Jigsaw
Documentation Overview

Jigsaw Home

This document has the following sections:

1. Short overview of **Jigsaw**
2. User's documentation
3. FAQ about **Jigsaw**
4. Programmer's documentation

Overview of Jigsaw

The **Jigsaw** server is entirely written in Java. Because of this, it offers the following features:

- Portability
- Extensibility
- Object Oriented design

Jigsaw will run on any platform that supports Java, with *no* changes! **Jigsaw** has been reported to run on several platforms **Jigsaw** is made of a core and a set of extension modules. You can add your own modules, *dynamically*, to the server. Moreover, because the Java runtime comes with both threads and garbage collection, your job, as an extension writer, is largely simplified. The full code of the server is object-oriented. The interested thing is that all the resources *are* objects. By opposition to most of the existing servers, which consider a resource as being either a **cgi script** or a **file**. **Jigsaw** allows *any* object to become accessible via HTTP or whatever protocol implemented.

User's Documentation

You will find here the basics on how to install and run the server to the description of the most complex configuration you can do with **Jigsaw**.

1. Installation of **Jigsaw**.
2. Command line options.
3. Basic concepts of **Jigsaw**.

4. JigAdmin 2.0 overview.
5. Configuration of **Jigsaw**.
6. Authentication in **Jigsaw**.
7. Pics in **Jigsaw**.
8. Page Compilation.
9. Redirections in **Jigsaw**.
10. **Tutorials**.
    1. How to use JigAdmin.
    2. How to use JigKill.
    3. Configuration of attributes.
    4. Creation of a resource.
    5. Creation of an indexer.
    6. How to add a new Mime Type in **Jigsaw**.
    7. How to use Server Side Include commands in **Jigsaw**.
    8. How to setup CGI scripts in **Jigsaw**.
    9. How to setup Servlets in **Jigsaw**.
    10. How to setup publication (PUT) in **Jigsaw**.
    11. How to setup **Jigsaw** as a proxy.
    12. How to setup authentication.
    13. How to setup virtual hosting in **Jigsaw**.
    14. Browsing zip files with **Jigsaw**
11. Reference information for frames and resources.

FAQ

Answers to all the Frequently Asked Questions about **Jigsaw** and its use.

Programmer's documentation

You will find here the description of Jigsaw from a programmer point of view, and indications on how to extend it to fulfill your needs.

1. Internal design.
2. JigXML, the internal format used to store the resources metadata.
3. **API** (frames of no-frames). This is the API of the developpement version of **Jigsaw** available from

the public CVS tree.

4. Tutorials.
    1. How to compile **Jigsaw**.
    2. How to write a Resource.
    3. How to write a Frame.
    4. How to write a server-side Filter.
    5. How to write a client-side Filter.
    6. How to write a new Server Side Include command.
5. Sample code.

Jigsaw Team

$Id: Overview.html,v 1.77 1999/09/20 13:52:33 bmahe Exp $

Jigsaw
Installation Procedure

Jigsaw Home / Documentation Overview

Prerequisite:

Before reading further make sure:

- That you have downloaded the latest **Jigsaw** distribution (2.0.1)
- That you have downloaded any implementation of **Java** for your platform (JDK1.1.x or JDK1.2)
- If you're using the zip distribution, make sure it's at least version 6.x

This document describes how to install **Jigsaw**. As an example, we are assuming that you are running either on Windows (be it 95 or NT) or on UNIX, even though **Jigsaw** will (should ?) run on *any* platform that supports **Java**.

The installation process involves the following steps:

Unpacking the distribution file

Pick a place to unpack the distribution: we'll call this directory the *installation directory* (abreviated as *INSTDIR*). This can be any directory, you just have to change your current working directory to it:

```
UNIX : cd INSTDIR

Windows : cd INSTDIR
```

Now unpack the file:

```
UNIX : cat jigsaw.tar.gz | gzip -d | tar xomvf -
```

```
Windows : unzip jigsaw.zip
```

This will create a number of directories under the **Jigsaw** directory:

`Jigsaw/src`
: Contains **Jigsaw** sources.

`Jigsaw/classes`
: Contains the pre-compiled classes.

`Jigsaw/lib`
: Contains some native code support for solaris.

`Jigsaw/Jigsaw`
: Is a sample root directory to run the server in. This directory in turns contain the following sub-directories:

  `Jigsaw/Jigsaw/config`
  : Is the configuration directory for the server

  `Jigsaw/Jigsaw/configadm`
  : Is the configuration directory for the administration server.

  `Jigsaw/Jigsaw/logs`
  : Is the normal directory for log files

  `Jigsaw/Jigsaw/bin`
  : Contains some shell scripts to help you start **Jigsaw**.

  `Jigsaw/Jigsaw/cache`
  : The directory to use for caching when using **Jigsaw** as a caching proxy.

  `Jigsaw/Jigsaw/WWW`
  : Is your exported file space

You are now ready for the next section, which explains how to setup your environment.

Setting up your environment

As **Jigsaw** is just a set of Java classes, you need to specify to the Java interpreter the place where **Jigsaw** classes are stored. This is usually done by setting some CLASSPATH environment variable. This is simply done by the following command for 2.1.0 and up:

```
 UNIX
           # This depends on the shell you are using, we're assuming /bin/sh

CLASSPATH=INSTDIR/Jigsaw/classes/jigsaw.jar:INSTDIR/Jigsaw/classes/sax.jar:INSTDIR/Jigsaw/classes/xp.jar:.
           export CLASSPATH
```

```
Windows
        set
CLASSPATH=INSTDIR\Jigsaw\classes\jigsaw.jar;INSTDIR\Jigsaw\classes\sax.jar;INSTDIR\Jigsaw\classes\xp.jar;.
```

And this one for version up to 2.0.x:

```
UNIX
        # This depends on the shell you are using, we're assuming /bin/sh
        CLASSPATH=INSTDIR/Jigsaw/classes/jigsaw.zip:.
        export CLASSPATH


Windows
        set CLASSPATH=INSTDIR\Jigsaw\classes\jigsaw.zip;.
```

Don't forget to change *INSTDIR* with the absolute path of the place you have unpacked the distribution. You should now be ready to run **Jigsaw**.

Warning: on some Windows Java implementation, prefixing the CLASSPATH with the disk drive letter may cause some confusion. If java complains about not being able to find some class, you may want to remove the disk drive letter from the CLASSPATH. Note also that if you are adding this line in a batch file, you must escape the '\' the line will be: `set CLASSPATH=INSTDIR\\Jigsaw\\classes\\jigsaw.zip`

Build the right property files

Be careful to have the right CLASSPATH (see above):

```
UNIX
       cd INSTDIR/Jigsaw/Jigsaw
       java Install


Windows
       cd INSTDIR\Jigsaw\Jigsaw
       java Install
```

Running Jigsaw

You are now all set to run **Jigsaw**. Just type in the following command:

```
 UNIX
        cd INSTDIR/Jigsaw/Jigsaw
        java org.w3c.jigsaw.Main -host host -root INSTDIR/Jigsaw/Jigsaw

Windows
        cd INSTDIR\Jigsaw\Jigsaw
        java org.w3c.jigsaw.Main -host host -root INSTDIR\Jigsaw\Jigsaw
```

Don't forget to susbstitute to *INSTDIR* the absolute path of the location where you have unpacked the distribution file, and to *host* the full IP hostname of the machine running **Jigsaw**.

Jigsaw should be running, and will probably have emited a message like:

```
jigsaw Sample/Jigsaw> java org.w3c.jigsaw.Main
loading properties from: /auto/tarantula/u/tarantula/0/w3c/ylafon/Sample/Jigsaw/config/server.props
Jigsaw[2.0beta1]: serving at http://jigsaw.inria.fr:8007/
*** Warning : no logger specified, not logging.
JigAdmin[2.0beta1]: serving at http://jigsaw.inria.fr:8009/
```

Further reading

At this point, it is recommended that you start reading the documentation, available from your server at `/Doc` (i.e. in the above example, the full URL would be `http://www24.w3.org:8001/Doc/`

Here is a roadmap to the documentation:

- You should first start by reading the architecture overview. More on the Jigsaw 2.0 design
- You should then configure your server to meet your need. There are two sources of documentation for this stage:
  - The configuration tutorial
  - The Jigsaw aministration tool **(JigAdmin).**
- At this point, you should have understood the configuration process, you can now browse the rest of the documentation in whatever order you want.

Jigsaw Team

$Id: installation.html,v 1.28 1999/09/09 09:29:27 ylafon Exp $

Jigsaw
Command Line Options

Jigsaw Home / Documentation Overview

Usage

`java org.w3c.jigsaw.Main [`*`options`*`]`

The available options are:

-root *root-directory*

The root directory is the one in which **Jigsaw** will run. By default, this is the directory in which **Jigsaw** is being run. That directory should (if no other command line options are provided) contain at least:

A `config` directory
: Containing the server's configuration

A `logs` directory
: Where **Jigsaw** will emit its log files.

A `WWW` directory
: Containing the document to serve.

-space *space-directory (deprecated)*

The space directory is the directory that contains the actual documents to be served by **Jigsaw**. If this command line option is not provided the space directory defaults to the `WWW` directory under the current root directory.

-config *config-directory*

The config directory is the directory containing **Jigsaw**'s configuration. It is possible for two different servers to share the same space directory, but only one server can be run within a given configuration.

This directory defaults to the `config` directory under the root directory.

-host *hostname*

Fully qualified host name of the host running **Jigsaw**. Java1.0.2 doesn't provide the necessary support to discover that information, and it might also be usefull if your machine supports several names (ie DNS aliases).

By default the hostname used by the server is the result of that expression:

```
InetAddress.getLocalHost().getHostName();
```

-port *port (deprecated)*

The TCP port number **Jigsaw** should be listening at. By default **Jigsaw** listen on port 8001. On UNIX, you may want to try the supported native code to run **Jigsaw** on port 80 (check the appropriate FAQ entry).

-trace

Enables tracing of all requests/replies. This command line option does *not* take precedence over the current value of the `w3c.jigsaw.trace` property, as defined in the `config/http-server.props` file.

By default this toggle is set of **false**.

-noupgrade

Prevent any automatic upgrade of **Jigsaw**'s configuration. By default **Jigsaw** will automatically upgrade the configuration if needed, if this flag is set, **Jigsaw** will just emit an error, stating that your configuration needs to be upgraded.

-maxstores *number*

Max number of stores in memory.

-help

Display online help for all above options.

Jigsaw Team

$Id: cmdline.html,v 1.8 1999/07/30 08:36:24 bmahe Exp $

Jigsaw
Basic concepts

Jigsaw Home / Documentation Overview

This page has not been updated since release 1.0alpha3. Even though the ideas haven't changed that much, some class have been renamed as part of an API cleanup (check the Release Notes). A new architecture overview is under construction here.

Jigsaw Architecture

Jigsaw is made of two distinct modules, linked through a set of interfaces:

- The daemon module deals with the HTTP protocol: it handles new incomming connections, create new client objects, decode requests, and send replies.
- The resource module is some representation of your information space. It is responsible for generating reply objects out of the incoming request objects.

As these two modules are linked through a set of Java interface specifications, you can replace each of them independantly of the other, provided they implement adequatly the interfaces.

The daemon module

As a server administrator, you probably won't have to deal much with this part of **Jigsaw**, although it might be a good idea to read this section (or at least the one on terminology), just to get a filling of what's happening behind the scene. This section will goes through a bit of terminology, it will then step through the life-time of a connection, and introduce the resource module.

Terminology

The protocol module deals with a number of objects. To get you into this world, we will start by describing the most important.

httpd

> This is the object whose `main` method will actually run the server. It has two purposes: the first one is to run the *accepting thread*, i.e. the thread that will loop waiting for new incomming

connections to come by. The second purpose of this object is to manage the set of other objects, responsible for handling part of the server behavior. Among them, there is the logger (responsible for loggin requests), the authentication realm manager (responsible for the list of authentication realms defined attached to the server), the client pool (responsible for handling accepted connections), the root resource of the server, and last but not least, the resource store manager. We will describe all these objects more precisely in the comming sections.

logger

Each time a request processing terminates (be it with success or not), the server will call back the logger so that it can keep track of all handled requests. The current version of **Jigsaw** comes with a simple logger, compliant with the Common Log file format (ie it will emit a one line record for each processed request).

realm manager

The realm manager keeps track of all the authentication realms defined by the server. Each created authentication realm is assigned a symbolic name, that the web admin will use to refer to it when configuring the server. This name will also be used as the HTTP *realm name*, so it should be uniq within the server scope. The sample implementation of this object manages a persistent catalog of realms, that can be edited through a special tool called JigAdmin (see **Jigsaw** configuration manual).

client pool

The client pool object is responsible for handling new incomming connections. It should make its best effort to guess what protocol the other end wants to speak on this connection, and create an appropriate client object, to handle it. The current sample implementation will always assume that new connections are for speaking the HTTP/1.0 protocol (with the addition of persistent connections). The other role of the client pool is to optimize as much as possible thread creations. Thread creation can be a costly process, wo its worth trying to avoid it as much as possible. The sample implementation will maintain a ready-to-run set of client objects, so that it won't re-create them from scratch upon each new connections.

root resource

The root resource is *the* object that will link the protocol module to the resource module. This object should implement the appropriate interface (right now, it should be an instance of the ContainerResource, but this is likely to change in the very near future).

resource store manager

As you will see in the next section, **Jigsaw** serves each file or directory by wrapping it into some Resource instance. As their number might become fearly large, the server will keep track of the one that haven't been accessed for a while, and unload them from memory. The resource store manager is responsible for this piece of the server behavior: it keeps track of all the loaded resources, and unload them when it thinks appropriate.

Given these definitions, we can now explain how the server handles new incomming connections.

Life-time of a connection

The life time of a connection can be divided into the following steps:

1. The accepting thread is notified of it.
2. It gets handled by the client pool object
3. A thread starts waiting for incoming requests
4. The request is handed out to the resource module, for actual processing
5. The resource module generated reply is emited
6. The server logger is called back to log the request

The first stage in processing a new connection, is to hand it out as quickly as possible to the client pool (so that the accepting thread can return as fast as possible to the `accept` system call). The client pool then look for an idle Client object, if one is found, it is `bind` to the accepted connection, which makes it run its main loop. If no client is available, if we have reached the maximum allowed number of connections the new connection gets rejected (by closing it), otherwise a fresh client is created and bound to the connection.

By the end of stage 2, the client pool has either rejected the connection, got a new client to handle it, or created a fresh client for this connection. At stage 3, the client object is bound to the connection, and awaken to actually process it. The client thread enters the client main loop.

The client main loop starts by getting any available request. When such a request has been read from the network, it is handed out to the resource module. This latest module is responsible for generating an appropriate Reply object, which is then emited (stage 5) by the client thread, back to the browser. Finally the server logger is invoked with the request, its reply and the number of bytes sent back to the browser.

At the end of this request processing, the client object tests to see if it can keep the connection alive. If so, it loops back to stage 2, otherwise, the client notify the client pool that it has become idle. The client pool, in turn, decides if this client object should be spared for future use or not.

The Resource module

The resource module is the one that manages the information space. In **Jigsaw** each exported object is mapped to an HTTPResource instance, which is created at configuration time, either manually or by the resource factory. We will describe here what are resources and then sketch the way **Jigsaw** looks up the specific target resource of a request. We finally present the *filter* concept.

Resources

Resources are full Java objects, defined by two characteristics:

- Their Java class define their *behavior* (how they implement the HTTP methods)
- Their *state* is described through a set of attributes

The AttributeRegistry keeps track of all the attributes of each classes. As instance variables, attributes are

inherited along the normal sub-class relationship. Each resource attribute is described by some instance of the Attribute class (or some of its sub-class). This description is made of

- the name of the attribute,
- some flags indicating if the attribute is mandatory, editable, computed or whatever
- the methods to pickle (dump) and unpickle (restore) this attribute values
- a method to check if some Object instance can be used as a value for this attribute.

Given this description, **Jigsaw** is able to make resources persistent, just by dumping their class-name, and all their attribute values. Unpickling (i.e. restoring) a resource is just creating an empty instance of its class, and filling its attributes with their saved values.

Resource instances are the basic resources. The HTTPResource class is the basic class for resources that are accessible through the HTTP protocol. Instances of this class define a number of attributes along with the method that implements the HTTP methods (e.g. GET, POST, PUT, etc. which are mapped resp. to the get, post put, etc methods of the class). These methods are all trigered through the perform method of the class, whose role is to *dispatch* a request to the appropriate handler.

Remember in the previous section, we said that request were handed out the the resource module. The `perform` method of HTTPResource are the one that get called by the daemon module once the target resource of the request has been looked up. The next section explains how this lookup is performed.

Looking up resources

In the paragraph about the life-time of a connection, we mention that at stage 4 the parsed request was handed out to the resource module. The first thing the resource module does when it receives a request, is to look up the target resource. This paragraph explains briefly how this is performed.

Jigsaw defines a special subclass of the HTTPResource class, the ContainerResource, whose role is to implement the look up strategy for the sub-space it controls. All servers (i.e. all instances of the httpd class) keeps a pointer to their *root resource*. This root resource **must** be a container resource: it must implement the lookup method. Given a request, this method must return a suitable target resource for processing it. However, there is no constraints on *how* the lookup is performed. We will briefly sketch how directory resources implement their lookup method.

The directory resource's lookup method starts by checking that it has an up-to-date list of children. What is meant by up-to-date here might not be what you expect: **Jigsaw** caching strategy can make this notion quiet complex. Anyway, once the directory resource thinks its list of children is up-to-date, it looks up the first component of the URL in its children set. For example, if the URL is `/foo/bar`, it starts by looking up `foo` in itself. This can lead to three cases, depending on the result of this:

- The directory resource doesn't have a child resource named `foo`. In this case, it throws an exception to signal a *not found* error.
- The directory resource has a child named `foo`. As the looked up URL contains more components, the directory resource check that the found resource is a container resource. If this is not the case, then a *not found* error is signaled by throwing an appropriate exception. Otherwise, the looked up component is removed from the look up state, and the directory resource calls up the found child

resource's lookup method.

This look up process is just one example of how the look-up operation can be implemented. It has several advantages in the specific case of handling directory resources, but other situations may require other algorithms. One important property of the directory resource's lookup algorithm is that it is able to *delegate sub-space* naming to the resource that actually handles the sub-space.

Resource filters

We have briefly described **Jigsaw** resource module. The last thing you need to understand is **Jigsaw**'s concept of resource *filters*. You might have been surprised that until now, we haven't said a word on authentication. In **Jigsaw** authentication is implemented as a special resource filter. Resource filters are a special kind of resource (i.e. they are persistent, and can define any kind of attributes), that are attached to some *target resource*. Filter instances are called back twice during request processing:

- At lookup stage, the filter's `ingoingFilter` method is called. It is provided with the request whose URL we are looking up.
- After the target resource has generated its reply, the filter's `outgoingFilter` method gets (optionally - see below) called, with both the original request and the reply has parameters.

For a resource to support filters, its class must be a subclass of the FilteredResource class. Most resource classes provided with **Jigsaw** distribution are sub-classes of it.

Back to authentication now. As we said above, authentication is handled by a special filter, whose ingoingFilter method tries to authenticate the request. If this succeeds, normal processing of the request continues: it is performed by its target resource, and the corresponding reply is emited back to the browser. In the case of the authentication filter, as all the work is done only in the ingoing way (while the target resource is being looked up), there is no need to have the outgoingFilter method called. A filter `ingoingFilter` method can return a special value DontCallOutgoing to indicate that it has performed all its job, in such cases, the server won't spend time invoking its empty `outgoingFilter` method. More return codes are available, see the api documentation for the ResourceFilter to get into the details.

Further reading

The best way to continue your **Jigsaw** tour now, is to install it, and to read the following tutorials:

- **Jigsaw** administration guide is the reference guide for **Jigsaw** configuration.
- The configuration tutorials will go through the configuration process, explaining you the basics of **Jigsaw** configuration.
- Once you are familiar with the configuration process, you might want to know how to define new frame classes. The frames tutorial walk through a full example of this.
- You might want to know also how to define a new resource classe. The resources tutorial walk through a full example of this.
- Finally, you may want to learn more about resource filters, by reading the filter tutorial, which explains how to write new resource filters.

Jigsaw Team

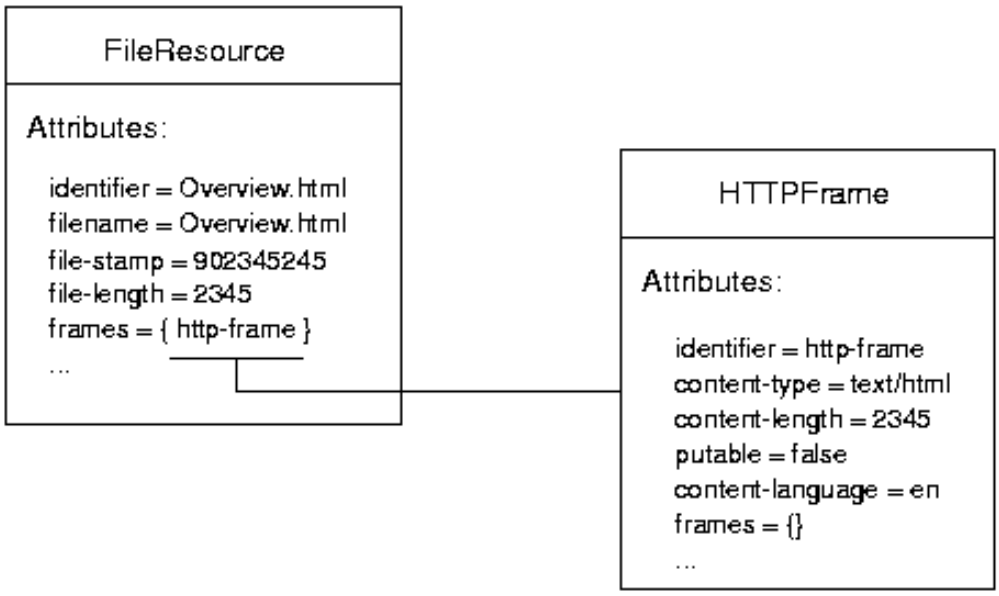
$Id: architecture.html,v 1.15 1999/09/08 14:58:11 ylafon Exp $

Jigsaw
Internal design of Jigsaw 2.0

Jigsaw Home / Documentation Overview

In **Jigsaw** each exported object is mapped to a FramedResource (FileResource, DirectoryResource...) associated to a ProtocolFrame (usually the HTTPFrame or any subclass) instance, which is created manually (with JigAdmin 1.0 or 2.0) or by an indexer.

This document has the following sections:



What is a Resource?

A resource is a full Java object, containing only information that the raw Resource (a file, a directory...) can provide (eg, for a file, the size, last modification date...)

A resource **MUST** be associated to a ProtocolFrame to be available on the server.

The list of all Resources is available.

What is a Frame?

A Frame is a full Java Object, containing all the information needed to serve this Resource using a specific Protocol (eg, HTTPFrame for **HTTP**).

The list of all Frames is available.

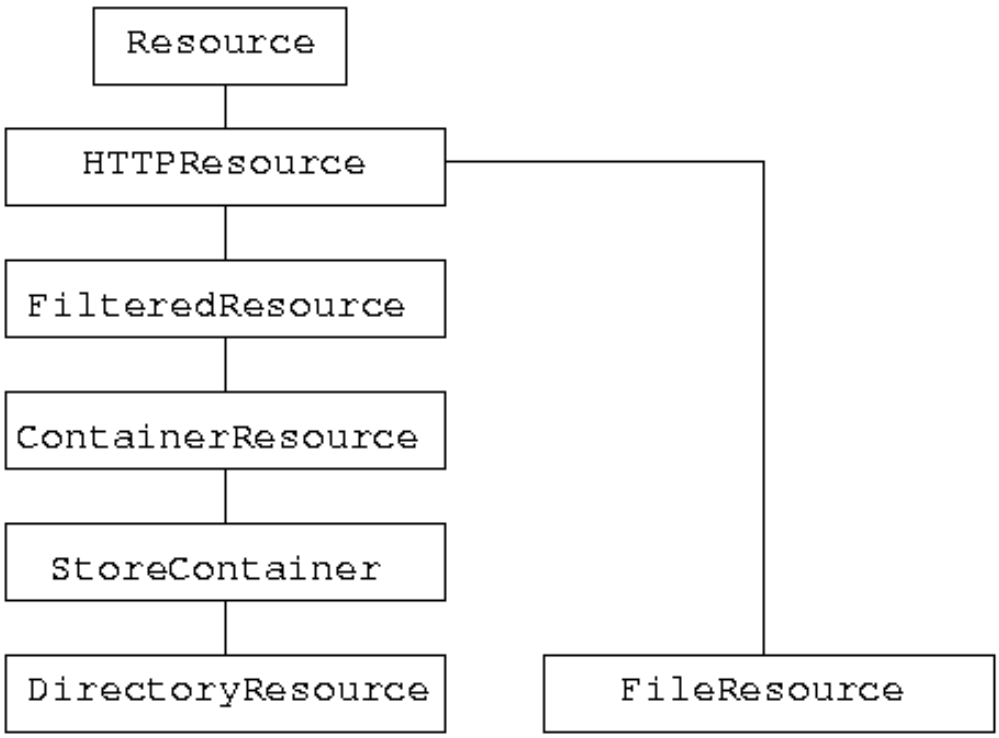
What is a Filter?

A Filter is a full Java Object, associated to a Frame, that can modify the Request and/or the Reply. For example the Authentication is handled by a special filter, the GenericAuthFilter (Note, as this filter is protocol dependant, it is placed on the protocol frame).

The list of all Filters is available.

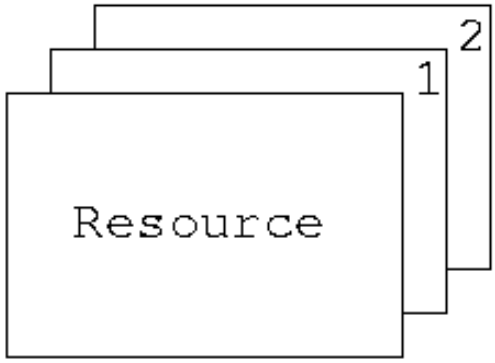
The new Resource

In the previous version of **Jigsaw** (1.0), the inheritance tree of Resources was:



All the basic Resources, such as FileResource and DirectoryResource, were heavily linked to HTTP as all the resources served

were extensions protocols that are not HTTP-related, we propose this new version of the Resource:



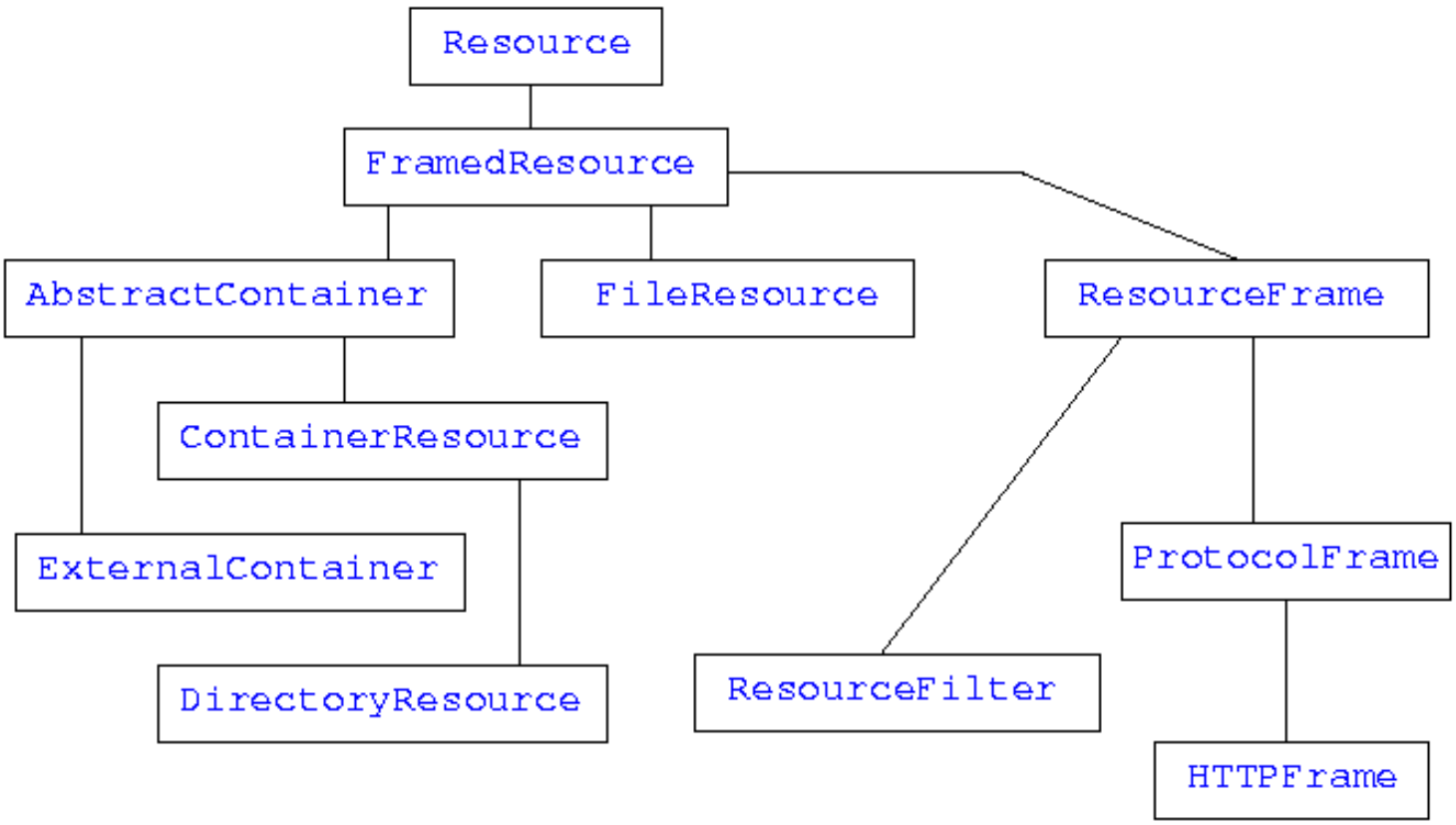
Where (1) and (2) are ResourceFrames. A Resource is now a very basic thing, containing only information that the raw Resource can provide (e.g., for a file, the size, last modification date, creation date if available, etc.), then, attached to that Resource, we find the ResourceFrames that extend Resource (they are handled the same way) and contain information about the Resource they are attached to.

To serve a resource using a protocol -- for instance, HTTP -- the Resource will have a protocol ResourceFrame, HTTPFrame, that contains all the information needed to serve this Resource using HTTP. This frame is like the old version of HTTPResource, but it contains more information than the previous version.

The filters are now divided in two categories: the filters on the Resource and the filters on the protocol Frames. The usual filtering scheme used in the previous version of **Jigsaw** is still valid. The main difference is that filters are no longer attached to the Resource itself but to its protocol frame. ResourceFrames can also have frames.

Other kind of frames can be attached, like RDF frame for metadata, PICS frame to rate this resource, etc...

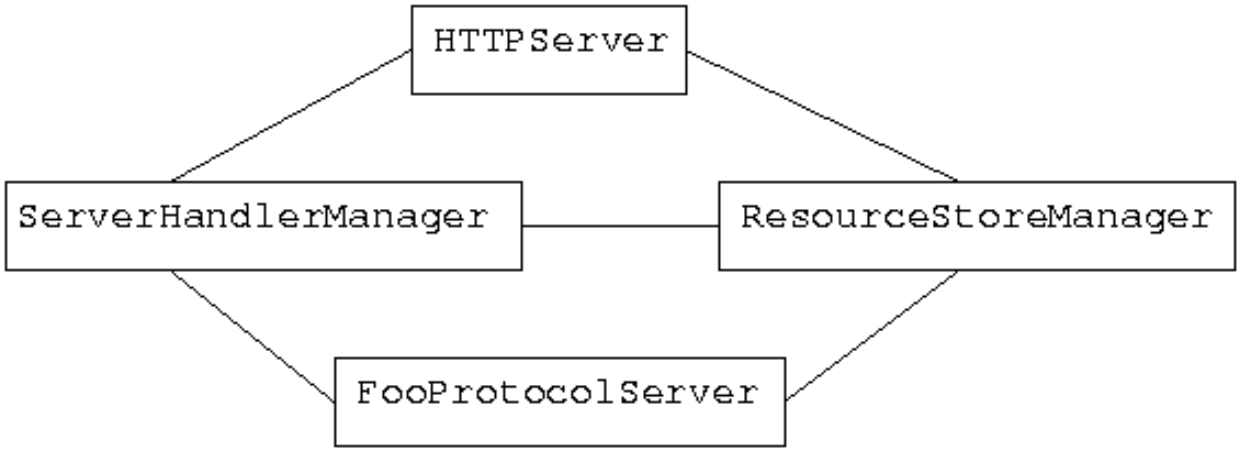
The new inheritance tree is:



more complex, but more flexible than the previous version.

The new ResourceStoreManager

In order to share all the Resources amongst different servers efficiently, we created a new central ResourceStoreManager. In the previous version the Resources were handled by other Resources. For example, the FileResource was handled by its DirectoryResource. This induced a number of bugs and was not very well-adapted to the new way of sharing Resources. There is now only one manager for the server handler so that each server will talk to this sole manager.



This RSM contains a hashtable associating a key (unique indentifier of a ResourceContainer) and a StoreEntry. The StoreEntry contains the store of the resource sons and a hashtable associating the identifier of the sons of the resource and the ResourceReference of those resources.

The ResourceReference is used like this:

```
ResourceReference rr;
....
try {
   Resource res = rr.lock();
   ....
} catch (InvalidResourceException ex) {
  /* InvalidResource means that the resource has been deleted */
   ....
} finally {
  rr.unlock();
}
...
```

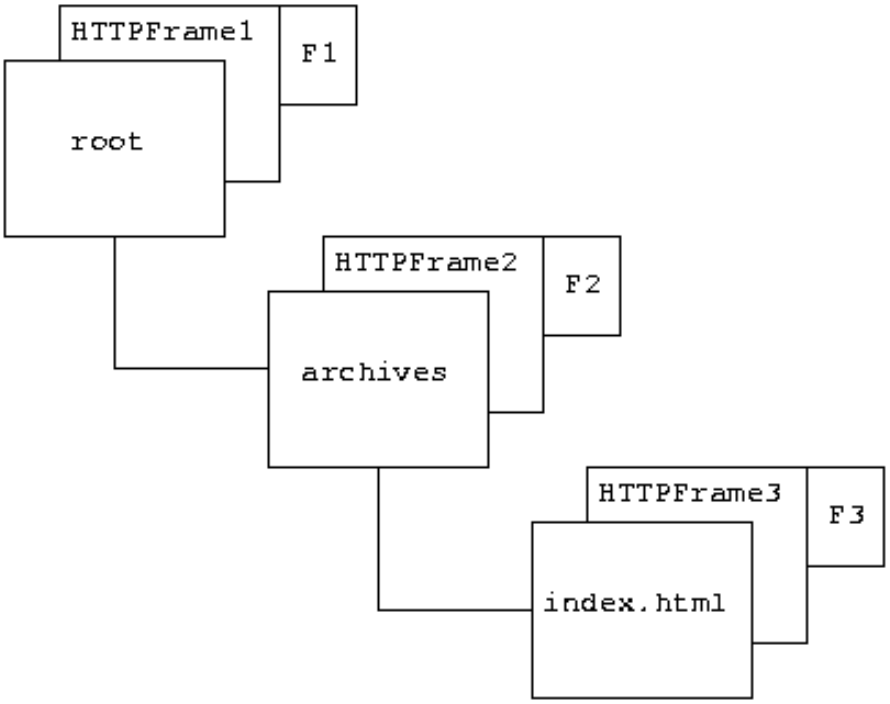
If the resource has been garbage-collected, the rr.lock() will load it again, and during the lock, it is guaranteed that the resource won't be deleted, unloaded or modified by someone else. This allows safe concurrent modification access to this resource.

Now the container is no longer responsible for the management of its son; it only has a key to the StoreEntry, which contains its sons. To get its own store, the resource has to ask its parent for the StoreEntry that contains it.

The lookup and perform algorithm

This part describe the lookup and the perform algorithm used by **Jigsaw**.

The following picture show a **Jigsaw** resources tree (relative to the URL `/archives/index.html`), where `root` and `archives` are DirectoryResource (`root` is the root resource) and `index.html` is a FileResource. `F1`, `F2` and `F3` are filters (ResourceFilter subclass instance).

