Enclosing class:
VelocityFilter

---

public class **VelocityFilter.Vserver**
extends Server

A helper class for Velocity that provides read only access to the public fields of the `Server` object. This class and it's methods should only be used by the Velocity engine.

Fields inherited from class sunlabs.brazil.server.Server
`acceptCount, bufsize, errorCount, handler, hostName, initFailure, listen, LOG_DIAGNOSTIC, LOG_ERROR, LOG_INFORMATIONAL, LOG_LOG, LOG_WARNING, logLevel, maxPost, maxRequests, maxThreads, name, prefix, props, protocol, requestCount, restrict, timeout`

Fields inherited from class java.lang.Thread
`MAX_PRIORITY, MIN_PRIORITY, NORM_PRIORITY`

Constructor Summary

`VelocityFilter.Vserver``(Server server)`	

Method Summary

Object	**get**(String fieldName)

Methods inherited from class sunlabs.brazil.server.Server
close, init, log, restart, run, setup

Methods inherited from class java.lang.Thread
activeCount, checkAccess, countStackFrames, currentThread, destroy, dumpStack, enumerate, getContextClassLoader, getName, getPriority, getThreadGroup, interrupt, interrupted, isAlive, isDaemon, isInterrupted, join, join, join, resume, setContextClassLoader, setDaemon, setName, setPriority, sleep, sleep, start, stop, stop, suspend, toString, yield

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

VelocityFilter.Vserver

```
public VelocityFilter.Vserver(Server server)
                        throws Exception
```

Method Detail

get

```
public Object get(String fieldName)
```

Class VelocityFilter

sunlabs.brazil.velocity
Class VelocityFilter

```
java.lang.Object
  |
  +--sunlabs.brazil.velocity.VelocityFilter
```

All Implemented Interfaces:
Filter, Handler, LogSystem

---

public class **VelocityFilter**
extends Object
implements Filter, LogSystem

A filter for processing markup that is a Velocity template. The filter will "merge" the content with a BeanShell script to produce output content.

The following server properties are used:
script
The name of a BeanShell script file.
session
The name of the propery containing the session. The default session, if none is found, is "common".

The BeanShell script is found by first looking in the filesystem. If not found, the classpath is searched. It's okay not to specify or use a BeanShell script. The content will still be processed by the Velocity engine.

The BeanShell interpreter is managed by session and is persistent during the session. The interpreter has access to the `Server` object, the prefix string, the `Request` object, and the Velocity context through the variables "server", "prefix", "request", and "context".

A new Velocity context is created each time the `filter` method is called. For convenience, the context is initially populated with the same objects as the BeanShell. They are referenced by the same names ("server", "prefix", "request", and "context"). However, due to the design of Velocity, the public fields of the `Server` and `Request` objects are read only. The field values can not be changed by the template code.

Each time the `filter` method is called, a new Velocity context is created and populated. Then the BeanShell script, if it exists, is passed through `Format.subst` for variable substitution and then passed to the BeanShell interpreter. Finally the content (i.e. a Velocity template) and the context are passed to the Velocity engine for processing.

Note: Velocity requires at least a 1.2 Java VM. Attempts to use Velocity with a 1.1 VM will most likely produce incorrect results. Also note that the Brazil system must be compiled with a 1.2+ compiler so that core components of Brazil are compatible with the Velocity engine.

Inner Class Summary

class	**VelocityFilter.Vrequest** A helper class for Velocity that provides read only access to the public fields of the `Request` object.
class	**VelocityFilter.Vserver** A helper class for Velocity that provides read only access to the public fields of the `Server` object.

Fields inherited from interface org.apache.velocity.runtime.log.LogSystem
`DEBUG_ID, DEBUG_ON, ERROR_ID, INFO_ID, WARN_ID`

Constructor Summary

VelocityFilter()	

Method Summary

byte[]	**filter**(Request request, MimeHeaders headers, byte[] content) Execute the BeanShell script if it exists and then process the content as a Velocity template.
void	**init**(RuntimeServices rsvc) Velocity LogSystem interface implementation.
boolean	**init**(Server server, String prefix) Initializes the handler.
void	**logVelocityMessage**(int level, String message)
boolean	**respond**(Request request) This is the request object before the content was fetched
boolean	**shouldFilter**(Request request, MimeHeaders headers) Only filter text/* documents

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

VelocityFilter

```
public VelocityFilter()
```

Method Detail

init

```
public boolean init(Server server,
                    String prefix)
```

Description copied from interface: Handler
Initializes the handler.
Specified by:
init in interface Handler
Tags copied from interface: Handler
Parameters:
`server` - The HTTP server that created this `Handler`. Typical `Handler`s will use `Server.props` to obtain run-time configuration information.
`prefix` - The handlers *name*. The string this `Handler` may prepend to all of the keys that it uses to extract configuration information from `Server.props`. This is set (by the `Server` and `ChainHandler`) to help avoid configuration parameter namespace collisions.
Returns:
`true` if this `Handler` initialized successfully, `false` otherwise. If `false` is returned, this `Handler` should not be used.

---

respond

```
public boolean respond(Request request)
```

This is the request object before the content was fetched
Specified by:
respond in interface Handler
Tags copied from interface: Handler

Parameters:

`request` - The `Request` object that represents the HTTP request.

Returns:

`true` if the request was handled. A request was handled if a response was supplied to the client, typically by calling `Request.sendResponse()` or `Request.sendError`.

Throws:

IOException - if there was an I/O error while sending the response to the client. Typically, in that case, the `Server` will (try to) send an error message to the client and then close the client's connection.

The `IOException` should not be used to silently ignore problems such as being unable to access some server-side resource (for example getting a `FileNotFoundException` due to not being able to open a file). In that case, the `Handler`'s duty is to turn that `IOException` into a HTTP response indicating, in this case, that a file could not be found.

---

shouldFilter

```
public boolean shouldFilter(Request request,
                            MimeHeaders headers)
```

Only filter text/* documents

Specified by:

shouldFilter in interface Filter

Tags copied from interface: Filter

Parameters:

`request` - The in-progress HTTP request.

`headers` - The MIME headers generated by the wrapped `Handler`.

Returns:

`true` if this filter would like to examine and possibly rewrite the content, `false` otherwise.

---

filter

```
public byte[] filter(Request request,
                     MimeHeaders headers,
                     byte[] content)
```

Execute the BeanShell script if it exists and then process the content as a Velocity template.

Specified by:

filter in interface Filter

Tags copied from interface: Filter

Parameters:

`request` - The finished HTTP request.

`headers` - The MIME headers generated by the `Handler`.

`content` - The output from the `Handler` that this `Filter` may rewrite.

Returns:

The rewritten content. The `Filter` may return the original `content` unchanged. The `Filter` may return `null` to indicate that the `FilterHandler` should stop processing the request and

should not return any content to the client.

---

init

```
public void init(RuntimeServices rsvc)
```

Velocity LogSystem interface implementation.

Specified by:

init in interface LogSystem

---

logVelocityMessage

```
public void logVelocityMessage(int level,
                               String message)
```

Specified by:

logVelocityMessage in interface LogSystem

Configuration Parameter Summary

The follow is an alphabetical summary of the configuration options by class. Most of the classes are Handlers, but some Templates use configuration parameters as well. Handlers or Templates with no configuration options are not listed. (Regrettably, there is no Template tag and attribute summary at this time).

AclSwitchHandler

Simple access control hander based on url prefixes or regexps.

authName
: The name of the request.props entry to find a white-space delimited list of url prefixes or regular expression patterns. (defaults to "roles"). If the items in the list don't start with "/", then the url prefix is prepended (only for prefix matching).

prefix, suffix, glob, match
: Sepcify the URL that triggers this handler. (See `MatchString`).

redirect
: Name of the url to re-direct to if permission is denied. If not specified, a simple message is sent to the client.

useRegexp
: If provided, the list of credentials is interpreted as regular expressions, otherwise url prefixes are used.

AsteriskHandler

Connect to asterisk manager api.

debug=true|false
: turn on more diagnostics on the console, depending on the current server logging level. at "3", keep alives are logged, at "4", all events are logged, and at "5" even more stuff is logged.

keepalive=n
: If set, this handler will issue a keep-alive every "n" seconds to the Asterisk Server. If the keep-alive fails, a new connection will be attempted with the Asterisk server.

queue
: The name of the Q to send manager commands to using the <enqueue name="queue" ...>. If not specifies, the "host:port" combination is used.

retry=n
: Set the number of seconds to wait before retrying a broken connection to an asterisk server (defaults to 10).

server
: The server:port to use to contact the asterisk server

userid, password
: The Manager credentials

BSLTemplate

The `BSLTemplate` takes an HTML document with embedded "BSL" markup tags in it and evaluates those special tags to produce a standard HTML document.

basematchelist.
: The list of matches (e.g. "1 2 3 ...").

basematches
: The number of times the regular expression was matched.

basesubmatches.
: The number of sub-expressions for this regular expression.

debug
: If this configuration parameter is present, this template replaces the BSL tags with comments, so the user can keep track of where the dynamically generated content is coming from by examining the comments in the resultant HTML document. By default, the BSL tags are completely eliminated from the HTML document rather than changed into comments.

BasicAuthHandler

The `BasicAuthHandler` obtains a Session ID by performing "basic" authentication, using either the "Authorization" or the "Proxy-Authorization" headers.

authenticate
: If specified, this is the response header that will be sent in the HTTP error response if the user is not authenticated.