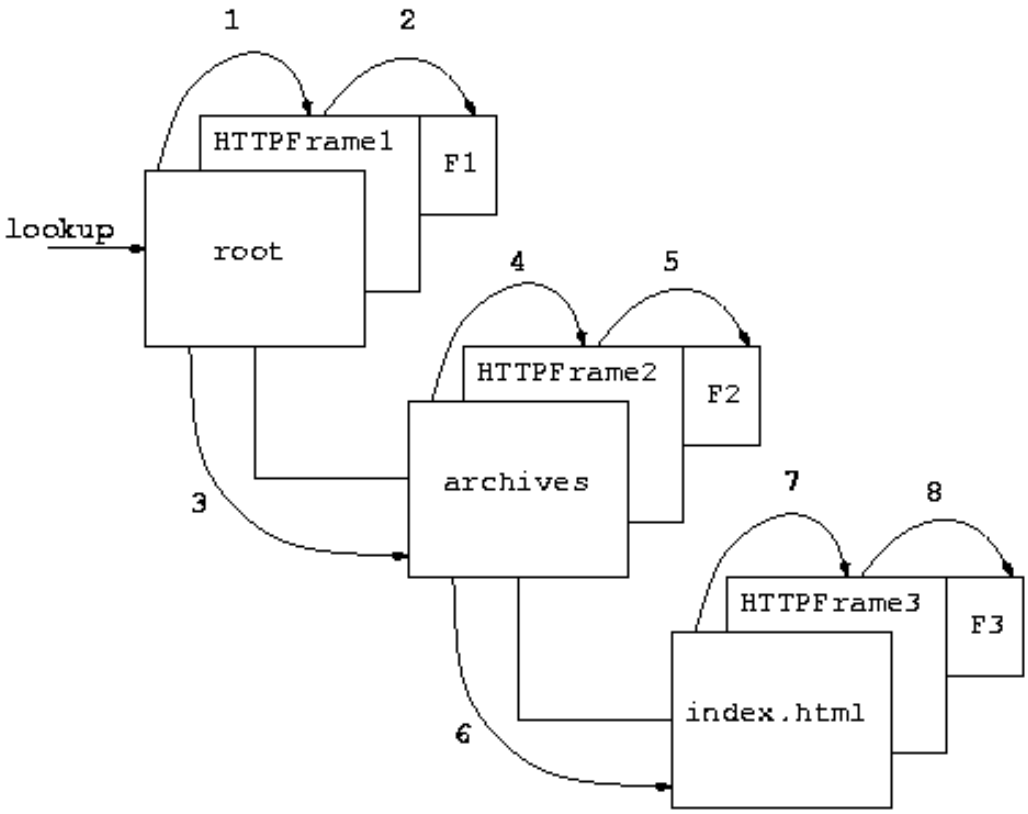


In the following description, **Jigsaw** receive an HTTP GET request for `/archives/index.html`. To handle the incomming request, **Jigsaw** will go through the following steps:

1. Lookup for /archives/index.html
2. Call the ingoingFilter method of filters
3. Perform the request
4. Call the outgoingFilter method of filters
5. Emit the reply

1) Lookup for `/archives/index.html`. The LookupState (ls) keeps the state info, and the LookupResult (lr) is the result of the lookup algorithm.

```
root lookup(ls,lr)
    -> HTTPFrame1 lookup(ls,lr)
        -> F1 lookup(ls,lr)
        -> HTTPFrame1 lookupDirectory(ls,lr)
    -> archives lookup(ls,lr)
        -> HTTPFrame2 lookup(ls,lr)
            -> F2 lookup(ls,lr)
            -> HTTPFrame2 lookupDirectory(ls,lr)
        -> index.html lookup(ls,lr)
            -> HTTPFrame3 lookup(ls,lr)
                -> F3 lookup(ls,lr)
                -> HTTPFrame3 lookupFile(ls,lr) => index.html
```



2) Call the ingoingFilter method of filters. Request is the incomming request.

```
F1 ingoingFilter(Request)
F2 ingoingFilter(Request)
F3 ingoingFilter(Request)
```

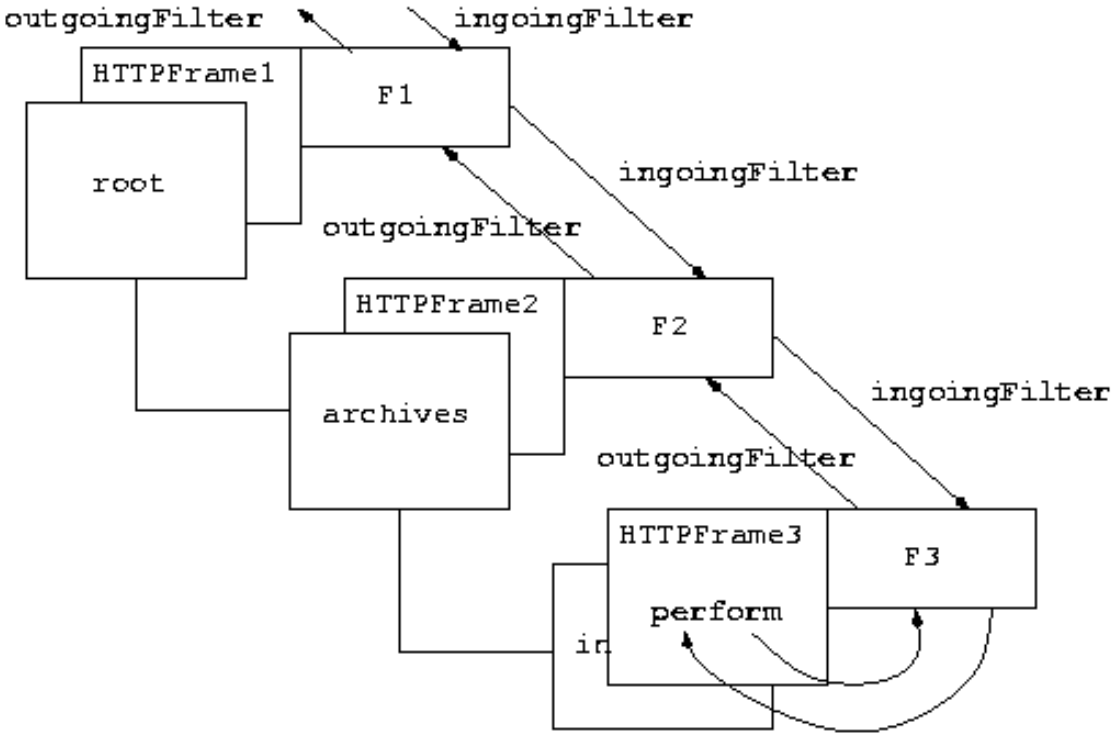
Note that if any filter answer with a non-null Reply, the process is stopped and the Reply is sent back to the client directly (like in the GenericAuthFilter)

3) Perform the request (GET) on the resource found at lookup time.

```
index.html perform(Request)
    -> HTTPFrame3 perform(Request)
        -> HTTPFrame3 get(Request)
            -> HTTPFrame3 getFileResource(Request) => Reply
```

4) Call the outgoingFilter method of filters. Request is the incomming request, Reply is the reply created by `HTTPFrame3`.

```
F3 outgoingFilter(Request, Reply)
F2 outgoingFilter(Request, Reply)
F1 outgoingFilter(Request, Reply)
```



5) Emit the reply created by `HTTPFrame3`.

Jigsaw Team

$Id: design.html,v 1.28 1999/07/30 13:42:40 bmahe Exp $

All resources All frames

FramedResource

The most simple resource that can be associated with a ResourceFrame.

Inherits

The FramedResource class inerits from the following classes:

- Resource

Attributes description

The FramedResource doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.tools.resources.FramedResource.html,v 1.3 1998/03/27 08:25:36 bmahe Exp $

Jigsaw
Resources Index

Jigsaw Home / Documentation Overview / Frames - Indexers - Client Side Components

This index briefly describes what resources are currently available in the **Jigsaw** release. See the indexer documentation to understand how these resources are created by default.

To be more readable, the list of available resources has been split into four groups:

Standard resources

Resources that you expect any server to support.

Extension resources

Basic resources, or resources that implement extra protocols.

Admin resources

Resources that allow you to administer Jigsaw.

Property resources

Resources that wraps a sub-set of Jigsaw properties in order to make themn editable.

Standard resources

org.w3c.tools.resources.FramedResource

This is the most basic resource, only used to attach frame.

org.w3c.tools.resources.FileResource

The FileResource handles files within a DirectoryResource.

org.w3c.tools.resources.ContainerResource

The ContainerResource is the most basic container fo resources.

org.w3c.tools.resources.DirectoryResource

The DirectoryResource handles file system directories.

org.w3c.jigsaw.resources.DirectoryResource

This DirectoryResource provides content negotiation

org.w3c.tools.resources.PassDirectory

A directory resource that emulates the CERN-server PASS rule.

org.w3c.jigsaw.resources.PassDirectory

This PassDirectory provides content negotiation.

org.w3c.jigsaw.resources.HttpDirectoryResource

This is a DirectoryResource which has a HTTPFrame by default.

org.w3c.jigsaw.resources.HttpPassDirectory

This is a PassDirectory which has a HTTPFrame by default.

org.w3c.jigsaw.resources.HttpFileResource

This is a FileResource which has a HTTPFrame by default.

org.w3c.jigsaw.resources.VirtualHostResource

This is just a base for VirtualHostFrame.

org.w3c.jigsaw.map.MapResource

A resource that handles both CERN-server style and NCSA style image maps.

Extension resources

org.w3c.jigsaw.resources.DirectoryLister

A resource which display the directory listing on its parent (DirectoryResource).

org.w3c.jigsaw.servlet.ServletWrapper

Implementation of Sun's servlet API, for local classes

org.w3c.jigsaw.servlet.RemoteServletWrapper

A servlet wrapper for remote servlet

org.w3c.jigsaw.filters.PutListResource

This is the PutListFrame associated resource.

org.w3c.jigsaw.cvs.AutoLookupDirectory

A resource that can fetch or update files directly from CVS.

org.w3c.jigsaw.cvs.ToolsLister

A resource that allow web site users to delete some files under a DirectoryResource.

org.w3c.jigsaw.pics.LabelBureauResource

A Jigsaw resource to query a Label bureau.

org.w3c.jigsaw.zip.ZipDirectoryResource

Allows you to browse a zip file.

org.w3c.jigsaw.zip.ZipFileResource

Allows you to browse a zip file.

Admin resources

org.w3c.jigsaw.resources.CheckpointResource

A resource that will periodically make sure that **Jigsaw** configuration has been saved back to disk.

org.w3c.jigsaw.resources.PasswordEditor

A resource that allow web site users to change their own password.

Property resources

org.w3c.jigsaw.http.UnixProp

UNIX specific properties editor. Allows you to chroot **Jigsaw** and run it on port 80.

org.w3c.jigsaw.http.ConnectionProp

Connection properties.

org.w3c.jigsaw.http.socket.SocketConnectionProp

Connection properties specific to the raw socket client and client factory.

org.w3c.jigsaw.http.LoggingProp

Gives editable access to **Jigsaw** logging properties.

org.w3c.jigsaw.http.GeneralProp

Gives editable access to **Jigsaw** general properties.

org.w3c.jigsaw.proxy.ProxyProp

Gives editable access to **Jigsaw** client-side specific properties to customize it when used by the proxy. This will only appear when you use the ProxyFrame.

org.w3c.jigsaw.proxy.CacheProp

Gives editable access to **Jigsaw** client-side cache, when the proxy resource is used.

org.w3c.jigsaw.cvs.CvsProp

Gives editable access to **Jigsaw** cvs properties, when CvsFrame is used somewhere.

org.w3c.jigsaw.servlet.ServletProps

This resource provides access to Jigsaw's servlet properties.

org.w3c.jigsaw.pagecompile.PageCompileProp

Gives editable access to Jigsaw Page Compilation properties when PageCompileFrame is used somewhere.

Jigsaw Team

$Id: resources.html,v 1.16 1999/03/30 09:39:22 bmahe Exp $

Jigsaw
Frames Index

Jigsaw Home / Documentation Overview / Resources - Indexers - Client Side Components

This index briefly describes what frames are currently available in the **Jigsaw** release.

To be more readable, the list of available frames has been split into six groups:

Standard frames
: Frames that you expect server to support.

Extension frames
: Frames that implements extra features.

Filter frames
: Frames that allow you to filter the requests and replies (authentication, log).

Proxy frames
: Frames relative to the proxy features of **Jigsaw**.

Admin frames
: Frames that allow you to administer **Jigsaw**.

Specific frames
: Frames that can be attached only to specific resources.

Metadata frames
: Frames used to store metadata.

Standard frames

org.w3c.jigsaw.frames.HTTPFrame
: The basic frame for all HTTP accessible resources.

org.w3c.jigsaw.frames.CgiFrame
: A frame that allows you to run CGI/1.1 compliant scripts. This is of course not the recommended way of extending Jigsaw.

org.w3c.jigsaw.frames.NegotiatedFrame

A frame that will handle negotiation among a given set of variant resources.

org.w3c.jigsaw.frames.PostableFrame

The basic frame class for handling the HTTP POST method.

org.w3c.jigsaw.frames.RedirecterFrame

A frame that handle internal redirection.

org.w3c.jigsaw.frames.RelocateFrame

A frame that handle HTTP redirection.

org.w3c.jigsaw.frames.SeeOtherFrame

generates a 303 See Other reply on a POST

org.w3c.jigsaw.frames.VirtualHostFrame

A top level frame that will handle virtual hosts without consuming IP addresses !

Extension frames

org.w3c.jigsaw.servlet.ServletDirectoryFrame

A context and container for servlets. All servlets should be made children of a DirectoryResource attached to a ServletDirectoryFrame instance.

org.w3c.jigsaw.servlet.ServletWrapperFrame

The specific frame of ServletWrapper.

org.w3c.jigsaw.ssi.SSIFrame

A frame that will serve file and run any server side include command.

org.w3c.jigsaw.pagecompile.PageCompileFrame

PageCompileFrame allows you to generate HTML pages from HTML/Java pages.

org.w3c.jigsaw.cvs.CvsFileFrame

A frames that gives you CVS access to the server's files through CVS. (with auto commit when modified)

org.w3c.jigsaw.cvs.CvsFrame

A frame that gives you CVS access to the server's files through CVS.

org.w3c.jigsaw.filters.PutListFrame

The specific frame of PutListResource. Manages a list of last puted documents and allows users to publish them.

org.w3c.jigsaw.pics.LabelBureauFrame

The HTTP interface of the LabelBureauResource.

org.w3c.jigsaw.zip.ZipFrame

Allows you to browse a zip file.

Filter frames

org.w3c.jigsaw.auth.GenericAuthFilter

This filter provides several ways of protecting part of your information space.

org.w3c.jigsaw.filters.AccessLimitFilter

Limit the number of simultaneous accesses to a resource.

org.w3c.jigsaw.filters.CacheFilter

A cache filter.

org.w3c.jigsaw.filters.CookieFilter

A demo for how to use cookies from Jigsaw.

org.w3c.jigsaw.filters.CounterFilter

Count the number of traversals or hits of its target.

org.w3c.jigsaw.filters.DebugFilter

Print incoming request and outgoing replies.

org.w3c.jigsaw.filters.ErrorFilter

The error filter allows you to redefine on the fly all error messages emitted by **Jigsaw** by using internal redirections: all errors are then emitted by some other resource (which can be any of the Jigsaw supported resources).

org.w3c.jigsaw.filters.GZIPFilter

This filter will compress "on the fly" the content of replies using GZIP.

org.w3c.jigsaw.filters.HeaderFilter

Enforces a specific header value on all replies.

org.w3c.jigsaw.filters.LogFilter

The log filter allows you to get very detailed logging of transactions for a particular sub-space of your web server.

org.w3c.jigsaw.filters.ProcessFilter

A filter that will process a reply's content through any external filter program.

org.w3c.jigsaw.filters.PutFilter

This filter update the PutListResource.

org.w3c.jigsaw.filters.GrepPutFilter

This PutFilter allows you to control the content of puted documents.

org.w3c.jigsaw.filters.PutSizeFilter

This filter allows you to limit the size of puted documents.

org.w3c.jigsaw.filters.SimpleCacheFilter

A simple cache filter.

org.w3c.jigsaw.filters.HourLimiterFilter

Allows to access some resources in a specific period of time.

org.w3c.jigsaw.filters.URISizeLimiterFilter

Limit the size of URI.

org.w3c.jigsaw.filters.UseProxyFilter

Restrict access to a proxy, to acces the protected resource, you must go to a specific proxy.

org.w3c.jigsaw.filters.TEFilter

This filter will compress the content of replies using GZIP.

org.w3c.jigsaw.acl.AclFilter

An authentication filter, must be associated to a Metadata frames implementing the Acl interface.

Proxy frames

org.w3c.jigsaw.proxy.ProxyFrame
A frame that will turn Jigsaw into a fully HTTP/1.1 compliant proxy.
org.w3c.jigsaw.proxy.MirrorFrame
A frame to mirror other web sites.

Admin frames

org.w3c.jigsaw.status.GcStatFrame

This frame implements a GC counter.

org.w3c.jigsaw.status.StatisticsFrame

A frame that displays internal server statistics (such as number of hits, etc).

org.w3c.jigsaw.status.ThreadStatFrame

A frame that will display the status of the threads running in the server process, and that will refresh it at a given interval of time.

Specific frames

org.w3c.jigsaw.resources.DirectoryListerFrame

Specific frame of DirectoryLister.

org.w3c.jigsaw.resources.PasswordEditorFrame

Specific frame of PasswordEditor.

org.w3c.jigsaw.cvs.ToolsListerFrame

Specific frame of ToolsLister.

org.w3c.jigsaw.map.MapFrame

Specific frame of MapResource.

Metadata frames

org.w3c.jigsaw.acl.AclRealm

Used to store the old realms in an ACL compliant way.

Jigsaw Team

$Id: frames.html,v 1.22 1999/03/30 09:39:01 bmahe Exp $

All resources All frames

ResourceFrame

The most basic frame class.

Inherits

The ResourceFrame class inerits from the following classes:

- Resource
- FramedResource

Attributes description

The ResourceFrame doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.tools.resources.ResourceFrame.html,v 1.3 1998/03/27 08:26:32 bmahe Exp $

All resources All frames

Resource

The basic class for all resources. This base class provides support for persistency.

Inherits

The Resource is the most simple resource in **Jigsaw**.

Attributes description

The Resource defines the following attributes:

- identifier
- last-modified

`identifier`

semantics

The name of that resource. This name is the key of the resource in its resource store. You can change it, the resource store will take the appropriate actions.

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

`last-modified`

semantics

Last modified time.

type

This attribute is an editable DateAttribute

default value

This attribute defaults to **null**.

Jigsaw Team
$Id: org.w3c.tools.resources.Resource.html,v 1.3 1998/03/27 08:26:10 bmahe Exp $

All resources All frames

FileResource

The basic resource to serve files. This resource allows you to export files.

Inherits

The FileResource class inerits from the following classes:

- Resource
- FramedResource

Attributes description

The FileResource defines the following attributes:

- filename
- backup

`filename`

semantics

The optional name of the file to be served by the file resource. By default, the file resource will serve the file's having the resource's name. You can define this attribute in order to change the URL to file mapping. E.g. you can serve the file foo.html through the name oof.html, by setting the foo.html filename attribute to off.html.

type

This attribute is an editable FilenameAttribute

default value

This attribute defaults to **null**.

`backup`

semantics

Should this resource create a backup file if the file content change (with the newContent method).

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **false**.

Jigsaw Team

$Id: org.w3c.tools.resources.FileResource.html,v 1.3 1998/03/27 08:25:25 bmahe Exp $

All resources All frames

DirectoryResource

The directory resource is the basic resource to export file-system directories. It keeps track of all its children resources, create them dynamically if needed. This class should be used as the basic class to export file system directories.

Inherits

The DirectoryResource class inerits from the following classes:

- Resource
- FramedResource
- AbstractContainer
- ContainerResource

Attributes description

The DirectoryResource defines the following attributes:

- indexer
- extensible

`indexer`

semantics

The name of this DirectoryResource indexer. The indexer is an object that given some global configuration informations, tries to build default resources for files that the server doesn't know about.

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

`extensible`

semantics

Should this directory automatically keep in sync with the underlying physical directory ? The directory resource maintains a cache of its list of children, which may be outdated if the directory is changed through direct file system access. When this flag is turned to true, the directory resource will make its best effort to keep in sync with it, by adopting the following lookup algorithm: first look up children in our cache list, if this fails, check if some appropriate file exists. If such a file exists, hand it to the indexer and install the resulting resource (if any) as a new child of the directory resource.

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **true**.

Jigsaw Team

$Id: org.w3c.tools.resources.DirectoryResource.html,v 1.3 1998/03/27 08:25:16 bmahe Exp $

All resources All frames

AbstractContainer

An Abstract class for resources containers (see ContainerResource).

Inherits

The AbstractContainer class inerits from the following classes:

- Resource
- FramedResource

Attributes description

The AbstractContainer doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.tools.resources.AbstractContainer.html,v 1.3 1998/03/27 08:24:55 bmahe Exp $

All resources All frames

ContainerResource

The most simple resource container.

Inherits

The ContainerResource class inerits from the following classes:

- Resource
- FramedResource
- AbstractContainer

Attributes description

The ContainerResource doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.tools.resources.ContainerResource.html,v 1.3 1998/03/27 08:25:04 bmahe Exp $

All resources All frames

ProtocolFrame

An empty class.

Inherits

The ProtocolFrame class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame

Attributes description

The ProtocolFrame doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.tools.resources.ProtocolFrame.html,v 1.3 1998/03/27 08:25:58 bmahe Exp $

All resources All frames

HTTPFrame

The basic frame class of all resources accessible through the HTTP protocol.

Inherits

The HTTPFrame class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ProtocolFrame

Attributes description

The HTTPFrame defines the following attributes:

- quality
- title
- content-language
- content-encoding
- content-type
- icon
- maxage
- send-md5
- putable
- relocate
- index
- icondir
- browsable

- style-sheet-link

`quality`

semantics

A rating of the quality of this resource's content. The rating is provided as a number between 0.0 and 1.0. It is used mainly by the NegotiatedFrame to negotiate among its set of variants.

type

This attribute is an editable DoubleAttribute

default value

This attribute defaults to **1.0**.

`title`

semantics

The title of this resource. This attribute can be either computed from the resource content (e.g. if the content is an HTML file which has some META tag), or provided for informational purposes (even if the resource's content type is not text/html.)

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

`content-language`

semantics

The natural language of the resource content. This is used mainly by the NegotiatedFrame to negotiate among its set of variant resources. The value of this attribute can be either extracted from the resource content (e.g. if it is an HTML file that includes some appropriate META tag), or provided for informational purposes.

type

This attribute is an editable LanguageAttribute

default value

This attribute defaults to **null**.

`content-encoding`

semantics

The encoding in which the resource's content is stored. Right now this can only be a single token (as described in the HTTP/1.1 specification).

type

This attribute is an editable EncodingAttribute

default value

This attribute defaults to **null**.

`content-type`

semantics

The (MIME) type of the resource's content.

type

This attribute is an editable MimeTypeAttribute

default value

This attribute defaults to **text/plain**.

`icon`

semantics

Any icon to be associated with the resource. This is used, for example, to produce nice directory listings.

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

`maxage`

semantics

This attribute defines the allowed drift between the real content of a resource, and the one that is sent as request replies. The bigger this value, the more efficient the server can be, since it will be able to reuse cachd request replies for a longer time. This attribute takes effect only if it is defined, and if the resource provides a meaningfull last-modified attribute value. The unit is milliseconds.

type

This attribute is an editable LongAttribute

default value

This attribute defaults to **null**.

`send-md5`

semantics

Should we add a Content-Md5 header in the reply?

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **false**.

`putable`

semantics

- Attached to a FileResource:

Should the file resource support PUT requets ? When this flag is true, the file resource object will handle appropriately the HTTP PUT method, by overriding the resource's file with the new content. The old content will be saved using the emacs convention (the ~ files). Care should be taken when turning this feature on: you probably want to use some authentication filter to ensure that only authors are allowed to change resources.

- Attached to a DirectoryResource:

If this flag is setted to true that will allow you to create new resources through the HTTP PUT method. When this resource is looked up, it uses the normal DirectoryResource alogorithm to find the appropriate resource. If this fails, it goes to the resource indexer, and asks it to create a resource having the given name. If this succeeds, the newly created resource is attached as a child of the directory resource, and the PUT method is delegated to its HTTPFrame. Notice that the indexer should be configured to create a resource (with its appropriate frame) that handles the PUT method, otherwise, the resource will be effectively created, but it will fail to save the put'ed content (the FileResource, for example, should have a HTTPFrame with its putable attribute turned to true).

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **false**.

`relocate`

semantics (when attached to a DirectoryResource)

Should the directory emit a relocation reply when accessed through an invalid URL. A common case of handling invalid directory access is to emit a relocation reply so that the browser gets access to the directory through a valid URL (e.g. http://www.w3.org/pub is invalid, because pub is a directory, the correct URL is http://www.w3.org/pub/). When this flag is set to true, the directory resource will emit the appropriate relocation reply.

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **true**.

`index`

semantics (when attached to a DirectoryResource)

The optional name of the directory child resource that is to be used as the directory index. This attribute should name an existing child resource, that will be used as the index resource of the directory (all accesses to the directory will be delegated to it).

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

`icondir`

semantics (when attached to a DirectoryResource)

The name of the directory that handles this directory's icons. Each HTTPFrame has an optional icon attribute. When a directory resource needs to produce a listing it will dereference each icons relative to its icon directory.

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

`browsable` (when attached to a DirectoryResource)

semantics

Should this directory handle the GNN BROWSE method ? When trun to true the putable directory will reply appropriately to the GNN BROWSE method (which allows it to get the directory content).

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **false**.

`style-sheet-link`

semantics

This attribute is a relative link to a Style Sheet. This could be use by the frame when it generates a HTML document "on the fly", it could add a link to this style sheet.

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

Jigsaw Team

$Id: org.w3c.jigsaw.frames.HTTPFrame.html,v 1.3 1998/03/27 08:19:12 bmahe Exp $

All resources All frames

NegotiatedFrame

A frame that implements HTTP negotiation among a set of variant resources. This frame implements the HTTP/1.0 negotiation algorithm, which is known to be subject to some bugs.

It maintains the list of resources (by their names, see the variants attribute) among which to negotiate. It uses only the standard HTTPFrame attributes to perform the actual negotiation, which can be done based on the variants types, their natural languages, their encodings, or their character sets.

Inherits

The NegotiatedFrame class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ProtocolFrame
- HTTPFrame

Attributes description

The NegotiatedFrame defines the following attributes:

- variants

`variants`

semantics

The set of variants to negotiate among. Each variant is given by its name, as registered in its directory resource. The variants should provide as much information as possible (e.g. their quality, their content encodings, content language, etc.)

type

This attribute is an editable StringArrayAttribute

default value

This attribute defaults to **null**.

Jigsaw Team

$Id: org.w3c.jigsaw.frames.NegotiatedFrame.html,v 1.3 1998/03/27 08:19:17 bmahe Exp $

All resources All frames

DirectoryResource

A DirectoryResource able to create negotiated resources on the fly (as needed).

This class should be used as the basic class to export file system directories.

Inherits

The DirectoryResource class inerits from the following classes:

- Resource
- FramedResource
- AbstractContainer
- ContainerResource
- DirectoryResource

Attributes description

The DirectoryResource defines the following attributes:

- negotiable

`negotiable`

semantics

Should the directory resource automatically create resource with NegotiatedFrame? When this flag is turned to true, the directory resource will automatically create negotiable resources on top of normal resources: each time a new resource is added to the directory, the resource looks up for a resource having the new child name, but possibly different extensions. If this succeeds, either the found resource is already a negotiated resource, in which case the new child is added as one of its variant resource; otherwise (the negotiated resource doesn't exist), the directory resource creates it with only one variant (the new child resource).

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **false**.

Jigsaw Team

$Id: org.w3c.jigsaw.resources.DirectoryResource.html,v 1.3 1998/03/27 08:22:13 bmahe Exp $

Jigsaw Administration Tools

Jigsaw Home / Documentation Overview

There is a new **JigAdmin**, working only with JDK1.2 (swing), it is more beautiful and user friendly. If there is a JDK1.2 available for your platform you should download it and run the new JigAdmin. Here is the new JigAdmin documentation.

There are, since **Jigsaw** 1.0beta, two different ways of administrating your server. The first one is the oldest one and is a per-server administration scheme, using HTML and form-based editors.
The new one is a graphical interface which communicate with an Administration server, called **JigAdmin Server**. This server can administer multiple Jigsaw servers running on the same machine, provided these servers has been launched at the same time (means by the same java VM), we will call this graphical interface **JigAdmin**.
Since **Jigsaw** 2.0alpha, the only way of administrating **Jigsaw** is **JigAdmin**.

To use JigAdmin, follow one of these link:

- How to run JigAdmin
- How to use JigAdmin
- How to navigate in the tree browser
- How to stop Jigsaw

Running JigAdmin

JigAdmin Server

This is very easy in fact. The default configuration files provided both by the upgrade of an earlier **Jigsaw** or by the default installation are designed to start two server, an instance of **Jigsaw** and one **JigAdmin Server**. The default port of **JigAdmin Server** is 8009. When you start the server with:

```
java org.w3c.jigsaw.Main
```

You must see, among the trace, this output:

```
JigAdmin[2.0beta2]: serving at http://arachne.inria.fr:8009/
```

JigAdmin Client

Now that the administration server is running, you can access it with the following command:

```
java org.w3c.jigadm.Main [-root root] [url]
```

The default root is your current directory, so if you are in the same directory where you started **Jigsaw**, you don't need the "-root" option.
If you are running the administration server on the same machine, using the default port 8009, you don't need to provide an URL. The URL is the one of the administration server.

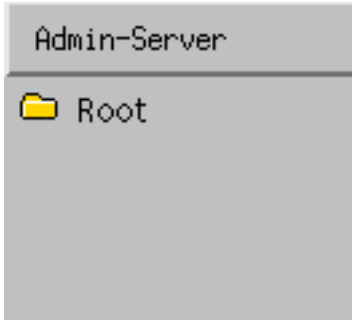
Using JigAdmin

Authentication

To access the administration server, you need to be authenticated for obvious reasons.

The realm used to access the server is "admin", the default user is "**admin**" and the default password is also "**admin**". After the first authentification, modify the user ot the password or both to avoid unwanted changes to your server!

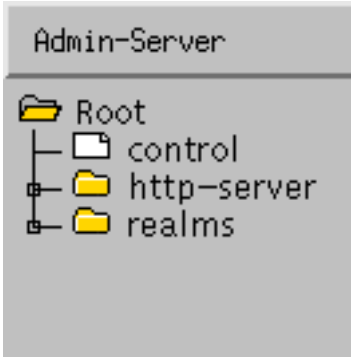
Now a new frame should popup and you must see this:



You are now ready to navigate through the server and configure your servers!

The administration zone

Expand the first node by clicking on the small icon. You must see something like this:



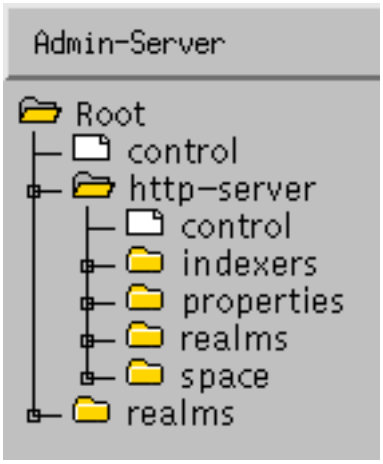
The "control" resource is used to control the server, the control resource of "Root" means that it controls the administration server itself.

The "http-server" resource is the first server administrated by **JigAdmin Server**. It corresponds to the server with the properties located in `http-server.props`.

The "realms" resource is used to allow access to the authentification realms used by the server. The "realms" resource of "Root" allows you to edit, modify or add new users to access **JigAdmin Server**.

The server zone

Expand one of the server node (in the default configuration, you will only have one server to edit: http-server). You must now see something like this:



The "control" resource has the same meaning and same role as in the administration zone.

The "indexers" resource is used to add, delete or edit the indexers of the server. Read more about indexers to understand how to edit them.

The "properties" resource corresponds to the general properties of **Jigsaw**. It correspond to the old `/Admin/Properties` directory of the form based editor.

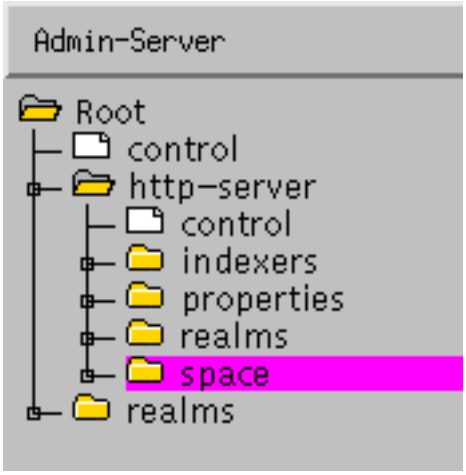
The "realms" resource is also the same as in the administration zone.

The "space" resource correspond to the root resource of your server. Under "space" you will find the resources actually served by **Jigsaw**. It is the place to go if you want to add, delete or edit resources, add, delete or edit filters and so on.

Note on the tree browser

Navigation helps

To expand or collapse a node, you must click on the icon.
To edit a resource, you must click on the label, the label will become magenta, like this:



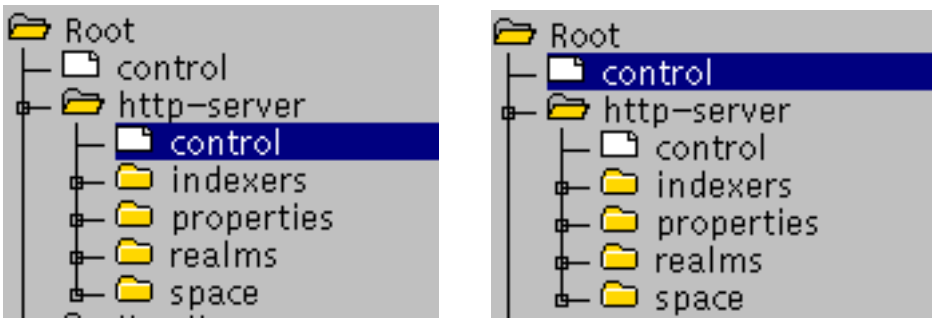
The menu "Admin-Server" has the following capabilities:

- Open - Go to another **JigAdmin Server**
- Open in new window - same as open, but in a new window
- Close window
- Exit - close all windows and exit

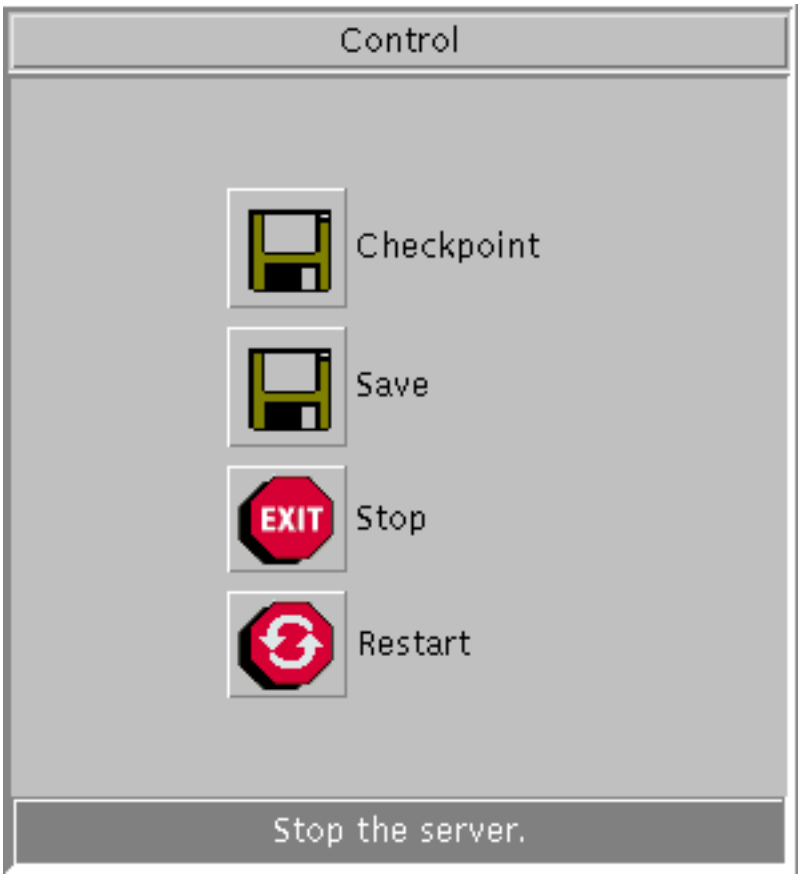
How to stop Jigsaw

First of all, **DO NOT USE Ctrl-c !** if you don't want to loose your changes**.**

You must stop the http server first, and then the administration server, for each server select the control node:



Then you must see this:



For each server click on the `Save` button and then on the `Stop` button, your configuration in saved and your server stopped.

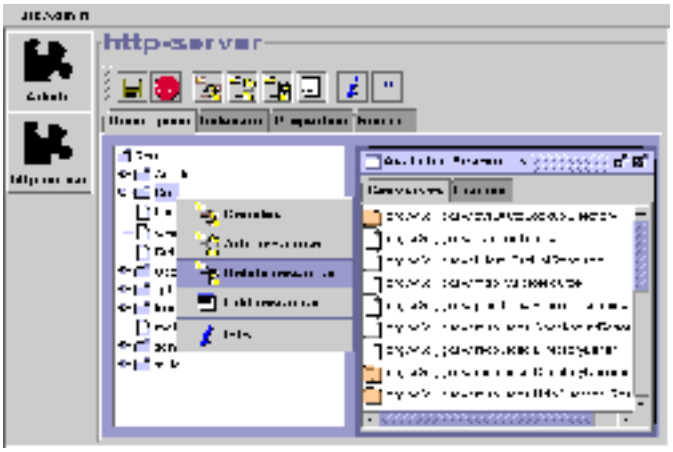
Jigsaw Team

$Id: AdminTools.html,v 1.15 1999/03/16 16:22:00 bmahe Exp $

JigAdmin
The Jigsaw Administration Tool

Jigsaw Home / Documentation Overview

JigAdmin is the **Jigsaw** Administration Tool, it's a graphical interface that communicates with an Administration server, called **JigAdmin Server**. This server can administer multiple Jigsaw servers running on the same machine, provided those servers has been launched at the same time (means by the same java VM). This version of **JigAdmin** is built with Swing Components, so it's more beautiful and user friendly (ie: Drag and Drop feature). Here is the documentation of the old **JigAdmin** (for JDK1.1.x users).



some snapshots (visual index).

- Running JigAdmin

  How to run JigAdmin? Command line options.
- The Main Menu

  Connect, close, exit.
- The Servers List

  Load server configuration.
- The ToolBar

  Save configuration, Stop server(s).
- Documents Space

  Add/Delete/Reindex Resources.
- Resource Editor

  Edit Resources/Frames, Add/Delete Frames.
- Indexers

  Create/Configure new Indexers.
- Properties

  Edit server Properties.
- Realms

  Configure Authentication, define username & password.
- Documentation on line

  Read the Reference documentation about resources.

Jigsaw Team

$Id: Overview.html,v 1.17 1999/03/31 12:47:16 bmahe Exp $

Indexer configuration

Jigsaw Home / Documentation Overview

 There is a new indexer documentation updated for the new JigAdmin.

In this section, the Jigsaw2.0 version of the indexing scheme will be presented. The Jigsaw1.0 scheme is easy to find out as it is much simpler than the 2.0 one (no protocol frames).

Goal of an indexer

The main goal of an indexer is to create and setup some resource automatically. The resources can be created depending on their name or their extension. Once the resource has been created, the indexer is also in charge of attaching the right frames to this resource, like the HTTP frame, the filters and so on.

Description of an indexer

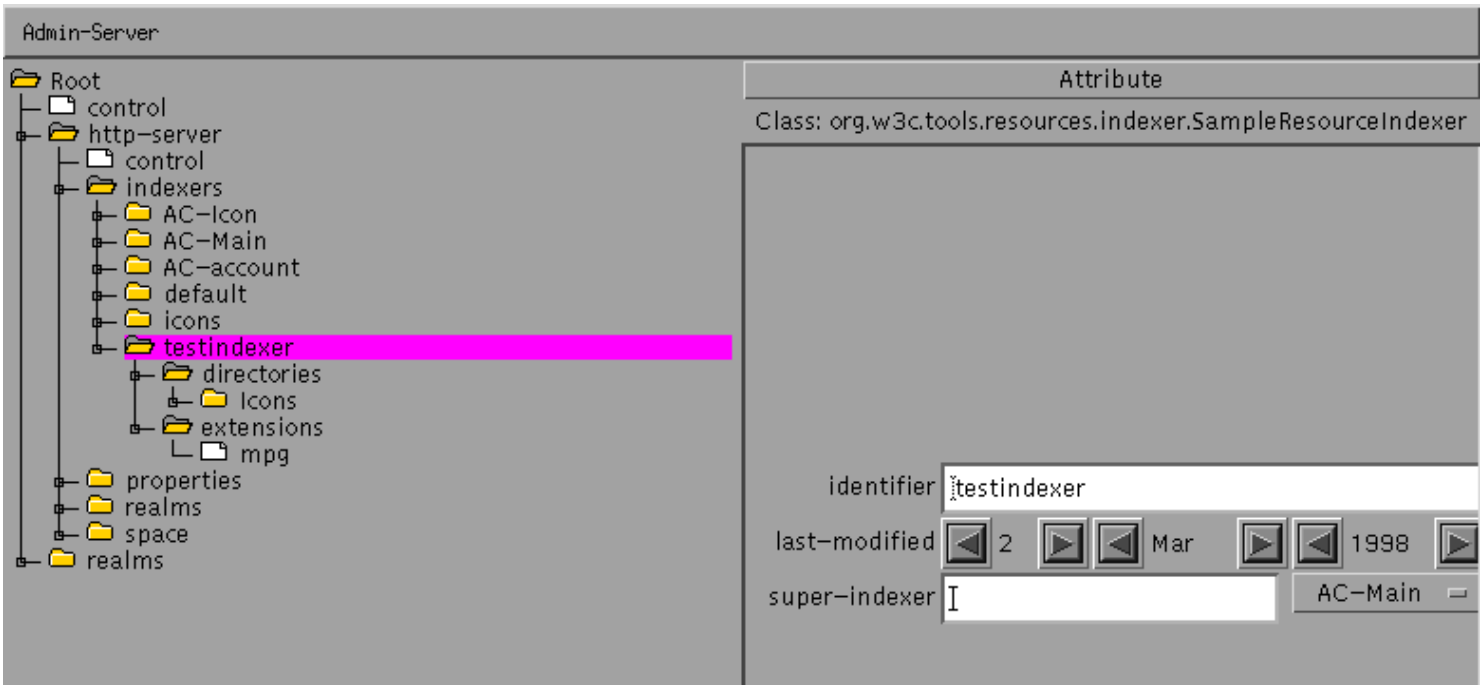
1. Class and attributes of an indexer.
   Class:
   Usually, the indexer's class is `org.w3c.tools.resources.indexer.SampleResourceIndexer`
   identifier
   The name of the indexer, ex: "icons"
   Last-modified
   Unused, but resent as, internally, it is a resource.
   super-indexer
   The name of the parent indexer used when the current indexer fails to index. By default, the super indexer is the "default" indexer.
2. The sons of an indexer
   directories
   Used to index files matching exactly a name, mainly used to index directories. You can specify that an "Icons" directory will always be negotiable, for example. The default name (ie: matching all directory names) is "*default*"
   extensions
   Used to index files with a specific extension. For example, "html" is a FileResource with an HTTPFrame set to give the "text/html" content type to this file. Then all the "foo.html" files will be indexed as "text/html" type object when accessed by HTTP. The default extension (ie: matching all the extension names) is "*default*". To index files with no extensions, you must use the name "*noextension*".

Setting up the indexer

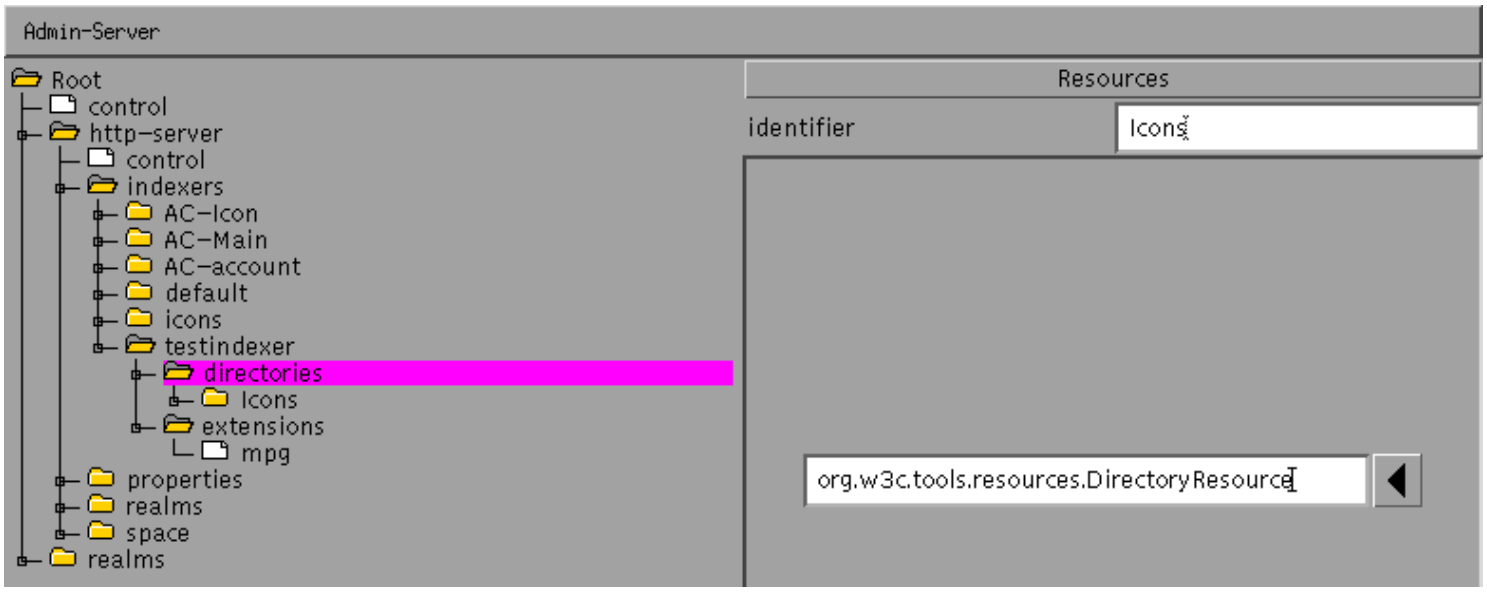
We will use a small example. The indexer will create all directories named "Icons" as a normal DirectoryResource, but using "icons" as its indexers. It means that all the Icons directories and their subdirectories will be indexed with "icons" instead of "default". Along with that, we will define a new extension "mpg" as a "video/mpeg" object.

1. Open a jigadmin window
2. Go to the server you want to add the indexer to (usually http-server), click on the "indexer" node. On the right side, you must see the indexer helper, like this.

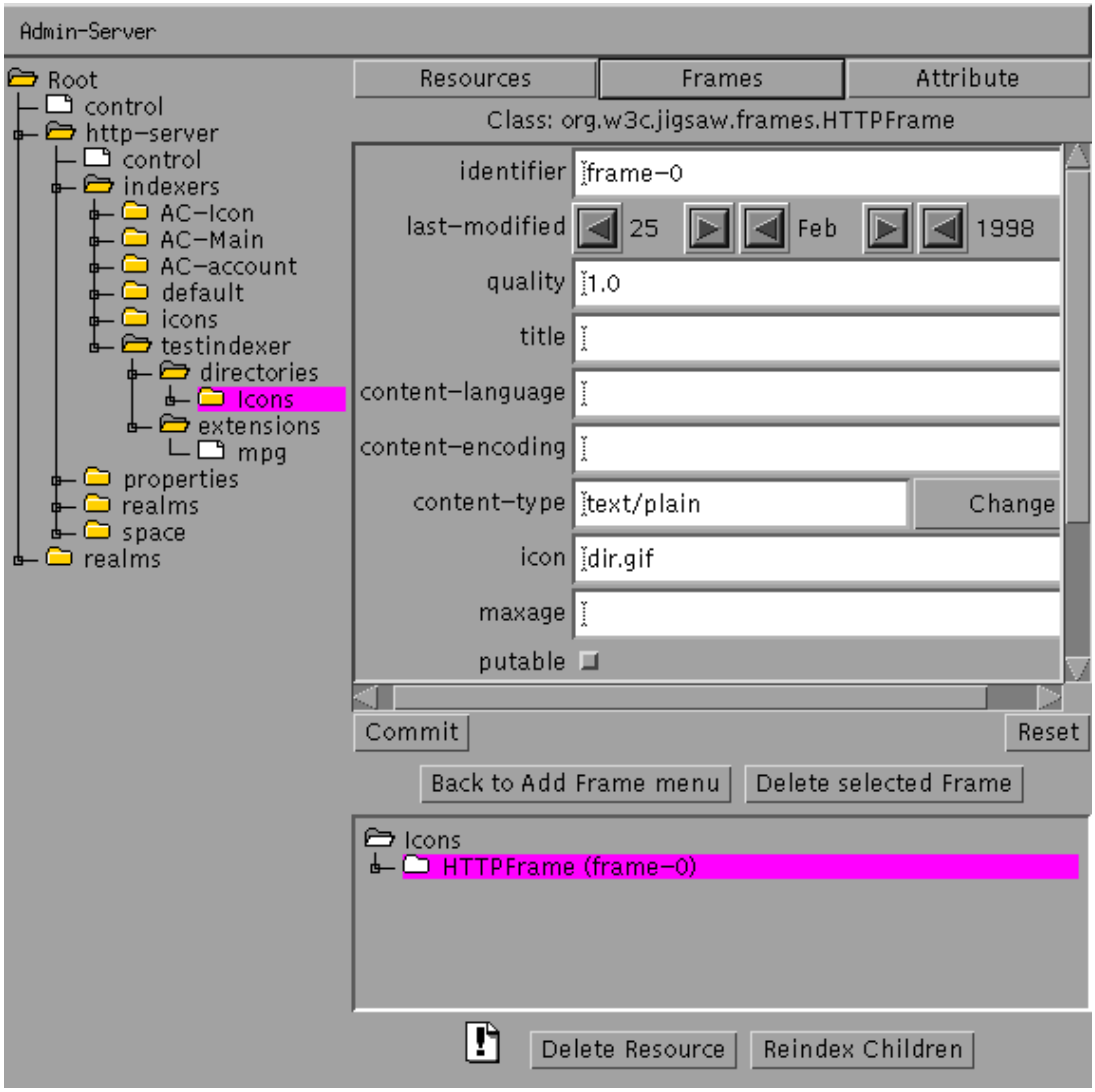
- Create an indexer. To do so, put a name in the identifier textfield (ex: "testindexer"), then, using the pulldown menu, select the class of indexer you want to use. The common indexer is `org.w3c.tools.resources.indexer.SampleResourceIndexer` click on the "Add Indexer" button and you are all set for the next step:
- You now have a "testindexer" node. Open this node and you will see the two directories "directories" and "extensions", as described above. The Attribute helper shows the super-indexer field. Let it blank as there is no specific indexer to call before asking the default indexer if this one fails.



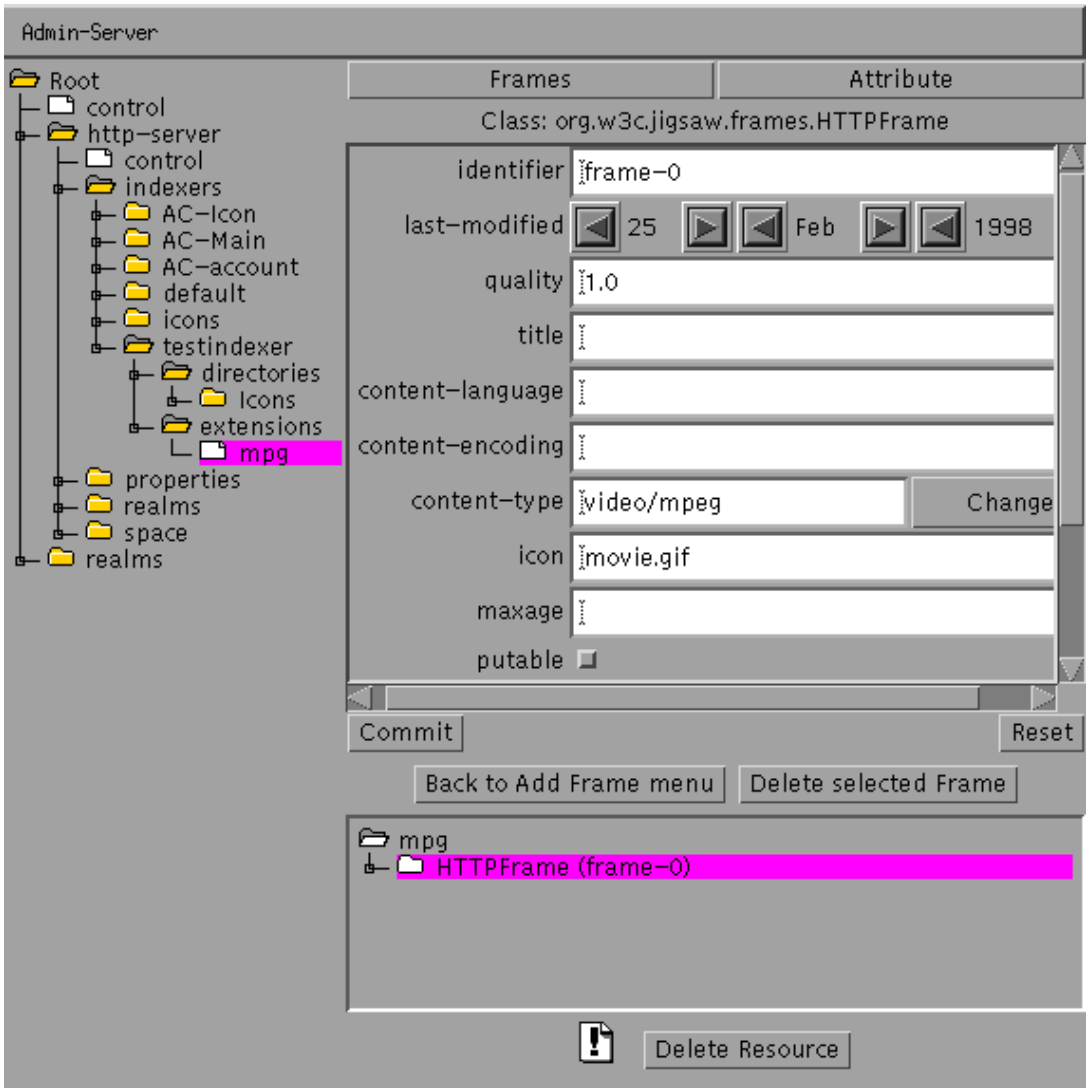
- Click on "directories", the resource helper appears on the right part of the window, type "Icons" in the indentifier textfield, and select `org.w3c.tools.resources.DirectoryResource` using the pull-down menu. It means that all the directories named "Icons" will be created as a DirectoryResource. You have now to configure this resource.



- Click on the new "Icons" directory under the "directories" node, the resource helper appears first. We want to change the indexer on this resource, so click on the Attribute button to show the right helper. You have now the attributes, amongst them, the indexer. Select "icons" using the option menu, then click on "Commit" to confirm your changes. BEWARE! The indexer is not yet completely setup, as this resource can't be accessed in any way! You need to add a ProtocolFrame to allow one protocol to access this resource. To do so, Click on the "Frames" button to invoke the frame helper. Select the most basic protocol frame: `org.w3c.jigsaw.frames.HTTPFrame` and add it to the resource. Click now in the small tree browser below, `HTTPFrame (frame-0)` should appear. Click on it and you will be able to configure the HTTPFrame (which looks like an old Jigsaw-1.0 Resource) set the icon to dir.gif to enhance it.



- Now you have to create the new extension. You must use the same process as above, except for some details. Add the "mpg" resource with the `org.w3c.tools.resources.FileResource` class to the "extensions" directory. Then add an HTTPFrame to it, open the "mpg" node in the little tree browser. Change the mime type, by selecting the right one (video/mpeg) in the content-type editor. Add movie.gif as the default icon for directory listing, commit.

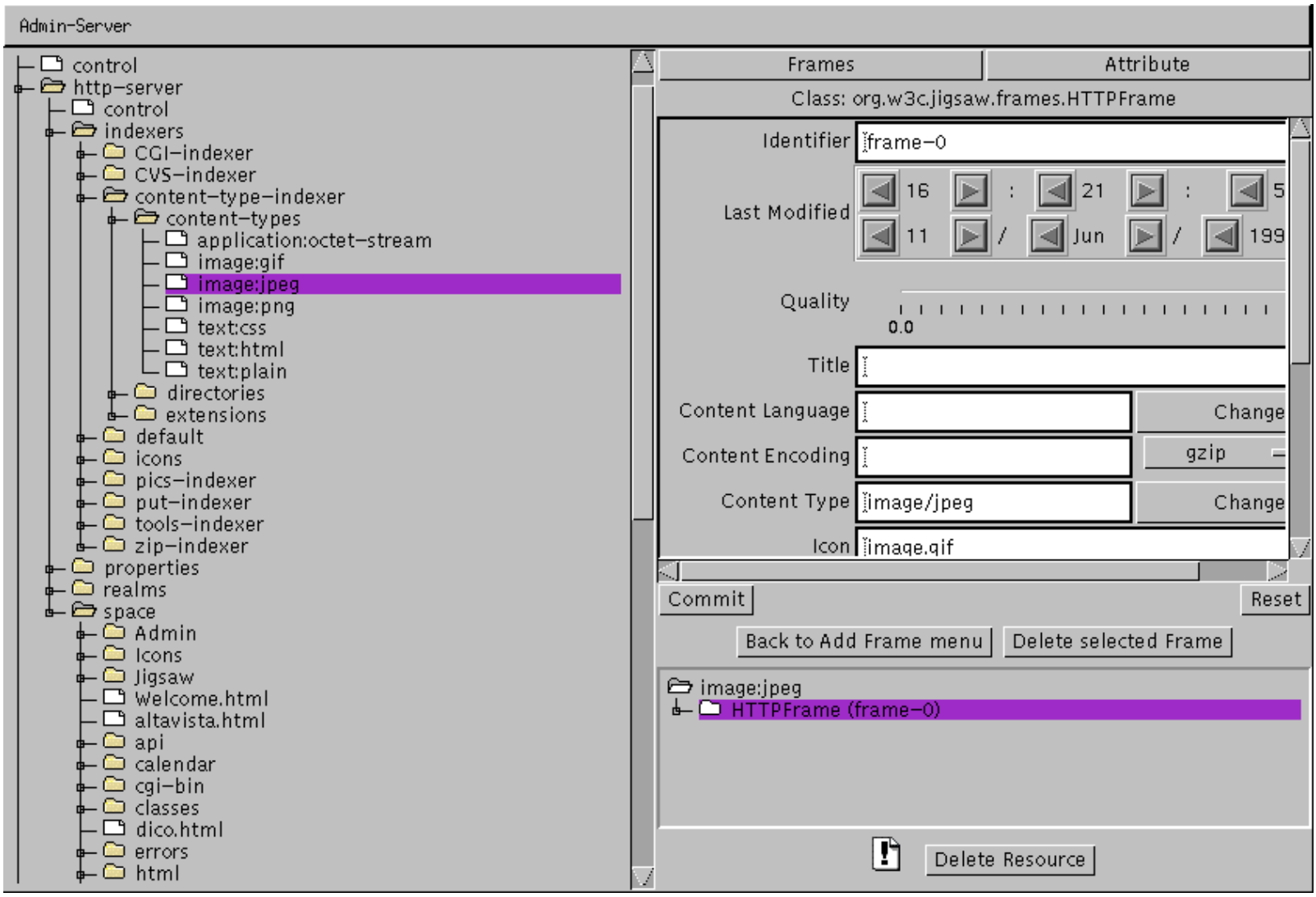


- You are all set now, don't forget to save your changes using the control resource of the server. You can now use this indexer.

Of course this is an example, if you want to add an extension for the whole server, the best way is to add it directly in the default indexer. Another thing, it is better to use `org.w3c.jigsaw.resources.DirectoryResource` than `org.w3c.tools.resources.DirectoryResource`, just to check if you read ALL the documentation ;)

The Content Type Indexer

In some cases the file extension is not the only criteria, for exemple when a PUT request occurs the indexer should use the Content-Type header comming with the request (if there is a content-type header). This is the job of the Content Type Indexer.



The Content Type Indexer (`org.w3c.jigsaw.indexer.ContentTypeIndexer`), has one more child, the `content-types` node. The associations between mime types and resources are stored in this new child.

Since 2.0.2 the ContentTypeIndexer accept generic mime types like **text:***, ***:xml** or even ***:***. For example, if you define **text:*** as a FileResource using a HTTPFrame (with a content-type set to ***none***) all content types like text/html, text/plain, text/xml will be accepted.

Note: As you can see in the screenshot, the mime types stored in the indexer are not "real" mime types, the `'/'` has been replaced by a `':'`. We decided that because the `'/'` can create some conflicts with the URLs in **Jigsaw**.

Jigsaw Team

$Id: indexers.html,v 1.25 1999/08/02 14:09:06 yves Exp $

Running JigAdmin

Jigsaw Home / Documentation Overview / Jigsaw Administration Tool

This document has the following sections:

Installation of JigAdmin

First of all, you need the **JDK1.2** to run **JigAdmin**, so make sure that you have downloaded it. If there is a no JDK1.2 available for you platform, use the old JigAdmin.

You just have to set your **CLASSPATH**.

```
   # This depends on the shell you are using, we're assuming /bin/sh
   UNIX

CLASSPATH=INSTDIR/Jigsaw/classes/jigsaw.jar:INSTDIR/Jigsaw/classes/jigadmin.jar:INSTDIR/Jigsaw/classes/sax.jar:INSTDIR/Jigsaw/classes/xp.jar
           export CLASSPATH

   Windows
           set
CLASSPATH=INSTDIR\Jigsaw\classes\jigsaw.zip;INSTDIR\Jigsaw\classes\jigadmin.jar;INSTDIR\Jigsaw\classes\sax.jar;INSTDIR\Jigsaw\classes\xp.jar
```

Running JigAdmin

JigAdmin Server

This is very easy in fact. The default configuration files provided by the default installation are designed to start two server, an instance of **Jigsaw** and one **JigAdmin** Server. The default port of **JigAdmin** Server is **8009**. When you start the server with:

java org.w3c.jigsaw.Main

You must see, among the trace, this output:

JigAdmin[2.0.1]: serving at http://ender.inria.fr:8009/

JigAdmin Client

Now that the administration server is running, you can access it with the following command:

java org.w3c.jigadmin.Main [-root root] [url]

The default root is your current directory, so if you are in the same directory where you started **Jigsaw**, you don't need the "-root" option. If you are running the administration server on the same machine, using the default port **8009**, you don't need to provide an URL. The URL is the one of the administration server.

If you are not in the root directory, you can access the administration server with:

UNIX

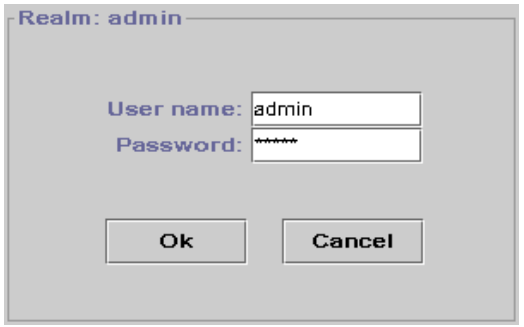
```
java org.w3c.jigadmin.Main -root INSTDIR/Jigsaw/Jigsaw/
```

Windows

```
java org.w3c.jigadmin.Main -root INSTDIR\Jigsaw\Jigsaw\
```

Authentication

To access the administration server, you need to be authenticated for obvious reasons.



The realm used to access the server is "**admin**", the default user is "**admin**" and the default password is also "**admin**". After the first authentication, modify the user of the password or both to avoid unwanted changes to your server!

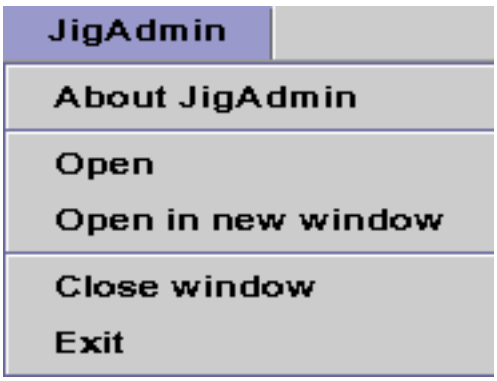
Now, JigAdmin is running!

Jigsaw Team

$Id: running.html,v 1.12 1999/09/09 09:30:35 bmahe Exp $

The Main Menu

Jigsaw Home / Documentation Overview / Jigsaw Administration Tool



About JigAdmin: Popup a "About" window.

Open: Connect to another JigAdmin Server

Open in new window: Create a new JigAdmin Window and connect to another JigAdmin Server.

Close: Close this window

Exit: Exit, close all JigAdmin windows.

Jigsaw Team

$Id: menu.html,v 1.2 1999/03/16 13:40:10 bmahe Exp $

The Servers list

Jigsaw Home / Documentation Overview / Jigsaw Administration Tool

The servers list allows you to select the server you want to configure. The default configuration has two servers, the Administration Server (**Admin**) and the HTTP Server (**http-server**).

When you launch **JigAdmin** the server selected is the Administration server.

So, click on the button relative to the server you want to configure, and enjoy!

Jigsaw Team

$Id: slist.html,v 1.9 1999/03/16 13:40:18 bmahe Exp $

The Toolbar

Jigsaw Home / Documentation Overview / Jigsaw Administration Tool

The ToolBar contains the following buttons:

Save the current configuration. If the current server is the Administration Server, save all the servers configuration including the Administration Server.

Stop the selected server. If the current server is the Administration Server, stop all servers including the Administration Server.

Reindex the selected containers. If there is no container selected, do nothing. If the resource selected is not a container, display an error dialog.

Add a resource to the selected container. If there is no container selected, do nothing. If the resource selected is not a container, display an error dialog.

Delete the selected resources. If there is no resources selected, do nothing. If you are not allowed to delete the selected resource, display an error dialog.

Edit the selected resource, display a ResourceEditor that allows you to edit the resource properties. If there is no resource selected, do nothing.

Popup a mini HTML browser that point to the "Resource Reference Documentation" of the selected resource. If there is no selected resource, do nothing.

Popup a mini HTML browser that point to the Jigsaw Documentation.

Jigsaw Team

$Id: toolbar.html,v 1.8 1999/03/16 13:40:24 bmahe Exp $

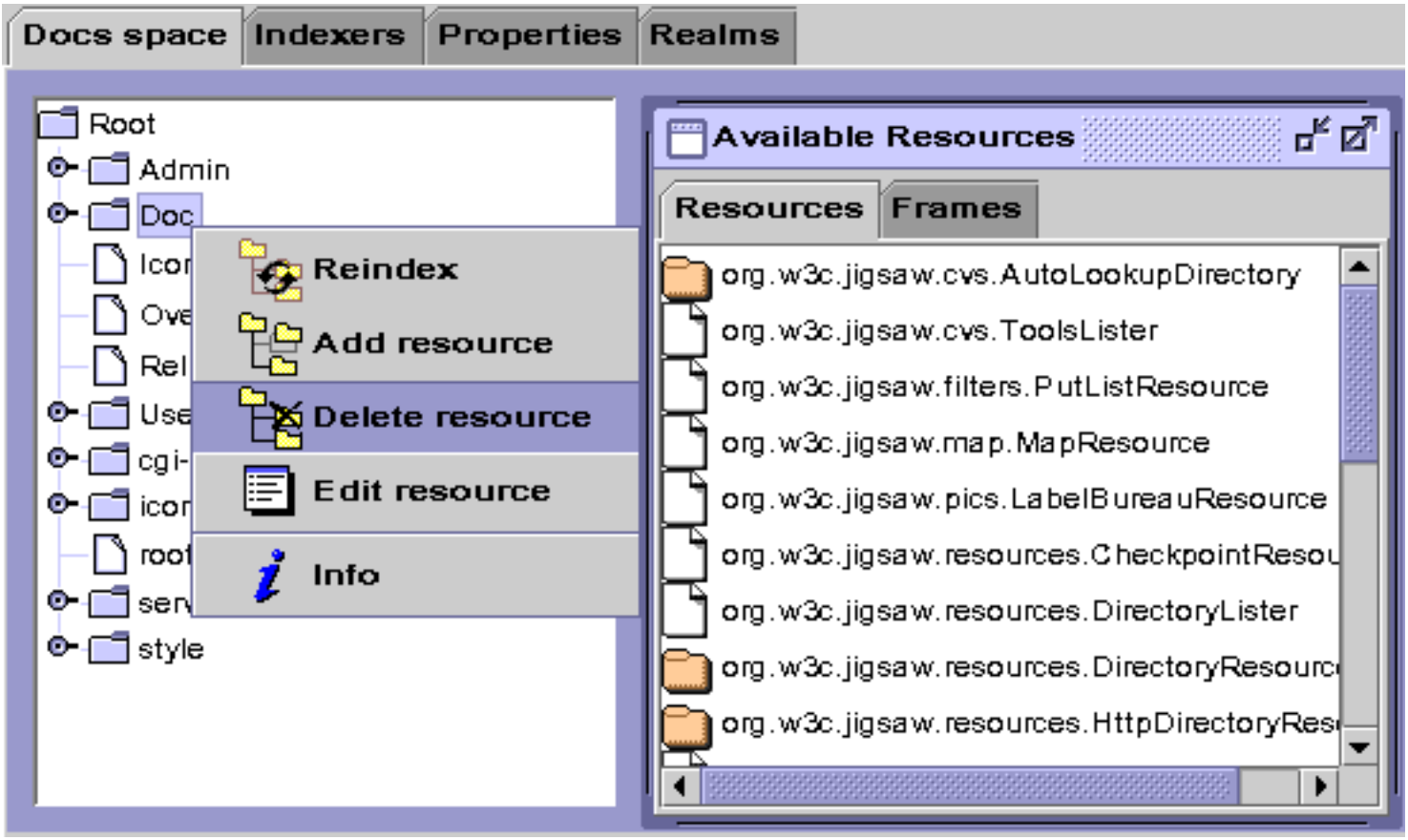
licensing rules apply. Your interactions with this site are in accordance with our public and Member privacy statements.

The Documents space
Managing the documents

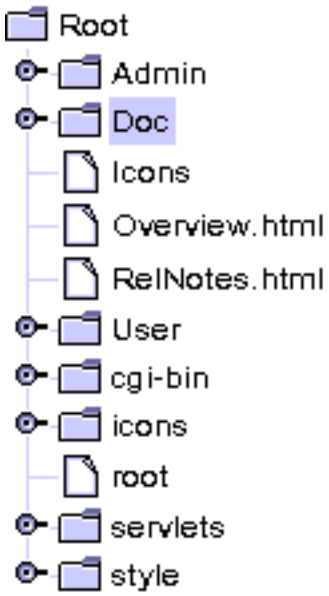
Jigsaw Home / Documentation Overview / Jigsaw Administration Tool

The Documents Space is the place where the documents available on your server are stored. Here you can add, delete and edit your resources

This document has the following sections:



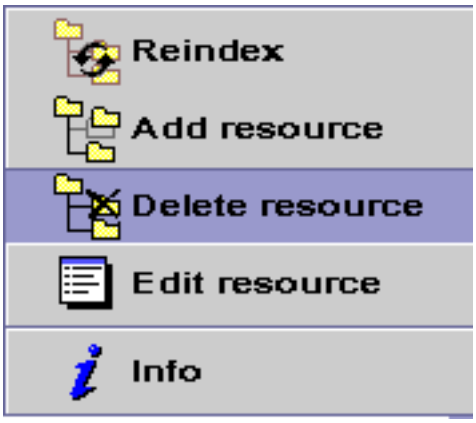
The Resource Tree



The Resource Tree is a view of the documents stored in your server. Click one time on the node to select it. By clicking on a you will expands the node, for example "Doc", and see the children of this resource.

By clicking another time on you will collapse the node. By double clicking on a you will popup a Resource Editor.

The Popup Menu



The Popup Menu appears if some resource(s) are selected and if you click on the third button of the mouse (or `Ctrl Mouse-button1` on some OS). Here is the description of the action associated to each menu item:

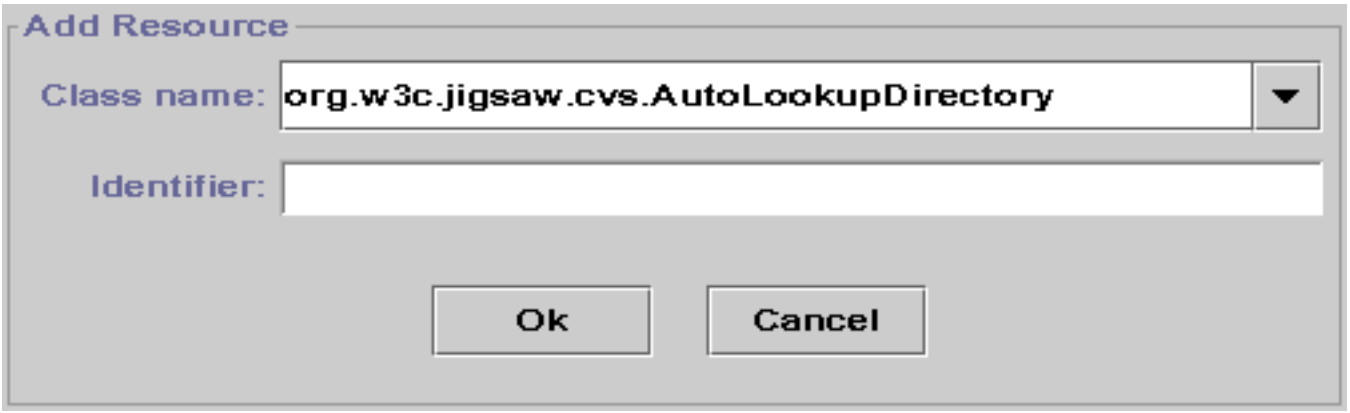
- Reindex
- Add Resource
- Delete Resource
- Edit Resource
- Info

Reindex

Reindex the selected containers. If there is no container selected, do nothing. If the resource selected is not a container, display an error dialog.

Add Resource

Add a resource to the selected container. Popup a dialog that allows you to enter (or select) the resource class and to enter the resource identifier. If there is no container selected, do nothing. If the resource selected is not a container, display an error dialog.



Delete Resource

Delete the selected resources. If there is no resources selected, do nothing. If you are not allowed to delete the selected resource, display an error dialog.

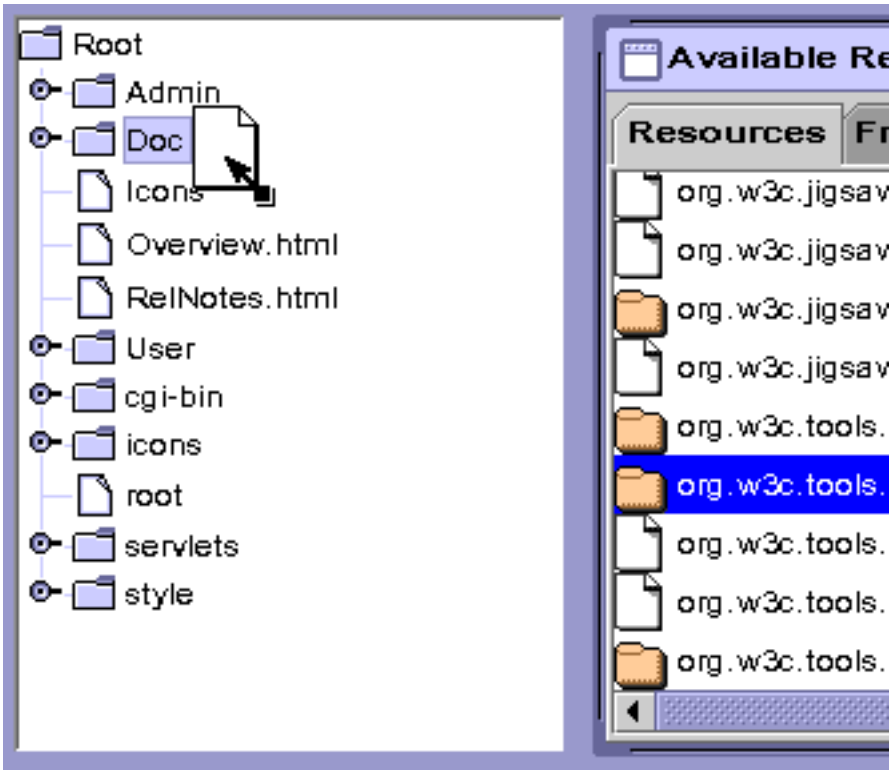
Edit Resource

Edit the selected resource, display a ResourceEditor that allows you to edit the resource properties. If there is no resource selected, do nothing.

Info

Popup a mini HTML browser that point to the "Resource Reference Documentation" of the selected resource. If there is no selected resource, do nothing.

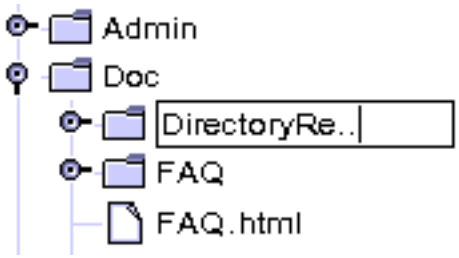
Drag And Drop



Now you can add a resource by dragging its class name and dropping it on its container.

Select the resource class in the "Available Resources" window, drag it by clicking on it and moving the cursor with the mouse button still pressed, then drop it on its (new) container.

The resource has been added, you can change its name by clicking one time on its name (in the resource tree).



Jigsaw Team

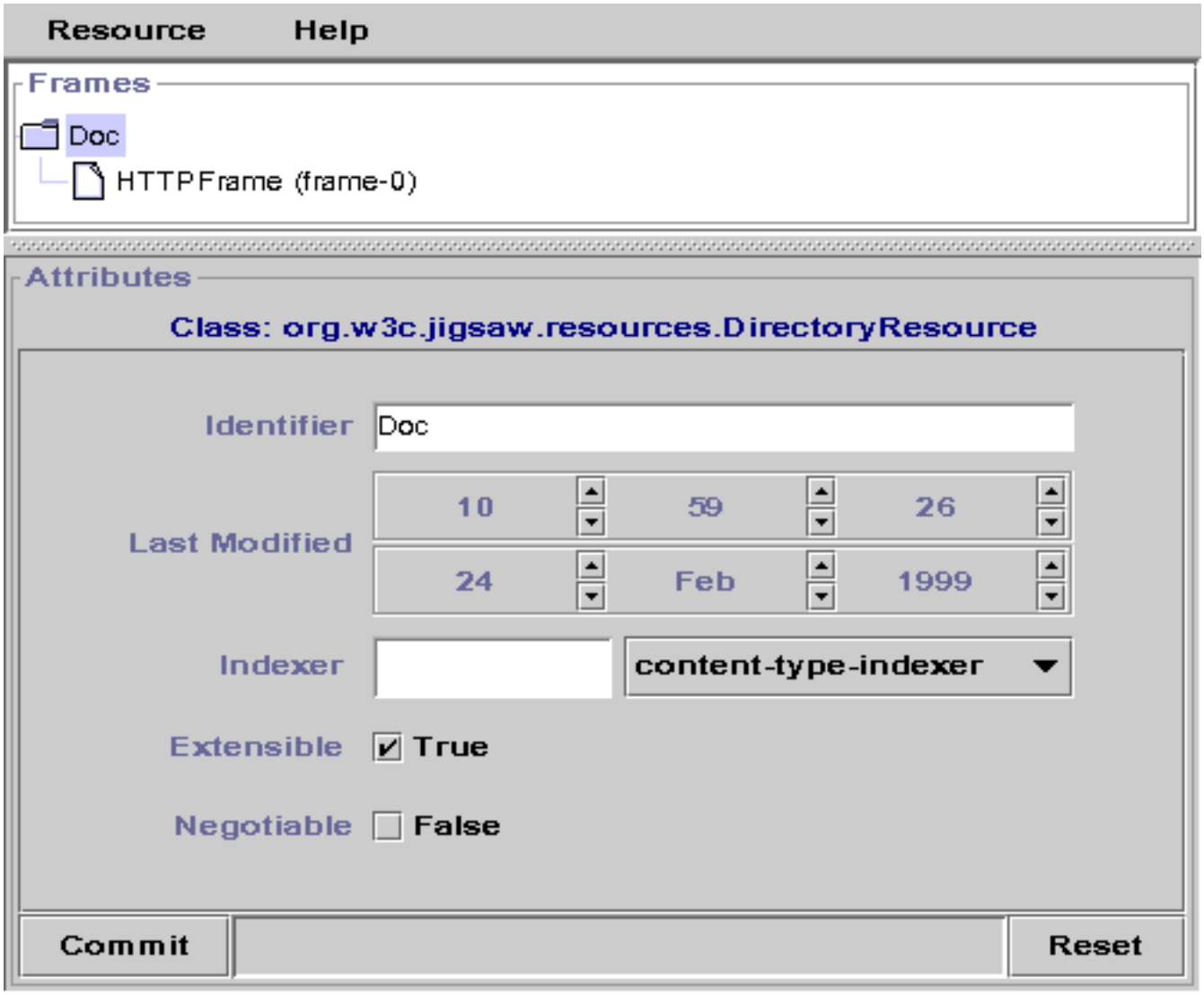
$Id: docs_space.html,v 1.9 1999/03/16 13:40:05 bmahe Exp $

The Resource Editor
Editing resource and frame properties

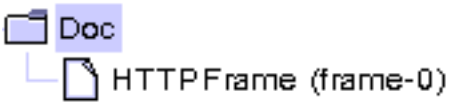
Jigsaw Home / Documentation Overview / Jigsaw Administration Tool

The Resource Editor allows you to edit the resource properties and to add, remove or configure the resource frames.