  If this string is "", then this handler will authenticate the request if the authorization header is present, but **will not** send an HTTP error message if the request could not be authenticated. This is useful if the web developer wants to do something more complex (such as invoking an arbitrary set of handlers) instead of just sending a simple error message if the request was not authenticated. In this case, the web developer can determine that the request was not authenticated because **no** Session ID will be present in the request properties.

authorization
: If specified, this is the request header that will contain the "basic" authentication string, instead of the "Authorization" or "Proxy-Authorization" header implied by `code`.

code
: The type of authentication to perform. The default value is 401.

  The value 401 corresponds to standard "basic" authentication. The "Authorization" request header is supposed to contain the authentication string. If the request was not authenticated, the "WWW-Authenticate" header is sent in the HTTP error response to cause the browser to prompt the client to authenticate.

  The value 407 corresponds to "basic" proxy/firewall authentication. The "Proxy-Authorization" request header is supposed to contain the authentication string. If the request was not authenticated, the "Proxy-Authenticate" header is sent in the HTTP error response to cause the browser to prompt the client to authenticate.

Any other value may also be specified. Whatever the value, it will be returned as the HTTP result code of the error message.

ident
: The `ident` argument to `SessionManager.getSession(java.lang.Object, java.lang.Object, java.lang.Class)` to get the table of valid sessions. The default value is "authorized". If `ident` is of the form `ident:session`, then the `session` portion is used as the `session` argument to SessionManager.get(). Otherwise the `session` argument is NULL. This table may be manipulated with the SetTemplate, using the "ident" namespace and "session" for the SetTemplate "sessionTable" parameter.

mapFile
: If specified, this is the initial Session ID file. This is expected to be a java properties file, whose keys are the authentication tokens, and whose values are the Session IDs that are inserted into the request properties.

  The keys in the file are basic authentication (base64) tokens with any trailing `"="` characters changed to `"!"`.

message
: The body of the HTTP authentication error message. This will be displayed by the browser if the client chooses not to authenticate. The default value is "". Patterns of the form *${xxx}* are replaced with the value of the *xxx* entry of `request.props`.

prefix, suffix, glob, match
: Sepcify the URL that triggers this handler.

realm
: The "realm" of the HTTP authentication error message. This is a string that the browser is supposed to present to the client when asking the client the authenticate. It provides a human-friendly name describing who wants the authentication.

session
: The name of the request property that the Session ID will be stored in, to be passed to downstream handlers. The default value is "SessionID".

BasicSSLHandler

Start an "ssl" server.

auth
: Require a valid client certificate (not useful).

enable
: Must be set to "true" to turn on ssl server.

password
: The certificate store password. If the password starts with "@", the rest of the password is taken to be the name of the file containing the password. "@-" causes the password to be read from stdin.

sslport
: The port to start the server on. If no port is specified, Then the existing server port is reused. **Note**: Do not use a prefix when specifying this property, it is precicely "sslport".

store
: The path name to the certificate keystore.

BeanShellHandler

The `BeanShellHandler` permits handlers to be written in "beanshell".

debug
: If this configuration parameter is present, the `script` is re-read on each request, and a new interperter is created and initialized. The call to `init` is deferred until *request* time, and called before each call to `respond`.

  This allows beanshell scripts to be debugged interatively from scratch.

root
: The script directory, if the script is a relative file name. If the "root" property under the prefix is not found, the global "root" property is used. If the global "root" property is not found, the current directory is used.

script
: The name of the BeanShell script to use as the handler. Normally, the script is read only once. (defaults to *prefix*.bsh)

BeanShellServerTemplate

The `BeanShellServerTemplate` looks for one of the starting tags `<server language="beanshell">`, `<beanshell>`, or `<bsh>` in an HTML page and treats the following data up to the corresponding ending tag (`</server>`, `</beanshell>`, or `</bsh>`) as a BeanShell script to evaluate.

debug
: If this configuration parameter is present, this class replaces the starting and ending tags with comments, so the user can keep track of where the dynamically generated content is coming from by examining the comments in the resultant HTML document. By default, the starting and ending tags are completely eliminated from the HTML document rather than changed into comments.

root
: The document root, if the script is a relative file name. If the "root" property under the template prefix is not found, the global "root" property is used. If the global "root" property is not found, the current directory is used.

script
: The name of the BeanShell script to evaluate when the interpreter is created. This script is only evaluated when the interpreter is created, not on every request. The variables `prefix` and `server` are set before this file is evaluated, and are references to the parameters passed to a `handler` init method.

BrazilServlet

This is the Brazil Toolkit Servlet Adapter.

context_path
: The servlet context path

servlet_container
: The name of the servlet container

servlet_name
: The servlet's name

url.servlet
: The original URL requested by the client.

CacheManager

This `SessionManager` associates an object with a Session ID to give Handlers the ability to maintain state that lasts for the duration of a session instead of just for the duration of a request.

size
: The max number of entries in each table (defaults to 1000).

tables
: The number of Hashtables in the pool (defaults to 6)

CertHandler

Handler for issuing A netscape certificate.

cert
: The file name containing the CA certificate

id
: If set, The the request property "id" MUST be set in order to generate a cert. This allows upstream handlers to do authentication before a cert is issued.

prefix
: The url prefix for this handler to generate a cert

CgiHandler

Handler for implementing cgi/1.1 interface.

custom
: set to "true" to enable custom environment variables. If set, all server properties starting with this handler's prefix are placed into the environment with the name: `CONFIG_`*`name`*, where *name* is the property key, in upper case, with the prefix removed. This allows cgi scripts to be customized in the server's configuration file.

prefix
: The prefix for all cgi files (e.g. /cgi-bin)

root
: The document root for cgi files

suffix
: The suffix for cgi files (defaults to .cgi)

url
: "o(riginal)" or "c(urrent)". If an upstream handler has changed the URL, this specifes which url to look for the cgi script relative to. The default is to use the original url.

ChainHandler

Allows multiple handlers to be invoked sequentially for a single HTTP request.

exitOnError
: If set, the server's `initFailure` will set any of the handlers fail to initialize. No handler prefix is required.

handlers
: A list of `Handler` names that will be invoked in the given order to handle the request. These are considered the "wrapped" handlers. These handlers will all be initialized at startup by `init(sunlabs.brazil.server.Server, java.lang.String)`. For each name in the list, the property *`name`*`.class` is examined to determine which class to use for this handler. Then `name` is used as the prefix in the handler's init() method.

prefix
: If set, All url's must start with this, or the request will be ignored

report
: If set, this property will be set to the name of the handler that handled the request (e.g. returned true).

ChainSawHandler

Variant of the chain handler for doing standard logging.

flush
: The number of requests between flushes to the file

logFile
: The name of the file to write the logs to.

ChangedTemplate

This `Template` adds an icon to HREFs to indicate when the file being referred to is new, changed, or unchanged with respect to the user's session.

always
: If this property is present, the `ChangedTemplate` always rewrites the HREFs, instead of just when they appear within the `<changed>` and `</changed>` tags.

changed
: The HTML to substitute into the document if the HREF refers to a file that has changed since the last time it was accessed by the user. If absent, the HREF for changed files will not be rewritten.

fileName
: A request property containing the full path name of the current file, used to keep track of the last time that file was accessed by the current user. A `Handler` or other code may set this property if it wishes the file to be tracked.

new
: The HTML to substitute into the document if the HREF refers to a file that has never been accessed by the user. If absent, the HREF for new files will not be rewritten.

root
: The root of the document hierarchy. An HREF must resolve to a file in this hierarchy so its last-modified time can be checked. If the file does not exist, the HREF will not be rewritten.

unchanged
: The HTML to substitute into the document if the HREF refers to a file that has not changed since the last time it was accessed by the user. If absent, the HREF for unchanged files will not be rewritten.

ChownHandler

Handler for changing the group and owner of the server.

groupName
: The name of the group to run as

userName
: name of the user to run as

ConfigFileHandler

Handler for manipulating per-user session state that can be represented as ASCII name/value pairs.

default
: The default properties file to "seed" session properties

glob
: Properties that match this "glob" pattern may be set using the "set" pattern. If this property is specified, the "default" property is optional.

name
: The name of the config file. the first "%" is replaced by the current SessionID.

noContent
: a url, matching the "set" pattern that causes a "204 no content" to be returned to the client (experimental).

prefix
: The URL prefix required for all documents

root
: The document root (no properties prefix required). If the "name" or "default" properties don’t start with a "/", this is used as the current directory.

set
: The url pattern to match setting properties. Currently, it must also match "prefix".

ContentTemplate

Template class for extracting content out of remote html pages.

all
: The entire content

bodyArgs
: The attributes to the body tag, if any

content-length
: The document content length, as fetched from the origin server

content
: The body, delimited by content.../content>. The text inside multiple <content> ... </content> pairs are concatenated together.

last-modified
: The document last modified time (if any) in std format

link-[rel]
: Every link tag "rel" and "href"

meta-[name]
: Every meta tag "name" and "content"

prepend
: Prepend this string to the property names define above, that are populated by this template. (defaults to "").

referer
: The user agent referrer (if any)

script
: All "<script>"..."</script>" tags found in the document head

style
: All "<style">..."</style"> tags found in the document head

title
: The document title

user-agent
: The origin user agent

CookieFilter

The `CookieFilter` keeps a record of all the browser *cookies* associated with a given session.

admin
: A URL prefix that causes status information to be placed in the request properties.

nosession
: The name of the session to use if no session id is found. defaults to "common".

session
: The request property to find the session id. Defaults to "SessionID"

CookieSessionHandler

Handler for creating browser sessions using cookies.

cookie
: the name of the cookie to use (defaults to "cookie").

exist
: If specified, this means that the Session ID corresponding to the cookie value must already exist in the `SessionManager`. Normally, if the cookie was not present, a new cookie is automatically created.

map
: If specified, the `ident` argument to `SessionManager.getSession(java.lang.Object, java.lang.Object, java.lang.Class)` to get the table of valid cookies, used to map the cookie value to a Session ID. By default, the Session ID stored in the request is the cookie value itself.

persist
: If set, cookies persist across browser sessions

prefix, suffix, glob, match
: Specify the URL that triggers this handler (See `MatchString`).

  If `prefix` is specified, it is also used to instruct the client to limit the scope of the browser cookie. to that prefix.

session
: The name of the request property that the Session ID will be stored in, to be passed to downstream handler. The default value is "SessionID". If the property already exists, and is not empty, no action will be taken.

CopyContentFilter

Filter to save content (of an entire site) to a disk file.

directoryName
: The root in the file system to save the content in

DeCommentTemplate

Template class for removing comments from html pages.

disable
: if true, disable comment removal. This is checked at each request.

DebugTemplate

Template class for printing stuff to stderr (for template debugging).

debug
: If this configuration parameter is true, debugging is enabled.

DefaultFileHandler

Handler for appending a url ending with '/' into the appropriate url based on a default file in the file system.

defaults
: The names of the default files to search for in the directory implied by the URL. The first one that exists will cause its name to be appended to the URL. Defaults to "index.html".

DirectoryName
: This property is set if the URL represents a valid directory in the document root.

fileName
: This property is set to the name of the default file, if one was found.

root
: The document root to look for files. If none is found with our prefix, then "root" is examined. Defaults to ".".

DeferredHandler

Wrap another handler, deferring its initialization until request time.

[handler.class]
: The name of the handler class.

[handler.prefix]
: Used to trigger the configuration

handler
: The token representing the handler to conditionally configure. This is used as the handler's prefix

requires
: The names of classes required to be resolvable before configuring the handler

DelayHandler

Handler or template for adding a delay into a response.

delay
: The delay, in ms (defaults to 1000).

prefix, suffix, match, glob
: Specify which url's to process.

DerbyServer

Template for starting derby in network server mode.

log
: If true, log connections

port
: The port the server will listen on (defaults to 1527)

DialogHandler

Sample handler for popping up a dialog box on the server.

default
: The message to appear in the dialog box. Defaults to **Request from Client**.

denied
: The message to appear in the "permission denied" spot.

prefix, suffix, glob, match
: Specify the URL that triggers this handler. (See `MatchString`).

DigestAuthHandler

Perform digest authentication.

allowBogusIE
: Internet Explorer does not use the query parameters as part of the "uri" calculation. This is a bug (and a security risk, as it allows replay attacts to other than the url requested). If this variable is set, then it allows IE to work in this case.

credentials
: A java-properties format file of credentials. The keys are the users, the values are either the "A1" values described above, or the user's password.

isDynamic
: If set (to anything), when authentication for a user is requested that is not in the credentials table and the credentials table has changed since last read, the table is re-read, in case the user has been added since the credentials were loaded.

prefix, suffix, glob, match
: Specify which url's this handler applies to.

realm
: The string presented to the user for validation. This must also match any "digested" passwords.

username
: If the user was validated, this field is filled out by the handler.

DirectoryHandler

This is a bare-bones handler for providing directory listings for web servers.

delim
: The delimeter separating the file names. Defaults to a single space.

DirectoryName
: This property is set by the `FileHandler` if the URL it was passed resolves to a directory, but no index file (e.g. index.html) was found.

mime.xxx
: Only documents ending in ".xxx" are considered. more than on mime.xxx parameters may be specified.

prefix, suffix, glob, match
: Specify the URL that triggers this handler. (See `MatchString`).

setProps
: If present, no content is returned. Instead, The properties "Directories" and "Files" are set in the request properties, so the format of the output may be generated dynamically. [Note: This feature is deprecated, use the `DirectoryTemplate` instead].

DirectoryTemplate

Put current directory information (based on the URL) into the request properties.

[prepend.Directories]
: List of sub-directories in current directory is set by this template.

[prepend.Files]
: List of files with valid suffixes in current directory. is set by this template.

debug
: if set, a comment is emitted indicating where the file-list entitiy was encountered.

delimiter
: Delimiter character to separate entries, defaults to " ".

directory
: The directory to use instead of the one implied by the URL. If it starts with "/", then it is resolved relative to the document root, otherwise it is resolved relative to the directory implied by the URL. The document root is found in the porperty `"[prefix].root"`, or in `"root"`, or (if neither exists), the current directory of the server.

DirectoryName
: If set (usually by an upstream handler, such as the FileHandler, or TemplateHandler), this is used as the directory name instead of deriving it from the URL.

mime.xxx
: An indication that suffix "xxx" is valid. Only valid file names are returned. See `FileHandler` for a description of how to set mime types for url suffixes.

prepend
: String to prepend to the properties "Directories" and "Files" That contain the directory and file lists respectively. Defaults to the Templates properties prefix.

select
: Specifies a "glob" pattern to restrict the names of files and directories returned. The form "![pattern]" selects the inverse of the glob pattern.

stats
: If specified, then for each file, the properties: **[prepend].[file].mod** and **[prepend].[file].size** are set, containing the file last modified time (seconds since epoch) and size (bytes) respectively.

DynamicConfigHandler

The `DynamicConfigHandler` allows the user to change the configuration of the server and its handlers on the fly.

config
: URLs beginning with this prefix can be used to upload or download new configuration properties to this handler, which will also dynamically change which sub-handler is installed. This property belongs to the `DynamicConfigHandler` and is **not** changed when new properties are uploaded. The default is "/config/".

  Properties may be uploaded by sending them as "name=value" pairs in the body of a POST or in the "?" query parameters. The URL for uploading properties is "*config*/set".

  The current set of properties may be retrieved from this handler by sending the URL "*config*/get"

handler
: The name of the initial sub-handler that this `DynamicConfigHandler` will use to process requests. When new properties are uploaded, the sub-handler will be replaced with whatever is specified in the newly uploaded `handler` property.

prefix
: Only URLs beginning with this string will be redirected to the sub-handler. This property belongs to the `DynamicConfigHandler` and is **not** changed when new properties are uploaded. The default is "/".

ExecFilter

Filter to Run all content through an external process filter.

command
: The command to exec. The content is supplied as stdin, and the filtered output is expected on stdout. ${...} substitutions Are done at each filter invocation.

error
: If the command failed, this property will contain the error message. If the command generated output on stderr, this property will contain the output.

newType
: This property, if set, is used as the new content type. It is evaluated for ${...} at each conversion.

type
: This property is set to the content type of the content just before *command* is evaluated.

types
: A regular expression that matches the content types for the content we wish to filter

ExecTemplate

template to exec a program, and return its arguments into request properties .

code
: The exit code for the program.

error
: The error message, if something went wrong.

stderr
: The standard error output produced by the program, converted to a String using the default encoding.

stdout
: The standard output produced by the program, converted to a String using the default encoding.

ExpContentTemplate

Allow extracted content to be filtered through regular expressions.

extract
: A regular expression to match the extracted content

replace
: A regular expression substitution string used to replace the content, if the expression matched.

urlPrefix
: A prefix the url must match to be considered for rewriting

ExprPropsHandler

The `ExprPropsHandler` installs an expression evaluator as a "smart properties" into the current request object, enabling arithmetic and logical expression evaluation in property name lookups.

prefix, suffix, glob, match
: Only URL's that match are allowed. (See `MatchString`).

FileHandler

Standard handler for fetching static files.

default
: The document to deliver if the URL ends in "/". (defaults to index.html.)

getOnly
: If defined, only "GET" requests will be processed. By default, all request types are handled. (Note: this is the inverse of the previous policy, defined by the undocumented "allow" parameter).

mime
: property for mime type For each file suffix .XX, the property mime.XX is used to determine the mime type. If no property exists (or its value is "unknown", the document will not be delivered.

mimePatterns
: List of glob patterns that match file name suffixes for matching mime types. For example:

```
mimePatterns=.x* .a?
mime.x*=text/xml
mime.a?=application/octet-stream
```

The types corrosponding to mime patterns are searched for in mimePattern order, first looking for `prefix.mime.pattern` then `mime.pattern`. If neither property exists, then the type is invalid.

prefix
: Only url's that start with this are allowed. defaults to "". The prefix is removed from the url before looking it up in the file system. So, if prefix is `/foo` then the the file `[root]/foo/bar.html` will be delivered in response to the url `/bar.html`.

root
: property for document root (.) Since the document root is common to many handlers, if no root property is found with the supplied prefix, then the root property with the empty prefix ("") is used instead. This allows many handlers to share the common property.

FilterHandler

The `FilterHandler` captures the output of another `Handler` and allows the ouput to be modified.

exitOnError
: If set, the server's `initFailure` will set any of the filters fail to initialize. No handler prefix is required.

filters
: A list of `Filter` names. The filters are applied in the specified order to the output of the wrapped handler.

handler
: The name of the `Handler` whose output will be captured and then filtered. This is called the "wrapped handler".

prefix, suffix, glob, match
: Specify the URL that triggers this handler. (See `MatchString`).