This document has the following sections:

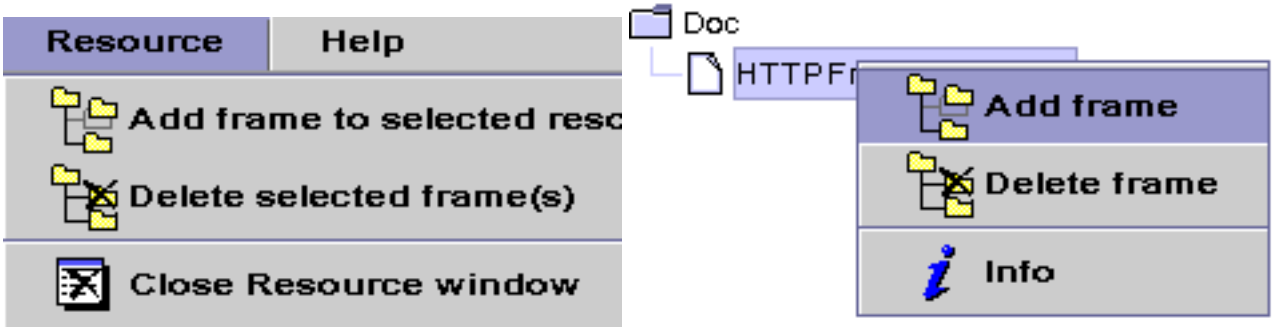


The Frames Tree



The Frames Tree works like the Resource Tree except that double clicking on a  expands or collapse the node instead of displaying the Resource Editor. When a resource is selected on the Frames Tree, the Resource Editor display its Attributes Editor.

The Resource Menu and the Popup Menu



Those Menus have the followings items:

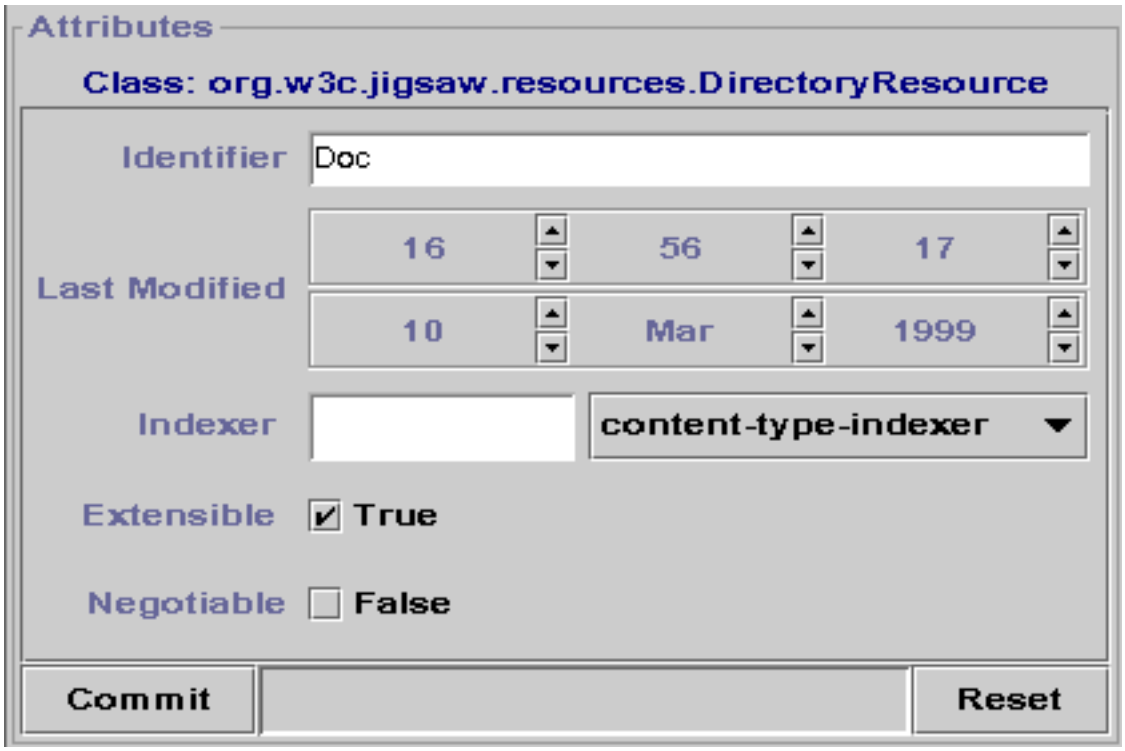
 Add a frame to the selected resource (or frame). If there is no resource selected, do nothing.

 Delete the selected resource/frames. If there is no resources selected, do nothing. If you are not allowed to delete the selected resource, display an error dialog.

 Popup a mini HTML browser that point to the "Resource Reference Documentation" of the selected resource/frame. If there is no selected resource, do nothing.

 Close the Resource Editor.

The Attributes Editor



The Attributes Editor allows you to edit the resource/frame properties like "Indexer", "Extensible",

"Negotiable"... Each attribute has its own graphical editor.

Commit	Commit your changes to the server. If you don't commit your changes, you will loose them.
Reset	Reset your changes (if not committed).

So, now you are able to configure the **Jigsaw** Resources.

Jigsaw Team

$Id: edit.html,v 1.7 1999/03/16 13:40:06 bmahe Exp $

Indexers
How to create resources automatically

Jigsaw Home / Documentation Overview / Jigsaw Administration Tool

This document has the following sections:

What is an Indexer?

Goal of an indexer

The main goal of an indexer is to create and setup some resource automatically. The resources can be created depending on their name or their extension. Once the resource has been created, the indexer is also in charge of attaching the right frames to this resource, like the HTTP frame, the filters and so on.

Each DirectoryResource (and subclasses) is associated to an indexer, if no indexer is specified the DirectoryResource is associated to the "default" indexer.

Description of an indexer

1. Class and attributes of an indexer

   Class:

   Usually, the indexer's class is `org.w3c.tools.resources.indexer.SampleResourceIndexer`

   Identifier

   The name of the indexer, ex: "icons"

   Last Modified

   Unused, but resent as, internally, it is a resource.

   Super Indexer

The name of the parent indexer used when the current indexer fails to index. By default, the super indexer is the "default" indexer.

2. The sons of an indexer

directories

Used to index files matching exactly a name, mainly used to index directories. You can specify that an "Icons" directory will always be negotiable, for example. The default name (ie: matching all directory names) is "*default*"

extensions

Used to index files with a specific extension. For example, "html" is a FileResource with an HTTPFrame set to give the "text/html" content type to this file. Then all the "foo.html" files will be indexed as "text/html" type object when accessed by HTTP. The default extension (ie: matching all the extension names) is "*default*". To index files with no extensions, you must use the name "*noextension*".

content-types (only for the Content Type Indexer)

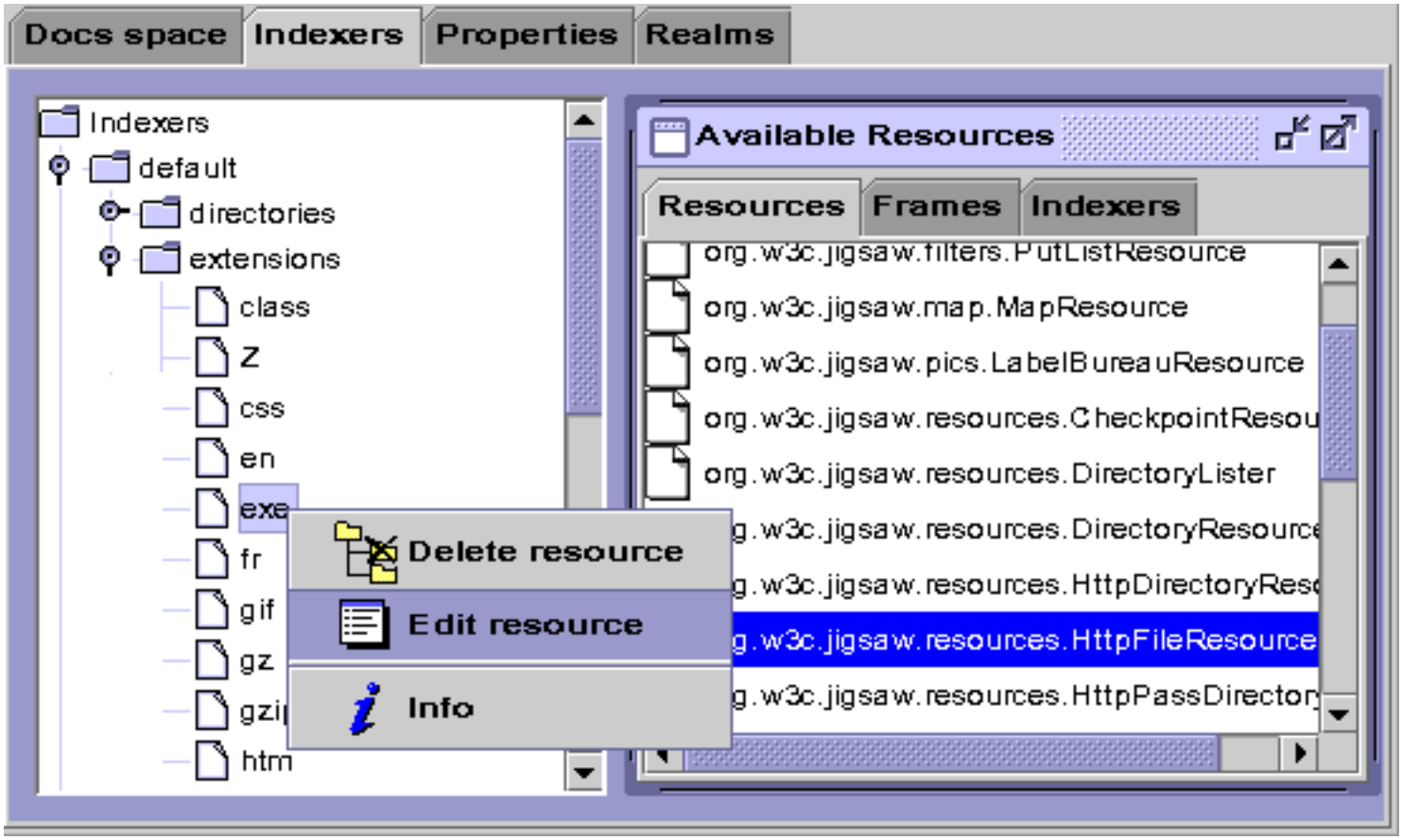
In some cases the file extension is not the only criteria, for example when a PUT request occurs the indexer should use the Content-Type header coming with the request (if there is a content-type header). This is the job of the Content Type Indexer. The Content Type Indexer (org.w3c.jigsaw.indexer.ContentTypeIndexer), has one more child, the content-types node. The associations between mime types and resources are stored in this new child.

Since 2.0.2 the ContentTypeIndexer accept generic mime types like **text:***, ***:xml** or even ***:***. For example, if you define **text:*** as a FileResource using a HTTPFrame (with a content-type set to ***none***) all content types like text/html, text/plain, text/xml will be accepted.

Note: The mime types stored in the indexer are not "real" mime types, the '/' has been replaced by a ':'. We decided that because the '/' can create some conflicts with the URLs in Jigsaw.

You can find a sample indexer configuration in this page.

Indexers in JigAdmin



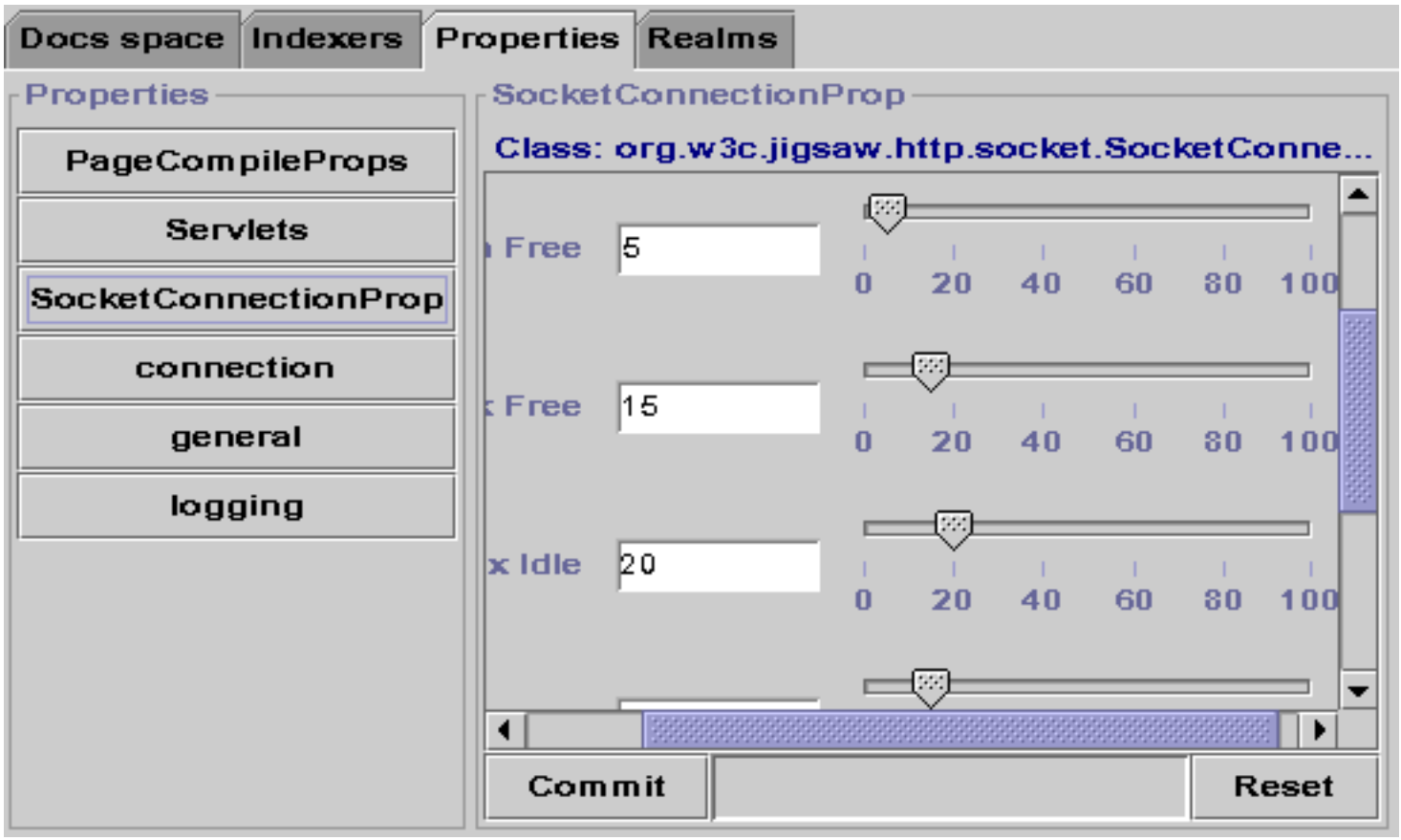
The Indexers Space is exactly the same thing than the Documents Space except that indexers classes are available in the "Available Resources" window. You are still able to add, delete, configure resources and frames but only in the indexers nodes (**directories**, **extensions** and sometimes **content-types**). Of course, you can also create new indexers (under the **Indexers** node).

Jigsaw Team

$Id: indexers.html,v 1.10 1999/03/30 09:39:33 bmahe Exp $

Properties

Jigsaw Home / Documentation Overview / Jigsaw Administration Tool



The Jigsaw properties are reachable here, select the properties you want to edit by clicking on a button on the left of the screen. Then, edit the attribute(s) you want to modify.

Commit	Commit your changes to the server. If you don't commit your changes, you will loose them.
Reset	Reset your changes (if not committed).

Jigsaw Team

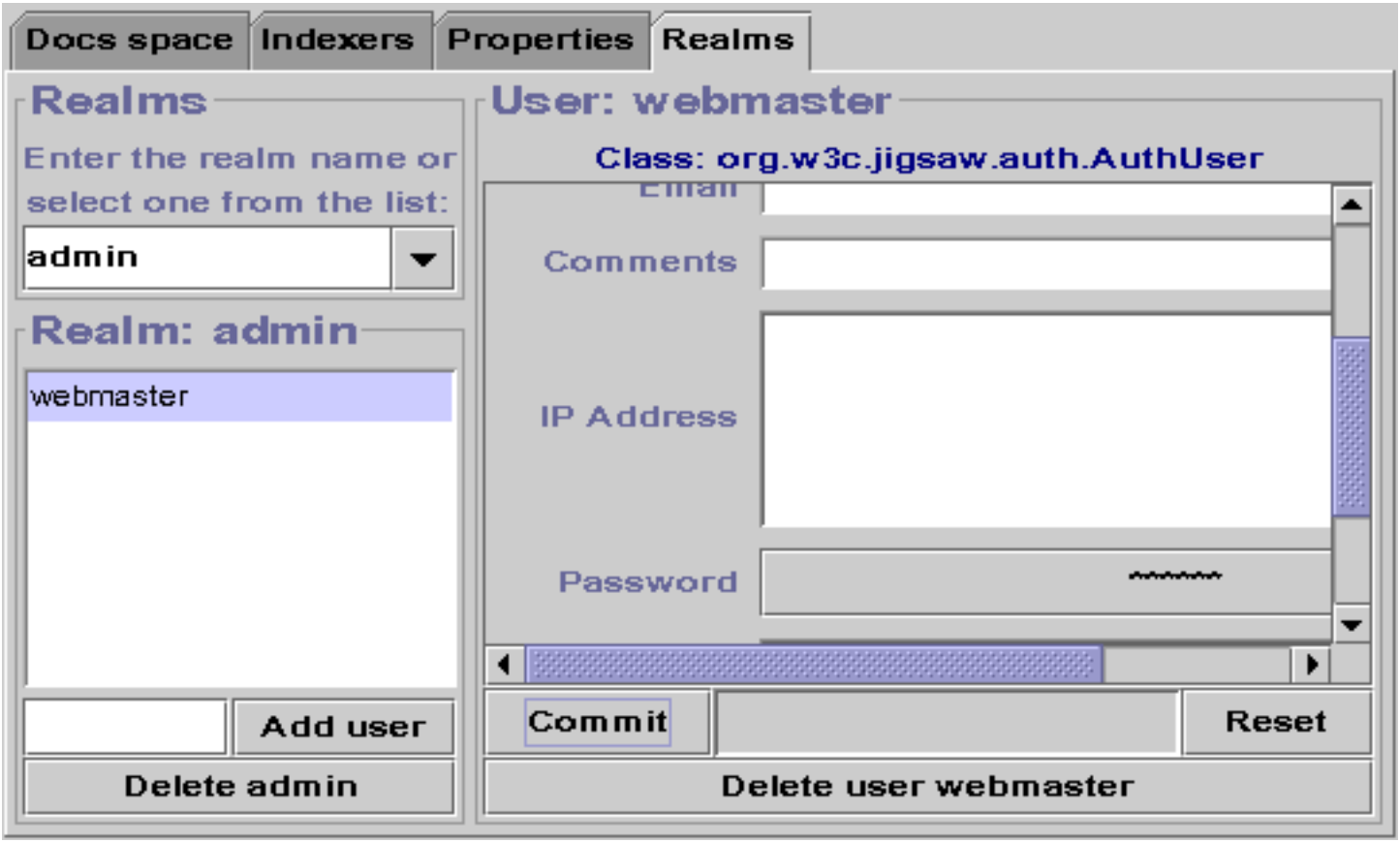
$Id: properties.html,v 1.9 1999/03/16 13:40:12 bmahe Exp $

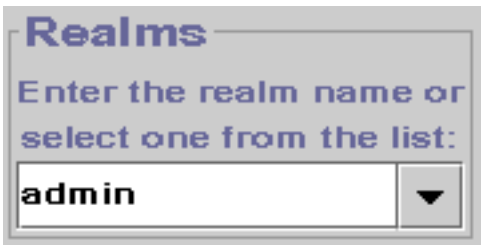
licensing rules apply. Your interactions with this site are in accordance with our public and Member privacy statements.

Realms
Authentication in Jigsaw

Jigsaw Home / Documentation Overview / Jigsaw Administration Tool

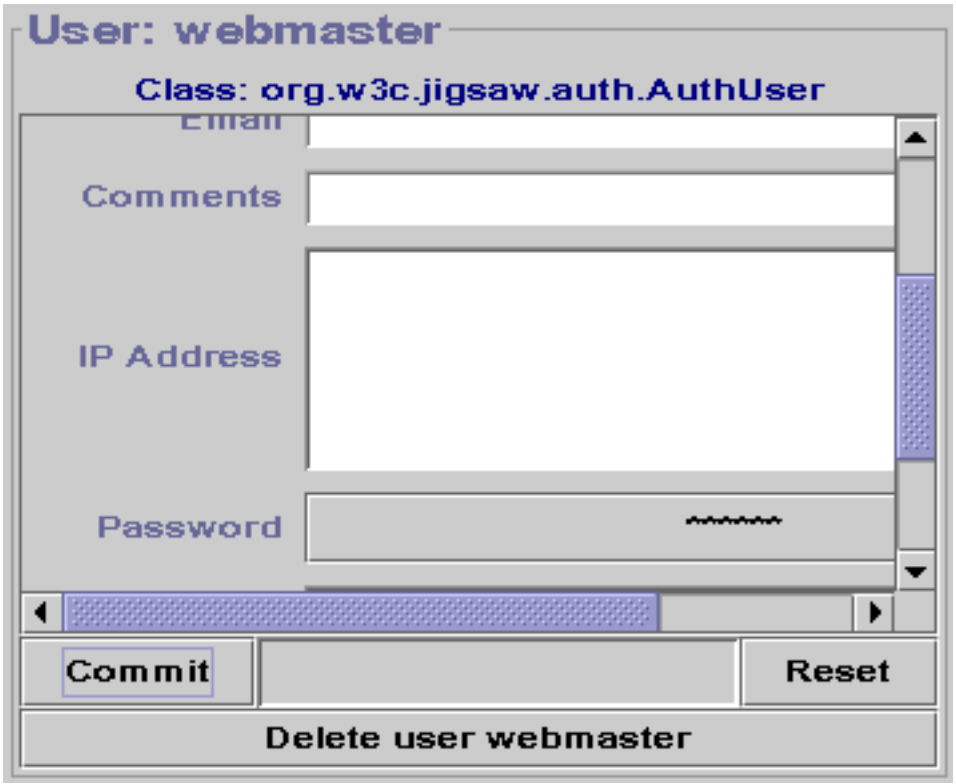
An authentication realm is a database that will contain the description of a set of users, along with their passwords and/or IP addresses. Read this page for more details on Authentication in **Jigsaw**.





Here you can select or create a Realm. Creating a realm is really simple, you just have to enter its name and hit the Enter key. In this example, the realm selected is "admin".

Here you can delete the selected realm ("admin" in this example) or add a new user in this realm by entering its name in the little textfield and clicking on "Add User".



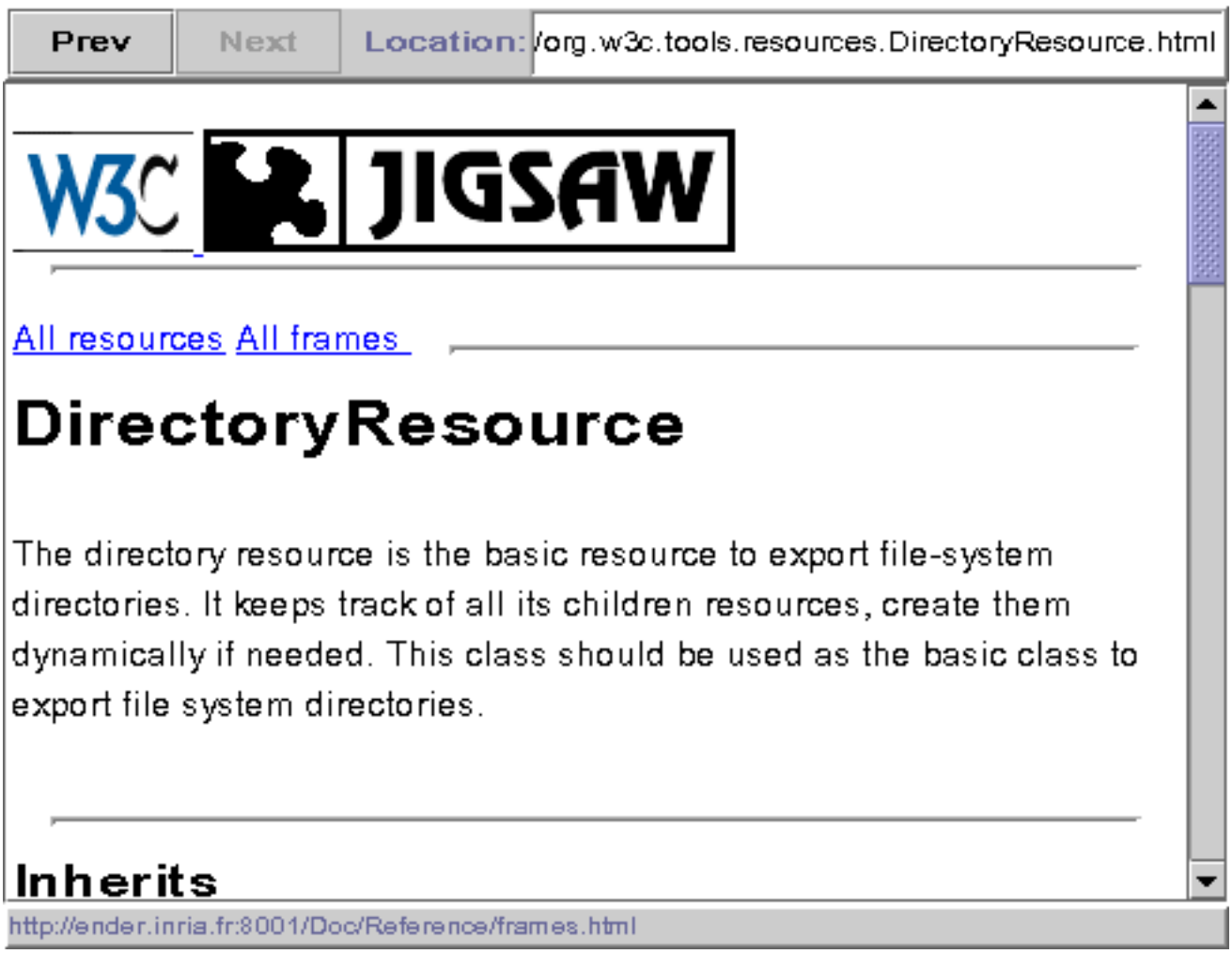
This is the Attribute Editor of the AuthUser resource. Here you can edit the user attributes like "Password", "Email" or "IP Adress". Read the AuthUser Reference Documentation for more details.

Jigsaw Team

$Id: realms.html,v 1.9 1999/03/16 13:40:14 bmahe Exp $

Documentation On Line
Browse the documentation from JigAdmin

Jigsaw Home / Documentation Overview / Jigsaw Administration Tool



This mini browser allows you to read the **Jigsaw** documentation from **JigAdmin** it has a very simple graphical interface, so I don't need to describe it. :-)

Jigsaw Team

$Id: doc.html,v 1.4 1999/03/16 13:40:03 bmahe Exp $

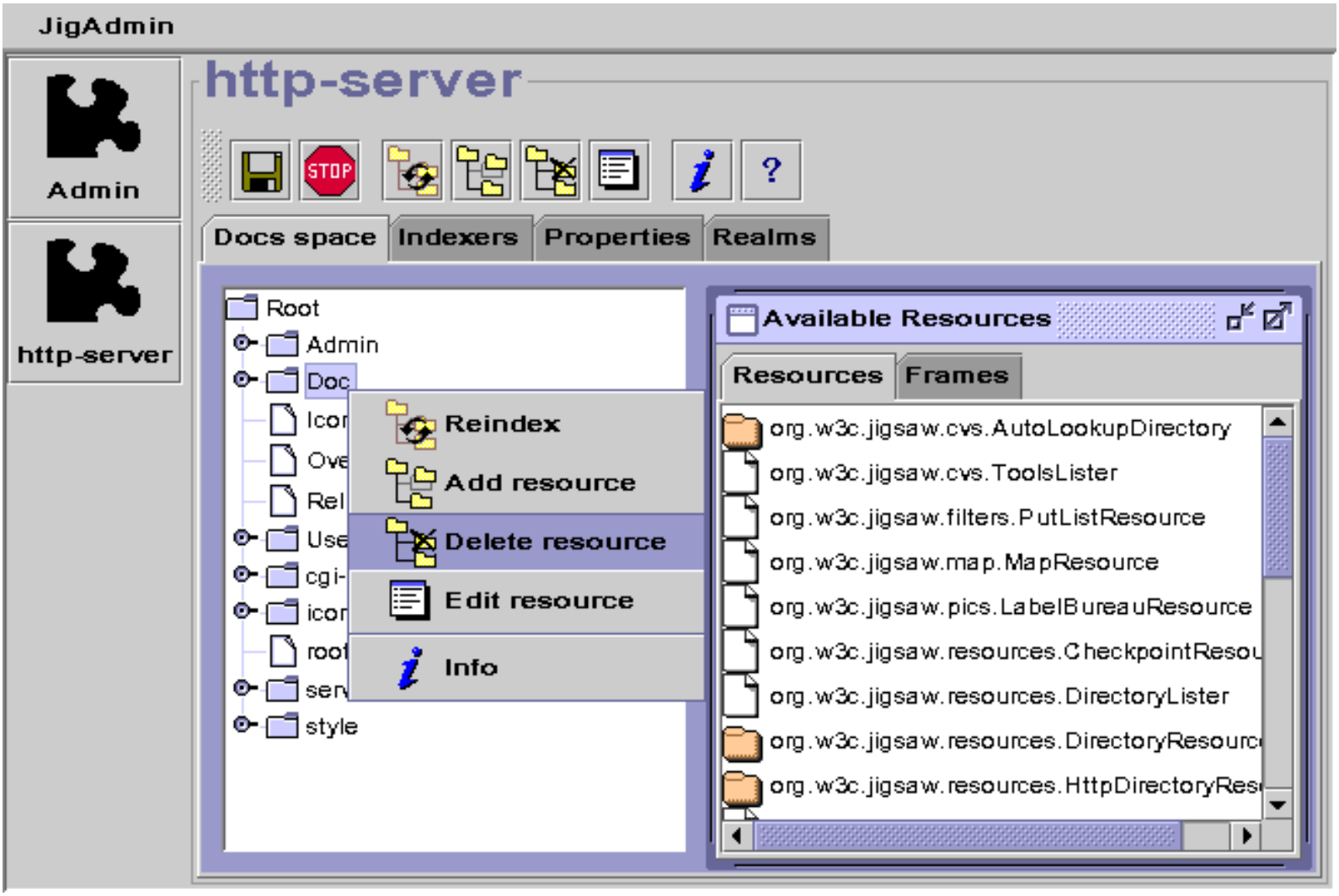
licensing rules apply. Your interactions with this site are in accordance with our public and Member privacy statements.

Snapshots
Visual Index

Jigsaw Home / Documentation Overview / Jigsaw Administration Tool

This is a visual index, click on a widget you will go to its documentation page.

The main window



The Resource Editor

Jigsaw Team

$Id: snapshots.html,v 1.8 1999/03/30 12:38:42 null Exp $

All resources All frames

GenericAuthFilter

The GenericAuthFilter provides a basic authentication mechanism. Depending on the realm it uses, and on its user description, it will use either IP filtering, or IP filtering with Basic authentication.

The filter is configured to allow only some users or some users belonging to a given set of groups to access the information it protects. Users are recorded in a realm database. If no allowed users or groups is specified, all users of the realm are allowed.

Inherits

The GenericAuthFilter class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ResourceFilter
- AuthFilter

Attributes description

The GenericAuthFilter defines the following attributes:

- users
- groups

`users`

semantics

The list of the names of the users allowed to access the information protected by the authentication filter.

type

This attribute is an editable StringArrayAttribute

default value

This attribute defaults to **null**.

`groups`

semantics

The list of the names of the groups allowed to access the information protected by the authentication filter.

type

This attribute is an editable StringArrayAttribute

default value

This attribute defaults to **null**.