GenericProxyHandler

Handler for implementing a virtual web site.

headers
: A list of white space delimited tokens that refer to additional HTTP headers that are added onto the polled request. For each token the server properties `[token].name` and `[token].value` define a new http header.

host
: name of host site to proxy to.

port
: Host port to proxy to (defaults to 80).

prefix
: URL prefix must match

proxyHost
: Which proxy host to use (if any) to contact "host".

proxyPort
: The proxy's port (defaults to 80)

GroovyServerTemplate

The `GroovyServerTemplate` looks for each `<groovy>` tag in an HTML page and treats the following data up to the next `</groovy>`) tag as a groovy script to evaluate.

debug
: If this configuration parameter is present, this class replaces the `<groovy>` and `</groovy>` tags with comments, so the user can keep track of where the dynamically generated content is coming from by examining the comments in the resultant HTML document. By default, the `<groovy>` and `</groovy>` are completely eliminated from the HTML document rather than changed into comments.

root
: The document root, if the script is a relative file name. If the "root" property under the template prefix is not found, the global "root" property is used. If the global "root" property is not found, the current directory is used.

script
: The name of the Groovy script to evaluate when the interpreter is created. This script only evaluated when the interp is created, not on every request. The variables `prefix` and `server` are set before this file is evaluated, and are references to the parameters passed to a `handler` init method.

HighlightTemplate

Template class for highlighting text that matches a regular expression.

exit
: If set, the template "init" method will return false, and no further processing will take place. This is useful if this template is used by itself.

highlight
: A regular expression that with match any text between entities.

matchCase
: If specifies, matches are case sensitive. The default is to ignore case when matching.

mustHighlight
: If not set, the entire document is surrounded by implicit `highlight` tags. If set no highlighting will take place until an actual `highlight` tag is present.

options
: the set of name=value options that will be added to the starting tag of the tag pair, above. The default is "color=red".

substitute
: The string to substitute for the matched text. This is for advanced uses. If specified, the values for `tag` and `options` are ignored. The default is: <${tag} ${options}>&</${tag}> The format of the string is a regular expression substitution string, which supports ${} style variable substitutions from the request properties.

tag
: the html/xml tag pair that will be added before and after all text maching "highlight", above. The default is " .....

HistoryFilter

The `HistoryFilter` is both a `Handler` and a `Filter` that keeps a record of all pages visited by a given session.

admin
: URLs beginning with this prefix cause the `HistoryFilter` to store the history information for the current Session in the request properties

filter
: If specified, then this is a `Regexp` pattern to match against the "Content-Type" of the result. Setting this also implies that the `HistoryFilter` will be invoked as a `Filter` and not a `Handler`. The default value is "", which indicates that the "Content-Type" is **not** examined and that this `HistoryFilter` will be invoked as a `Handler`.

nosession
: The Session ID to use if the Session ID was not specified. The default value is "common".

prefix
: This handler will only process URLs beginning with this string. The default value is "", which matches all URLs.

session
: The name of the request property that holds the Session ID. The default value is "SessionID".

HomeDirHandler

Handler for converting ~username queries.

home
: The mount-point for home directories, defaults to "/home/".

prefix
: The url prefix used to identify home directory queries. Defaults to "/~".

root
: The name of the root property to set. Defaults to "root".

subdir
: Name of the directory in the user's home directory that represents the user's "doc root"

JavaScriptTemplate

The `JavaScriptTemplate` looks for each `<server language="javascript">` (or `<javascript>`) tag in an HTML page and treats the following data up to the next `</server>` (or `</javascript>`) tag as a JavaScript script to evaluate.

debug
: If this configuration parameter is present, this class replaces the `<server>` and `</server>` tags with comments, so the user can keep track of where the dynamically generated content is coming from by examining the comments in the resultant HTML document. By default, the `<server>` and `</server>` are completely eliminated from the HTML document rather than changed into comments.

root
: The document root, if the script is a relative file name. If the "root" property under the template prefix is not found, the global "root" property is used. If the global "root" property is not found, the current directory is used.

script
: The name of the JavaScript script to evaluate when the interpreter is created. This script is only evaluated when the interp is created, not on every request. The variables `prefix` and `server` are set before this file is evaluated, and are references to the parameters passed to a `handler` init method.

JunkBusterHandler

Remove *junk* images from web pages.

host
: The regular expression matching url's to reject. If the expression starts with a '@', it interpreted as a file name (minus the @) that contains a new-line separated list of regular exporessions. See `Regexp` for more information on regular expressions.

image
: The file to contain the replacement image.

LDAPTemplate

The `LDAPTemplate` is invoked to process LDAP tags embedded in a document.

attributes
: The space-delimited list of attribute names to return from the LDAP "dn" or "search" operation. If empty or unspecified, all attributes for the record are returned. Not all records in the LDAP database have the same attributes. Defaults to "".

authenticate
: The Distinguished Name used for authenticating to the LDAP server, if necessary. Defaults to "". This would be a good option to specify in the configuration file rather than in the LDAP tag.

base
: The Distinguished Name of the base record that forms the root of the search tree in the LDAP database. Used only with the "search" option. Defaults to "". This would be a good option to specify in the configuration file rather than in the LDAP tag.

dn
: The Distinguished Name (DN) to lookup in the LDAP server. The format of a DN is described in RFC-1779. The "dn" and "search" options are mutually exclusive. When "dn" is specified, only zero or one result will be returned from the LDAP database. The result (if any) will be stored in the request properties as follows:

```
<ldap dn="uid=6105,ou=people,o=WebAuth" prefix=name>
<property name.dn>
<property name.cn>
<property name.sn>
<property name.objectclass>
```

etc. The property `name.dn` is the DN that was found. Other properties will be defined as shown, based on the attributes present in the LDAP record.

host
: The hostname of the LDAP server, of the form `"host"` or `"host:port"` if the server is not running on the standard LDAP port. Defaults to "". This would be a good option to specify in the configuration file rather than in the LDAP tag.

password
: The password sent when the "authenticate" option is used. Defaults to "".

prefix
: The string prefix for the property names that will be stored in the request properties to hold the results. If not specified, defaults to the prefix of this template as specified in the configuration file.

scope
: The scope of the LDAP search, one of
  - "base" Search only in base record (specified by the "base" option).
  - "one" Search only records one level below the base record.
  - "sub" Search the entire subtree below the base record.

  Used only with the "search" option. Defaults to "sub". This would be a good option to specify in the configuration file rather than in the LDAP tag.

search
: The search filter to use when searching the LDAP server. The format of a search filter is described in RFC-1558. The "search" and "dn" options are mutually exclusive. When "search" is specified, zero or more results will be returned from the LDAP database. The results will be stored in the request properties as follows:

```
<ldap search="(givenname=scott)" prefix=name>
<property name.rows>
<property name.rowcount>

<property name.0.dn>
<property name.0.cn>
<property name.0.mail>

<property name.1.dn>
<property name.1.cn>
<property name.1.pager>
```

etc. The property `name.rows` is set to the list of record indices found, and can be used by the BSL tag `<foreach name=x property=name.rows>` to iterate over all records. Other properties will be defined for each of the records found as shown, based on the attributes present in the each of the LDAP records.

LogHandler

Handler for logging information about requests.

flush
: The number of lines of logging output that may be buffered in memory before being written out to the log file. default to 25.

format
: The format of the output string. Embedded strings of the form "%X" are replaced, based on the following values for "X":
  - %
    A single "%"
  - b
    Bytes written to the client for this request.
  - d
    Time to service this request (ms).
  - i
    Client ip address.
  - m
    Request method (GET, POST, etc)
  - M
    Memory utilization (%).
  - q
    query string (if any)
  - r
    Requests used for this connection.
  - s
    HTTP result code.
  - t
    TimeStamp (ms since epoch).
  - T
    Number of active threads.
  - u
    URL for this request.
  - v
    HTTP protocol version (10 or 11).

  Defaults to "%u;%t:%d:%b".

handler
: The name of the handler to wrap. This can either be the token for the class, or the class name itself.

headers
: If specified This string is tacked onto the end of the "props" string. Entries in the HTTPrequest headers may be included using ${...} substitutions.

logFile
: The name of the file to log the output to. If the file already exists, data is appended to it. If the file is removed, a new one is created. If no name is specified, one is invented that contains the name and port

of the server. Unless an absolute path is specified, the log file is placed in the current directory.

props
: If specified This string is tacked onto the end of the "format" string. Entries in the Request Properties may be included using ${...} substitutions.

title
: if present, this is output as the first line of the file

MD5Filter

Filter to compute the MD5 checksum of the content, and generate the appropriate "Content-MD5" http header.

prefix, suffix, glob, match
: Specify the URLs that trigger this filter (See `MatchString`).

MacroTemplate

Template class for defining macros.

init
: The name of the file (or resource) to read a default set of macro definitions from. If an absolute path isn't specified, the file is taken relative to the document root. The default macros are kept in the SessionManager on a per-server basis. All macros defined in the "init" file are global. All markup in this file outside of a macro definition is ignored. If "subst" is present as an attribute of a macro definition in this file, then all ${...} are evaluated relative to "server.props" before the macro is defined.

subst
: If specified, then any tags that are not processed by any templates will have all ${..} contructs in attribute values substituted. This subsumes the function of the "SubstAllTemplate".

MatchString

Utility class for handlers to determine, based on the URL, if the current request should be processed.

glob
: The glob pattern the url must match. If defined, this overrides both `prefix` and `suffix`.

ignoreCase
: If present and `match` is defined, this causes the regular expression match to be case insensitive. By default, case counts.

match
: The reqular expression pattern the url must match. If defined, this overrides `glob`.

prefix
: The url prefix the url must match (defaults to "/").

suffix
: The url suffix the url must match (defaults to "").

ModifiedTemplate

Template class for computing `last-modified` times for content that is processed through templates.

debug
: If this configuration parameter is present, `modified` tag is replaced by a comment. Otherwise it is removed from the document.