Jigsaw Team

$Id: org.w3c.jigsaw.auth.GenericAuthFilter.html,v 1.3 1998/03/27 08:15:08 bmahe Exp $

```
// HTTPFrame.java
// $Id: HTTPFrame.html,v 1.4 1998/07/09 13:38:36 benoit Exp $
// (c) COPYRIGHT MIT and INRIA, 1997.
// Please first read the full copyright statement in file COPYRIGHT.html

package org.w3c.jigsaw.frames;

import java.io.*;
import java.net.*;
import java.util.*;

import org.w3c.tools.codec.*;
import org.w3c.tools.sorter.*;
import org.w3c.tools.resources.*;
import org.w3c.tools.resources.event.*;
import org.w3c.jigsaw.http.* ;
import org.w3c.jigsaw.html.* ;
import org.w3c.www.mime.*;
import org.w3c.www.http.*;
import org.w3c.tools.crypt.Md5;

import org.w3c.tools.resources.ProtocolException;
import org.w3c.tools.resources.NotAProtocolException;

/**
 * Default class to handle the HTTP protocol, manage FileResource and
 * DirectoryResource.
 */
public class HTTPFrame extends ProtocolFrame {

    /**
     * The special class of filter.
     */
    protected static Class filterClass = null;

    /**
     * Condition check return code - Condition existed but failed.
     */
    public static final int COND_FAILED = 1;
    /**
     * Condition check return code - Condition existed and succeeded.
     */
    public static final int COND_OK = 2;

    // Methods allowed by that class in general:
    protected static HttpTokenList _allowed        = null;
    private   static HttpTokenList _put_allowed    = null;
    private   static HttpTokenList _browse_allowed = null;
    private   static HttpTokenList _accept_ranges  = null;
```

```
    static {
        // allowed shares _allowed value, that's a feature.
        String str_allowed[] = { "GET", "HEAD", "OPTIONS", "TRACE"};
        _allowed = HttpFactory.makeStringList(str_allowed);
        String str_allow[] = { "HEAD" , "GET" , "PUT" , "OPTIONS", "TRACE" };
        _put_allowed = HttpFactory.makeStringList(str_allow);
        String str_all[] = { "HEAD" , "GET" , "BROWSE" , "OPTIONS","TRACE" };
        _browse_allowed = HttpFactory.makeStringList(str_allow);
        String accept_ranges[] = { "bytes" };
        _accept_ranges = HttpFactory.makeStringList(accept_ranges);
    }
    // Methods allowed by instances of that class in particular:
    protected        HttpTokenList  allowed = _allowed;


    /**
     * Attributes index - The index for the quality attribute.
     */
    protected static int ATTR_QUALITY = -1 ;
    /**
     * Attribute index - The index for the title attribute.
     */
    protected static int ATTR_TITLE = -1 ;
    /**
     * Attribute index - The index for the content languages attribute.
     */
    protected static int ATTR_CONTENT_LANGUAGE = -1 ;
    /**
     * Attribute index - The index for the content encodings attribute.
     */
    protected static int ATTR_CONTENT_ENCODING = -1 ;
    /**
     * Attribute index - The index for the content type attribute.
     */
    protected static int ATTR_CONTENT_TYPE = -1 ;
    /**
     * Attribute index - The index for the content length attribute.
     */
    protected static int ATTR_CONTENT_LENGTH = -1 ;
    /**
     * Attribute index - The icon (if any) associated to the resource.
     */
    protected static int ATTR_ICON = -1 ;
    /**
     * Attribute index - Max age: the maximum drift allowed from reality.
     */
    protected static int ATTR_MAXAGE = -1 ;
    /**
     * Attribute index - Send MD5 Digest: the md5 digest of the resource sent
     */
    protected static int ATTR_MD5 = -1;
```

```
    //
    // Attribute relative to FileResource
    //

    /**
     * Attribute index - Do we allow PUT method on this file.
     */
    protected static int ATTR_PUTABLE = -1 ;

    //
    // Attribute relative to DirectoryResource
    //

    /**
     * Attribute index - The index for our relocate attribute.
     */
    protected static int ATTR_RELOCATE = -1 ;
    /**
     * Attribute index - our index resource name.
     */
    protected static int ATTR_INDEX = -1 ;
    /**
     * Attribute index - The icon directory to use in dir listing.
     */
    protected static int ATTR_ICONDIR = -1 ;
    /**
     * Attribute index - Allow the GNN browse method.
     */
    protected static int ATTR_BROWSABLE = -1 ;
    /**
     * Attribute index - Style sheet for directory listing
     */
    protected static int ATTR_STYLE_LINK = -1 ;

    static {
        Attribute a   = null ;
        Class     cls = null ;

        try {
            filterClass =
                Class.forName("org.w3c.tools.resources.ResourceFilter");
        } catch (Exception ex) {
            throw new RuntimeException("No ResourceFilter class found.");
        }

        // Get a pointer to our class:
        try {
            cls = Class.forName("org.w3c.jigsaw.frames.HTTPFrame") ;
        } catch (Exception ex) {
```

```
            ex.printStackTrace() ;
            System.exit(1) ;
        }
        // The quality attribute:
        a = new DoubleAttribute("quality"
                                , new Double(1.0)
                                , Attribute.EDITABLE);
        ATTR_QUALITY = AttributeRegistry.registerAttribute(cls, a) ;
        // The title attribute:
        a = new StringAttribute("title"
                                , null
                                , Attribute.EDITABLE) ;
        ATTR_TITLE = AttributeRegistry.registerAttribute(cls, a) ;
        // The content language attribute:
        a = new LanguageAttribute("content-language"
                                  , null
                                  , Attribute.EDITABLE) ;
        ATTR_CONTENT_LANGUAGE = AttributeRegistry.registerAttribute(cls,a);
        // The content encoding attribute:
        a = new EncodingAttribute("content-encoding"
                                  , null
                                  , Attribute.EDITABLE) ;
        ATTR_CONTENT_ENCODING = AttributeRegistry.registerAttribute(cls,a);
        // The content type attribute:
        a = new MimeTypeAttribute("content-type"
                                  , MimeType.TEXT_PLAIN
                                  , Attribute.EDITABLE) ;
        ATTR_CONTENT_TYPE = AttributeRegistry.registerAttribute(cls,a);
        // The content length attribute:
        a = new IntegerAttribute("content-length"
                                 , null
                                 , Attribute.COMPUTED);
        ATTR_CONTENT_LENGTH = AttributeRegistry.registerAttribute(cls,a);
        // The icon attribute:
        a = new StringAttribute("icon"
                                , null
                                , Attribute.EDITABLE) ;
        ATTR_ICON = AttributeRegistry.registerAttribute(cls, a) ;
        // The max age attribute (in ms)
        a = new LongAttribute("maxage"
                              , null
                              , Attribute.EDITABLE) ;
        ATTR_MAXAGE = AttributeRegistry.registerAttribute(cls, a) ;
        // Should we send MD5 digest?
        a = new BooleanAttribute("send-md5"
                                 , Boolean.FALSE
                                 , Attribute.EDITABLE);
        ATTR_MD5 = AttributeRegistry.registerAttribute(cls, a) ;

        //
        // Attribute relative to a FileResource
```

```
        //

        // The putable flag:
        a = new BooleanAttribute("putable"
                                 , Boolean.FALSE
                                 , Attribute.EDITABLE) ;
        ATTR_PUTABLE = AttributeRegistry.registerAttribute(cls, a) ;

        //
        // Attribute relative to a DirectoryResource
        //

        //Should we relocate invalid request to this directory ?
        a = new BooleanAttribute("relocate"
                                 , Boolean.TRUE
                                 , Attribute.EDITABLE);
        ATTR_RELOCATE = AttributeRegistry.registerAttribute(cls, a) ;
        // Our index resource name (optional).
        a = new StringAttribute("index"
                                , null
                                , Attribute.EDITABLE) ;
        ATTR_INDEX = AttributeRegistry.registerAttribute(cls, a) ;
        // Our icon directory.
        a = new StringAttribute("icondir"
                                , null
                                , Attribute.EDITABLE) ;
        ATTR_ICONDIR = AttributeRegistry.registerAttribute(cls,a);
        // The browsable flag:
        a = new BooleanAttribute("browsable"
                                 , Boolean.FALSE
                                 , Attribute.EDITABLE) ;
        ATTR_BROWSABLE = AttributeRegistry.registerAttribute(cls, a) ;
        // The style sheet attribute:
        a = new StringAttribute("style-sheet-link"
                                , null
                                , Attribute.EDITABLE) ;
        ATTR_STYLE_LINK = AttributeRegistry.registerAttribute(cls, a) ;
    }

    protected DirectoryResource dresource = null;
    protected FileResource      fresource = null;

    public void registerResource(FramedResource resource) {
        super.registerResource(resource);
        if (resource instanceof FileResource)
            fresource = (FileResource) resource;
        else if (resource instanceof DirectoryResource)
            dresource = (DirectoryResource) resource;
    }

    public FileResource getFileResource() {
```

```
    return fresource;
}

public DirectoryResource getDirectoryResource() {
    return dresource;
}

/**
 * use this one instead of registerResource if the resource type
 * doesn't matter or if this is not a file or a directory resource.
 * In subclasses you should have to do that:
 * <pre>
 *  public void registerResource(FramedResource resource) {
 *   super.registerOtherResource(resource);
 *  }
 * </pre>
 * @param the resource to register.
 */
public void registerOtherResource(FramedResource resource) {
    super.registerResource(resource);
    dresource = null;
    fresource = null;
}

// The HTTPResource keeps a cache of ready to use Http values. This
// allows to save converting to/from wire rep these objects. Not
// much CPU time, but also memory is spared.
HttpMimeType  contenttype     = null;
HttpInteger   contentlength   = null;
HttpDate      lastmodified    = null;
HttpTokenList contentencoding = null;
HttpTokenList contentlanguage = null;

// The Http entity tag for this resource (for FileResource only)
HttpEntityTag etag   = null;
// the MD5 digest for this resource (for FileResource only)
HttpString md5Digest = null;

/**
 * Get this resource's help url.
 * @return An URL, encoded as a String, or <strong>null</strong> if not
 * available.
 */

public String getHelpURL() {
    httpd server = (httpd) getServer();
    if ( server == null )
        return null;
    String docurl = server.getDocumentationURL();
    if ( docurl == null )
        return null;
```

```
        return docurl + "/" + getClass().getName() + ".html";
    }

    /**
     * Get the help URL for that resource's attribute.
     * @param topic The topic (can be an attribute name, or a property, etc).
     * @return A String encoded URL, or <strong>null</strong>.
     */

    public String getHelpURL(String topic) {
        httpd server = (httpd) getServer();
        if ( server == null )
            return null;
        String docurl = server.getDocumentationURL();
        if ( docurl == null )
            return null;
        Class defines = AttributeRegistry.getAttributeClass(getClass(), topic);
        if ( defines != null )
            return docurl + "/" + defines.getName() + ".html";
        return null;
    }

    /**
     * give the md5 digest from cache or calculate it
     * @return the HttpString version of the digest
     */

    private HttpString getMd5Digest() {
        if (md5Digest != null)
            return md5Digest;
        // not found, compute it if necessary!
        Resource r = getResource();
        if (r instanceof FileResource) {
            try {
                Md5 md5 = new Md5 (
                    new FileInputStream(((FileResource)r).getFile()));
                String s = null;
                try {
                    byte b[] = md5.getDigest();
                    Base64Encoder b64;
                    ByteArrayOutputStream bos = new ByteArrayOutputStream();
                    b64 = new Base64Encoder(new ByteArrayInputStream(b), bos);
                    b64.process();
                    s = bos.toString();
                    md5Digest = HttpFactory.makeString(s);
                } catch (Exception mdex) {
                    // error, set it to null
                    md5Digest = null;
                }
                return md5Digest;
            } catch (FileNotFoundException ex) {
```

```
                // silent fail
                md5Digest = null;
            }
        }
        return null;
    }

    /**
     * Listen its resource.
     */
    public void attributeChanged(AttributeChangedEvent evt) {
        super.attributeChanged(evt);
        String name = evt.getAttribute().getName();
        if (name.equals("file-stamp")) {
            etag = null;
            lastmodified = null;
            md5Digest = null;
        } else if (name.equals("file-length")) {
            setValue(ATTR_CONTENT_LENGTH, evt.getNewValue());
        } else if (name.equals("last-modified")) {
            setValue(ATTR_LAST_MODIFIED, evt.getNewValue());
        } else {
            lastmodified = null;
        }
    }

    /**
     * Catch setValue, to maintain cached header values correctness.
     * @param idx The index of the attribute to be set.
     * @param value The new value for the attribute.
     */

    public synchronized void setValue(int idx, Object value) {
        super.setValue(idx, value);
        if (idx == ATTR_CONTENT_TYPE)
            contenttype = null;
        if (idx == ATTR_CONTENT_LENGTH)
            contentlength = null;
        if ( idx == ATTR_CONTENT_ENCODING )
            contentencoding = null;
        if ( idx == ATTR_CONTENT_LANGUAGE )
            contentlanguage = null;
        if ( idx == ATTR_PUTABLE) {
            if (fresource != null) {
                if (value == Boolean.TRUE)
                    allowed = _put_allowed;
                else
                    allowed = _allowed;
            }
        }
        if (idx == ATTR_MD5) {
```

```
            md5Digest = null; // reset the digest state
        }
        // Any attribute setting modifies the last modified time:
        lastmodified = null;
    }

    /**
     * Get the full URL for that resource.
     * @return An URL instance.
     */
    public URL getURL(Request request) {
        try {
            return new URL(request.getURL(), resource.getURLPath());
        } catch (MalformedURLException ex) {
            throw new RuntimeException("unable to build "+
                                       getURLPath()+
                                       " full URL, from server "+
                                       getServer().getURL());
        }
    }

    /**
     * Get this resource quality.
     * @return The resource quality, or some negative value if not defined.
     */

    public double getQuality() {
        return getDouble(ATTR_QUALITY, -1.0) ;
    }

    /**
     * Get this resource title.
     * @return This resource's title, or <strong>null</strong> if not
     *    defined.
     */

    public String getTitle() {
        return getString(ATTR_TITLE, null) ;
    }

    /**
     * Get this resource content language.
     * Language are stored as a comma separated String of tokens.
     * @return A comma separated string of language tokens, or
     *    <strong>null</strong> if undefined.
     */

    public String getContentLanguage() {
        return (String) getValue(ATTR_CONTENT_LANGUAGE, null) ;
    }
```

```
    /**
     * Get this resource content encoding.
     * The content encoding of a resource is stored as a comma separated
     * list of tokens (as decribed in the Content_encoding header of the
     * HTTP specification, and in the order they should appear in the header).
     * @return A string of comma separated encoding tokens, or
     *    <strong>null</strong> if not defined.
     */

    public String getContentEncoding() {
        String def = (String) attributes[ATTR_CONTENT_ENCODING].getDefault();
        System.out.println("Default ["+def+"]");
        String s =  (String) getString (ATTR_CONTENT_ENCODING, def) ;
        System.out.println("Defaulted to ["+s+"]");

        return (String) getString (ATTR_CONTENT_ENCODING, def) ;
    }

    /**
     * Get this resource content type.
     * @return An instance of MIMEType, or <strong>null</strong> if not
     *    defined.
     */

    public MimeType getContentType() {
        return (MimeType) getValue(ATTR_CONTENT_TYPE, null);
    }

    /**
     * Get this resource content length.
     * @return The resource content length, or <strong>-1</strong> if not
     *    defined.
     */

    public int getContentLength() {
        return getInt(ATTR_CONTENT_LENGTH, -1) ;
    }

    /**
     * Get this resource's icon.
     */

    public String getIcon() {
        return getString(ATTR_ICON, null) ;
    }

    /**
     * Get this resource's max age.
     * The max age of a resource indicates how much drift is allowed between
     * the physicall version of the resource, and any in-memory cached version
     * of it.
```

```
     * <p>The max age attribute is a long number giving the number of
     * milliseconds of allowed drift.
     */

    public long getMaxAge() {
        return getLong(ATTR_MAXAGE, (long) -1) ;
    }

    //
    // Relative to FileResource ...
    //

    /**
     * Does this resource support byte ranges.
     */
    protected boolean acceptRanges = false;

    /**
     * Get the PUT'able flag (are we allow to PUT to the resource ?)
     */
    public boolean getPutableFlag() {
        return getBoolean(ATTR_PUTABLE, false) ;
    }

    /**
     * Do we send the MD5 digest?
     */
    public boolean getMD5Flag() {
        return getBoolean(ATTR_MD5, false) ;
    }

    /**
     * handles a Range Request
     * @param request, the request
     * @param r, the HttpRange
     * @return a Reply if range is valid, or null if there is a change in the
     * resource, or if the HttpRange is not valid ( 4-2, for example).
     */

    public Reply handleRangeRequest(Request request, HttpRange r)
        throws ProtocolException
    {
        // Should we check against a IfRange header ?
        HttpEntityTag t = request.getIfRange();

        if ( t != null ) {
            if (t.isWeak() || ! t.getTag().equals(etag.getTag()))
                return null;
        }
        // Check the range:
        int cl = getContentLength();
```

```
        int fb = r.getFirstPosition();
        int lb = r.getLastPosition();
        int sz;

        if (fb > cl-1) { // first byte already out of range
            HttpContentRange cr = HttpFactory.makeContentRange("bytes", 0,
                                                               cl - 1, cl);
            Reply rr;
            rr = createDefaultReply(request,
                                    HTTP.REQUESTED_RANGE_NOT_SATISFIABLE);
            rr.setContentLength(-1);
            rr.setHeaderValue(rr.H_CONTENT_RANGE, cr);
            if (getMD5Flag())
                rr.setContentMD5(null);
            return rr;
        }

        if ((fb < 0) && (lb >= 0)) { // ex: bytes=-20 final 20 bytes
            if (lb >= cl)   // cut the end
                lb = cl;
            sz = lb;
            fb = cl - lb;
            lb = cl - 1;
        } else if (lb < 0) {  // ex: bytes=10- the last size - 10
            lb = cl-1;
            sz = lb-fb+1;
        } else {              // ex: bytes=10-20
            if (lb >= cl)  // cut the end
                lb = cl-1;
            sz = lb-fb+1;
        }
        if ((fb < 0) || (lb < 0) || (fb <= lb)) {
            HttpContentRange cr = null;
            fb = (fb < 0) ? 0 : fb;
            lb = ((lb > cl) || (lb < 0)) ? cl : lb;
            cr = HttpFactory.makeContentRange("bytes", fb, lb, cl);
            // Emit reply:
            Reply rr = createDefaultReply(request, HTTP.PARTIAL_CONTENT);
            // FIXME check for MD5 of only the subpart
            try { // create the MD5 for the subpart
                if (getMD5Flag()) {
                    String s = null;
                    try {
                        ByteRangeOutputStream br;
                        br = new ByteRangeOutputStream(fresource.getFile(),
                                                       fb, lb+1);
                        Md5 md5 = new Md5 (br);
                        byte b[] = md5.getDigest();
                        Base64Encoder b64;
                        ByteArrayOutputStream bs = new ByteArrayOutputStream();
                        b64 = new Base64Encoder(new ByteArrayInputStream(b),
```

```
                                            bs);
                    b64.process();
                    s = bs.toString();
                } catch (Exception md_ex) {
                    // default to null, no action here then
                }
                if (s == null)
                    rr.setContentMD5(null);
                else
                    rr.setContentMD5(s);
            }
            rr.setContentLength(sz);
            rr.setHeaderValue(rr.H_CONTENT_RANGE, cr);
            rr.setStream(new ByteRangeOutputStream(fresource.getFile(),
                                                   fb,
                                                   lb+1));
            return rr;
        } catch (IOException ex) {
        }
    }
    return null;
}

//
// Relative to DirectoryResource ...
//

/**
 * Get this class browsable flag.
 */
public boolean getBrowsableFlag() {
    return getBoolean (ATTR_BROWSABLE, false) ;
}

/**
 * Get this frame style sheet link
 */
public String getStyleSheetURL() {
    return getString (ATTR_STYLE_LINK, null);
}

/**
 * Our current (cached) directory listing.
 */
protected HtmlGenerator listing = null ;
/**
 * The time at which we generated the directory index.
 */
protected long listing_stamp = -1 ;

private String getUnextendedName(String name) {
```

```
        int strlen = name.length() ;
        for (int i = 0 ; i < strlen ; i++) {
            // FIXME: Should use the system props to get the right sep
            if ( name.charAt(i) == '.' ) {
                if ( i == 0 )
                    return null ;
                return name.substring(0, i) ;
            }
        }
        return null ;
    }

    /**
     * Get the optional icon directory.
     */
    public String getIconDirectory() {
        return getString(ATTR_ICONDIR, "/icons") ;
    }



    /**
     * Should we relocate invalid requests to this directory.
     * @return A boolean <strong>true</strong> if we should relocate.
     */
    public boolean getRelocateFlag() {
        return getBoolean(ATTR_RELOCATE, true) ;
    }

    /**
     * Get the optinal index name for this directory listing.
     * @return The name of the resource responsible to list that container.
     */
    public String getIndex() {
        return (String) getValue(ATTR_INDEX, null) ;
    }

    /**
     * Add our own Style Sheet to the HtmlGenerator.
     * @param g The HtmlGenerator.
     */
    protected void addStyleSheet(HtmlGenerator g) {
        // Add style link
        String css_url = getStyleSheetURL();
        if (css_url != null) {
            g.addLink( new HtmlLink("STYLESHEET",
                                    css_url,
                                    MimeType.TEXT_CSS));
        }
    }
```

```
    /**
     * Reply with an HTML doc listing the resources of this directory.
     * This function takes special care not to regenerate a directory listing
     * when one is available. It also caches the date of the directory
     * listing, so that it can win big with NOT_MODIFIED.
     * <p>Using a modem, I know that each place I can reply with an
     * NOT_MODIFIED, <strong>is</strong> a big win.
     * @param request The request to handle.
     * @exception ProtocolException If processsing the request failed.
     */

    public synchronized Reply getDirectoryListing(Request request)
        throws ProtocolException, NotAProtocolException
    {
        if (dresource == null)
            throw new NotAProtocolException("this frame is not attached to a "+
                                            "DirectoryResource. ("+
                                            resource.getIdentifier()+")");
        // delete us if the directory was deleted
        if (! dresource.getDirectory().exists()) {
            //delete us and emit an error
            String msg = dresource.getIdentifier()+
                ": deleted, removing the DirectoryResource";
            getServer().errlog(dresource, msg);
            try {
                dresource.delete();
            } catch (MultipleLockException ex) {
            }
            // Emit an error back:
            Reply error = request.makeReply(HTTP.NOT_FOUND) ;
            error.setContent ("<h1>Document not found</h1>"+
                              "<p>The document "+
                              request.getURL()+
                              " is indexed but not available."+
                              "<p>The server is misconfigured.") ;
            throw new HTTPException (error) ;
        }
        // Have we already an up-to-date computed a listing ?
        if ((listing == null)
            || (dresource.getDirectory().lastModified() > listing_stamp)
            || (dresource.getLastModified() > listing_stamp)
            || (getLastModified() > listing_stamp)) {

            Class http_class = null;
            try {
                http_class = Class.forName("org.w3c.jigsaw.frames.HTTPFrame");
            } catch (ClassNotFoundException ex) {
                http_class = null;
            }

            Enumeration   enum      =
```

```
            dresource.enumerateResourceIdentifiers() ;
        Vector        resources = Sorter.sortStringEnumeration(enum) ;
        HtmlGenerator g         =
            new HtmlGenerator("Directory of "+
                              dresource.getIdentifier());
        // Add style link
        addStyleSheet(g);
        g.append("<h1>"+dresource.getIdentifier()+"</h1>");
        // Link to the parent, when possible:
        if ( dresource.getParent() != null )
            g.append("<p><a href=\"..\">Parent</a><br>");
        // List the children:
        for (int i = 0 ; i < resources.size() ; i++) {
            String       name     = (String) resources.elementAt(i);
            ResourceReference rr = null;
            rr = dresource.lookup(name);
            FramedResource resource = null;
            if (rr != null) {
                try {
                    resource = (FramedResource) rr.lock();
                    HTTPFrame itsframe = null;
                    if (http_class != null)
                        itsframe =
                            (HTTPFrame) resource.getFrame(http_class);
                    if (itsframe != null) {
                        // Icon first, if available
                        String icon = itsframe.getIcon() ;
                        if ( icon != null )
                            g.append("<img src=\""+
                                     getIconDirectory() +"/" + icon+
                                     "\">");
                        // Resource's name with link:
                        g.append("<a href=\""
                                 , resource.getURLPath()
                                 , "\">"+name+"</a>");
                        // resource's title, if any:
                        String title = itsframe.getTitle();
                        if ( title != null )
                            g.append(" "+title);
                        g.append("<br>");
                    } else {
                        // Resource's name with link:
                        g.append(name+" (<i>Not available via HTTP.</i>)");
                        g.append("<br>");
                    }
                } catch (InvalidResourceException ex) {
                    g.append(name+
                             " cannot be loaded (server misconfigured)");
                    g.append("<br>");
                    continue;
                } finally {
```

```
                    rr.unlock();
                }
            }
        }
        g.close() ;
        listing_stamp = getLastModified() ;
        listing       = g ;
    } else if ( checkIfModifiedSince(request) == COND_FAILED ) {
        // Is it an IMS request ?
        Reply reply = createDefaultReply(request, HTTP.NOT_MODIFIED) ;
        reply.setContentMD5(null);
        return reply;
    }
    // New content or need update:
    Reply reply = createDefaultReply(request, HTTP.OK) ;
    reply.setLastModified(listing_stamp) ;
    reply.setStream(listing) ;
    // check MD5
    return reply ;
}

//
// Commom part.
//



/**
 * Update the cached headers value.
 * Each resource maintains a set of cached values for headers, this
 * allows for a nice sped-up in headers marshalling, which - as the
 * complexity of the protocol increases - becomes a bottleneck.
 */

protected void updateCachedHeaders() {
    // Precompute a set of header values to keep by:
    if ( contenttype == null )
        contenttype = HttpFactory.makeMimeType(getContentType());
    if (contentlength == null) {
        int cl = -1;
        if (fresource != null)
            cl = fresource.getFileLength();
        if ( cl >= 0 ) {
            setValue(ATTR_CONTENT_LENGTH, new Integer(cl));
            contentlength = HttpFactory.makeInteger(cl);
        }
    }
    if ( lastmodified == null ) {
        long lm = getLastModified();
        if ( lm > 0 )
            lastmodified = HttpFactory.makeDate(getLastModified());
```

```
        }
        if (definesAttribute(ATTR_CONTENT_ENCODING) &&(contentencoding==null))
            contentencoding = HttpFactory.makeStringList(getContentEncoding());
        if (definesAttribute(ATTR_CONTENT_LANGUAGE) &&(contentlanguage==null))
            contentlanguage = HttpFactory.makeStringList(getContentLanguage());

        if (fresource != null) {
            // We only take car eof etag here:
            if ( etag == null ) {
                long lstamp = fresource.getFileStamp();
                if ( lstamp >= 0L ) {
                    String soid  = Integer.toString(getOid(), 32);
                    String stamp = Long.toString(lstamp, 32);
                    etag = HttpFactory.makeETag(false, soid+":"+stamp);
                }
            }
            if (getMD5Flag() && (md5Digest == null)) {
                getMd5Digest();
            }
        }
    }

    /**
     * Create a reply to answer to request on this file.
     * This method will create a suitable reply (matching the given request)
     * and will set all its default header values to the appropriate
     * values.
     * @param request The request to make a reply for.
     * @return An instance of Reply, suited to answer this request.
     */

    public Reply createDefaultReply(Request request, int status) {
        Reply reply = request.makeReply(status);
        updateCachedHeaders();
        if ( status != HTTP.NOT_MODIFIED ) {
            if ( contentlength != null )
                reply.setHeaderValue(Reply.H_CONTENT_LENGTH, contentlength);
            if ( contenttype != null )
                reply.setHeaderValue(Reply.H_CONTENT_TYPE, contenttype);
            if ( lastmodified != null )
                reply.setHeaderValue(Reply.H_LAST_MODIFIED, lastmodified);
            if ( contentencoding != null ) {
System.out.println("Content Encoding [" + contentencoding +"]");
                reply.setHeaderValue(Reply.H_CONTENT_ENCODING,contentencoding);
            }
            if ( contentlanguage != null )
                reply.setHeaderValue(Reply.H_CONTENT_LANGUAGE,contentlanguage);

        }
        long maxage = getMaxAge();
        if ( maxage >= 0 ) {
```

```
            if (reply.getMajorVersion() >= 1 ) {
                if (reply.getMinorVersion() >= 1) {
                    reply.setMaxAge((int) (maxage / 1000));
                }
                // If max-age is zero, say what you mean:
                long expires = (System.currentTimeMillis()
                                + ((maxage == 0) ? -1000 : maxage));
                reply.setExpires(expires);
            }
        }
        // Set the date of the reply (round it to secs):
        reply.setDate((System.currentTimeMillis() / 1000L) * 1000L);

        if (fresource != null) {
            // Set the entity tag:
            if ( etag != null )
                reply.setHeaderValue(reply.H_ETAG, etag);
            if ( acceptRanges )
                reply.setHeaderValue(reply.H_ACCEPT_RANGES, _accept_ranges);
            if ( getMD5Flag()) {
                reply.setHeaderValue(reply.H_CONTENT_MD5, getMd5Digest());
            }
        }

        return reply;
    }

    /**
     * Check the <code>If-Match</code> condition of that request.
     * @param request The request to check.
     * @return An integer, either <code>COND_FAILED</cond> if condition
     * was checked, but failed, <code>COND_OK</code> if condition was checked
     * and succeeded, or <strong>0</strong> if the condition was not checked
     * at all (eg because the resource or the request didn't support it).
     */

    public int checkIfMatch(Request request) {
        if (fresource != null) {
            HttpEntityTag tags[] = request.getIfMatch();
            if ( tags != null ) {
                // Good, real validators in use:
                if ( etag != null ) {
                    // Note: if etag is null this means that the resource has
                    // changed and has not been even emited since then...
                    for (int i = 0 ; i < tags.length ; i++) {
                        HttpEntityTag t = tags[i];
                        if ((!t.isWeak()) && t.getTag().equals(etag.getTag()))
                            return COND_OK;
                    }
                }
                return COND_FAILED;
```

```
            }
        }
        return 0;
    }

    /**
     * Check the <code>If-None-Match</code> condition of that request.
     * @param request The request to check.
     * @return An integer, either <code>COND_FAILED</cond> if condition
     * was checked, but failed, <code>COND_OK</code> if condition was checked
     * and succeeded, or <strong>0</strong> if the condition was not checked
     * at all (eg because the resource or the request didn't support it).
     */

    public int checkIfNoneMatch(Request request) {
        if (fresource != null) {
            // Check for an If-None-Match conditional:
            HttpEntityTag tags[] = request.getIfNoneMatch();
            if ( tags != null ) {
                if ( etag == null )
                    return COND_OK;
                for (int i = 0 ; i < tags.length ; i++) {
                    HttpEntityTag t = tags[i];
                    if (( ! t.isWeak()) && t.getTag().equals(etag.getTag()))
                        return COND_FAILED;
                }
                return COND_OK;
            }
        }
        return 0;
    }

    /**
     * Check the <code>If-Modified-Since</code> condition of that request.
     * @param request The request to check.
     * @return An integer, either <code>COND_FAILED</cond> if condition
     * was checked, but failed, <code>COND_OK</code> if condition was checked
     * and succeeded, or <strong>0</strong> if the condition was not checked
     * at all (eg because the resource or the request didn't support it).
     */

    public int checkIfModifiedSince(Request request) {
        // Check for an If-Modified-Since conditional:
        long ims = request.getIfModifiedSince();
        if (dresource != null) {
            if (ims >= 0)
                return (((listing_stamp > 0) && (listing_stamp - 1000 <= ims))
                        ? COND_FAILED
                        : COND_OK);
        } else if (fresource != null) {
            long cmt = getLastModified();
```

```
            if ( ims >= 0 )
                return ((cmt > 0) && (cmt - 1000 <= ims))
                    ? COND_FAILED : COND_OK;
        }
        return 0;
    }

    /**
     * Check the <code>If-Unmodified-Since</code> condition of that request.
     * @param request The request to check.
     * @return An integer, either <code>COND_FAILED</cond> if condition
     * was checked, but failed, <code>COND_OK</code> if condition was checked
     * and succeeded, or <strong>0</strong> if the condition was not checked
     * at all (eg because the resource or the request didn't support it).
     */

    public int checkIfUnmodifiedSince(Request request) {
        if (fresource != null) {
            // Check for an If-Unmodified-Since conditional:
            long iums = request.getIfUnmodifiedSince();
            long cmt = getLastModified();
            if ( iums >= 0 )
                return ((cmt > 0) && (cmt - 1000) >= iums)
                    ? COND_FAILED : COND_OK;
        }
        return 0;
    }

    public boolean lookup(LookupState ls, LookupResult lr)
        throws ProtocolException
    {
        RequestInterface req = ls.getRequest();
        if (! checkRequest(req))
            return false;
        if (lookupFilters(ls,lr))
            return true;
        return lookupResource(ls,lr);
    }

    protected boolean lookupResource(LookupState ls, LookupResult lr)
        throws ProtocolException
    {
        if (fresource != null) {
            return lookupFile(ls,lr);
        } else if (dresource != null) {
            return lookupDirectory(ls,lr);
        } else {
            return lookupOther(ls,lr);
        }
    }
```

```
    protected boolean lookupFilters(LookupState ls, LookupResult lr)
        throws ProtocolException
    {
        ResourceFilter filters[] = getFilters();
        if ( filters != null ) {
            // Mark filters, for them to be called at outgoing time:
            lr.addFilters(filters);
            // Some clever filter around ?
            for (int i = 0 ; i < filters.length ; i++) {
                if ( filters[i] == null )
                    continue;
                if ( filters[i].lookup(ls, lr) )
                    return true;
            }
        }
        return false;
    }

    protected boolean lookupDirectory(LookupState ls, LookupResult lr)
        throws ProtocolException
    {
        // Give a chance to our super-class to run its own lookup scheme:
        // do we have to create a resource (PUT) ?
        if ((ls.hasMoreComponents()) && getPutableFlag()) {
            Request request = (Request) ls.getRequest() ;
            if ((request == null) || request.getMethod().equals("PUT")) {
                // We might well want to create a resource:
                String            name = ls.peekNextComponent() ;
                ResourceReference rr   = dresource.lookup(name);
                if ((rr == null) && (dresource.getExtensibleFlag())) {
                    if (ls.countRemainingComponents() == 1)
                        rr = dresource.createResource(name, request);
                    else
                        rr = dresource.createDirectoryResource(name);
                    if (rr == null) {
                        Reply error =
                            request.makeReply(HTTP.UNSUPPORTED_MEDIA_TYPE);
                        error.setContent(
                            "Failed to create resource "+
                            name +" : "+
                            "Unable to create the appropriate file:"+
                            request.getURLPath()+
                            " this media type is not supported");
                        throw new HTTPException (error);
                    }
                } else if (rr == null) {
                    Reply error = request.makeReply(HTTP.FORBIDDEN) ;
                    error.setContent("You are not allowed to create resource "+
                                     name +" : "+
                                     dresource.getIdentifier()+
                                     " is not extensible.");
```

```
                throw new HTTPException (error);
            }
        }
    }
    if ( super.lookup(ls, lr) ) {
        if ( ! ls.isDirectory() && ! ls.isInternal() ) {
            // The directory lookup URL doesn't end with a slash:
            Request request = (Request)ls.getRequest() ;
            if ( request == null ) {
                lr.setTarget(null);
                return true;
            }
            URL url = null;
            try {
                url = (ls.hasRequest()
                       ? getURL(request)
                       : new URL(getServer().getURL(),
                                 resource.getURLPath()));
            } catch (MalformedURLException ex) {
                getServer().errlog(this, "unable to build full URL.");
                throw new HTTPException("Internal server error");
            }
            String msg = "Invalid requested URL: the directory resource "+
                " you are trying to reach is available only through "+
                " its full URL: <a href=\""+
                url + "\">" + url + "</a>.";
            if ( getRelocateFlag() ) {
                // Emit an error (with reloc if allowed)
                Reply reloc = request.makeReply(HTTP.FOUND);
                reloc.setContent(msg) ;
                reloc.setLocation(url);
                lr.setTarget(null);
                lr.setReply(reloc);
                return true;
            } else {
                Reply error = request.makeReply(HTTP.NOT_FOUND) ;
                error.setContent(msg) ;
                lr.setTarget(null);
                lr.setReply(error);
                return true;
            }
        } else if ( ! ls.isInternal() ) {
            // return the index file.
            String index = getIndex();
            if ( index != null && index.length() > 0) {
                DirectoryResource dir = (DirectoryResource) resource;
                ResourceReference rr = dir.lookup(index);
                if (rr != null) {
                    try {
                        FramedResource rindex = (FramedResource) rr.lock();
                        return rindex.lookup(ls,lr);
```

```
                    } catch (InvalidResourceException ex) {
                    } finally {
                        rr.unlock();
                    }
                }
            }
        }
        return true;
    }
    return false;
}

protected boolean lookupFile(LookupState ls, LookupResult lr)
    throws ProtocolException
{
    return super.lookup(ls,lr);
}

protected boolean lookupOther(LookupState ls, LookupResult lr)
    throws ProtocolException
{
    return super.lookup(ls,lr);
}

public boolean checkRequest(RequestInterface request) {
    return ((request == null)
            ? true
            :(request instanceof org.w3c.jigsaw.http.Request));
}

protected ReplyInterface performFrames(RequestInterface request)
    throws ProtocolException, NotAProtocolException
{
    return super.performFrames(request);
}

public ReplyInterface perform(RequestInterface req)
    throws ProtocolException, NotAProtocolException
{
    ReplyInterface repi = super.perform(req);
    if (repi != null)
        return repi;

    if (! checkRequest(req))
        return null;

    Reply  reply  = null;
    Request request = (Request) req;
    String method = request.getMethod () ;
    // Perform the request:
    if ( method.equals("GET") ) {
```

```
            reply = get(request) ;
        } else if ( method.equals("HEAD") ) {
            reply = head(request) ;
        } else if ( method.equals("POST") ) {
            reply = post(request) ;
        } else if ( method.equals("PUT") ) {
            reply = put(request) ;
        } else if ( method.equals("OPTIONS") ) {
            reply = options(request);
        } else if ( method.equals("DELETE") ) {
            reply = delete(request) ;
        } else if ( method.equals("LINK") ) {
            reply = link(request) ;
        } else if ( method.equals("UNLINK") ) {
            reply = unlink(request) ;
        } else if ( method.equals("TRACE") ) {
            reply = trace(request) ;
        } else {
            reply = extended(request) ;
        }
        return reply;
    }

    /**
     * The default GET method.
     * @param request The request to handle.
     * @exception ProtocolException if processing the request failed.
     * @exception NotAProtocolException If the client instance controling the
     * request processing got a fatal error.
     */

    public Reply get(Request request)
        throws ProtocolException, NotAProtocolException
    {
        if (dresource != null) {
            // we manage a DirectoryResource
            return getDirectoryResource(request) ;
        } else if (fresource != null) {
            // we manage a FileResource
            return getFileResource(request);
        } else {
            return getOtherResource(request);
        }
    }

    protected Reply getOtherResource(Request request)
        throws ProtocolException, NotAProtocolException
    {
        // we don't manage this kind of resource
        Reply error = request.makeReply(HTTP.NOT_IMPLEMENTED) ;
        error.setContent("Method GET not implemented.") ;
```

```
        throw new HTTPException (error) ;
    }

    /**
     * Create the reply relative to the given file.
     * @param request the incomming request.
     * @return A Reply instance
     */
    protected Reply createFileReply(Request request)
        throws ProtocolException, NotAProtocolException
    {
        File file = fresource.getFile() ;
        Reply reply = null;
        // Check for a range request:
        HttpRange ranges[] = request.getRange();
        if ((ranges != null) && (ranges.length == 1)) {
            Reply rangereply = handleRangeRequest(request, ranges[0]);
            if ( rangereply != null )
                return rangereply;
        }
        // Default to full reply:
        reply = createDefaultReply(request, HTTP.OK) ;
        try {
            reply.setStream(new FileInputStream(file));
        } catch (IOException ex) {
            // I hate to have to loose time in tries
        }
        return reply ;
    }

    protected Reply getFileResource(Request request)
        throws ProtocolException, NotAProtocolException
    {
        if (fresource == null)
            throw new NotAProtocolException("this frame is not attached to a "+
                                            "FileResource. ("+
                                            resource.getIdentifier()+")");
        Reply reply = null;
        File file = fresource.getFile() ;
        fresource.checkContent();
        updateCachedHeaders();
        // Check validators:
        if ( checkIfMatch(request) == COND_FAILED ) {
            reply = request.makeReply(HTTP.PRECONDITION_FAILED);
            reply.setContent("Pre-conditions failed.");
            reply.setContentMD5(null);
            return reply;
        }
        if ( checkIfNoneMatch(request) == COND_FAILED ) {
            reply = createDefaultReply(request, HTTP.NOT_MODIFIED);
            reply.setContentMD5(null);
```

```
            return reply;
        }
        if ( checkIfModifiedSince(request) == COND_FAILED ) {
            reply = createDefaultReply(request, HTTP.NOT_MODIFIED);
            reply.setContentMD5(null);
            return reply;
        }
        if ( checkIfUnmodifiedSince(request) == COND_FAILED ) {
            reply = request.makeReply(HTTP.PRECONDITION_FAILED);
            reply.setContent("Pre-conditions failed.");
            reply.setContentMD5(null);
            return reply;
        }
        // Does this file really exists, if so send it back
        if ( file.exists() ) {
            return createFileReply(request);
        } else {
            // Delete the resource if parent is extensible:
            boolean extensible = false;
            ResourceReference rr = fresource.getParent();
            ResourceReference rrtemp = null;
            Resource p = null;
            while ( true ) {
                try {
                    if (rr == null)
                        break;
                    p = rr.lock();
                    if (p instanceof DirectoryResource) {
                        extensible =
                            ((DirectoryResource)p).getExtensibleFlag();
                        break;
                    }
                    rrtemp = p.getParent();
                } catch (InvalidResourceException ex) {
                    break;
                } finally {
                    if (rr != null)
                        rr.unlock();
                }
                rr = rrtemp;
            }
            if (extensible) {
                // The resource is indexed but has no file, emit an error
                String msg = fresource.getFile()+
                    ": deleted, removing the FileResource.";
                getServer().errlog(fresource, msg);
                try {
                    fresource.delete();
                } catch (MultipleLockException ex) {
                    Reply error = request.makeReply(HTTP.NOT_FOUND) ;
                    error.setContentMD5(null); // FIXME must compute it!
```

```
                error.setContent ("<h1>Document not found</h1>"+
                                  "<p>The document "+
                                  request.getURL()+
                                  " is indexed but not available."+
                                  "<p>"+ex.getMessage()+
                                  "<p>The server is misconfigured.") ;
                throw new HTTPException (error) ;
            }
        }
        // Emit an error back:
        Reply error = request.makeReply(HTTP.NOT_FOUND) ;
        error.setContentMD5(null);
        error.setContent ("<h1>Document not found</h1>"+
                          "<p>The document "+
                          request.getURL()+
                          " is indexed but not available."+
                          "<p>The server is misconfigured.") ;
        throw new HTTPException (error) ;
    }
}

protected Reply getDirectoryResource(Request request)
    throws ProtocolException, NotAProtocolException
{
    return getDirectoryListing(request) ;
}

/**
 * The default HEAD method replies does a GET and removes entity.
 * @param request The request to handle.
 * @exception ProtocolException Always thrown, to return a NOT_IMPLEMENTED
 *    error.
 * @exception NotAProtocolException If the client instance controling the
 * request processing got a fatal error.
 */

public Reply head(Request request)
    throws ProtocolException, NotAProtocolException
{
    if (dresource != null) {
        return headDirectoryResource(request);
    } else if (fresource != null) {
        return headFileResource(request);
    } else {
        return headOtherResource(request);
    }
}

protected Reply headOtherResource(Request request)
    throws ProtocolException, NotAProtocolException
{
```

```
        Reply reply = null;
        reply = getOtherResource(request) ;
        reply.setStream((InputStream) null);
        return reply;
    }

    protected Reply headDirectoryResource(Request request)
        throws ProtocolException, NotAProtocolException
    {
        Reply reply = null;
        reply = getDirectoryResource(request) ;
        reply.setStream((InputStream) null);
        return reply;
    }

    protected Reply headFileResource(Request request)
        throws ProtocolException, NotAProtocolException
    {
        if (fresource == null)
            throw new NotAProtocolException("this frame is not attached to a "+
                                            "FileResource. ("+
                                            resource.getIdentifier()+")");
        Reply reply = null;
        fresource.checkContent();
        updateCachedHeaders();
        // Conditional check:
        if ( checkIfMatch(request) == COND_FAILED ) {
            Reply r = request.makeReply(HTTP.PRECONDITION_FAILED);
            r.setContent("Pre-conditions failed.");
            return r;
        }
        if ( checkIfNoneMatch(request) == COND_FAILED )
            return createDefaultReply(request, HTTP.NOT_MODIFIED);
        if ( checkIfModifiedSince(request) == COND_FAILED )
            return createDefaultReply(request, HTTP.NOT_MODIFIED);
        if ( checkIfUnmodifiedSince(request) == COND_FAILED ) {
            Reply r = request.makeReply(HTTP.PRECONDITION_FAILED);
            r.setContent("Pre-conditions failed.");
            return r;
        }
        if (! fresource.getFile().exists()) {
            // Delete the resource if parent is extensible:
            boolean extensible = false;
            ResourceReference rr = fresource.getParent();
            ResourceReference rrtemp = null;
            Resource p = null;
            while ( true ) {
                try {
                    if (rr == null)
                        break;
                    p = rr.lock();
```

```
                if (p instanceof DirectoryResource)
                    extensible =
                        ((DirectoryResource)p).getExtensibleFlag();
                rrtemp = p.getParent();
            } catch (InvalidResourceException ex) {
                break;
            } finally {
                if (rr != null)
                    rr.unlock();
            }
            rr = rrtemp;
        }
        if (extensible) {
            // The resource is indexed but has no file, emit an error
            String msg = fresource.getFile()+
                ": deleted, removing the FileResource.";
            getServer().errlog(fresource, msg);
            try {
                fresource.delete();
            } catch (MultipleLockException ex) {
                Reply error = request.makeReply(HTTP.NOT_FOUND) ;
                error.setContent ("<h1>Document not found</h1>"+
                                  "<p>The document "+
                                  request.getURL()+
                                  " is indexed but not available."+
                                  "<p>"+ex.getMessage()+
                                  "<p>The server is misconfigured.") ;
                throw new HTTPException (error) ;
            }
        }
        // Emit an error back:
        Reply error = request.makeReply(HTTP.NOT_FOUND) ;
        error.setContent ("<h1>Document not found</h1>"+
                          "<p>The document "+
                          request.getURL()+
                          " is indexed but not available."+
                          "<p>The server is misconfigured.") ;
        throw new HTTPException (error) ;
    }
    return createDefaultReply(request, HTTP.OK);
}

/**
 * The default POST method replies with a not implemented.
 * @param request The request to handle.
 * @exception ProtocolException Always thrown, to return a NOT_IMPLEMENTED
 *    error.
 * @exception NotAProtocolException If the client instance controling the
 * request processing got a fatal error.
 */
```

```
    public Reply post(Request request)
        throws ProtocolException, NotAProtocolException
    {
        Reply error = request.makeReply(HTTP.NOT_ALLOWED) ;
        if ( allowed != null )
            error.setHeaderValue(Reply.H_ALLOW, allowed);
        error.setContent("Method POST not allowed on this resource.") ;
        throw new HTTPException (error) ;
    }

    /**
     * The default PUT method replies with a not implemented.
     * @param request The request to handle.
     * @exception ProtocolException Always thrown, to return a NOT_IMPLEMENTED
     *    error.
     * @exception NotAProtocolException If the client instance controling the
     * request processing got a fatal error.
     */

    public Reply put(Request request)
        throws ProtocolException, NotAProtocolException
    {
        if (fresource != null) {
            return putFileResource(request);
        } else {
            return putOtherResource(request);
        }
    }

    protected Reply putOtherResource(Request request)
        throws ProtocolException
    {
        Reply error = request.makeReply(HTTP.NOT_IMPLEMENTED) ;
        error.setContent("Method PUT not implemented.") ;
        throw new HTTPException (error) ;
    }

    protected Reply putFileResource(Request request)
        throws ProtocolException, NotAProtocolException
    {
        Reply reply = null;
        int status = HTTP.OK;
        fresource.checkContent();
        updateCachedHeaders();
        // Is this resource writable ?
        if ( ! getPutableFlag() ) {
            Reply error = request.makeReply(HTTP.NOT_ALLOWED) ;
            error.setContent("Method PUT not allowed.") ;
            throw new HTTPException (error) ;
        }
        // Check validators:
```

```
        if ((checkIfMatch(request) == COND_FAILED)
            || (checkIfNoneMatch(request) == COND_FAILED)
            || (checkIfModifiedSince(request) == COND_FAILED)
            || (checkIfUnmodifiedSince(request) == COND_FAILED)) {
            Reply r = request.makeReply(HTTP.PRECONDITION_FAILED);
            r.setContent("Pre-condition failed.");
            return r;
        }
        // Check the request:
        InputStream in = null;
        try {
            in = request.getInputStream();
            if ( in == null ) {
                Reply error = request.makeReply(HTTP.BAD_REQUEST) ;
                error.setContent ("<p>Request doesn't have a valid content.");
                throw new HTTPException (error) ;
            }
        } catch (IOException ex) {
            throw new ClientException(request.getClient(), ex);
        }
        // We do not support (for the time being) put with ranges:
        if ( request.hasContentRange() ) {
            Reply error = request.makeReply(HTTP.BAD_REQUEST);
            error.setContent("partial PUT not supported.");
            throw new HTTPException(error);
        }
        // Check that if some type is provided it doesn't conflict:
        if ( request.hasContentType() ) {
            MimeType rtype = request.getContentType() ;
            MimeType type  = getContentType() ;
            if ( type == null ) {
                setValue (ATTR_CONTENT_TYPE, rtype) ;
            } else if ( rtype.match (type) < 0 ) {
                Reply error = request.makeReply(HTTP.UNSUPPORTED_MEDIA_TYPE) ;
                error.setContent ("<p>Invalid content type: "+type.toString());
                throw new HTTPException (error) ;
            }
        }
        // Write the body back to the file:
        try {
            // We are about to accept the put, notify client before continuing
            Client client = request.getClient();
            if ( client != null  && request.getExpect() != null ) {
                client.sendContinue();
            }
            if ( fresource.newContent(request.getInputStream()) )
                status = HTTP.CREATED;
            else
                status = HTTP.NO_CONTENT;
        } catch (IOException ex) {
            throw new ClientException(request.getClient(), ex);
```

```
        }
        if ( status == HTTP.CREATED ) {
            reply = request.makeReply(status);
            reply.setLocation(getURL(request));
            reply.setContent ("<p>Entity body saved succesfully !") ;
        } else {
            reply = createDefaultReply(request, status);
        }
        return reply ;
    }

    /**
     * The default OPTIONS method replies with a not implemented.
     * @param request The request to handle.
     * @exception ProtocolException In case of errors.
     * @exception NotAProtocolException If the client instance controling the
     * request processing got a fatal error.
     */

    public Reply options(Request request)
        throws ProtocolException, NotAProtocolException
    {
        Reply reply = createDefaultReply(request, HTTP.OK);
        // Removed unused headers:
        reply.setContentLength(-1);
        reply.setContentType(null);
        // Add the allow header:
        if ( allowed != null )
            reply.setHeaderValue(Reply.H_ALLOW, allowed);
        return reply;
    }

    /**
     * The default DELETE method replies with a not implemented.
     * @param request The request to handle.
     * @exception ProtocolException Always thrown, to return a NOT_IMPLEMENTED
     *    error.
     * @exception NotAProtocolException If the client instance controling the
     * request processing got a fatal error.
     */

    public Reply delete(Request request)
        throws ProtocolException, NotAProtocolException
    {
        Reply error = request.makeReply(HTTP.NOT_IMPLEMENTED) ;
        error.setContent("Method DELETE not implemented.") ;
        throw new HTTPException (error) ;
    }

    /**
     * The default LINK method replies with a not implemented.
```

```
     * @param request The request to handle.
     * @exception ProtocolException Always thrown, to return a NOT_IMPLEMENTED
     *    error.
     * @exception NotAProtocolException If the client instance controling the
     * request processing got a fatal error.
     */

    public Reply link(Request request)
        throws ProtocolException, NotAProtocolException
    {
        Reply error = request.makeReply(HTTP.NOT_IMPLEMENTED) ;
        error.setContent("Method LINK not implemented.") ;
        throw new HTTPException (error) ;
    }

    /**
     * The default UNLINK method replies with a not implemented.
     * @param request The request to handle.
     * @exception ProtocolException Always thrown, to return a NOT_IMPLEMENTED
     *    error.
     * @exception NotAProtocolException If the client instance controling the
     * request processing got a fatal error.
     */

    public Reply unlink(Request request)
        throws ProtocolException, NotAProtocolException
    {
        Reply error = request.makeReply(HTTP.NOT_IMPLEMENTED) ;
        error.setContent("Method UNLINK not implemented.") ;
        throw new HTTPException (error) ;
    }

    /**
     * The default TRACE method replies with a not implemented
     * @param request The request to handle.
     * @exception HTTPException In case of errors.
     * @exception ClientException If the client instance controling the
     * request processing got a fatal error.
     */

    public Reply trace(Request request)
        throws HTTPException, ClientException
    {
        Reply reply = createDefaultReply(request, HTTP.OK);
        reply.setNoCache(); // don't cache this
        reply.setMaxAge(-1); //
        reply.setContentMD5(null);
        // Dump the request as the body
        // Removed unused headers:
        // FIXME should be something else for chuncked stream
        ByteArrayOutputStream ba = new ByteArrayOutputStream();
```

```
        try {
            reply.setContentType(new MimeType("message/http"));
            request.dump(ba);
            reply.setContentLength(ba.size());
        } catch (Exception ex) {
            ex.printStackTrace();
        }
        reply.setStream(new ByteArrayInputStream(ba.toByteArray()));
        return reply;
    }

    /**
     * The handler for unknown method replies with a not implemented.
     * @param request The request to handle.
     * @exception ProtocolException Always thrown, to return a NOT_IMPLEMENTED
     *    error.
     * @exception NotAProtocolException If the client instance controling the
     * request processing got a fatal error.
     */

    public Reply extended(Request request)
        throws ProtocolException, NotAProtocolException
    {
        String method = request.getMethod() ;
        if ((method != null) && method.equals("BROWSE") && getBrowsableFlag())
            return browse(request) ;

        Reply error = request.makeReply(HTTP.NOT_IMPLEMENTED) ;
        error.setContent("Method "+method+" not implemented.") ;
        throw new HTTPException (error) ;
    }

    /**
     * Handle the browse method.
     * @param request The request to handle.
     */

    protected static MimeType  browsetype = null;

    protected synchronized MimeType getBrowseType() {
        if ( browsetype == null ) {
            try {
                browsetype = new MimeType("application/x-navibrowse");
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        }
        return browsetype;
    }

    /**
```

```
    * A present to GNNPress users !
    * This method implements the <code>BROWSE</code> method that
    * AOL press (or GNN press, or whatever its last name is) expects.
    * @param request The request to process.
    * @exception ProtocolException If some error occurs.
    * @return A Reply instance.
    */
   public Reply browse (Request request)
       throws ProtocolException
   {
       if (dresource == null) {
           Reply error = request.makeReply(HTTP.NOT_IMPLEMENTED) ;
           error.setContent("Method "+request.getMethod()+
                            " not implemented.") ;
           throw new HTTPException (error) ;
       }

       Enumeration  enum      = dresource.enumerateResourceIdentifiers() ;
       Vector       resources = Sorter.sortStringEnumeration(enum) ;
       int          rsize     = ((resources == null) ? 0 : resources.size()) ;
       StringBuffer sb        = new StringBuffer() ;

       // As we have enumerated all resources, just looking the store is ok
       for (int i = 0 ; i < rsize ; i++) {
           String            rname = (String) resources.elementAt(i) ;
           ResourceReference rr    = null;
           Resource          r     = null;
           try {
               rr = dresource.lookup(rname) ;
               r = rr.lock();
               // may throw InvalidResourceException

               if ( r instanceof DirectoryResource ) {
                   sb.append("application/x-navidir "+
                             rname+
                             "\r\n") ;
               } else {
                   HTTPFrame itsframe =
                       (HTTPFrame) resource.getFrame(getClass());
                   if (itsframe != null) {
                       sb.append(itsframe.getContentType().toString()+
                                 " "+
                                 rname+
                                 "\r\n") ;
                   }
               }
           } catch (InvalidResourceException ex) {
               continue;
           } finally {
               rr.unlock();
           }
```

```
        }
        Reply reply = request.makeReply(HTTP.OK) ;
        reply.addPragma("no-cache");
        reply.setNoCache();
        reply.setContent(sb.toString()) ;
        reply.setContentType(getBrowseType());
        return reply ;
    }

    //
    // Filtered part
    //

    /**
     * Get our whole list of filters.
     */

    public synchronized ResourceFilter[] getFilters() {
        ResourceFrame frames[] = collectFrames(filterClass);
        if ( frames != null ) {
            // FIXME Normally a simple cast should suffice (?)
            ResourceFilter f[] = new ResourceFilter[frames.length];
            for (int i = 0 ; i < frames.length ; i++)
                f[i] = (ResourceFilter) frames[i];
            return f;
        }
        return null;
    }

    /**
     * Get the list of filters of this class.
     * @param cls The class of filters requested.
     * @return An array of filters, which are instances of the given class.
     */

    public synchronized ResourceFilter[] getFilters(Class cls) {
        ResourceFrame frames[] = collectFrames(cls);
        if ( frames != null ) {
            // FIXME Normally a simple cast should suffice (?)
            ResourceFilter f[] = new ResourceFilter[frames.length];
            for (int i = 0 ; i < frames.length ; i++)
                f[i] = (ResourceFilter) frames[i];
            return f;
        }
        return null;
    }

}
```

```
// DirectoryResource.java
// $Id: DirectoryResource.html,v 1.1 1998/07/09 10:48:27 benoit Exp $
// (c) COPYRIGHT MIT and INRIA, 1996.
// Please first read the full copyright statement in file COPYRIGHT.html

package org.w3c.tools.resources ;

import java.util.*;
import java.io.*;

import org.w3c.tools.resources.indexer.*;
import org.w3c.tools.resources.event.*;

/**
 * A simple, and reasonably efficient directory resource.
 */
public class DirectoryResource extends ContainerResource {
    /**
     * Attribute index - The index for our directory attribute.
     */
    protected static int ATTR_DIRECTORY = -1 ;
    /**
     * Attribute index - The last time we physically visited the directory.
     */
    protected static int ATTR_DIRSTAMP = -1 ;
    /**
     * Attribute index - The indexer to use for that directory, if any.
     */
    protected static int ATTR_INDEXER = -1;
    /**
     * Attribute index - The index of wether we are extensible.
     */
    protected static int ATTR_EXTENSIBLE = -1 ;

    static {
        Attribute a   = null ;
        Class     cls = null ;
        // Get a pointer to our class.
        try {
            cls = Class.forName("org.w3c.tools.resources.DirectoryResource") ;
        } catch (Exception ex) {
            ex.printStackTrace() ;
            System.exit(1) ;
        }
        // The directory attribute.
        a = new FileAttribute("directory"
                              , null
                              , Attribute.COMPUTED|Attribute.DONTSAVE);
        ATTR_DIRECTORY = AttributeRegistry.registerAttribute(cls, a) ;
        // The last time we visited the directory
```

```
        a = new DateAttribute("dirstamp"
                              , null
                              , Attribute.COMPUTED) ;
        ATTR_DIRSTAMP = AttributeRegistry.registerAttribute(cls, a) ;
        // Our indexer name (optional).
        a = new StringAttribute("indexer"
                                , null
                                , Attribute.EDITABLE) ;
        ATTR_INDEXER = AttributeRegistry.registerAttribute(cls, a) ;
        // Are we extensible (can we create resources on the fly):
        a = new BooleanAttribute("extensible"
                                 , Boolean.TRUE
                                 , Attribute.EDITABLE) ;
        ATTR_EXTENSIBLE = AttributeRegistry.registerAttribute(cls, a) ;
    }

    /**
     * Get the indexer out of the given context.
     * @return A ResourceIndexer instance, guaranteeed not to be <strong>
     * null</strong>.
     */
    protected ResourceReference getIndexer(ResourceContext c) {
        IndexerModule   m = (IndexerModule) c.getModule(IndexerModule.NAME);
        ResourceReference rr = m.getIndexer(c);
        return rr;
    }

    public void setValue(int idx, Object value) {
        super.setValue(idx, value);
        if ( idx == ATTR_INDEXER ) {
            String indexer = getString(ATTR_INDEXER, null);
            if ( indexer != null ) {
                ResourceContext c = null;
                IndexerModule   m = null;
                c = getContext();
                m = (IndexerModule) c.getModule(IndexerModule.NAME);
                m.registerIndexer(c, indexer);
            }
        }
    }

    /**
     * Get the physical directory exported by this resource.
     * @return A non-null File object giving the directory of this resource.
     */

    public File getDirectory() {
        return (File) getValue(ATTR_DIRECTORY, null) ;
    }

    /**
```

```
     * Get the absolute time at which we examined the physicall directory.
     * @return The date (as a long number of ms since Java epoch), or
     * <strong>-1</strong> if we never examined it before.
     */

    public long getDirStamp() {
        return getLong(ATTR_DIRSTAMP, -1) ;
    }

    /**
     * Get the extensible flag value.
     * A DirectoryResource is extensible, if it is allowed to create new
     * resources out of the file system knowledge on the fly.
     * <p>Setting this flag might slow down the server. It unfortunatelly
     * defaults to <strong>true</strong> until I have a decent admin
     * program.
     * @return A boolean <strong>true</strong> if the directory is
     *    extensible.
     */

    public boolean getExtensibleFlag() {
        return getBoolean(ATTR_EXTENSIBLE, true) ;
    }

    /**
     * A resource is about to be removed
     * This handles the <code>RESOURCE_REMOVED</code> kind of events.
     * @param evt The event describing the change.
     */

    public void resourceRemoved(StructureChangedEvent evt) {
        super.resourceRemoved(evt);
        markModified();
    }


    /**
     * Create a DirectoryResource and the physical directory too.
     * @param name the name of the resource.
     * @return A ResourceReference instance.
     */
    public ResourceReference createDirectoryResource(String name) {
        // Create an empty file:
        File    file          = new File(getDirectory(), name) ;
        boolean created       = false ;
        boolean exists_before = false ;

        try {
            if (file.exists()) {
                if (! file.isDirectory())
                    created = false;
```

```
                else
                    exists_before = true;
            } else {
                file.mkdir();
                created = true;
            }
        } catch (Exception ex) {
            created = false;
        }

        if (! created)
            return null;

        ResourceReference rr = createDefaultResource(name);
        if (rr == null) {
            if (!exists_before)
                file.delete();
            return null;
        }

        try {
            Resource r = rr.lock();
            if (! (r instanceof DirectoryResource)) {
                try {
                    r.delete();
                } catch (MultipleLockException ex) {
                    //OUCH!
                    //manual delete
                }
                if (!exists_before)
                    file.delete();
                return null;
            }
        } catch (InvalidResourceException ex) {
            if (!exists_before)
                file.delete();
            return null;
        } finally {
            rr.unlock();
        }
        return rr;
    }

    /**
     * Create a Resource and the physical file too.
     * @param name the name of the resource.
     * @return A ResourceReference instance.
     */
    public ResourceReference createResource(String name) {
        return createResource(name, null);
    }
```

```
    /**
     * Create a Resource and the physical file too.
     * @param name the name of the resource.
     * @param req the protocol request.
     * @return A ResourceReference instance.
     */
    public ResourceReference createResource(String name,
                                            RequestInterface req)
    {
        // Create an empty file:
        File    file    = new File(getDirectory(), name) ;
        boolean created = false ;

        if ( ! file.exists() ) {
            try {
                (new RandomAccessFile(file, "rw")).close() ;
                created = true ;
            } catch (Exception ex) {
                created = false ;
            }
        }
        if (! created)
            return null;

        ResourceReference rr = createDefaultResource(name, req);
        if (rr == null)
            file.delete();
        return rr;
    }

    /**
     * Index a Resource. Call the indexer.
     * @param name The name of the resource to index.
     * @param defs The defaults attributes.
     * @return A resource instance.
     * @see org.w3c.tools.resources.indexer.SampleResourceIndexer
     */
    private Resource index(String name, Hashtable defs) {
        return index(name, defs, null);
    }

    /**
     * Index a Resource. Call the indexer.
     * @param name The name of the resource to index.
     * @param defs The defaults attributes.
     * @param req The protocol request.
     * @return A resource instance.
     * @see org.w3c.tools.resources.indexer.SampleResourceIndexer
     */
    protected Resource index(String name,
```

```
                        Hashtable defs,
                        RequestInterface req)
    {
        // Prepare a set of default parameters for the resource:
        defs.put("identifier", name);
        updateDefaultChildAttributes(defs);
        ResourceContext context = getContext();
        // Try to get the indexer to create the resource:
        Resource    resource = null;
        ResourceReference rr_indexer  = null;
        ResourceReference rr_lastidx  = null;
        while ( context != null ) {
            // Lookup for next indexer in hierarchy:
            do {
                rr_indexer = getIndexer(context);
                context = context.getParent();
            } while ((rr_indexer == rr_lastidx) && (context != null));
            // Is this a usefull indexer ?
            if ((rr_lastidx = rr_indexer) != null ) {
                try {
                    ResourceIndexer indexer =
                        (ResourceIndexer)rr_indexer.lock();
                    resource = indexer.createResource(this,
                                                      req,
                                                      getDirectory(),
                                                      name,
                                                      defs) ;
                    if ( resource != null )
                        break;
                } catch (InvalidResourceException ex) {
                    resource = null;
                } finally {
                    rr_indexer.unlock();
                }
            }
        }
        return resource;
    }


    public synchronized ResourceReference createDefaultResource(String name) {
        return createDefaultResource(name, null);
    }
    /**
     * Try creating a default resource having the given name.
     * This method will make its best effort to create a default resource
     * having this name in the directory. If a file with this name exists,
     * it will check the pre-defined admin extensions and look for a match.
     * If a directory with this name exists, and admin allows to do so, it
     * will create a sub-directory resource.
     * @param name The name of the resource to try to create.
```

```
     * @param req The incomming request
     * @return A Resource instance, if possible, <strong>null</strong>
     *    otherwise.
     */

    protected synchronized
        ResourceReference createDefaultResource(String name,
                                                RequestInterface req)
    {
        // Don't automagically create resources of name '..' or '.'
        if (name.equals("..") || name.equals("."))
            return null ;

        Hashtable defs = new Hashtable(10) ;
        Resource resource = index(name, defs, req);
        // Did we finally create a resource ?
        ResourceReference rr = null;
        if ( resource != null ) {
            // Register this child in our store:
            rr = addResource(resource, defs) ;
            markModified() ;
        }
        return rr ;
    }

    /**
     * Initialize and register a new resource into this directory.
     * @param resource The uninitialized resource to be added.
     */
    protected ResourceContext updateDefaultChildAttributes(Hashtable attrs) {
        ResourceContext context = null;
        context = super.updateDefaultChildAttributes(attrs);
        String name = (String) attrs.get("identifier");
        if (( name != null ) && (getDirectory() !=  null)) {
            attrs.put("directory", new File(getDirectory(), name));
            return context;
        } else {
            return null;
        }
    }

    /**
     * Reindex recursivly all the resources from this DirectoryResource.
     */
    public synchronized void reindex() {
        if (getExtensibleFlag()) {
            Enumeration       e    = enumerateAllResourceIdentifiers();
            String            name = null;
            ResourceReference rr   = null;
            Resource          r    = null;
            while (e.hasMoreElements()) {
```

```
                name = (String) e.nextElement();
                rr = lookup(name);
                if (rr != null) {
                    try {
                        r = rr.lock();
                        // forbid cycles
                        if (r == this)
                            continue;
                        if (r instanceof DirectoryResource) {
                            //launch reindex
                            DirectoryResource dir = (DirectoryResource) r;
                            //reindex directory itself
                            //the new diretory must have the same context
                            Hashtable defs = new Hashtable(10);
                            defs.put("context", dir.getContext());
                            //indexing ...
                            Resource newdir = index(name, defs);
                            // do we want it to keep its indexer?
                            if (newdir == null) {
                                dir.reindex();
                            } else {
                                if (! (newdir instanceof DirectoryResource)) {
                                    throw new RuntimeException(
                                                "Reindex Error : "+
                                                name+" can't be reindexed. "+
                                                "The reindexed resource is "+
                                                "no more a DirectoryResource.");
                                }
                                DirectoryResource reindexed =
                                    (DirectoryResource) newdir;
                                String indexer =
                                    reindexed.getString(ATTR_INDEXER, "");
                                if (indexer.equals("")) {
                                    dir.reindex();
                                    indexer =
                                        dir.getString(ATTR_INDEXER, null);
                                    reindexed.setValue(ATTR_INDEXER, indexer);
                                } else {
                                    dir.setValue(ATTR_INDEXER, indexer);
                                    dir.reindex();
                                }
                                //move children to the reindexed directory
                                reindexed.setValue(ATTR_KEY, dir.getKey());
                                dir.setValue(ATTR_IDENTIFIER,
                                             name+"-bakindex");
                                addResource(reindexed, defs);
                                // Now replace the old DirectoryResource
                                // by the new one
                                try {
                                    dir.replace(reindexed);
                                } catch (MultipleLockException ex) {
```

```
                                    throw new RuntimeException(
                                                                "Reindex Error : "+
                                                                ex.getMessage());
                                }
                            }
                        } else if (! (r instanceof AbstractContainer)) {
                            //leaf
                            Hashtable resdefs = new Hashtable(10);
                            Resource resource = index(name, resdefs);
                            if (resource != null) {
                                try {
                                    r.delete();
                                } catch (MultipleLockException ex) {
                                    throw new RuntimeException(
                                                                 "Reindex Error : "+
                                                                 ex.getMessage());
                                }
                                addResource(resource, resdefs);
                            }
                        }
                    } catch (InvalidResourceException ex) {
                        System.out.println(ex.getMessage());
                    } finally {
                        rr.unlock();
                    }
                }
            }
            markModified();
        }
    }

    /**
     * Enumerate all available children resource identifiers.
     * This method <em>requires</em> that we create all our pending resources.
     * @return An enumeration of all our resources.
     */
    protected synchronized Enumeration enumerateAllResourceIdentifiers() {
        File directory = getDirectory() ;
        if ( directory != null ) {
            synchronized(this) {
                String lst[] = directory.list() ;
                if ( lst != null ) {
                    for (int i = 0 ; i < lst.length ; i++) {
                        if (lst[i].equals(".") || lst[i].equals(".."))
                            continue ;
                        if (lookup(lst[i]) == null)
                            createDefaultResource(lst[i]) ;
                    }
                }
            }
        }
```

```
        return super.enumerateResourceIdentifiers(true);
    }

    /**
     * Enumerate all available children resource identifiers.
     * This method <em>requires</em> that we create all our pending resources
     * if we are in the extensible mode...too bad !
     * @return An enumeration of all our resources.
     */
    public synchronized Enumeration enumerateResourceIdentifiers(boolean all) {
        // If extensible, update if needed:
        if (all && getExtensibleFlag() ) {
            File directory = getDirectory() ;
            if ( directory != null ) {
                synchronized(this) {
                    long dirstamp  = directory.lastModified() ;
                    if ( dirstamp > getDirStamp() ) {
                        String lst[] = directory.list() ;
                        if ( lst != null ) {
                            for (int i = 0 ; i < lst.length ; i++) {
                                if (lst[i].equals(".") || lst[i].equals(".."))
                                    continue ;
                                if (lookup(lst[i]) == null)
                                    createDefaultResource(lst[i]) ;
                            }
                        }
                        setLong(ATTR_DIRSTAMP, dirstamp) ;
                    }
                }
            }
        }
        return super.enumerateResourceIdentifiers(all);
    }

    /**
     * Lookup the resource having the given name in this directory.
     * @param name The name of the resource.
     * @return A resource instance, or <strong>null</strong>.
     */
    public ResourceReference lookup(String name)
    {
        ResourceReference rr = null;
        // Try our store:
        rr = super.lookup(name);
        if (rr != null)
            return rr;
        // If allowed, than try a default fallback:
        return getExtensibleFlag() ? createDefaultResource(name) : null ;
    }

    /**
```

```
     * Delete this directory resource, for ever.
     * This method will delete the directory resource, and its associated
     * store, <strong>along</strong> with any of the sub-resources it contains.
     * Deleting the root directory of your server might take sometime...
     * <p>Once the resource is deleted, it isx1 removed from its inital store
     * and will not be unpickleable any more.
     */

    public synchronized void delete()
        throws MultipleLockException
    {
        disableEvent();
        // Remove all the defined resources in this directory
        // Set the extensible flag to false, otherwise, the directory grows
        // as we shrink it :-)
        setBoolean(ATTR_EXTENSIBLE, false);
        super.delete();
    }

    /**
     * Was return false (don't khow why)
     */
    public synchronized boolean verify() {
        return getDirectory().exists();
    }

    /**
     * Initialize this directory resource with the given set of attributes.
     * @param values The attribute values.
     */
    public void initialize(Object values[]) {
        super.initialize(values) ;
        disableEvent();
        // Get our parent resource and compute our directory:
        File dir = null ;
        if ( ! definesAttribute(ATTR_DIRECTORY) ) {
            // Get our parent:
            ResourceReference rr = getParent();
            if (rr != null) {
                try {
                    Resource parent = rr.lock();
                    if (parent.definesAttribute("directory")) {
                        File pdir = (File) parent.getValue("directory", null);
                        if ( pdir != null ) {
                            // Compute and set our directory attribute:
                            dir = new File(pdir, getIdentifier()) ;
                            setValue(ATTR_DIRECTORY, dir) ;
                        }
                    }
                } catch (InvalidResourceException ex) {
```

```
                } finally {
                    rr.unlock();
                }
            }
        } else {
            dir = getDirectory();
        }
        // Register our specific indexer, if any:
        ResourceContext c = getContext();
        String indexer = getString(ATTR_INDEXER, null);

        if (( indexer != null ) && (!indexer.equals(""))) {
            IndexerModule   m = (IndexerModule)c.getModule(IndexerModule.NAME);
            m.registerIndexer(c, indexer);
        }
        enableEvent();
    }

}
```

```
// ResourceFilter.java
// $Id: ResourceFilter.html,v 1.1 1998/07/09 10:49:32 benoit Exp $
// (c) COPYRIGHT MIT and INRIA, 1996.
// Please first read the full copyright statement in file COPYRIGHT.html

package org.w3c.tools.resources ;

import java.util.*;
import java.io.*;

public class ResourceFilter extends ResourceFrame
    implements FilterInterface {

    /**
     * Get our target resource.
     */

    public Resource getTargetResource() {
        Resource target = (Resource) getResource();
        while (target instanceof ResourceFrame) {
            target = ((ResourceFrame)target).getResource();
        }
        return target;
    }

    /**
     * Lookup time filtering.
     * This method is called while Jigsaw performs resource lookup. Each time
     * a lookup end up on the target resource of that filter, this method
     * will be called.
     * @param ls The current lookup state.
     * @param lr The current lookup result.
     * @return A boolean, <strong>true</strong> if this filter has performed
     * the whole lookup, and side-effect the lookup result appropriatelly,
     * <strong>false</strong> otherwise.
     */

    public boolean lookup(LookupState ls, LookupResult lr)
        throws ProtocolException
    {
        return false;
    }

    /**
     * Simplified ingoingFilter API.
     * This is a default, simplified API to the ingoing filter method.
     * @param request The request to filter.
     * @return A Reply instance, or <strong>null</strong> if processing
     * should continue normally.
     * @exception HTTPException If processing should be interrupted, because
```

```
     * an abnormal situation occured.
     */

    public ReplyInterface ingoingFilter(RequestInterface request)
        throws ProtocolException
    {
        return null;
    }

    /**
     * The ingoingFilter method.
     * This is the method that really gets called by Jigsaw core. By default
     * it will invoke the simpler <code>ingoingFilter</code> method,
     * taking only the request has a parameter.
     * @param request The request that is about to be handled.
     * @param filters The whole filter list to be applied to the resource.
     * @param i A pointer into the above array, indicating which filters
     * have already been triggered (the one whose index are lower than
     * <strong>i</strong>), and what filters have to be triggered (the one
     * whose index are greater or equal to <strong>i+1</strong>).
     * @return A Reply instance, if the filter did know how to answer
     * the request without further processing, <strong>null</strong>
     * otherwise.
     */

    public ReplyInterface ingoingFilter(RequestInterface request,
                                        FilterInterface filters[],
                                        int i)
        throws ProtocolException
    {
        return ingoingFilter(request);
    }


    /**
     * Simplified API to the outgoing filter metho.
     * This is a simplified API to the ougoing filter method, you can either
     * overide this method, or the more powerfull one.
     * @param request The original request.
     * @param reply It's original reply.
     * @return A Reply instance, or <strong>null</strong> if processing
     * should continue normally.
     * @exception HTTPException If processing should be interrupted, because
     * an abnormal situation occured.
     */

    public ReplyInterface outgoingFilter(RequestInterface request,
                                         ReplyInterface reply)
        throws ProtocolException
    {
        return null;
```

```
    }

    public ReplyInterface exceptionFilter(RequestInterface request,
                                          ProtocolException ex,
                                          FilterInterface filters[],
                                          int i) {
        return null;
    }

    /**
     * The outgoingFilter method.
     * This method is the one that gets called by Jigsaw core. By default it
     * will call the simpler <code>outgoingFilter</code> method that takes
     * only the request and the reply as parameters.
     * @param request The request that has been processed.
     * @param reply The original reply as emitted by the resource.
     * @param filters The whole filter that applies to the resource.
     * @param i The current index of filters. The <em>i</em> filter is ourself,
     * filters with lower indexes have already been applied, and filters with
     * greater indexes are still to be applied.
     * @return A Reply instance, if that filter know how to complete the
     * request processing, or <strong>null</strong> if reminaing filters
     * are to be called by Jigsaw engine.
     */

    public ReplyInterface outgoingFilter(RequestInterface request,
                                         ReplyInterface reply,
                                         FilterInterface filters[],
                                         int fidx)
        throws ProtocolException
    {
        ReplyInterface fr = outgoingFilter(request, reply);
        return (fr != reply) ? fr : null;
    }

    public OutputStream outputFilter(RequestInterface request,
                                     ReplyInterface reply,
                                     OutputStream output) {
        return output;
    }
}
```

```
// ExternalContainer.java
// $Id: ExternalContainer.html,v 1.1 1998/07/09 10:49:19 benoit Exp $
// (c) COPYRIGHT MIT and INRIA, 1996.
// Please first read the full copyright statement in file COPYRIGHT.html

package org.w3c.tools.resources ;

import java.util.*;
import java.io.*;

/**
 * A Container which manage an external store, outside the space.
 */
public abstract class ExternalContainer extends ContainerResource {

  /**
   * Our transientFlag, is true that container must not be saved.
   */
  protected boolean transientFlag = false;

  /**
   * Our external repository.
   */
  protected File    repository    = null;

  public  ResourceReference createDefaultResource(String name) {
    throw new RuntimeException("not extensible");
  }

  /**
   * Mark this resource as having been modified.
   */
  public void markModified() {
    if (transientFlag) {
      setValue(ATTR_LAST_MODIFIED, new Long(System.currentTimeMillis()));
    } else {
      super.markModified();
    }
  }

  /**
   * acquire children and notify space if we will be saved.
   */
  protected synchronized void acquireChildren() {
    if (!acquired) {
      ResourceSpace space = getSpace();
      if (repository != null) {
```

```
      space.acquireChildren( getChildrenSpaceEntry() ,
                             repository,
                             transientFlag );
    } else {
      // if we have been saved one time yet.
      space.acquireChildren( getChildrenSpaceEntry() );
    }
    acquired = true;
  }
}

/**
 * Delete this Resource instance , and remove it from its store.
 * This method will erase definitely this resource, for ever, by removing
 * it from its resource store (when doable).
 */

public synchronized void delete()
  throws MultipleLockException
{
  if (transientFlag) {
    // transient, so don't try to delete myself.
    ResourceSpace space = getSpace();
    if (space != null) {
      acquireChildren();
      // check for lock on children
      Enumeration       e        = enumerateResourceIdentifiers();
      ResourceReference rr       = null;
      Resource          resource = null;
      while (e.hasMoreElements()) {
        rr = lookup((String) e.nextElement());
        if (rr != null) {
          try {
            synchronized (rr) {
              resource = rr.lock();
              resource.delete();
            }
          } catch (InvalidResourceException ex) {
            // nothing, remove invalid resource.
          } finally {
            rr.unlock();
          }
        }
      }
      space.deleteChildren(getChildrenSpaceEntry());
    }
  } else {
    super.delete();
```

```
    }
  }


  /**
   * Get The repository for this external container.
   * Warning: called in the constructor!
   * @param context The container context.
   * @return A File instance
   */
  abstract public File getRepository(ResourceContext context);


  public void initialize(Object values[]) {
    super.initialize(values);
    if (repository == null)
      repository = getRepository(getContext());
  }

  /**
   * @param id The identifier.
   * @param context The default context.
   * @param transientFlag The transient flag.
   */

  public ExternalContainer (String id,
                            ResourceContext context,
                            boolean transientFlag)
  {
    Hashtable h = new Hashtable(3);
    h.put("identifier", id);
    h.put("context", context);
    initialize(h);
    this.acquired      = false;
    this.transientFlag = transientFlag;
    if (transientFlag)
      context.setResourceReference( new DummyResourceReference(this));
  }

  public ExternalContainer () {
    super();
    this.acquired      = false;
    this.transientFlag = false;
    this.repository    = null;
  }
}
```

```
// ProtocolFrame.java
// $Id: ProtocolFrame.html,v 1.1 1998/07/09 10:48:55 benoit Exp $
// (c) COPYRIGHT MIT and INRIA, 1996.
// Please first read the full copyright statement in file COPYRIGHT.html

package org.w3c.tools.resources ;

public class ProtocolFrame extends ResourceFrame {}
```

```
// ContainerResource.java
// $Id: ContainerResource.html,v 1.1 1998/07/09 10:48:18 benoit Exp $
// (c) COPYRIGHT MIT and INRIA, 1996.
// Please first read the full copyright statement in file COPYRIGHT.html

package org.w3c.tools.resources ;

import java.util.*;

import org.w3c.tools.resources.event.*;

/**
 * This resource manage children resources.
 */
public class ContainerResource extends AbstractContainer {

    /**
     * Attribute index - The index of the resource key.
     */
    protected static int ATTR_KEY = -1;

    static {
        Attribute a   = null ;
        Class     cls = null ;
        // Get a pointer to our own class:
        try {
            cls  = Class.forName("org.w3c.tools.resources.ContainerResource") ;
        } catch (Exception ex) {
            ex.printStackTrace() ;
            System.exit(1) ;
        }
        // The identifier attribute:
        a = new IntegerAttribute("key",
                                 null,
                                 Attribute.COMPUTED);
        ATTR_KEY = AttributeRegistry.registerAttribute(cls, a);
    }

    public Object getClone(Object values[]) {
        values[ATTR_KEY] = null;
        return super.getClone(values);
    }

    /**
     * Get the container Key. This key must be unique and unchanged
     * during the container life.
     * @return a String instance.
     */
    public Integer getKey() {
        Integer key = (Integer) getValue(ATTR_KEY, null);
```

```
        if (key == null) {
            key = new Integer(getIdentifier().hashCode() ^
                              (new Date().hashCode()));
            setValue(ATTR_KEY, key);
        }
        return key;
    }

    protected SpaceEntry getSpaceEntry() {
        ResourceReference rr = getParent();
        if (rr == null) //I'm root or external!!
            return new SpaceEntryImpl(this);
        try {
            //FIXME sure that is a containerResource?
            ContainerResource cont = (ContainerResource) rr.lock();
            return new SpaceEntryImpl(cont);
        } catch (InvalidResourceException ex) {
            System.out.println(ex.getMessage());
            ex.printStackTrace();
            return null;
        } finally {
            rr.unlock();
        }
    }

    /**
     * Get the SpaceEntry of our children resources.
     * @return A SpaceEntry instance.
     */
    protected SpaceEntry getChildrenSpaceEntry() {
        return new SpaceEntryImpl( this );
    }

    /**
     * This handles the <code>RESOURCE_MODIFIED</code> kind of events.
     * @param evt The StructureChangeEvent.
     */

    //FIXME EVENT, we don't know if this resource is valid.
    public void resourceModified(StructureChangedEvent evt) {
        super.resourceModified(evt);
    }

    /**
     * A new resource has been created in some space.
     * This handles the <code>RESOURCE_CREATED</code> kind of events.
     * @param evt The event describing the change.
     */

    public void resourceCreated(StructureChangedEvent evt) {
        super.resourceCreated(evt);
```

```
    }

    /**
     * A resource is about to be removed
     * This handles the <code>RESOURCE_REMOVED</code> kind of events.
     * @param evt The event describing the change.
     */

    public void resourceRemoved(StructureChangedEvent evt) {
        super.resourceRemoved(evt);
    }

    /**
     * Update default child attributes.
     * A parent can often pass default attribute values to its children,
     * such as a pointer to itself (the <em>parent</em> attribute).
     * <p>This is the method to overide when you want your container
     * to provide these kinds of attributes. By default this method will set
     * the following attributes:
     * <dl><dt>name<dd>The name of the child (it's identifier) -
     * String instance.
     * <dt>parent<dd>The parent of the child (ie ourself here) -
     * a ContainerResource instance.
     * <dt>url<dd>If a <em>identifier</em> attribute is defined, that
     * attribute is set to the full URL path of the children.
     * </dl>
     */

    protected ResourceContext updateDefaultChildAttributes(Hashtable attrs) {
        ResourceContext context = super.updateDefaultChildAttributes(attrs);
        if (context == null) {
            context = new ResourceContext(getContext());
            attrs.put("context", context) ;
        }
        String name = (String) attrs.get("identifier");
        if ( name != null )
            attrs.put("url", getURLPath()+name);
        return context;
    }

    /**
     * Enumerate children resource identifiers.
     * @param all Should all resources be enumerated ? Resources are often
     * created on demand only, this flag allows the caller to tell the
     * container about wether it is interested only in already created
     * resources, or in all resources (even the one that have not yet been
     * created).
     * @return An String enumeration, one element per child.
     */

    public synchronized Enumeration enumerateResourceIdentifiers(boolean all) {
```

```
        ResourceSpace space = getSpace();
        acquireChildren();
        return space.enumerateResourceIdentifiers( getChildrenSpaceEntry() );
    }

    /**
     * Create a default child resource in that container.
     * This method is called by the editor to add a default resource
     * in the container under the given name. The meaning of <em>default</em>
     * is left up to the container here.
     * @param name The identifier for the new resource.
     */
    public  ResourceReference createDefaultResource(String name) {
        return null;
    }

    /**
     * Lookup a children in the container.
     * @param name The name of the children to lookup.
     */

    public ResourceReference lookup(String name) {
        acquireChildren();
        SpaceEntry sp = getChildrenSpaceEntry();
        ResourceSpace space = getSpace();
        ResourceReference rr = space.lookupResource(sp, name);
        if (rr != null)
            return rr;
        Hashtable defs = new Hashtable(5) ;
        defs.put("identifier", name);
        ResourceContext context = updateDefaultChildAttributes(defs);

        rr = space.loadResource(sp, name, defs);
        if (rr != null) {
            context.setResourceReference(rr);
            try {
                Resource resource = rr.lock();
                if (resource instanceof FramedResource) {
                    FramedResource fres = (FramedResource) resource;
                    fres.addStructureChangedListener(this);
                    // send event
                }
            } catch (InvalidResourceException ex) {
                // nothing here
            } finally {
                rr.unlock();
            }
        }
        return rr;
    }
```

```
    /**
     * Lookup the next component of this lookup state in here.
     * @param ls The current lookup state.
     * @param lr The lookup result under construction.
     * @exception ProtocolException If an error occurs.
     * @return A boolean, <strong>true</strong> if lookup has completed,
     * <strong>false</strong> if it should be continued by the caller.
     */
    public boolean lookup(LookupState ls, LookupResult lr)
        throws ProtocolException
    {
        // Give a chance to our super-class to run its own lookup scheme:
        if ( super.lookup(ls, lr) )
            return true;
        // Perform our own lookup on the next component:
        String name = ls.getNextComponent() ;
        ResourceReference rr = null;
        rr = lookup(name);
        if (rr == null) {
            lr.setTarget(null);
            return false;
        }
        try {
            lr.setTarget(rr);
            FramedResource resource = (FramedResource) rr.lock();
            return (resource != null ) ? resource.lookup(ls, lr) : false;
        } catch (InvalidResourceException ex) {
            return false;
        } finally {
            rr.unlock();
        }
    }

    /**
     * Remove a child resource from that container.
     * @param name The name of the child to remove.
     * @exception MultipleLockException If somone else has locked the
     * resource.
     */

    public void delete(String name)
        throws MultipleLockException
    {
        ResourceReference rr = null;
        rr = lookup(name);

        if (rr != null) {
            try {
                synchronized (rr) {
                    Resource resource = rr.lock();
                    if (resource instanceof FramedResource)
```

```
                    ((FramedResource)resource).
                        removeStructureChangedListener(this);
                resource.delete();
            }
        } catch (InvalidResourceException ex) {
            // FIXME ??
        } finally {
            rr.unlock();
        }
    }
}



/**
 * Delete that container and its children if children is true
 * @exception MultipleLockException If somone else has locked one
 * of the resource child.
 */
public synchronized void replace(DirectoryResource newdir)
    throws MultipleLockException
{
    Enumeration       e        = enumerateResourceIdentifiers();
    ResourceReference rr       = null;
    Resource          resource = null;
    while (e.hasMoreElements()) {
        rr = lookup((String) e.nextElement());
        if (rr != null) {
            try {
                resource = rr.lock();
                ResourceContext ctxt = new ResourceContext(
                    newdir.getContext());
                resource.setContext(ctxt, true);
                if (resource instanceof FramedResource) {
                    ((FramedResource)resource).
                        removeStructureChangedListener(this);
                    ((FramedResource)resource).
                        addStructureChangedListener(newdir);
                }
            } catch (InvalidResourceException ex) {
                // do nothing , continue
            } finally {
                rr.unlock();
            }
        }
    }
    super.delete();
}

/**
 * Delete that resource container.
```

```
     * @exception MultipleLockException If somone else has locked the
     * resource.
     */
    public synchronized void delete()
        throws MultipleLockException
    {
        disableEvent();
        ResourceSpace space = getSpace();
        //delete our children
        acquireChildren();
        // check for lock on children
        Enumeration       e        = enumerateResourceIdentifiers();
        ResourceReference rr       = null;
        Resource          resource = null;

        while (e.hasMoreElements())
            delete((String)e.nextElement());

        SpaceEntry sentry = getChildrenSpaceEntry();
        //delete myself
        super.delete();
        space.deleteChildren(sentry);
    }

    /**
     * This resource is being unloaded.
     * The resource is being unloaded from memory, perform any additional
     * cleanup required.
     */
    public void notifyUnload() {
        super.notifyUnload();
        // anything else?
    }

    protected boolean acquired = false;

    /**
     * Acquire our children. Must be called before all child manipulation.
     */
    protected synchronized void acquireChildren() {
        if (!acquired) {
            ResourceSpace space = getSpace();
            space.acquireChildren( getChildrenSpaceEntry() );
            acquired = true;
        }
    }

    /**
     * Add an initialized resource into this store container instance.
     * @param resource The resource to be added to the store.
     */
```

```
    protected synchronized ResourceReference addResource(Resource resource,
                                                         Hashtable defs) {
        acquireChildren();
        ResourceReference rr = getSpace().addResource(getChildrenSpaceEntry() ,
                                                      resource,
                                                      defs);
        resource.getContext().setResourceReference(rr);
        if (resource instanceof FramedResource) {
            FramedResource fres = (FramedResource) resource;
            fres.addStructureChangedListener(this);
        }
        markModified() ;
        postStructureChangedEvent(rr, Events.RESOURCE_CREATED);
        return rr;
    }

    /**
     * Initialize and register the given resource within that container.
     * @param name The identifier for the resource.
     * @param resource An unitialized resource instance.
     * @param defs A default set of init attribute values (may be
     * <strong>null</strong>).
     * @exception InvalidResourceException If an error occurs during
     * the registration.
     */

    public void registerResource(String name,
                                 Resource resource,
                                 Hashtable defs)
        throws InvalidResourceException
    {
        acquireChildren();
        // Create a default set of attributes:
        if ( defs == null )
            defs = new Hashtable(11) ;
        defs.put("identifier", name);
        ResourceContext context =  updateDefaultChildAttributes(defs);
        if (context != null) {
            resource.initialize(defs);
            ResourceReference rr;
            rr = getSpace().addResource(getChildrenSpaceEntry(),
                                        resource,
                                        defs);
            context.setResourceReference(rr);
            if (resource instanceof FramedResource) {
                FramedResource fres = (FramedResource) resource;
                fres.addStructureChangedListener(this);
                // send event
            }
            markModified();
```

```
            postStructureChangedEvent(rr, Events.RESOURCE_CREATED);
        } else {
            throw new InvalidResourceException(getIdentifier(),
                                               name,
                                               "unable to get context");
        }
    }

    /**
     * Initialize ourself.
     * As we are a container resource that really contains something, we make
     * sure our URL ends properly with a slash.
     * @param values Our default attribute values.
     */
    public void initialize(Object values[]) {
        super.initialize(values);
        disableEvent();
        // If my URL doesn't end with a slah, correct it:
        String url = getURLPath() ;
        if ((url != null) && ! url.endsWith("/") )
            setValue(ATTR_URL, url+"/") ;
        acquired = false;
        enableEvent();
    }
}
```

```
// FileResource.java
// $Id: FileResource.html,v 1.1 1998/07/09 10:48:58 benoit Exp $
// (c) COPYRIGHT MIT and INRIA, 1996.
// Please first read the full copyright statement in file COPYRIGHT.html

package org.w3c.tools.resources ;

import java.io.* ;

/**
 * A simple file resource.
 */
public class FileResource extends FramedResource {

    /**
     * Attributes index - The filename attribute.
     */
    protected static int ATTR_FILENAME = -1 ;

    /**
     * Attribute index - The date at which we last checked the file content.
     */
    protected static int ATTR_FILESTAMP = -1 ;

    /**
     * Attribute index - The index for the content length attribute.
     */
    protected static int ATTR_FILE_LENGTH = -1 ;

    /**
     * Attribute index - The index for the backup flag
     */
    protected static int ATTR_FILE_BACKUP = -1 ;

    static {
        Attribute a   = null ;
        Class     cls = null ;
        try {
            cls = Class.forName("org.w3c.tools.resources.FileResource") ;
        } catch (Exception ex) {
            ex.printStackTrace();
            System.exit(0);
        }
        // The filename attribute.
        a = new FilenameAttribute("filename"
                                  , null
                                  , Attribute.EDITABLE) ;
        ATTR_FILENAME = AttributeRegistry.registerAttribute(cls, a) ;
        // The file stamp attribute
```

```
        a = new DateAttribute("file-stamp"
                              , new Long(-1)
                              , Attribute.COMPUTED) ;
        ATTR_FILESTAMP = AttributeRegistry.registerAttribute(cls, a) ;
        // The file length attribute:
        a = new IntegerAttribute("file-length"
                                 , null
                                 , Attribute.COMPUTED);
        ATTR_FILE_LENGTH = AttributeRegistry.registerAttribute(cls,a);
        // The backup attribute:
        a = new BooleanAttribute("backup"
                                 , Boolean.FALSE
                                 , Attribute.EDITABLE);
        ATTR_FILE_BACKUP = AttributeRegistry.registerAttribute(cls,a);
    }

    /**
     * The file we refer to.
     * This is a cached version of some attributes, so we need to override
     * the setValue method in order to be able to catch any changes to it.
     */
    protected File file = null ;

    /**
     * Get this resource filename attribute.
     */
    public String getFilename() {
        return (String) getValue(ATTR_FILENAME, null);
    }

    /**
     * Get this file length
     */
    public int getFileLength() {
        return ((Integer) getValue(ATTR_FILE_LENGTH,
                                   new Integer(0))).intValue();
    }

    /**
     * Get the date at which we last examined the file.
     */

    public long getFileStamp() {
        return getLong(ATTR_FILESTAMP, (long) -1) ;
    }

    /**
     * Get the backup flag, create a backup file when content change
     * if true.
     */
```

```
    public boolean getBackupFlag() {
        return getBoolean(ATTR_FILE_BACKUP, false) ;
    }

    /**
     * Get the name of the backup file for this resource.