MultiHostHandler

The `MultiHostHandler` allows the user to handle a set of host names that are all running on the same IP address.

prefix.config
: Read in the file specified by "config" to initialize this sub-server's server properties. The file is expected to be in java properties format. If not specified, this sub-server shares a copy of the main server's properties, otherwise, the main server's properties are used as the "default". If this property is specified and no config file is found, then the sub-server isn't started.

  The property "root", if included in the "config" file, is treated specially: If it does not represent an absolute path, then it is resolved relative to the main server's root.

prefix.handler
: The main handler for the server with the given *prefix*. If this property is not specified, it defaults to the `FileHandler`.

prefix.host
: Each server is started with a given *prefix*. The property *prefix*.host specifies a Glob pattern for a virtual hostname the server will be expected to handle. If this property is not specified, the server's virtual hostname will just be *prefix*. If multiple host patterns could match a given "Host" header, the first match in the "servers" list matches first.

servers
: The list of prefixes for the other servers. Each server will be initialized from the main `server.props` with the specified prefix. In this way, the configuration parameters for all the sub-servers can be stored in the same `Properties` object.

MultipartSetTemplate

Version of the SetTemplate that reflects form/multipart data in Request.props.

noEncode
: If present, no encoding is performed on file uploads.

query
: If present, The form data is translated from form/multipart and placed into the request properties, prefixed by the value of `query`.

saveName=name
: The name to use to save the file. May contain ${...} substitutions. The variables ${fileName}, ${fieldName) and ${prefix} may be used here as "special" variables to make creating a file name easier. `saveName`defaults to: `${prefix}-${fieldName}-${fileName}`

savePattern=[glob pattern]
: If set, then the form is scanned for field names that match *glob pattern*. If a match is found, then the *next* form element of type *file* is saved to a file in the document root instead of being loaded as a property. The name of the file is specified by the value of the `saveName` entry.

NoImageTemplate

Sample template class for removing all images from a web page, and replacing them with their alt strings.

template
: The text used to replace the image. The first "%" will contain the image "alt" string, if any.

NotFoundHandler

Handler for returning "file not found" errors back to the client.

fileName
: The name of the file to send for missing files. Defaults to "notfound.html"

prefix, suffix, glob, match
: Specify the URL that triggers this handler. (See `MatchString`).

root
: The location of the document root for locating the default "not found" file (also looks using prefix of "").

type
: The file type, defaults to text/html

PlainFilter

Filter to turn text/plain into html.

template
: The string to use as an html template. The string should contain a single "%", which is replaced by the text/plain content. The default stuff the content between <pre>...</pre>.

PollHandler

Handler for periodically polling another web site, whose results are (optionally) added to the server's properties.

count.attempts
: The total number of polls attemped.

count.errors
: The total number of poll failures.

error.at
: The poll attempt # for the last failure.

error.msg
: The message describing the last failure.

error.time
: The timestamp of the last failure.

fast
: If set, don't wait "interval" before 1st poll.

format
: a `date format specifier` to use for matching "match" patterns. Defaults to "EE-MM-dd-HH-mm".

headers
: A list of white space delimited tokens that refer to additional HTTP headers that are added onto the polled request. For each token the server properties `[token].name` and `[token].value` define a new http header.

interval
: The interval (in seconds) to fetch the url. Defaults to 10 seconds. If *match* is specified, this is the interval used to check for a time/date match. At each "interval", the current time format is computed, based on "format", below. If the computed format has not changed since the previous poll, then no poll is done. The interval is recalculated after each poll.

match
: If specified, a regular expression that must match the current time for this URL to run. The format to match is specified by the "format" parameter, below. "EEE-dd-HH-mm" (eg: Thu-Dec, 14, 14:12 pm).

namespace
: The namespace to use to store the properties to. If the `sessionTable` (see below)parameter is identical to the `sessionTable` parameter of the `SetTemplate`, then this specifies the `namespace` parameter that may be used with the `SetTemplate` "namespace" parameter to obtain the extracted data. Defaults to the "prepend" parameter.

post
: The "post" data, if any. ${...} are evaluates as per `url` above.

prepend
: The string to prepend to the properties. If not supplied no properties are loaded.

proxy
: If specified, connect through a proxy. This should be in the form `host:port`, or `host` it the desired port is `80`.

sessionTable
: The name of the SessionManager table to use for storing values. By default, properties are stored in server.props. The value should match the `sessionTable` used by the `SetTemplate` to allow values obtained by this handler to be accessable from within templates.

If the `sessionTable` is set, the `namespace` value is used to name the table (e.g. the `namespace` specified by `SetTemplate`. If no namespace parameter is given, then `prepend` is used as the namespace parameter.

timestamp
: The timestamp for the last successful poll.

url
: The fully qualified URL to fetch periodically. any ${...} constructs are evaluated at each poll, with the values in the server properties object.

PropertiesCacheManager

A version of the CacheManager that saves out any session state that is either a "java properties" object, or implements "Saveable" into a directory in the filesystem, one file per entry, then restores them on server startup.

defer
: If set, the saved session information is not reconstructed upon startup. Only the list of sessions is read in; the session information is restored only when needed.

filePrefix
: A prefix pattern to use for all session files. Defaults to the handler prefix.

match
: A glob pattern that matches the url (or url?query if a query is used. Defaults to "*?*save=true".

storeDir
: The directory to use to store the state files. It is created as needed when the state is saved. Defalts to "store".

PropertiesHandler

Handler for returning selected request properties as a text/plain document in java properties format.

comment
: Comment to put on output (defaults to *select*).

prefix, suffix, glob, match
: Specify the URL that triggers this handler. (See `MatchString`).

select
: Glob pattern to match properties selected (Defaults to *). This is re-examined at every request.

type
: Type of output to generate (defaults to text/plain).

PropsTemplate

[*Deprecated, use the the SetTemplate*.

headers
: The mime headers are placed into the request object, prefixed by the value assigned to "headers". The values: url, query, method, and version are copied from the request object into the properties. The

clients IP address is saved in the "address" property.

query
: The query parameters are placed into the request object, prefixed by the value assigned to "query".

url.orig
: If set and "headers" are requested, this value is used as the `url` instead of the one in request.url.

ProxyHandler

Handler for implementing a web proxy.

auth
: The value of the proxy-authenticate header (if any) sent to the upstream proxy

proxyHost
: If specified, the name of the upstream proxy

proxyPort
: The up stream proxys port, if a proxyHost is specified (defaults to 80)

useproxy
: The name of the SocketFactory class to use for this handler. If additional properties are required to set up the SocketFactory, it should be configured as a handler instead. This is here for convenience only.

ProxyPropertiesHandler

Obtain properties format content from remote websites, and add it to the current request properties.

prepend
: The prefix that should be prepended to each property before it is inserted into the request properties

type
: The document type for files to process as java properties (defaults to text/plain)

url
: The url that should be used to fetch the remote content. If not specified, the curent url is used instead. Any ${...} constructs in the url are evaluated at each request.

PublishHandler

Handler for supporting publishing from Communicator.

prefix, suffix, glob, match
: Specify the URL that triggers this handler. (See `MatchString`).

session
: The the name of request property holding the session information to provide the credentials for posting. The default is "SessionID".

PushHandler

Skeleton Handler for uploading files using multipart/form-data.

prefix, suffix, glob, match
: Specify the URL that triggers this handler. (See `MatchString`).

PythonServerTemplate

The `PythonServerTemplate` looks for each `<server language="python">` (or `<python>`) tag in an HTML page and treats the following data up to the next `</server>` (or `</python>`) tag as a python script to evaluate.

debug
: If this configuration parameter is present, this class replaces the `<server>` and `</server>` tags with comments, so the user can keep track of where the dynamically generated content is coming from by examining the comments in the resultant HTML document. By default, the `<server>` and `</server>` are completely eliminated from the HTML document rather than changed into comments.

root
: The document root, if the script is a relative file name. If the "root" property under the template prefix is not found, the global "root" property is used. If the global "root" property is not found, the current directory is used.

script
: The name of the Python script to evaluate when the interpreter is created. This script only evaluated when the interp is created, not on every request. The variables `prefix` and `server` are set before this file is evaluated, and are references to the parameters passed to a `handler` init method.

QueueTemplate

Template class for Managing simple Queues, allowing text communication among sessions.

age
: Set how long the message has been q's (in seconds)

error
: Something went wrong. Set an error message.

from
: Set the (unauthenticated) sender.

items
: Set the number of Q'd items.

sent
: Set the timestamp (in sec) of when the item was q'd.

ReFilter

Filter to replace text content via a reqular expression substitution.

noCase
: If set, case-insensitive matchins is performed.

oneOnly
: If set, only replace the first match. by default, all matches are replaced.

re
: The regular expression to match the content

sub
: The replacement expression. If not specified, the matched content is deleted.

RePollHandler

Do regsub processing on content to extract properties.

encoding
: The character set encoding to use when converting the request results to a string. Defaults to the default encoding.

prepend
: The string to prepend to all properties. Extracted properties will contain the the "re" token as an additional prefix.

re.exp
: The regular expression to search for.

re.flags
: One or more ASCII flags to control how this "re" is processed. Consists of one or more of The following (defaults to "SFE"):. Characters not on this list are ignored.
  - **E** Extract current result into server properties. See the rules for naming the properties, below. At least one regular expression Must have an "E" flag.
  - **F** Process if previous "RE" failed.
  - **I** Ignore case in expression
  - **O** only do one substitution or extraction, not all
  - **R** Reset content to original before proceeding Otherwise, the result of the previous substitution (if any) is used.
  - **S** Process if previous "RE" succeeded

re.key
: The index of the sub-match (starting at 1) that will be used to name the row number portion of the property name instead of a counter. This is useful if one of the sub-matches will be unique for each matching pattern. This option is ignored if the "O" flag is specified, as there will be only one match so no "key" is required.

re.names
: A white-space delimited set of tokens to use instead of numerical indices to name the properties. The first name in the list names the entire match, the remaining names name the sub-expressions. If there are more properties extracted than names provided, the "left over" properties will have numerical indeces. This implies ’E’.

If the name "X" is used, no property will be extracted for that match.

re.sub
: The regular expression substitution pattern. If ’E’ is specified, the substitution is done after the extraction.

re
: the list of "re" tokens to process in order. Each "re" token has the following attributes:

ReplaceFilter

Filter to replace current content with a static form, or template.

debug
: If set, the template is re-read each time. Otherwise a cached copy is used.

fileName
: Name of the file to use as the form or template. The file is searched for as a *Resource* if not found in the filesystem.

root
: The document root used to find the template file. If not found, "root" with no prefix is used instead.

type
: Text subtype of content to filter. Defaults to "html"

ResourceHandler

Handler for serving documents out of the jar file.

default
: The default file name for url references ending in /

mime.xxx
: The mime type for suffix xxx. See `FileHandler` for a description of how to set mime types for url suffixes.

prefix, suffix, glob, match
: Specify the URL that triggers this handler. (See `MatchString`).

root
: The document root path within the jar file

ResourceLimitHandler

Handler for server resource management.

file
: The file name or resource of the html file to return if resources run low. Defaults to "busy.html".

memory
: The minimum # of remaining bytes available to the vm

retry
: The number of seconds to request the client wait before retrying the request.

threads
: The Max number of active threads

RestartHandler

Restart the server when a sigHUP is received.

config
: The name of the configuration file to use for this server. Relative paths are resolved relative to the current directory. If no file is specified, the server continues to use its existing configuration [which has presumably been modified in-place]. If a config file is specified and the config file hasn't changed, then no restart is done.

keep
: If set, and a config file is specified, the existing configuration (server.props) is not cleared first.

RestrictClientHandler

Simple access control hander based on source ip addresses.

allow
: The regular expression that matches the ip addresses of clients (in xxx.xxx.xxx.xxx format) that are permitted to access url's starting with `prefix`.

deny
: The regular expression that matches the set of ip **names** that should be denied access. This is to make complying with silly EAR requirements easier. The use of this option implies a reverse DNS lookup, which could be expensive, as DNS names (and not ip addresses) are used for the comparison. Case insensitive matching is used.

prefix, suffix, glob, match
: Specify the URL that triggers this handler. (See `MatchString`).

redirect
: Name of the url to re-direct to if permission is denied. If not specified, a simple message is sent to the client.

RolesHandler

Handler for associating roles with an id.

mapFile
: The absolute path to the java properties file containing the it to role mapping.

prefix, suffix, glob, match
: Specify the URL that triggers this handler. (See `MatchString`).

roleName
: The property to place the result of the id lookup into. Defaults to "roleName";

SessionID
: The property to use to look up the id. Defaults to "SessionID".

SMTPHandler

Handler (or template) for Sending an email message via SMTP.

host
: The mail host (e.g. listening on the SMTP port). Defaults to "localhost".

prefix, suffix, glob, match
: Specify the URL that triggers this handler. (See `MatchString`).

SessionFilter

Filter to manage browser sessions using browser cookies or URL rewriting as needed.

cookie
: The name of the cookie to use (defaults to "cookie"). If the name is "none", then no cookies are used. Instead, session rewriting will occur for every session.

cookiePrefix
: The URL prefix for which the cookie applies. Defaults to "/".

persist
: If set, cookies persist across browser sessions. If cookies are disabled, no persistence is available.

session
: The name of the request property that the Session ID will be stored in, to be passed to downstream handler. The default value is "SessionID". If the session property is set, and not empty, then no processing is done.

suffix
: A regular expression that matches url suffix we process. Defaults to `html|xml|txt`.

SetTemplate

Template (and handler) class for setting and getting values to and from the current (or other) request context.

autoImport
: If set to "1", the namespace for the session is automatically imported. (defaults to "1");

debug
: If set, the original tag is included in a comment, otherwise it is removed entirely.

headers
: The mime headers are inserted into the request object, prefixed by the value assigned to "headers".

  In addition, the following properties (prefixed with "headers") are also set:

  - **address** The ip address of the client
  - **counter** A monotonically increasing counter (# of client requests accepted since the server started).
  - **method** The request method (typically GET or POST).
  - **protocol** The client HTTP protocol level (for 1.0 or 1.1)
  - **query** The current query data, if any.

- **timestamp** A timestamp (in ms since epoch) from when this request was first accepted.
- **url** The current url.
- **hostname, hostport** The name and port parts of the "host" header, if set.

imports
: Defines a set of (white space delimited) namespaces that will automatically be imported at the beginning of each page. Each namespace name will be processed for ${...} substitutions before an import is attempted. If the namespace doesn't already exist, the import is ignored.

mustMatch
: Set to a glob pattern that all names must match in order to be set. This may be used to prevent malicious html pages (what a concept) from changing inappropriate values.

noserver
: The "server" namespace will no longer be mapped to `server.props`

noSet
: If set, then the "set" tag will be disabled.

query
: The query parameters are inserted into the request object, prefixed by the value assigned to "query".

querySet
: If set, then properties may be set in query parameters, to the "handler" portion, but only if they match the glob pattern.

saveOk
: This must be specified in order for the "namespace store" or "namespace load" functions to operate.

session=value
: The request property to find the session information in. Normally this should be the same as the session property used by the container calling this as a template.

sessionTable
: The name of the SessionManager table to use for storing values. Defaults to the template handler's prefix. When configured in one server both as a handler and a template, the sessionTable should be set to the same value in both cases to get a useful result.

url.orig
: If set and "headers" are requested, this value is used as the `url` instead of the one in request.url.

SimpleSessionHandler

Handler for creating browser sessions based on information found in the http request.

digest
: If set, the "value" is replaced by the base64 encoding of the MD5 checksum of `value`.

extract
: If specified, a string to use as the session-id. ${...} values will be searched for first in the HTTP header values, and then in the request properties.

  In addition to the actual HTTP headers, the pseudo http headers `ipaddress, url, method, and query` are made available for ${...} substitutions.

force
: If set (to anything), a session ID is set even if one already exists.

prefix, suffix, glob, match
: Specify the URL that triggers this handler (See `MatchString`).

re
: If specified, a regular expression that the extracted data must match. if it doesn't match, no session id is installed. The default is ".", which matches any non-empty string. If the first character is "!" then the sense of the match is inverted, But only for determining whether a match "succeeded" or not. no sub-matches may be used in computing the key value in this case.

session
: The name of the request property that the Session ID will be stored in, to be passed to downstream handlers. The default value is "SessionID". If the property already exists, and is not empty, no session will be defined (unless force=true).

value
: The value of the session ID. May contain & or (n=0,1,2...) constructs to substitute matched sub-expressions of `re`. The default is "&" , which uses the entire string "extract" as the `session` id. ${...} are substituted (but not 's) for `value` before looking for ' ' sequences that are part of the regular expression matches.

SnarfTemplate

Template class for extracting content out of <snarf property=xxx> .

debug
: If set, the *snarf* tags will be replaced by comments.

prepend
: The string to prepend all properties with. Defaults to the handler's prefix.

SqlTemplate

Sample Template class for running SQL queries via jdbc and placing the results into the request properties for further processing.

driver
: The name of the jdbc driver class for the desired database. Currently, only one driver may be specified. (e.g. *`prefix`*`.driver=org.gjt.mm.mysql.Driver`).

sqlPrefix
: The properties prefix for any additional parameters that are required for this connection. For example:

```
prefix.sqlPrefix=params
params.user=my_name
params.password=xxx
```

All of the parameters are supplied to the jdbc connection at connection time.

url
: The jdbc url used to establish a connection with the database. (e.g. *`prefix`*`.url=jdbc:mysql://host/db?user=xxx&password=yyy`).

SslHandler

Handler for installing SSL into the server.

certDir
: The absolute pathname of the directory containing the server's certificates. The "main" in this class should generate a representitive sample (but doesn't). In the mean time, you can run "java CreateCertificates" with the Iaik jar in your path, to create a set of server test certificates in the `certs` directory.

certRequired
: If set, the server will ask the user for a client cert.

fingerprint.*
: The base64 encoded fingerprints of the clients certificates

issuer.*
: Information about the certificate issuer.

owner.*
: Information about the certificate owner.

session
: The property to put the certificate serial number into. Defaults to "cert.id".

StunnelHandler

Handler to enable proper interaction with a protocol conversion gateway, by rewriting "redirect" directives properly.

protocol
: The protocol to replace "http" with when redirection via a gateway (defaults to "https").

ssl
: The regexp to match client ip addresses that are coming from ssl gateways (such as stunnel).

SubstPropsHandler

Handler that performs value conversions on ${.

[token.code]
: The name to match the "token" in the property name. The default is "[token]".

key
: The regular expression substitution string used to represent the actual property name to filter. The default is `\2`

match
: A regular expression that matches a property name that is a candidate for filtering. This expression should have at least 2 sets of ()'s in order to gather values for "key" and "token" below. The default value is `^([a-z]+)([^)]+)$`

token
: The regular expression substitution string used to represent the filter name or "token". The default is `\1`

Using the defaults for "match", "key", and "token", a property named "foo" would be represented as `${xxx(foo)}` where "xxx" is the name of the conversion filter.

tokens
: A witespace separated list of filter names or "token"s that map the conversion filters to conversion classes. For each token (e.g. foo), there should be a property of the form "foo.class" which specifies the name of the class that implements the filter, (and implements the Convert interface described below). Any additional properties (e.g. x, y, z) needed to initialize a filter should be present in the properties file as "foo.x, foo.y...".