     * @return A File object suitable to receive the backup version of this
     *    file.
     */

    public File getBackupFile() {
        File   file = getFile() ;
        String name = file.getName() ;
        return new File(file.getParent(), name+"~") ;
    }

    /**
     * Save the given stream as the underlying file content.
     * This method preserve the old file version in a <code>~</code> file.
     * @param in The input stream to use as the resource entity.
     * @return A boolean, <strong>true</strong> if the resource was just
     * created, <strong>false</strong> otherwise.
     * @exception IOException If dumping the content failed.
     */

    public synchronized boolean newContent(InputStream in)
        throws IOException
    {
        File   file     = getFile() ;
        boolean created = (! file.exists() | (file.length() == 0));
        String name     = file.getName() ;
        File   temp     = new File(file.getParent(), "#"+name+"#") ;
        String iomsg    = null ;

        // We are not catching IO exceptions here, except to remove temp:
        try {
            FileOutputStream fout  = new FileOutputStream(temp) ;
            byte             buf[] = new byte[4096] ;
            for (int got = 0 ; (got = in.read(buf)) > 0 ; )
                fout.write(buf, 0, got) ;
            fout.close() ;
        } catch (IOException ex) {
            iomsg = ex.getMessage() ;
        } finally {
            if ( iomsg != null ) {
                temp.delete() ;
                throw new IOException(iomsg) ;
            } else {
                if (getBackupFlag()) {
```

```
                File backup = getBackupFile();
                if ( backup.exists() )
                    backup.delete();
                file.renameTo(getBackupFile()) ;
            }
            temp.renameTo(file) ;
            // update our attributes for this new content:
            updateFileAttributes() ;
        }
    }
    return created;
}

/**
 * Check this file content, and update attributes if needed.
 * This method is normally called before any perform request is done, so
 * that we make sure that all meta-informations is up to date before
 * handling a request.
 * @return The time of the last update to the resource.
 */

public long checkContent() {
    File file = getFile() ;
    // Has this resource changed since last queried ?
    long lmt = file.lastModified() ;
    long cmt = getFileStamp() ;
    if ((cmt < 0) || (cmt < lmt)) {
        updateFileAttributes() ;
        return getLastModified() ;
    } else {
        return cmt;
    }
}

/**
 * Set some of this resource attribute.
 * We just catch here any write access to the filename's, to update
 * our cache file object.
 */

public synchronized void setValue(int idx, Object value) {
    super.setValue(idx, value) ;
    if ((idx == ATTR_FILENAME) || (idx == ATTR_IDENTIFIER))
        file = null;
}

/**
 * Get this file resource file name.
 */
```

```
    public synchronized File getFile() {
        // Have we already computed this ?
        if ( file == null ) {
            // Get the file name:
            String name = getFilename() ;
            if ( name == null )
                name = getIdentifier() ;
            // Get the file directory:
            ResourceReference rr = getParent();
            ResourceReference rrtemp = null;
            Resource p = null;

            while ( true ) {
                try {
                    if (rr == null)
                        return null;
                    p = rr.lock();
                    if (p instanceof DirectoryResource) {
                        file = new File(((DirectoryResource) p)
                                        .getDirectory(), name);
                        return file;
                    }
                    rrtemp = p.getParent();
                } catch (InvalidResourceException ex) {
                    return null;
                } finally {
                    if (rr != null)
                        rr.unlock();
                }
                rr = rrtemp;
            }
        }
        return file ;
    }

    /**
     * Is that resource still wrapping an existing file ?
     * If the underlying file has disappeared <string> and if</strong> the
     * container directory is extensible, remove the resource.
     */

    public synchronized boolean verify()
        throws MultipleLockException
    {
        File file = getFile();
        if ( ! file.exists() ) {
            // Is the parent extensible:
            ResourceReference rr = getParent();
            ResourceReference rrtemp = null;
            Resource p = null;
```

```
            while ( true ) {
                try {
                    if (rr == null)
                        return false;
                    p = rr.lock();
                    if (p instanceof DirectoryResource) {
                        DirectoryResource d = (DirectoryResource) p;
                        if ( ! d.getExtensibleFlag() )
                            return false;
                        else {
                            // Emit an error message, and delete the resource:
                            String msg = file+
                                ": deleted, removing the FileResource.";
                            getServer().errlog(this, msg);
                            delete();
                            return false;
                        }
                    }
                    rrtemp = p.getParent();
                } catch (InvalidResourceException ex) {
                    return false;
                } finally {
                    if (rr != null)
                        rr.unlock();
                }
                rr = rrtemp;
            }
        } else {
            return true;
        }
    }

    /**
     * Update the file related attributes.
     * The file we serve has changed since the last time we checked it, if
     * any of the attribute values depend on the file content, this is the
     * appropriate place to recompute them.
     */

    public void updateFileAttributes() {
        File file = getFile() ;
        setValue(ATTR_FILESTAMP, new Long(file.lastModified()));
        setValue(ATTR_FILE_LENGTH, new Integer((int)file.length()));
        return ;
    }

    /**
     * Update our computed attributes.
     */
```

```
    public void updateAttributes() {
        long fstamp = getFile().lastModified() ;
        long stamp  = getLong(ATTR_FILESTAMP, -1) ;

        if ((stamp < 0) || (stamp < fstamp))
            updateFileAttributes() ;
    }

    /**
     * Initialize the FileResource instance.
     */

    public void initialize(Object values[]) {
        super.initialize(values);
        disableEvent();
        // If we have a filename attribute, update url:
        String filename = getFilename();
        if ( filename != null ) {
            ResourceReference rr = getParent();
            try {
                Resource parent = rr.lock();
                setValue(ATTR_URL, parent.getURLPath()+filename);
            } catch (InvalidResourceException ex) {
                //FIXME
            } finally {
                rr.unlock();
            }
        }
        enableEvent();
    }

}
```

```
// ContainerInterfaceImpl.java
// $Id: AbstractContainer.html,v 1.1 1998/07/09 10:49:28 benoit Exp $
// (c) COPYRIGHT MIT and INRIA, 1996.
// Please first read the full copyright statement in file COPYRIGHT.html

package org.w3c.tools.resources ;

import java.util.*;

import org.w3c.tools.resources.event.*;


/**
 * The top level class for Resource Container.
 */
public abstract class AbstractContainer extends FramedResource
                                        implements ContainerInterface,
                                                   StructureChangedListener,
                                                   AttributeChangedListener
{

    /**
     * This handles the <code>RESOURCE_MODIFIED</code> kind of events.
     * @param evt The StructureChangeEvent.
     */

    public void resourceModified(StructureChangedEvent evt) {
        displayEvent( this, evt );
    }

    /**
     * A new resource has been created in some space.
     * This handles the <code>RESOURCE_CREATED</code> kind of events.
     * @param evt The event describing the change.
     */

    public void resourceCreated(StructureChangedEvent evt) {
        displayEvent( this, evt );
    }

    /**
     * A resource is about to be removed
     * This handles the <code>RESOURCE_REMOVED</code> kind of events.
     * @param evt The event describing the change.
     */

    public void resourceRemoved(StructureChangedEvent evt) {
        displayEvent( this, evt );
    }
```

```
    /**
     * A resource is about to be unloaded
     * This handles the <code>RESOURCE_UNLOADED</code> kind of events.
     * @param evt The event describing the change.
     */

    public void resourceUnloaded(StructureChangedEvent evt){
        // don't display event here, because the resource is about
        // to be unloaded.
    }

    /**
     * Gets called when a property changes.
     * @param evt The AttributeChangeEvent describing the change.
     */

    public void attributeChanged(AttributeChangedEvent evt) {
        displayEvent( this, evt );
    }

    /**
     * Enumerate children resource identifiers.
     * @param all Should all resources be enumerated ? Resources are often
     * created on demand only, this flag allows the caller to tell the
     * container about wether it is interested only in already created
     * resources, or in all resources (even the one that have not yet been
     * created).
     * @return An String enumeration, one element per child.
     */

    abstract public Enumeration enumerateResourceIdentifiers(boolean all);

    public Enumeration enumerateResourceIdentifiers() {
        return enumerateResourceIdentifiers(true);
    }

    /**
     * Ask our frames to update default child attributes.
     * @param attrs A hashtable.
     */
    protected ResourceContext updateDefaultChildAttributes(Hashtable attrs) {
        ResourceFrame frames[] = getFrames();
        if ( frames != null ) {
            for (int i = 0 ; i < frames.length ; i++) {
                if ( frames[i] == null )
                    continue;
                frames[i].updateDefaultChildAttributes(attrs);
            }
        }
```

```
        return (ResourceContext)attrs.get("context");
    }

    /**
     * Lookup a children in the container.
     * @param name The name of the children to lookup.
     * @exception InvalidResourceException If the container could not restore
     * the resource from its store.
     */

    abstract public ResourceReference lookup(String name);

    /**
     * Remove a child resource from that container.
     * @param name The name of the child to remove.
     * @exception MultipleLockException If someone else
     * has locked the resource.
     */

    abstract public void delete(String name)
        throws MultipleLockException;

    /**
     * Create a default child resource in that container.
     * This method is called by the editor to add a default resource
     * in the container under the given name. The meaning of <em>default</em>
     * is left up to the container here.
     * @param name The identifier for the new resource.
     */
    abstract public ResourceReference createDefaultResource(String name);

    /**
     * Initialize and register the given resource within that container.
     * @param name The identifier for the resource.
     * @param resource An unitialized resource instance.
     * @param defs A default set of init attribute values (may be
     * <strong>null</strong>).
     * @exception InvalidResourceException If an error occurs during the
     * registration.
     */

    abstract public void registerResource(String name,
                                          Resource resource,
                                          Hashtable defs)
        throws InvalidResourceException;
}
```

```
// ResourceFrame.java
// $Id: ResourceFrame.html,v 1.1 1998/07/09 10:49:41 benoit Exp $
// (c) COPYRIGHT MIT and INRIA, 1996.
// Please first read the full copyright statement in file COPYRIGHT.html

package org.w3c.tools.resources ;

import java.util.* ;
import java.io.* ;

import org.w3c.tools.resources.event.*;

/**
 * The resource frame class. A ResourceFrame can be attached to a
 * resource.
 */
public class ResourceFrame extends FramedResource
    implements AttributeChangedListener
{

    /**
     * Our FrameEventListener.
     */
    protected transient FrameEventListener frameListener = null;

    /**
     * Our target resource.
     */
    protected FramedResource resource = null;

    static {
        Attribute a   = null ;
        Class     cls = null ;
        // Get a pointer to our own class:
        try {
            cls  = Class.forName("org.w3c.tools.resources.ResourceFrame") ;
        } catch (Exception ex) {
            ex.printStackTrace() ;
            System.exit(1) ;
        }
    }

    /**
     * Get the file part of the URL this resource is attached to.
     * @return An URL object specifying the location in the information
     *    space of this resource.
     */
    public String getURLPath() {
        return getString(ATTR_URL, getResource().getURLPath()) ;
    }
```

```
    /**
     * Get the space entry for that resource. This Object is use to
     * retrieve the resource in the resource space.
     * A ResourceFrame has no SpaceEntry.
     * @return always null.
     */
    protected SpaceEntry getSpaceEntry() {
        return null;
    }

    private ResourceReference self = null;
    /**
     * Get The FrameReference of this frame, or <strong>null</strong>
     * if this frame is not registered.
     * @return A ResourceReference instance.
     */
    public ResourceReference getFrameReference() {
        if ((self == null) &&  (resource != null)) {
            self = resource.getFrameReference(this);
        }
        return self;
    }

    public ResourceReference getResourceReference() {
        return getFrameReference();
    }

    /**
     * If our target resource has some children, we could have
     * some attribute to give to them.
     * @param attrs A Hashtable.
     */
    protected void updateDefaultChildAttributes(Hashtable attrs) {
        //nothing here
    }

    /**
     * Check if this kind of request can be perform by this resource.
     * @param request A RequestInterface instance
     * @return a boolean.
     */
    public boolean checkRequest(RequestInterface request) {
        return true;
    }

    /**
     * FIXME doc
     */
    public ReplyInterface perform(RequestInterface request)
        throws ProtocolException, NotAProtocolException
```

```
        {
            return super.perform(request);
        }

    /**
     * FIXME doc
     */
    public boolean lookup(LookupState ls, LookupResult lr)
        throws ProtocolException
        {
            //FIXME does a frame can have frames other than filters?
            //exclude filters?
            ResourceFrame frames[] = getFrames();
            if (frames != null) {
                for (int i = 0 ; i < frames.length ; i++) {
                    if (frames[i] == null)
                        continue;
                    if (frames[i].lookup(ls,lr))
                        return true;
                }
            }
            //
            // FIXME unuseful.
            //
            if ( ls.hasMoreComponents() ) {
                // We are not a container resource, and we don't have children:
                lr.setTarget(null);
                return false;
            } else {
                // We are done !
                //    org.w3c.util.Trace.showTrace("lookup done : "+
                //                  this+", "+resource.getResourceReference());
                lr.setTarget(resource.getResourceReference());
                return true;
            }
        }

    public void processEvent(ResourceEvent evt) {
        if (evt instanceof FrameEvent) {
            fireFrameEvent((FrameEvent)evt);
        } else if (evt instanceof AttributeChangedEvent) {
            fireAttributeChangeEvent((AttributeChangedEvent)evt);
        }
    }

    /**
     * Add a frame event listener.
     * @param l The new frame event listener.
     */

    public void addFrameEventListener(FrameEventListener l) {
```

```
        frameListener = ResourceEventMulticaster.add(frameListener, l);
    }

    /**
     * Remove a frame event listener.
     * @param l The listener to remove.
     */

    public void removeFrameEventListener (FrameEventListener l) {
        frameListener = ResourceEventMulticaster.remove(frameListener, l);
    }

    /**
     * Post a frameEvent.
     * @param the frame event type.
     */
    protected void postFrameEvent(int type) {
        if (frameListener != null) {
            FrameEvent evt = new FrameEvent(this, type);
            postEvent(evt);
        }
    }

    /**
     * Fire a frameEvent.
     * @param the frame event type.
     */
    protected void fireFrameEvent(FrameEvent evt) {
        if (frameListener != null) {
            int type = evt.getID();
            switch (type) {
            case Events.FRAME_ADDED :
                frameListener.frameAdded(evt);
                break;
            case Events.FRAME_MODIFIED :
                frameListener.frameModified(evt);
                break;
            case Events.FRAME_REMOVED :
                frameListener.frameRemoved(evt);
                break;
            }
        }
    }

    /**
     * Listen its resource.
     */
    public void attributeChanged(AttributeChangedEvent evt) {
        displayEvent( this, evt );
        setValue(ATTR_LAST_MODIFIED, new Long(System.currentTimeMillis()));
    }
```

```
    /**
     * This handles the <code>FRAME_MODIFIED</code> kind of events.
     * @param evt The event describing the change.
     */

    public void frameModified(FrameEvent evt) {
        displayEvent( this, evt );
        markModified();
        postFrameEvent(evt.getID());
    }

    /**
     * We overide setValue, to fire event.
     * @param idx The index of the attribute to modify.
     * @param value The new attribute value.
     */
    public synchronized void setValue(int idx, Object value) {
        super.setValue(idx,value);
        if (idx != ATTR_LAST_MODIFIED)
            postFrameEvent(Events.FRAME_MODIFIED);
    }

    /**
     * Get the target resource.
     * @return a resource instance.
     */
    public FramedResource getResource() {
        return resource;
    }

    /**
     * Register a target resource. Called after initialize,
     * set the context. getServer() can be call only after
     * this method call.
     * @parame resource The resource to register.
     */
    public void registerResource(FramedResource resource) {
        this.resource = resource;
        postFrameEvent(Events.FRAME_ADDED);
        setValue(ATTR_CONTEXT, resource.getContext());
    }

    /**
     * Register a target resource.
     * @parame resource The resource to register.
     */
    public void unregisterResource(Resource resource) {
        //FIXME (can we have more than one resource? )
        this.resource = null;
        postFrameEvent(Events.FRAME_REMOVED);
```

```
    }

}
```

```
// FramedResource.java
// $Id: FramedResource.html,v 1.1 1998/07/09 10:49:23 benoit Exp $
// (c) COPYRIGHT MIT and INRIA, 1996.
// Please first read the full copyright statement in file COPYRIGHT.html

package org.w3c.tools.resources ;

import java.util.* ;
import java.io.* ;

import org.w3c.tools.resources.event.*;

/**
 * A FramedResource manage frames which are called during the
 * lookup and the perform.
 */
public class FramedResource extends Resource
                            implements FrameEventListener
{

    /**
     * The ResourceReference of frames.
     */
    class FrameReference implements ResourceReference {

        Class             frameClass = null;
        ResourceReference framedr    = null;

        int lockCount = 0;

        public void updateContext(ResourceContext ctxt) {
            //nothing to do
        }

        public int nbLock() {
            return lockCount;
        }

        /**
         * Lock the refered resource in memory.
         * @return A real pointer to the resource.
         */

        public Resource lock()
            throws InvalidResourceException
        {
            FramedResource res = (FramedResource)framedr.lock();
            lockCount++;
            return res.getFrame(frameClass);
```

```
        }

        /**
         * Unlock that resource from memory.
         */

        public void unlock() {
            framedr.unlock();
            lockCount--;
        }

        /**
         * Is that resource reference locked ?
         */

        public boolean isLocked() {
            return lockCount != 0;
        }

        FrameReference (ResourceFrame rframe, ResourceReference framedr) {
            this.frameClass = rframe.getClass();
            this.framedr = framedr;
        }
    }

    /**
     * Our frames references.
     */
    protected transient Hashtable framesRef = null;
    //<ResourceFrame, Reference>

    /**
     * Our AttributeChangedListener.
     */
    protected transient AttributeChangedListener attrListener = null;

    /**
     * Our StructureChangedListener.
     */
    protected transient StructureChangedListener structListener = null;

    protected transient boolean debugEvent = false;

    protected transient boolean event_disabled = false;

    protected void disableEvent() {
        event_disabled = true;
    }

    protected void enableEvent() {
```

```
        event_disabled = false;
    }

    /**
     * Attribute index - The object identifier.
     */
    protected static int ATTR_OID = -1;

    static {
        Attribute a   = null ;
        Class     cls = null ;
        // Get a pointer to our class:
        try {
            cls = Class.forName("org.w3c.tools.resources.FramedResource") ;
        } catch (Exception ex) {
            ex.printStackTrace() ;
            System.exit(1) ;
        }
        // The object identifier, *should* be uniq (see below)
        a = new IntegerAttribute("oid",
                                 null,
                                 Attribute.COMPUTED);
        ATTR_OID = AttributeRegistry.registerAttribute(cls, a);
    }


    public Object getClone(Object values[]) {
        FramedResource clone   = (FramedResource) super.getClone(values);
        clone.framesRef      = new Hashtable(3);
        return clone;
    }

    /**
     * Get the server this resource is served by.
     * @return The first instance of Jigsaw this resource was attached to.
     */
    public ServerInterface getServer() {
        return ((ResourceContext) getValue(ATTR_CONTEXT, null)).getServer();
    }

    /**
     * Get this resource's object identifier.
     * An object identifier is to be used specifically in etags. It's purpose
     * is to uniquify the etag of a resource. It's computed as a random number
     *, on demand only.
     * @return A uniq object identifier for that resource, as an inteeger.
     */
    public int getOid() {
        int oid = getInt(ATTR_OID, -1);
        if ( oid == -1 ) {
```

```
            double d = Math.random() * ((double) Integer.MAX_VALUE);
            setInt(ATTR_OID, oid = (int) d);
        }
        return oid;
    }

    protected void displayEvent(FramedResource fr, EventObject evt) {
        if (debugEvent) {
            System.out.println(">>> ["+fr.getIdentifier()+
                               "] has receive "+evt);
        }
    }

    /**
     * This handles the <code>FRAME_ADDED</code> kind of events.
     * @param evt The FrameEvent.
     */

    public void frameAdded(FrameEvent evt) {
        displayEvent( this, evt );
        markModified();
    }

    /**
     * This handles the <code>FRAME_MODIFIED</code> kind of events.
     * @param evt The event describing the change.
     */

    public void frameModified(FrameEvent evt) {
        displayEvent( this, evt );
        markModified();
    }

    /**
     * A frame is about to be removed
     * This handles the <code>FRAME_REMOVED</code> kind of events.
     * @param evt The event describing the change.
     */

    public void frameRemoved(FrameEvent evt) {
        displayEvent( this, evt );
        markModified();
    }

    /**
     * Initialize and attach a new ResourceFrame to that resource.
     * @param frame An uninitialized ResourceFrame instance.
     * @param defs A default set of attribute values.
     */
```

```
    public void registerFrame(ResourceFrame frame, Hashtable defs) {
        super.registerFrame(frame,defs);
        frame.addFrameEventListener(this);
        addAttributeChangedListener(frame);
        frame.registerResource(this);
    }

    /**
     * Register a new ResourceFrame if none (from the same class) has been
     * registered.
     * @param classname The ResourceFrame class
     * @param identifier The ResourceFrame identifier
     * @exception ClassNotFoundException if the class can't be found
     * @exception IllegalAccessException if the class or initializer is not
     * accessible
     * @exception InstantiationException if the class can't be instanciated
     * @exception ClassCastException if the class is not a ResourceFrame
     */
    protected void registerFrameIfNone(String classname, String identifier)
        throws ClassNotFoundException,
               IllegalAccessException,
               InstantiationException,
               ClassCastException
    {
        Class frameclass =
            Class.forName(classname);
        ResourceFrame frame = getFrame(frameclass);
        if (frame == null) {
            Hashtable defs = new Hashtable(3);
            defs.put("identifier" , identifier);
            registerFrame( (ResourceFrame)frameclass.newInstance() , defs );
        }
    }


    /**
     * Unregister a resource frame from the given resource.
     * @param frame The frame to unregister from the resource.
     */

    public synchronized void unregisterFrame(ResourceFrame frame) {
        super.unregisterFrame(frame);
        frame.unregisterResource(this);
        frame.removeFrameEventListener(this);
        removeAttributeChangedListener(frame);
    }

    private ResourceReference[] getReferenceArray(ResourceFrame[] frames) {
        if (frames == null)
            return null;
```

```
        ResourceReference[] refs = new ResourceReference[frames.length];
        ResourceReference rr = null;
        for (int i=0 ; i < frames.length ; i++) {
            rr = (ResourceReference)framesRef.get(frames[i]);
            if (rr == null) {
                rr = (ResourceReference)
                    new FrameReference(frames[i],
                                       getResourceReference());
                framesRef.put(frames[i],rr);
            }
            refs[i] = rr;
        }
        return refs;
    }

    /**
     * Collect all frames references.
     * @return An array of ResourceReference, containing a set of
     * FrameReference instances or <strong>null</strong> if no resource
     * frame is available.
     */
    public synchronized ResourceReference[] getFramesReference() {
        return getReferenceArray(getFrames());
    }

    /**
     * Collect any frame reference pointing to an instance of the given class.
     * @param cls The class of frames we are looking for.
     * @return An array of ResourceReference, containing a set of
     * FrameReference pointing to instances of the given class, or
     * <strong>null</strong> if no resource frame is available.
     */
    public synchronized ResourceReference[] collectFramesReference(Class c) {
        return getReferenceArray(collectFrames(c));
    }

    /**
     * Get the first occurence of a frame of the given class.
     * @param cls The class of te frame to look for.
     * @return A ResourceReference instance, or <strong>null</strong>.
     */
    public synchronized ResourceReference getFrameReference(Class c) {
        ResourceFrame     frame = getFrame(c);
        if (frame == null)
            return null;
        ResourceReference  rr = (ResourceReference)framesRef.get(frame);
        if (rr == null) {
            rr = (ResourceReference)
                new FrameReference(frame,
                                   getResourceReference());
```

```
            framesRef.put(frame,rr);
        }
        return rr;
    }


    /**
     * Get The FrameReference of the given frame, or <strong>null</strong>
     * if the frame is not registered.
     * @param frame The ResourceFrame.
     * @return A ResourceReference instance.
     */
    public synchronized
        ResourceReference getFrameReference(ResourceFrame frame) {
        ResourceReference rr = (ResourceReference)framesRef.get(frame);
        if (rr == null) {
            rr = (ResourceReference)
                new FrameReference(frame,
                                   getResourceReference());
            framesRef.put(frame,rr);
        }
        return rr;
    }

    /**
     * (AWT Like), dspatch the Event to all our listeners.
     * @param evt The resourceEvent to dispatch.
     */
    public void processEvent(ResourceEvent evt) {
        if (evt instanceof StructureChangedEvent) {
            fireStructureChangedEvent((StructureChangedEvent)evt);
        } else if (evt instanceof AttributeChangedEvent) {
            fireAttributeChangeEvent((AttributeChangedEvent)evt);
        }
    }

    /**
     * Post an Event in the Event Queue.
     * @param evt The Event to post.
     */
    public void postEvent(ResourceEvent evt) {
        if (event_disabled)
            return;
        ResourceSpace space = getSpace();
        if (space != null)
            space.getEventQueue().sendEvent(evt);
    }

    /**
     * Add an attribute change listener.
```

```
     * @param l The new attribute change listener.
     */

    public void addAttributeChangedListener(AttributeChangedListener l) {
        attrListener = ResourceEventMulticaster.add(attrListener, l);
    }

    /**
     * Remove an attribute change listener.
     * @param l The listener to remove.
     */

    public void removeAttributeChangedListener(AttributeChangedListener l) {
        attrListener = ResourceEventMulticaster.remove(attrListener, l);
    }


    /**
     * post an attribute change event. Actually this kind of event should
     * not be posted. So fire them!
     * @param idx The index of the attribute that has changed.
     * @param newvalue The new value for that attribute.
     */

    protected void postAttributeChangeEvent(int idx, Object newvalue) {
        if (( attrListener != null ) && (getResourceReference() != null)) {
            AttributeChangedEvent evt =
                new AttributeChangedEvent(getResourceReference(),
                                          attributes[idx],
                                          newvalue);
            fireAttributeChangeEvent(evt);
        }
    }

    /**
     * Fire an attribute change event.
     * @param evt the AttributeChangedEvent to fire.
     */
    protected void fireAttributeChangeEvent(AttributeChangedEvent evt) {
        if ( attrListener != null )
            attrListener.attributeChanged(evt);
    }


    /**
     * Add a structure change listener.
     * @param l The new structure change listener.
     */

    public void addStructureChangedListener(StructureChangedListener l) {
```

```
        structListener = ResourceEventMulticaster.add(structListener, l);
    }

    /**
     * Remove a structure change listener.
     * @param l The listener to remove.
     */

    public void removeStructureChangedListener(StructureChangedListener l) {
        structListener = ResourceEventMulticaster.remove(structListener, l);
    }

    /**
     * post an structure change event.
     * @param rr the ResourceReference of the source.
     * @param type The type of the event.
     */
    protected void postStructureChangedEvent(ResourceReference rr, int type) {
        if ((structListener != null) && (rr != null)) {
            StructureChangedEvent evt =
                new StructureChangedEvent(rr, type);
            postEvent(evt);
        }
    }

    /**
     * post an structure change event.
     * @param type The type of the event.
     */
    protected void postStructureChangedEvent(int type) {
        if ((structListener != null) && (getResourceReference() != null)) {
            StructureChangedEvent evt =
                new StructureChangedEvent(getResourceReference(), type);
            postEvent(evt);
        }
    }

    /**
     * Fire an structure change event.
     * @param type The type of the event.
     */
    protected void fireStructureChangedEvent(int type) {
        if ((structListener != null) && (getResourceReference() != null)) {
            StructureChangedEvent evt =
                new StructureChangedEvent(getResourceReference(), type);
            fireStructureChangedEvent(evt);
        }
    }
```

```
    /**
     * Fire an structure change event.
     * @param evt the StructureChangedEvent to fire.
     */
    protected void fireStructureChangedEvent(StructureChangedEvent evt) {
        if (structListener != null) {
            int type = evt.getID();
            switch (type) {
            case Events.RESOURCE_MODIFIED :
                structListener.resourceModified(evt);
                break;
            case Events.RESOURCE_CREATED :
                structListener.resourceCreated(evt);
                break;
            case Events.RESOURCE_REMOVED :
                structListener.resourceRemoved(evt);
                break;
            case Events.RESOURCE_UNLOADED :
                structListener.resourceUnloaded(evt);
                break;
            }
        }
    }

    /**
     * This resource is being unloaded.
     * The resource is being unloaded from memory, perform any additional
     * cleanup required.
     */
    public void notifyUnload() {
        fireStructureChangedEvent(Events.RESOURCE_UNLOADED);
        super.notifyUnload();
    }
    /**
     * Delete this Resource instance, and remove it from its store.
     * This method will erase definitely this resource, for ever, by removing
     * it from its resource store (when doable).
     */
    public synchronized void delete()
        throws MultipleLockException
    {
        disableEvent();
        // fire and not post because we don't want this resource
        // to be locked() during the delete.
        fireStructureChangedEvent(Events.RESOURCE_REMOVED);
        ResourceFrame frames[] = getFrames();
        if ( frames != null ) {
            for (int i = 0 ; i < frames.length ; i++) {
                if ( frames[i] == null )
                    continue;
```

```
                frames[i].removeFrameEventListener(this);
                this.removeAttributeChangedListener(frames[i]);
                frames[i].unregisterResource(this);
            }
        }
        try {
            super.delete();
        } catch (MultipleLockException ex) {
            enableEvent();
            throw ex;
        }
    }

    /**
     * Mark this resource as having been modified.
     */
    public void markModified() {
        super.markModified();
        postStructureChangedEvent(Events.RESOURCE_MODIFIED);
    }

    /**
     * Set some of this resource attribute. We overide setValue to post
     * events.
     */
    public synchronized void setValue(int idx, Object value) {
        super.setValue(idx, value) ;
        if (idx != ATTR_LAST_MODIFIED) {
            postAttributeChangeEvent(idx, value);
            postStructureChangedEvent(Events.RESOURCE_MODIFIED);
        }
    }

    /**
     * FIXME doc
     */
    public boolean lookup(LookupState ls, LookupResult lr)
        throws ProtocolException
    {
        ResourceFrame frames[] = getFrames();
        if (frames != null) {
            for (int i = 0 ; i < frames.length ; i++) {
                if (frames[i] == null)
                    continue;
                if (frames[i].lookup(ls,lr))
                    return true;
            }
        }
        if ( ls.hasMoreComponents() ) {
            // We are not a container resource, and we don't have children:
```

```
            lr.setTarget(null);
            return false;
        } else {
            //we are done!
            lr.setTarget(getResourceReference());
            return true;
        }
    }

    /**
     * Perform the request on all the frames of that resource. The
     * Reply returned is the first non-null reply.
     * @param request A RequestInterface instance.
     * @return A ReplyInterface instance.
     */
    protected ReplyInterface performFrames(RequestInterface request)
        throws ProtocolException, NotAProtocolException
    {
        ResourceFrame frames[] = getFrames();
        if (frames != null) {
            for (int i = 0 ; i < frames.length ; i++) {
                if (frames[i] == null)
                    continue;
                ReplyInterface reply  = frames[i].perform(request);
                if (reply != null)
                    return reply;
            }
        }
        return null;
    }

    /**
     * FIXME doc
     */
    public ReplyInterface perform(RequestInterface request)
        throws ProtocolException, NotAProtocolException
    {
        return performFrames(request);
    }

    /**
     * Initialize the frames of that framed resource.
     * @param values Default attribute values.
     */

    public void initialize(Object values[]) {
        this.attrListener   = null;
        this.structListener = null;
        disableEvent();
        super.initialize(values);
```

```
        // Initialize the frames if any.
        ResourceFrame frames[] = getFrames();
        if ( frames != null ) {
            this.framesRef = new Hashtable(frames.length);
            Hashtable defs = new Hashtable(3);
            for (int i = 0 ; i < frames.length ; i++) {
                if ( frames[i] == null )
                    continue;
                frames[i].registerResource(this);
                frames[i].initialize(defs);
                frames[i].addFrameEventListener(this);
                this.addAttributeChangedListener(frames[i]);
            }
        } else {
            this.framesRef = new Hashtable(3);
        }
        enableEvent();
    }

}
```

```
// Resource.java
// $Id: Resource.html,v 1.1 1998/07/09 10:49:36 benoit Exp $
// (c) COPYRIGHT MIT and INRIA, 1996.
// Please first read the full copyright statement in file COPYRIGHT.html

package org.w3c.tools.resources ;

import java.util.*;
import java.lang.ArrayIndexOutOfBoundsException;

/**
 * The resource class describes an object, accessible through the server.
 * Resource objects are required to have the following properties:
 * <ul>
 * <li>They must be persistent (their life-time can span multiple server
 * life-time).
 * <li>They must be editable, so that one can change some of their aspects
 * (such as any associated attribute).
 * <li>They must be self-described: each resource must now what kind
 * of attribute it <em>understands</em>.
 * <li>They must be able to update themselves: some of the meta-information
 * associated with a resource may require lot of CPU to compute.
 * <li>They must implement some name-service policy.
 * </ul>
 * <p>These resource objects do not define how they are accessed. See the
 * sub-classes for specific accesses.
 */

public class Resource extends AttributeHolder {
    private static final boolean debugunload = false;

    /**
     * Attribute index - The index of the resource store entry attribute.
     */
    protected static int ATTR_STORE_ENTRY = -1;
    /**
     * Attribute index - The index for the identifier attribute.
     */
    protected static int ATTR_IDENTIFIER = -1 ;
    /**
     * Attribute index - Associated resource frames
     */
    protected static int ATTR_RESOURCE_FRAMES = -1;
    /**
     * Attribute index - The index for our parent attribute.
     */
    protected static int ATTR_PARENT = -1 ;
    /**
     * Attribute index - The hierarchical context of the resource.
     */
```

```
    protected static int ATTR_CONTEXT = -1;
    /**
     * Attribute index - The index for our URL attribute.
     */
    protected static int ATTR_URL = -1;
    /**
     * Attribute index - The index for the last-modified attribute.
     */
    protected static int ATTR_LAST_MODIFIED = -1 ;

    static {
        Attribute a   = null ;
        Class     cls = null ;
        // Get a pointer to our own class:
        try {
            cls  = Class.forName("org.w3c.tools.resources.Resource") ;
        } catch (Exception ex) {
            ex.printStackTrace() ;
            System.exit(1) ;
        }
        // Our parent resource (the one that created us)
        a = new ObjectAttribute("parent",
                                "org.w3c.tools.resources.Resource",
                                null,
                                Attribute.COMPUTED|Attribute.DONTSAVE);
        ATTR_PARENT = AttributeRegistry.registerAttribute(cls, a) ;
        // Our runtime context
        a = new ObjectAttribute("context",
                                "org.w3c.tools.resources.ResourceContext",
                                null,
                                Attribute.COMPUTED|Attribute.DONTSAVE);
        ATTR_CONTEXT = AttributeRegistry.registerAttribute(cls, a) ;
        // The resource store entry for that resource:
        a = new ObjectAttribute("store-entry"
                                , "java.lang.Object"
                                , null
                                , Attribute.DONTSAVE);
        ATTR_STORE_ENTRY = AttributeRegistry.registerAttribute(cls, a);
        // The identifier attribute:
        a = new StringAttribute("identifier"
                                , null
                                , Attribute.MANDATORY|Attribute.EDITABLE);
        ATTR_IDENTIFIER = AttributeRegistry.registerAttribute(cls, a);
        // The frames associated to that resource:
        a = new FrameArrayAttribute("frames"
                                    , null
                                    , Attribute.COMPUTED);
        ATTR_RESOURCE_FRAMES = AttributeRegistry.registerAttribute(cls, a);
        // Our URL
        a = new StringAttribute("url",
                                null,
```

```
                                Attribute.COMPUTED|Attribute.DONTSAVE);
        ATTR_URL = AttributeRegistry.registerAttribute(cls, a) ;
        // The last modified attribute:
        a = new DateAttribute("last-modified",
                              null,
                              Attribute.COMPUTED|Attribute.EDITABLE) ;
        ATTR_LAST_MODIFIED = AttributeRegistry.registerAttribute(cls,a);
    }

    public Object getClone(Object values[]) {
        // The frame attribute needs one more level of cloning:
        ResourceFrame f[] = (ResourceFrame[]) values[ATTR_RESOURCE_FRAMES];
        if ( f != null ) {
            ResourceFrame c[] = new ResourceFrame[f.length] ;
            for (int i = 0 ; i < f.length ; i++) {
                if ( f[i] == null )
                    c[i] = null;
                else
                    c[i] = (ResourceFrame) f[i].getClone();
            }
            values[ATTR_RESOURCE_FRAMES] = c;
        }
        return super.getClone(values);
    }

    /**
     * Get this resource parent resource.
     * The parent of a resource can be either <strong>null</strong> if it is
     * the server root resource, or any Resource.
     * @return An instance of ResourceReference, or <strong>null</strong>
     */
    public ResourceReference getParent() {
        ResourceContext context = getContext();
        if (context == null) //are we external?
            return null;
        return context.getContainer();
    }

    /**
     * Get the file part of the URL this resource is attached to.
     * @return An URL object specifying the location in the information
     *    space of this resource.
     */
    public String getURLPath() {
        return getString(ATTR_URL, null) ;
    }

    /**
     * Get this resource's help url.
     * @return An URL, encoded as a String, or <strong>null</strong> if not
     * available.
```

```
     */
    public String getHelpURL() {
        return null;
    }

    /**
     * Get the help URL for that resource's topic.
     * @param topic The topic you want help for.
     * @return A String encoded URL, or <strong>null</strong> if none
     * was found.
     */
    public String getHelpURL(String topics) {
        return null;
    }

    /**
     * Get the hierarchical context for that resource.
     * @return A ResourceContext instance, guaranteed not to be <strong>null
     * </strong>
     */
    protected ResourceContext getContext() {
        return (ResourceContext) getValue(ATTR_CONTEXT, null);
    }

    /**
     * Set the given context as the current context of this resource.
     * @param context The new context.
     */
    protected void setContext(ResourceContext context) {
        context.setResourceReference(getResourceReference());
        setValue(ATTR_CONTEXT, context);
    }

    /**
     * Set the given context as the current context of this resource.
     * @param context The new context.
     * @param keepmodules If true the new context will have the same
     * modules than the old one.
     */
    protected void setContext(ResourceContext context, boolean keepmodules) {
        context.setResourceReference(getResourceReference());
        if (keepmodules) {
            ResourceContext ctxt = getContext();
            if (ctxt != null)
                context.modules = ctxt.modules;
        }
        setValue(ATTR_CONTEXT, context);
    }

    /**
     * Get the store entry for that resource.
```

```
     * Only the resource store in charge of this resource knows about the
     * type of the resulting object. Buy declaring the class of that object
     * private, the resource store can assume some private access to it.
     * @return A java Object instance, or <strong>null</strong> if no
     * store entry is attached to that resource.
     */

    public Object getStoreEntry() {
        return getValue(ATTR_STORE_ENTRY, null);
    }

    /**
     * Get this resource identifier.
     * @return The String value for the identifier.
     */
    public String getIdentifier() {
        return getString(ATTR_IDENTIFIER, null) ;
    }

    /**
     * Get the space entry for that resource. This Object is use to
     * retrieve the resource in the resource space.
     * @return A spaceEntry instance.
     */
    protected SpaceEntry getSpaceEntry() {
        ResourceReference rr = getParent();
        if (rr != null) {
            try {
                ContainerResource cont = (ContainerResource) rr.lock();
                return new SpaceEntryImpl(cont);
            } catch (InvalidResourceException ex) {
                return null;
            } finally {
                rr.unlock();
            }
        }
        return null;
    }

    /**
     * Get the ResourceSpace where this resource is stored.
     * @return A ResourceSpace instance.
     */
    protected ResourceSpace getSpace() {
        ResourceContext context = getContext();
        if (context != null)
            return context.getSpace();
        return null;
    }

    /**
```

```
     * Get the ResourceReference of that resource. ResourceReference is the
     * only public way to access a resource.
     * @return a ResourceReference instance.
     */
    public ResourceReference getResourceReference() {
        ResourceContext context = getContext();
        if (context != null)
            return context.getResourceReference();
        return null;
    }

    /**
     * Initialize and attach a new ResourceFrame to that resource.
     * @param frame An uninitialized ResourceFrame instance.
     * @param defs A default set of attribute values.
     */
    public void registerFrame(ResourceFrame frame, Hashtable defs) {
        synchronized (this) {
            ResourceFrame frames[] = null;
            frames = (ResourceFrame[]) getValue(ATTR_RESOURCE_FRAMES, null);
            // Initialize the frame with given default attributes:
            if ( defs.get("identifier") == null )
                defs.put("identifier"
                         , "frame-"+((frames == null) ? 0 : frames.length));
            frame.initialize(defs);
            // Look for a free slot frame:
            if ( frames == null ) {
                frames    = new ResourceFrame[1];
                frames[0] = frame;
            } else {
                int slot = -1;
                // Look for a free slot:
                for (int i = 0 ; i < frames.length ; i++) {
                    if ( frames[i] == null ) {
                        slot = i;
                        break;
                    }
                }
                if ( slot >= 0 ) {
                    // Free slot available:
                    frames[slot] = frame;
                } else {
                    // Resize frames:
                    ResourceFrame nf[] = new ResourceFrame[frames.length+1];
                    System.arraycopy(frames, 0, nf, 0, frames.length);
                    nf[frames.length] = frame;
                    frames = nf;
                }
            }
            // Set the frames:
            setValue(ATTR_RESOURCE_FRAMES, frames);
```

```
        }
    }

    /**
     * Unregister a resource frame from the given resource.
     * @param frame The frame to unregister from the resource.
     */
    public synchronized void unregisterFrame(ResourceFrame frame) {
        ResourceFrame frames[] = null;
        frames = (ResourceFrame[]) getValue(ATTR_RESOURCE_FRAMES, null);
        if ( frames != null ) {
            ResourceFrame f[] = new ResourceFrame[frames.length-1];
            int j=0;
            for (int i = 0; i < frames.length ; i++) {
                if ( frames[i] == frame ) {
                    // got it, copy the end of the array
                    System.arraycopy(frames, i+1, f, j, frames.length-i-1);
                    setValue(ATTR_RESOURCE_FRAMES, f);
                    return;
                } else {
                    try {
                        f[j++] = frames[i];
                    } catch (ArrayIndexOutOfBoundsException ex) {
                        return; // no modifications, return
                    }
                }
            }
        }
    }

    /**
     * Collect all frames.
     * @return An array of ResourceFrame, containing a set of frames instances
     * or <strong>null</strong> if no resource frame is available.
     */
    public synchronized ResourceFrame[] getFrames() {
        return (ResourceFrame[]) getValue(ATTR_RESOURCE_FRAMES, null);
    }

    /**
     * Collect any frame that's an instance of the given class.
     * @param cls The class of frames we are looking for.
     * @return An array of ResourceFrame, containing a set of frames instances
     * of the given class, or <strong>null</strong> if no resource frame is
     * available.
     */
    public synchronized ResourceFrame[] collectFrames(Class c) {
        ResourceFrame frames[] = null;
        frames = (ResourceFrame[]) getValue(ATTR_RESOURCE_FRAMES, null);
        if ( frames != null ) {
            Vector v = new Vector();
```

```
            for (int i = 0 ; i < frames.length ; i++) {
                if ( c.isInstance(frames[i]) )
                    v.addElement(frames[i]);
            }
            int sz = v.size();
            if ( sz > 0 ) {
                ResourceFrame ret[] = new ResourceFrame[sz];
                v.copyInto(ret);
                return ret;
            }
        }
        return null;
    }

    /**
     * Get the first occurence of a frame of the given class.
     * @param cls The class of te frame to look for.
     * @return A ResourceFrame instance, or <strong>null</strong>.
     */
    public synchronized ResourceFrame getFrame(Class c) {
        ResourceFrame frames[] = null;
        frames = (ResourceFrame[]) getValue(ATTR_RESOURCE_FRAMES, null);
        if ( frames != null ) {
            for (int i = 0 ; i < frames.length ; i++) {
                if ( c.isInstance(frames[i]) )
                    return frames[i];
            }
        }
        return null;
    }


    /**
     * Get an attached frame attribute value.
     * This method really is a short-hand that combines a <code>getFrame</code>
     * and <code>getValue</code> method call.
     * @param cls The class of the frame we want the value of.
     * @param idx The attribute index.
     * @param def The default value (if the attribute value isn't defined).
     * @return The attribute value as an Object instance, or the provided
     * default value if the attribute value isn't defined.
     */
    public synchronized Object getValue(Class c, int idx, Object def) {
        ResourceFrame frame = getFrame(c);
        if ( frame != null )
            return frame.getValue(idx, def);
        throw new IllegalAttributeAccess(this, idx);
    }

    /**
     * Set an attached frame attribute value.
```

```
     * This method really is a short-hand that combines a <code>getFrame</code>
     * and a <code>setValue</code> method call.
     * @param cls The class of the frame we want to modify.
     * @param idx The attribute to modify.
     * @param val The new attribute value.
     */
    public synchronized void setValue(Class c, int idx, Object val) {
        ResourceFrame frame = getFrame(c);
        if ( frame != null ) {
            frame.setValue(idx, val);
            markModified();
            return;
        }
        throw new IllegalAttributeAccess(this, idx);
    }


    /**
     * Get this resource last modification time.
     * @return A long giving the date of the last modification time, or
     *    <strong>-1</strong> if undefined.
     */
    public long getLastModified() {
        return getLong(ATTR_LAST_MODIFIED, (long) -1) ;
    }


    /**
     * Mark this resource as having been modified.
     */
    public void markModified() {
        ResourceSpace space = getSpace();
        if ((space != null) && (getSpaceEntry() != null))
            space.markModified(getSpaceEntry(), this);
        super.setValue(ATTR_LAST_MODIFIED,
                       new Long(System.currentTimeMillis()));
    }

    /**
     * We overide setValue, to mark the resource as modified.
     * @param idx The index of the attribute to modify.
     * @param value The new attribute value.
     */
    public void setValue(int idx, Object value) {
        // Changing the identifier of a resource needs some special tricks:
        if ( idx == ATTR_IDENTIFIER ) {
            String oldid = getIdentifier();
            try {
                super.setValue(idx, value);
            } catch (IllegalAttributeAccess ex) {
                // We were not able to change the identifier, rethrow
```

```
                throw ex;
            }
            // Change was successfull, update the resource space:
            if (getSpaceEntry() != null) {
                ResourceSpace space = getSpace();
                space.renameResource(getSpaceEntry(), oldid, (String) value);
            }
            markModified();
            return;
        }
        // Normal setValue, but markModified before leaving:
        super.setValue(idx, value) ;
        if (( ! attributes[idx].checkFlag(Attribute.DONTSAVE) ) &&
            ( idx != ATTR_LAST_MODIFIED))
            markModified() ;
    }

    /**
     * Is that resource willing to be unloaded.
     * This method is a bit tricky to implement. The guideline is that you
     * should not dynamically change the returned value (since you can't
     * control what happens between a call to that method and a call to
     * the <code>notifyUnload</code> method).
     * <p>Returning <strong>false</strong> should never be needed, except
     * for very strange resources.
     * @return A boolean <strong>true</strong> if the resource can be unloaded
     * <strong>false</strong> otherwise.
     */

    public boolean acceptUnload() {
        if ( debugunload ) {
            try {
                System.out.println(getValue("url","<??>")+": acceptUnload");
            } catch (IllegalAttributeAccess ex) {
            }
        }
        return true;
    }

    /**
     * This resource is being unloaded.
     * The resource is being unloaded from memory, perform any additional
     * cleanup required.
     */
    public void notifyUnload() {
        if ( debugunload ) {
            try {
                System.out.println(getValue("url","<??>")+": unloaded.");
            } catch (IllegalAttributeAccess ex) {
            }
        }
```

```
        values = null ;
    }

    /**
     * The web admin wants us to update any out of date attribute.
     */
    public void updateAttributes() {
        return ;
    }

    /**
     * Check if this resource is loked more than one time.
     * @exception MultipleLockException is thrown if true.
     */
    protected void checkMultipleLock(ResourceReference rr)
        throws MultipleLockException
    {
        if (rr.nbLock() > 1)
            throw new MultipleLockException(rr.nbLock(), this, "can't delete");
    }

    /**
     * Delete this Resource instance, and remove it from its store.
     * This method will erase definitely this resource, for ever, by removing
     * it from its resource store (when doable).
     */
    public synchronized void delete()
        throws MultipleLockException
    {
        ResourceSpace space = getSpace();

        if ((space != null) && (getSpaceEntry() != null)) {
            ResourceReference self = getResourceReference();
            if (self != null) {
                synchronized (self) {
                    checkMultipleLock(self);
                    space.deleteResource(getSpaceEntry(), this);
                }
            } else {
                space.deleteResource(getSpaceEntry(), this);
            }
        }
    }

    public boolean isInitialized() {
        return (values != null);
    }

    /**
     * Create an empty resource instance.
     * Initialize the instance attributes description, and its values.
```

```
     */
    public Resource() {
        super() ;
    }
}
```