SubstPropsHandler.Resub

Do a regexp substitution on a value.

match
: A Regular expression that matches the string value.

sub
: The regular expression substitution to perform. All occurances of "match" are substututed.

SupplyHandler

Sample Handler for dispatching different users to different url's based on a special http authentication header.

default
: default map

header
: http header (authentication)

mapFile
: properties file

prefix
: url prefix

realm
: The authentication realm (basic)

TclFilter

Wrapper for writing FilterHandler filters in TCL.

script
: The name of the TCL file sourced on startup. The `init(sunlabs.brazil.server.Server, java.lang.String)` parameters are made available as the global variables `prefix` and `server`.

TclHandler

Handler for writing handlers in tcl.

callback
: The name of the TCL script to call at each request. Defaults to `respond`.

prefix, suffix, glob, match
: Specify the URL that triggers this handler. (See `MatchString`).

script
: The name of the TCL file sourced on startup. The `init(sunlabs.brazil.server.Server, java.lang.String)` parameters a make available as the global variables `prefix` and `server`.

TclRePollHandler

Post-process all "polled" properties with tcl code.

debug
: If set, the "script" is sourced each time.

script
: The name of the TCL script to call at each request. The interpreter is created with the following global variables. The script is "evaluted" once on startup.
  - **prefix** The handler prefix
  - **server** The current server instance
  - **logLevel** The current server log level setting
  - **argv0** The name of the running script

  The tcl procedure "process" is called with the java properties object. Any modifications to that object are done here.

TclServerTemplate

The `TclServerTemplate` looks for each `<server language="tcl">` (or `<"tcl">`) tag in an HTML page and treats the following data up to the next `</server>` tag as a Tcl script to evaluate.

debug
: If this configuration parameter is present, this class replaces the `<server>` and `</server>` tags with comments, so the user can keep track of where the dynamically generated content is coming from by examining the comments in the resultant HTML document. By default, the `<server>` and `</server>` are completely eliminated from the HTML document rather than changed into comments.

root
: The document root, if the script is a relative file name. If the "root" property under the template prefix is not found, the global "root" property is used. If the global "root" property is not found, the current directory is used.

script
: The name of the Tcl script to evaluate when the interpreter is created. This script only evaluated when the interp is created, not on every request. The variables `prefix` and `server` are set before this file

is evaluated, and are references to the parameters passed to a `handler` init method.

TemplateFilter

The `TemplateFilter` sends HTML content through an Html/XML parser to a set of `Template`s.

encoding
: The charset encoding to use to represent the content as text. If none is specified, the default encoding is used.

outputEncoding
: The character encoding to use to interpret the template results. If no "outputEncoding" is specified, then "encoding" is used. Once template processing is complete, the results are converted into a byte stream for transmission to the next filter, using "outputEncoding", if specified. If not specified then the default encoding is used.

session
: The request property that contains the session ID. If no "session" property is found with the supplied prefix, then the global "session" property is used instead. The default value is "SessionID".

subtype
: Restrict this template to only handle specified text sub-types. defaults to the empty string, which implies any text sub-type.

tagPrefix
: If specified, all tag names defined for each template class are prefixed with *tagPrefix*. This parameter only takes effect if the `tagPrefix` option is not specified for an individual template.

templates
: A list of template names. For each name in the list, the property *`name`*`.class` is examined to determine which class to use for each template. Then `name` is used as the prefix for other template specific properties if any. If *`name`*`.class` does not exist, then `name` is assumed to be the class name, and there are no template specific properties for the template. Methods in the template classes will be invoked to process the XML/HTML tags present in the content.

UrlMapperHandler

Handler for mapping or redirecting URLs.

export
: If set, use this as a properties prefix, and set request properties for each sub-expression in "match". (E.g. [export]1 [export]2 ...).

ignoreCase
: If set, the case of the expression is ignored.

match
: The regexp to match a url. May contain constructs of the form ${xxx}, which are replaced by the value of `request.props` for the key *xxx*

redirect
: If set, the request is redirected instead of being rewritten

replace
: The url to replace it with. This may contain both regular expression sub-patterns, such as "", or variables of the form ${..} which are replaced with the equivalent request properties.

source
: If set, then this string is used instead of the url as the *source* of the match. Variable substitution using ${xxx} is performed on *source*, which, if unset, defaults to "${url}". If set, the request properties "url", "host" and "user-agent" are set first. The *source* property is obtained at init time, but evaluated (for ${...}) at every request.

As an example, the configuration:

```
prefix.source=${user-agent}!${url}
prefix.match=Lynx.*!(.*)
prefix.replace=/text\1
```

could cause all browsers with "Lynx" in their user agent header to the "text" sub-directory.

UrlNavBarTemplate

Template class for dynamically generating a navigation bar by looking at portions of the url.

includeDir
: Normally, if the URL refers to the directory (e.g. it ends with a /), no nav bar entry is generated. If this property is set, the entry is generated.

prepend
: Use as a prefix on the property name, instead of "NAV.".

VelocityFilter

A filter for processing markup that is a Velocity template.

script
: The name of a BeanShell script file.

session
: The name of the propery containing the session. The default session, if none is found, is "common".

VirtualHostHandler

Handler for managing virtual hosts using the same server configuration.

[prefix.[host].]
: If the "mapping" property exists that matches the incoming "host" name, then that value is used instead of [host] to rewrite the "url" or "root".

addlevel=true|false
: If "true", "levels" is specified, and the number of tokens (levels) in the hostname exceeds "levels", then all the extra tokens in the hostname are prepended to the URL as initial directories: If "levels" is 3, and "addlevel=true" then: host `http://a.b.c.d.e/foo.html` will be mapped to `http://c.d.e/b/a/foo.html`, and the file "foo.html" should be at

[docroot]/c.d.e/b/a/foo.html].

If "addlevel=false", then `http://a.b.c.d.e/foo.html` will be mapped to `http://c.d.e/foo.html`, and the file "foo.html" should be at [docroot]/c.d.e/foo.html. In this case, the "a.b" part of the host is available as part of the host property, which retains its original value.

default
: If set, then all hosts for which no mappings are defined are mapped to the value of this property.

levels
: If defined, then for the purpose of host matching, only "levels" of hostnames are considered. If levels=3, then for host: a.b.c.d.e, the host is considered to be "c.d.e". This enables support for wildcard-host matching within a virtual domain.

maproot
: If set upon server startup, this handler changes the "root" property instead of the "url" property, by appending the "host" onto the document root, instead of prepending the "host" to the url.

(end of summary)

Template Tag Summary by class

class: sunlabs.brazil.asterisk.AsteriskAGIHandler
<agi>
- **String attributes**: command

class: sunlabs.brazil.asterisk.AsteriskHandler
<asterisk>
<amicommand>
- **String attributes**: action, prepend, server, timeout

<register>
- **String attributes**: context, exp, key, queue, server

<unregister>
- **String attributes**: exp, key, queue
- **Boolean attributes**: all

class: sunlabs.brazil.beanshell.BeanShellServerTemplate
<server>
- **String attributes**: language

<beanshell>
- **Boolean attributes**: eval

<bsh>

class: sunlabs.brazil.email.EmailTemplate
<forcetimeout>
<email>
- **String attributes**: action

<folder>
- **String attributes**: action, delim, dir, foldername, glob

<message>
- **String attributes**: action, bcc, body, cc, foldername, from, headerlist, msgnum, parturl, refilefolder, subject, to

<sendmail>
<part>
- **String attributes**: body, content-type, encoding, msglimit, name, password, server, startmsg, user

class: sunlabs.brazil.handler.SMTPHandler
<sendmail>
- **String attributes**: body, from, subject, to

<param>
- **String attributes**: name, value

class: sunlabs.brazil.javascript.JavaScriptTemplate
<server>
- **String attributes**: language

<javascript>
- **Boolean attributes**: eval

class: sunlabs.brazil.ldap.LDAPTemplate
<ldap>
- **String attributes**: attributes, authenticate, base, dn, host, limit, password, prefix, scope, search

class: sunlabs.brazil.python.PythonServerTemplate

<server>
- **String attributes**: language
<python>
- **Boolean attributes**: eval

class: sunlabs.brazil.sql.SqlTemplate
<sql>
- **String attributes**: max, na, prefix, timeout, type
- **Boolean attributes**: debug, eval, index, zeroIndex

class: sunlabs.brazil.sunlabs.DateTemplate
<date>
- **String attributes**: format, ms, scan, set, time, zone

class: sunlabs.brazil.sunlabs.DelayHandler
<delay>
- **String attributes**: delay

class: sunlabs.brazil.sunlabs.DigestTemplate
<digest>
- **String attributes**: name, value

class: sunlabs.brazil.sunlabs.ExecTemplate
<exec>
- **String attributes**: command, encoding, prepend, stdin
- **Boolean attributes**: usesh

class: sunlabs.brazil.sunlabs.ListTemplate
<list>
- **String attributes**: append, chunk, chunksize, delete, delim, insert, location, max, name, namespace, overlap, remove
- **Boolean attributes**: clear, front, next, previous, sort, unique

class: sunlabs.brazil.sunlabs.MiscTemplate
<append>
- **String attributes**: delim, name, value
<increment>
- **String attributes**: incr, name
<sequence>
- **String attributes**: count, delim, incr, name, start
<expr>
- **String attributes**: name, value
<eval>
- **String attributes**: markup
- **Boolean attributes**: atEnd
<push>
- **String attributes**: name, value
- **Boolean attributes**: clear
<pop>
- **String attributes**: name
- **Boolean attributes**: clear
<inline>
- **String attributes**: name
- **Boolean attributes**: eval

<stringop>
- **String attributes**: case, convert, name, newname, range, trim
<stringlength>
- **String attributes**: name, value
<mapnames>
- **String attributes**: case, convert, dst, range, src, trim
<random>
- **String attributes**: count, name, start, type

class: sunlabs.brazil.sunlabs.PlainTemplate
<plain>

class: sunlabs.brazil.sunlabs.RemoteHostTemplate
<remotehost>
- **String attributes**: name

class: sunlabs.brazil.sunlabs.SnarfTemplate
<snarf>
- **String attributes**: property

class: sunlabs.brazil.sunlabs.SourceTemplate
<source>
- **String attributes**: encoding, name, src
- **Boolean attributes**: eval, reprocess

class: sunlabs.brazil.sunlabs.XmlTreeTemplate
<xmltree>
- **String attributes**: attribute, cdata, default, delim, dflt, key, modify, name, namespace, root, src, tags, value, xml
- **Boolean attributes**: eval

class: sunlabs.brazil.tcl.TclServerTemplate
<server>
- **String attributes**: language
<tcl>
- **Boolean attributes**: eval

class: sunlabs.brazil.template.AddHeaderTemplate
<addheader>

class: sunlabs.brazil.template.BSLTemplate
<abort>
<break>
<continue>
<foreach>
- **String attributes**: delim, glob, list, match, name, property, sort
- **Boolean attributes**: nocase, numeric, reverse
<if>
- **String attributes**: expr, glob, if, match, name, value
- **Boolean attributes**: any, nocase, not
<extract>
- **String attributes**: glob, map, match, name, null, prepend, replace
- **Boolean attributes**: all, nocase

class: sunlabs.brazil.template.ChangedTemplate
<changed>

<a>
- **String attributes**: href

class: sunlabs.brazil.template.ContentTemplate
<title>
<script>
<style>
<content>
<body>
<meta>
- **String attributes**: content, http-equiv, name
<link>
- **String attributes**: href, rel

class: sunlabs.brazil.template.DebugTemplate
<debug>

class: sunlabs.brazil.template.DirectoryTemplate
<filelist>
- **String attributes**: delimiter, directory, prepend, stats

class: sunlabs.brazil.template.FormClientTemplate
<input>
- **String attributes**: name
<property>
- **String attributes**: default, name
<subst>
- **String attributes**: default, name
<tag>

class: sunlabs.brazil.template.FormTemplate
<form>
- **String attributes**: prepend
<input>
- **String attributes**: name, size, type, value
<select>
- **String attributes**: name
<option>
- **String attributes**: value

class: sunlabs.brazil.template.HighlightTemplate
<head>
<script>
<style>
<server>
<highlight>
<nohighlight>

class: sunlabs.brazil.template.IncludeTemplate
<include>
- **String attributes**: addheaders, alt, encoding, getheaders, href, name, post, proxy, queryList
- **Boolean attributes**: encodeQuery, postinline

class: sunlabs.brazil.template.MacroTemplate
<definemacro>

- **String attributes**: name
- **Boolean attributes**: defer, global

class: sunlabs.brazil.template.ModifiedTemplate

<modified>

class: sunlabs.brazil.template.MultipartSetTemplate

<set>

<property>

<get>

<import>

<tag>

class: sunlabs.brazil.template.NoImageTemplate

- **String attributes**: alt, src

class: sunlabs.brazil.template.PropsTemplate

<property>

- **String attributes**: convert, default, name

<tag>

class: sunlabs.brazil.template.QueueTemplate

<enqueue>

- **String attributes**: data, from, glob, message, meta, name
- **Boolean attributes**: force, nocreate

<dequeue>

- **String attributes**: name, prepend, timelimit

<queueinfo>

- **String attributes**: closed, name, prepend
- **Boolean attributes**: clear, closed, notify, remove

class: sunlabs.brazil.template.ScriptEvalTemplate

<script>

- **Boolean attributes**: esc, eval

<style>

class: sunlabs.brazil.template.SetTemplate

<set>

- **String attributes**: default, name, namespace, value
- **Boolean attributes**: local

<property>

<get>

- **String attributes**: convert, default, max, name, namespace, set

<import>

- **String attributes**: namespace

<namespace>

- **String attributes**: load, name, store
- **Boolean attributes**: clear, remove

<map>

- **String attributes**: convert, match, namespace, replace
- **Boolean attributes**: local

<tag>

- **String attributes**: name

class: sunlabs.brazil.template.TOCTemplate

<h1>

class: sunlabs.brazil.template.TemplateTemplate

<template>

- String attributes: data, prefix, token

- Boolean attributes: ignore

Tag Index by tag

- **a**: ChangedTemplate [p ??]
- **abort**: BSLTemplate [p ??]
- **addheader**: AddHeaderTemplate [p ??]
- **agi**: AsteriskAGIHandler [p ??]
- **amicommand**: AsteriskHandler [p ??]
- **append**: MiscTemplate [p ??]
- **asterisk**: AsteriskHandler [p ??]
- **beanshell**: BeanShellServerTemplate [p ??]
- **body**: ContentTemplate [p ??]
- **break**: BSLTemplate [p ??]
- **bsh**: BeanShellServerTemplate [p ??]
- **changed**: ChangedTemplate [p ??]
- **content**: ContentTemplate [p ??]
- **continue**: BSLTemplate [p ??]
- **date**: DateTemplate [p ??]
- **debug**: DebugTemplate [p ??]
- **definemacro**: MacroTemplate [p ??]
- **delay**: DelayHandler [p ??]
- **dequeue**: QueueTemplate [p ??]
- **digest**: DigestTemplate [p ??]
- **email**: EmailTemplate [p ??]
- **enqueue**: QueueTemplate [p ??]
- **eval**: MiscTemplate [p ??]
- **exec**: ExecTemplate [p ??]
- **expr**: MiscTemplate [p ??]
- **extract**: BSLTemplate [p ??]
- **filelist**: DirectoryTemplate [p ??]
- **folder**: EmailTemplate [p ??]
- **forcetimeout**: EmailTemplate [p ??]
- **foreach**: BSLTemplate [p ??]
- **form**: FormTemplate [p ??]
- **get**: MultipartSetTemplate [p ??] SetTemplate [p ??]
- **h1**: TOCTemplate [p ??]
- **head**: HighlightTemplate [p ??]
- **highlight**: HighlightTemplate [p ??]
- **if**: BSLTemplate [p ??]
- **img**: NoImageTemplate [p ??]
- **import**: MultipartSetTemplate [p ??] SetTemplate [p ??]
- **include**: IncludeTemplate [p ??]
- **increment**: MiscTemplate [p ??]
- **inline**: MiscTemplate [p ??]

- **input**: FormClientTemplate [p ??] FormTemplate [p ??]
- **javascript**: JavaScriptTemplate [p ??]
- **ldap**: LDAPTemplate [p ??]
- **link**: ContentTemplate [p ??]
- **list**: ListTemplate [p ??]
- **map**: SetTemplate [p ??]
- **mapnames**: MiscTemplate [p ??]
- **message**: EmailTemplate [p ??]
- **meta**: ContentTemplate [p ??]
- **modified**: ModifiedTemplate [p ??]
- **namespace**: SetTemplate [p ??]
- **nohighlight**: HighlightTemplate [p ??]
- **option**: FormTemplate [p ??]
- **param**: SMTPHandler [p ??]
- **part**: EmailTemplate [p ??]
- **plain**: PlainTemplate [p ??]
- **pop**: MiscTemplate [p ??]
- **property**: FormClientTemplate [p ??] MultipartSetTemplate [p ??] PropsTemplate [p ??] SetTemplate [p ??]
- **push**: MiscTemplate [p ??]
- **python**: PythonServerTemplate [p ??]
- **queueinfo**: QueueTemplate [p ??]
- **random**: MiscTemplate [p ??]
- **register**: AsteriskHandler [p ??]
- **remotehost**: RemoteHostTemplate [p ??]
- **script**: ContentTemplate [p ??] HighlightTemplate [p ??] ScriptEvalTemplate [p ??]
- **select**: FormTemplate [p ??]
- **sendmail**: EmailTemplate [p ??] SMTPHandler [p ??]
- **sequence**: MiscTemplate [p ??]
- **server**: BeanShellServerTemplate [p ??] HighlightTemplate [p ??] JavaScriptTemplate [p ??] PythonServerTemplate [p ??] TclServerTemplate [p ??]
- **set**: MultipartSetTemplate [p ??] SetTemplate [p ??]
- **snarf**: SnarfTemplate [p ??]
- **source**: SourceTemplate [p ??]
- **sql**: SqlTemplate [p ??]
- **stringlength**: MiscTemplate [p ??]
- **stringop**: MiscTemplate [p ??]
- **style**: ContentTemplate [p ??] HighlightTemplate [p ??] ScriptEvalTemplate [p ??]
- **subst**: FormClientTemplate [p ??]
- **tag**: FormClientTemplate [p ??] MultipartSetTemplate [p ??] PropsTemplate [p ??] SetTemplate [p ??]
- **tcl**: TclServerTemplate [p ??]
- **template**: TemplateTemplate [p ??]
- **title**: ContentTemplate [p ??]

- **unregister**: AsteriskHandler [p ??]
- **xmltree**: XmlTreeTemplate [p ??]

NOTES:

- This summary is generated dynamically; some classes may have tags or attributes that aren't here.
- If two templates are used at once implement the same tag, only the first one specified in the template list will see the tag. This is usually the bad idea.

(end)