Jigsaw
Indexers Index

Jigsaw Home / Documentation Overview / Resources - Frames - Client Side Components

The main goal of an indexer is to create and setup some resource automatically. The resources can be created depending on their name or their extension. Once the resource has been created, the indexer is also in charge of attaching the right frames to this resource, like the HTTP frame, the filters and so on.

Standard Indexers

SampleResourceIndexer

The Basic Indexer class

GhostResourceIndexer

deprecated

ContentTypeIndexer

Indexation based on the Content-Type Header of incomming PUT requests

Extension Indexers

ZipIndexer

Used only for zip browzing

ServletIndexer

Indexer dedicated to servlets

Jigsaw Team

$Id: indexers.html,v 1.1 1999/03/30 12:12:08 bmahe Exp $

Jigsaw
Client Side Component Index

Jigsaw Home / Documentation Overview / Resources - Frames - Indexers

HttpManager

The HttpManager is the class that handles HTTP requests.

Cookie Support

The Cookie Filter provides client side cookie support for the HTTP protocol.

Cache Support

The Cache Filter provides client side caching support for the HTTP protocol.

ICP Support

The ICP client side filter add support for the Internet Cache Protocol to the Jigsaw HTTP client side implementation.

Proxy Dispatcher

The ProxyDispatcher is a filter that allows some rule to be applied to some given request before the HTTP client side API emits out a request.

Jigsaw Team

$Id: client-side.html,v 1.3 1999/03/30 09:38:58 bmahe Exp $

All resources All frames

PassDirectory

The PassDirectory resource implements exactly the same functionality than the DirectoryResource, except that it is able to serve a file system directory not located below the server's space directory.

Its pass-target attribute indicates the location of the physical directory it should export. Using this resource means that the server exported space is no more confined to its space directory.

Inherits

The PassDirectory class inerits from the following classes:

- Resource
- FramedResource
- AbstractContainer
- ContainerResource
- DirectoryResource

Attributes description

The PassDirectory defines the following attributes:

- pass-target

`pass-target`

semantics

The physical directory that this resource exports. If the provided path is not absolute, then the server will turn it into an absolute path by taking the provided relative path and absolutizing it from the server's root directory.

type

This attribute is an editable FileAttribute

default value

This attribute defaults to **null**.

Jigsaw Team
$Id: org.w3c.tools.resources.PassDirectory.html,v 1.3 1998/03/27 08:25:46 bmahe Exp $

All resources All frames

PassDirectory

The PassDirectory resource implements exactly the same functionality than the DirectoryResource, except that it is able to serve a file system directory not located below the server's space directory.

Its pass-target attribute indicates the location of the physical directory it should export. Using this resource means that the server exported space is no more confined to its space directory.

Inherits

The PassDirectory class inerits from the following classes:

- Resource
- FramedResource
- AbstractContainer
- ContainerResource
- DirectoryResource
- DirectoryResource

Attributes description

The PassDirectory defines the following attributes:

- pass-target

`pass-target`

semantics

The physical directory that this resource exports. If the provided path is not absolute, then the server will turn it into an absolute path by taking the provided relative path and absolutizing it from the server's root directory.

type

This attribute is an editable FileAttribute

default value

This attribute defaults to **null**.

Jigsaw Team

$Id: org.w3c.jigsaw.resources.PassDirectory.html,v 1.3 1998/03/27 08:22:56 bmahe Exp $

All resources All frames

HttpDirectoryResource

A DirectoryResource that auto register a HTTPFrame.

Inherits

The HttpDirectoryResource class inerits from the following classes:

- Resource
- FramedResource
- AbstractContainer
- ContainerResource
- DirectoryResource
- DirectoryResource

Attributes description

The HttpDirectoryResource doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.jigsaw.resources.HttpDirectoryResource.html,v 1.3 1998/03/27 08:22:24 bmahe Exp $

All resources All frames

HttpPassDirectory

A PassDirectory that auto register a HTTPFrame.

Inherits

The HttpPassDirectory class inerits from the following classes:

- Resource
- FramedResource
- AbstractContainer
- ContainerResource
- DirectoryResource
- DirectoryResource
- PassDirectory

Attributes description

The HttpPassDirectory doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.jigsaw.resources.HttpPassDirectory.html,v 1.3 1998/03/27 08:22:46 bmahe Exp $

All resources All frames

HttpFileResource

A FileResource that auto register a HTTPFrame.

Inherits

The HttpFileResource class inerits from the following classes:

- Resource
- FramedResource
- FileResource

Attributes description

The HttpFileResource doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.jigsaw.resources.HttpFileResource.html,v 1.3 1998/03/27 08:22:34 bmahe Exp $

All resources All frames

VirtualHostResource

Just a base for the VirtualHostFrame.

Inherits

The VirtualHostResource class inerits from the following classes:

- Resource
- FramedResource
- AbstractContainer
- ContainerResource

Attributes description

The VirtualHostResource doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.jigsaw.resources.VirtualHostResource.html,v 1.3 1999/09/08 14:58:11 ylafon Exp $

All resources All frames

VirtualHostFrame

This frame will allow you to do multihoming. This technic is a way to host multiple logical web servers on the same physical process. Jigsaw supports only Host header based virtual hosting, which does not consume IP addresses.

As for other frames that are suposed to be attached to the root resource of your server, you should be very cautious when using that kind of frame. In particular make sure the followup attribute is set to a resource which can be a root resource.

Each child of the resource associated to the VirualHostFrame must be named under the host name it will serve. If you have a machine A with names name.foo.com and alias.foo.com, and if the server is running on port 8001, then requests prefixed by http://name.foo.com:8001/ will be dispatched to the child resource of the associated resource named name.foo.com:8001 and requests prefixed by http://alias.foo.com:8001 to child resource alias.foo.com:8001.

Warning: Unless you understand what you are doing, children of that resources should only be either MirrorDirectory or PassDirectory instances.

Note: Must be associated with a VirtualHostResource.

Inherits

The VirtualHostFrame class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ProtocolFrame
- HTTPFrame

Attributes description

The VirtualHostFrame defines the following attributes:

- followup

`followup`

semantics

The deafult resource for handling unknown hosts. This attribute should provide the name of a resource existing within the root resource store of the server.

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

Jigsaw Team

$Id: org.w3c.jigsaw.frames.VirtualHostFrame.html,v 1.3 1998/03/27 08:19:39 bmahe Exp $

All resources All frames

MapResource

Implements handling of image map requests. Understands both the NCSA and the W3C (CERN) format for map description files, with the "nosearch" extension for both. Allways associated with MapFrame.

Inherits

The MapResource class inerits from the following classes:

- Resource
- FramedResource
- FileResource

Attributes description

The MapResource defines the following attributes:

- useNCSA

`useNCSA`

semantics

Should this resource handles NCSA image maps instead of CERN?

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **false**.

Jigsaw Team

$Id: org.w3c.jigsaw.map.MapResource.html,v 1.3 1998/03/27 08:20:29 bmahe Exp $

All resources All frames

DirectoryLister

A resource which display the directory listing on its parent (DirectoryResource). Allways associated with a DirectoryListerFrame.

Inherits

The DirectoryLister class inerits from the following classes:

- Resource
- FramedResource

Attributes description

The DirectoryLister doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.jigsaw.resources.DirectoryLister.html,v 1.3 1998/03/27 08:21:50 bmahe Exp $

All resources All frames

ServletWrapper

This resource allows you to embed Servlets in Jigsaw. The ServletWrapper resource works in conjunction with a DirectoryResource (or a subclass) associated with a ServletDirectoryFrame , that acts as a context for a set of related servlets. You can install multiple servlet directory resources within the same Jigsaw process. The servletWrapper auto register a ServletWrapperFrame.

To install a servlet within Jigsaw, you need to install an instance of the ServletWrapper resource in the Jigsaw URL space, and then configure it and tell it what servlet class it should embed (by using the servlet-class parameter).

Inherits

The ServletWrapper class inerits from the following classes:

- Resource
- FramedResource

Attributes description

The ServletWrapper defines the following attributes:

- servlet-class
- servlet-parameters
- auto-reload
- servlet-timeout

`servlet-class`

semantics

The name of the main class for the servlet. That name should resolve to a subclass of java.servlet.Servlet. The servlet wrapper will only run the servlet when doable, it will otherwise (when the servlet class is not instantiable, or some other servlet initialization error occurs) emit an error message back to the client and report the problem in **Jigsaw**'s main error log.

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

`servlet-parameters`

semantics

This attribute lists the servlet initialization parameters. Each attribute has a name and a value, both of them are instances of String.

type

This attribute is an editable PropertiesAttribute

default value

This attribute defaults to **null**.

`auto-reload`

semantics

Should this wrapper reload modified servlet class?

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **true**.

`servlet-timeout`

semantics

Max idle time before a servlet is destroyed (in milliseconds), the servlet will never be destroyed if equals to -1.

type

This attribute is an editable LongAttribute

default value

This attribute defaults to **null**.

Jigsaw Team

$Id: org.w3c.jigsaw.servlet.ServletWrapper.html,v 1.3 1998/03/27 08:23:42 bmahe Exp $

All resourcesAll frames

RemoteServletWrapper

This ServletWrapper allows you to call a remote servlet.

Inherits

The RemoteServletWrapper class inerits from the following classes:

- Resource
- FramedResource
- ServletWrapper

Attributes description

The RemoteServletWrapper defines the following attributes:

- servlet-base

`servlet-base`

semantics

The URL of the remote sevlet class, if the servlet class URL is `http://remotehost/servlet/RemoteServlet.class`, the servlet-base is `http://remotehost/servlet/.`

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

Jigsaw Team

$Id: org.w3c.jigsaw.servlet.RemoteServletWrapper.html,v 1.2 1999/09/08 14:58:11 ylafon Exp $

All resources All frames

PutListResource

This is the PutListFrame associated resource (auto-register the PutListFrame). This resource manages a list of last puted documents and provides a way to publish them with CVS.

Inherits

The PutListResource class inerits from the following classes:

- Resource
- FramedResource

Attributes description

The PutListResource defines the following attributes:

- file
- space
- root

`file`

semantics

The file used to store the modification list.

type

This attribute is an editable FileAttribute

default value

This attribute defaults to **null**.

`space`

semantics

The user's local space.

type

This attribute is an editable FileAttribute

default value

This attribute defaults to **null**.

`root`

semantics

The web server public space.

type

This attribute is an editable FileAttribute

default value

This attribute defaults to **null**.

Jigsaw Team

$Id: org.w3c.jigsaw.filters.PutListResource.html,v 1.3 1998/03/27 08:18:33 bmahe Exp $

All resources All frames

AutoLookupDirectory

A directory resource that will allow you to fetch non-existing files from CVS. If a resource is not present, it will try to get it from cvs, if it does not exist, it will act like a normal DirectoryResource.

Inherits

The AutoLookupDirectory class inerits from the following classes:

- Resource
- FramedResource
- AbstractContainer
- ContainerResource
- DirectoryResource
- DirectoryResource

Attributes description

The AutoLookupDirectory defines the following attributes:

- autoupdate

`autoupdate`

semantics

Should the entries be always refreshed when requested? Note that if set to true, it will consume time and the performance of the server will collapse.

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **false**.

Jigsaw Team

$Id: org.w3c.jigsaw.cvs.AutoLookupDirectory.html,v 1.3 1998/03/27 08:15:30 bmahe Exp $

All resources All frames

ToolsLister

This is the ToolsListerFrame associated resource. Its only feature is to auto-register the ToolsListerFrame.

Inherits

The ToolsLister class inerits from the following classes:

- Resource
- FramedResource

Attributes description

The ToolsLister doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.jigsaw.cvs.ToolsLister.html,v 1.3 1998/03/27 08:16:07 bmahe Exp $

All resources All frames

LabelBureauResource

A **Jigsaw** resource to query a Label bureau. This conforms to the PICS protocol specification. Read the pics documentation for more details on setting up a LabelBureau.

Inherits

The LabelBureauResource class inerits from the following classes:

- Resource
- FramedResource

Attributes description

The LabelBureauResource defines the following attributes:

- bureau
- services
- debug

`bureau`

semantics

The absolute path of the services/labels file system database.

type

This attribute is an editable FileAttribute

default value

This attribute defaults to **null**.

`services`

semantics

The list of all the services available in this bureau.

type

This attribute is an editable StringArrayAttribute

default value

This attribute defaults to **null**.

---

debug

semantics

The debug flag.

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **false**.

Jigsaw Team

$Id: org.w3c.jigsaw.pics.LabelBureauResource.html,v 1.5 1999/09/08 14:58:11 ylafon Exp $

All resources All frames

ZipDirectoryResource

This resource allows you to browse a zip file. See the tutorial.

Inherits

The ZipDirectoryResource class inerits from the following classes:

- Resource
- FramedResource
- AbstractContainer
- ContainerResource
- DirectoryResource

Attributes description

The ZipDirectoryResource doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.jigsaw.zip.ZipDirectoryResource.html,v 1.2 1999/09/08 14:58:11 ylafon Exp $

All resources All frames

ZipFileResource

This resource allows you to browse a zip file. See the tutorial.

Inherits

The ZipFileResource class inerits from the following classes:

- Resource
- FramedResource
- FileResource

Attributes description

The ZipFileResource doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.jigsaw.zip.ZipFileResource.html,v 1.2 1999/09/08 14:58:11 ylafon Exp $

All resources All frames

CheckpointResource

...description...

Inherits

The CheckpointResource class inerits from the following classes:

- Resource
- FramedResource

Attributes description

The CheckpointResource defines the following attributes:

- interval
- thread-priority
- flush-logs
- flush-properties
- flush-configuration
- trace-checkpoint

`interval`

semantics

....

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **60**.

`thread-priority`

semantics

....

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **2**.

`flush-logs`

semantics

....

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **false**.

`flush-properties`

semantics

....

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **false**.

`flush-configuration`

semantics

....

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **true**.

`trace-checkpoint`

semantics

....

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **false**.

Jigsaw Team

$Id: org.w3c.jigsaw.resources.CheckpointResource.html,v 1.5 1999/09/08 14:58:11 ylafon Exp $

All resources All frames

PasswordEditor

This is the PasswordEditorFrame associated resource. Its only feature is to auto-register the PasswordEditorFrame.

Inherits

The PasswordEditor class inerits from the following classes:

- Resource
- FramedResource

Attributes description

The PasswordEditor doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.jigsaw.resources.PasswordEditor.html,v 1.3 1998/03/27 08:23:09 bmahe Exp $

All resources All frames

UnixProp

This resource provides editable access to UNIX specific properties. As for the other property sheets, the resource just provides a nice way of describing a set of resources, and the way they should be edited, so each attribute below, really corresponds to a property defined in the property file of **Jigsaw**.

Warning: because failure to set any of the below properties can result in severe security problems, **Jigsaw** will abort whenever one of them is set, but the appropriate system call has failed. Check **Jigsaw**'s error log for more informations about fixing any troubles.

Inherits

The UnixProp class inerits from the following classes:

- Resource
- PropertySet

Attributes description

The UnixProp defines the following attributes:

- org.w3c.jigsaw.unix.group
- org.w3c.jigsaw.unix.user

`org.w3c.jigsaw.unix.group`

semantics

The name of the group that the server should run as.

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

`org.w3c.jigsaw.unix.user`

semantics

The name of the user that the server should run as.

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **null**.

Jigsaw Team

$Id: org.w3c.jigsaw.http.UnixProp.html,v 1.2 1998/03/27 08:20:07 bmahe Exp $

All resources All frames

ConnectionProp

This resource class provides access to **Jigsaw** connection properties. It allows you to change the connection properties of **Jigsaw**.

Inherits

The ConnectionProp class inerits from the following classes:

- Resource
- PropertySet

Attributes description

The ConnectionProp defines the following attributes:

- org.w3c.jigsaw.keepAlive
- org.w3c.jigsaw.client.priority
- org.w3c.jigsaw.client.bufsize
- org.w3c.jigsaw.client.debug
- org.w3c.jigsaw.request.timeout

`org.w3c.jigsaw.keepAlive`

semantics

The Keep-Alive flag.

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **null**.

`org.w3c.jigsaw.client.priority`

semantics

Client's threads priority.

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **null**.

`org.w3c.jigsaw.client.bufsize`

semantics

Client's buffer size.

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **null**.

`org.w3c.jigsaw.client.debug`

semantics

Client's debug flag.

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **null**.

`org.w3c.jigsaw.request.timeout`

semantics

Request time out.

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **null**.

Jigsaw Team

$Id: org.w3c.jigsaw.http.ConnectionProp.html,v 1.4 1999/09/08 14:58:11 ylafon Exp $

All resources All frames

SocketConnectionProp

This resource provides editable access to connection properties specific to the raw socket access to **Jigsaw**. As for the other property sheets, the resource just provides a nice way of describing a set of resources, and the way they should be edited, so each attribute below, really corresponds to a property defined in the property file of **Jigsaw**.

Tunning these resources can be a very difficult exercise. It is recommended that you try very hard to understand these properties before playing with them.

Inherits

The SocketConnectionProp class inerits from the following classes:

- Resource
- PropertySet

Attributes description

The SocketConnectionProp defines the following attributes:

- org.w3c.jigsaw.http.socket.SocketClientFactory.minFree
- org.w3c.jigsaw.http.socket.SocketClientFactory.maxFree
- org.w3c.jigsaw.http.socket.SocketClientFactory.maxIdle
- org.w3c.jigsaw.http.socket.SocketClientFactory.maxClients
- org.w3c.jigsaw.http.socket.SocketClientFactory.idleTimeout
- org.w3c.jigsaw.http.socket.SocketClientFactory.maxThreads

`org.w3c.jigsaw.http.socket.SocketClientFactory.minFree`

semantics

The number of free (not working) clients the server has to have in order to consider itself under very light load. When the server is running under that load, the server will keep connection opens

as long as possible (with no time out).

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **5**.

`org.w3c.jigsaw.http.socket.SocketClientFactory.maxFree`

semantics

The number of free (not working) clients the server has to have in order to go from the normal load state back into the light load state. The difference between that number and minFree avoids having the server yo-yoing from the light load state to the normal load state.

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **10**.

`org.w3c.jigsaw.http.socket.SocketClientFactory.maxIdle`

semantics

The maximum number of idle client. An idle client thread is a thread waiting for the next request on a persistent connection. When the maxIdle number of idle client threads is reached, the server turns itself either into high load (if it still has free clients), or dead load otherwise. Under normal load, the client pool will make sure to shutdown at least one persistent connection before accepting a new one. Under high load the server will start adjusting thread prorities (it will lower the accepting thread's priority under the normal client's thread priority, to give more CPU to client threads). On dead load, the server will reject incomming connections.

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **20**.

`org.w3c.jigsaw.http.socket.SocketClientFactory.maxClients`

semantics

The maximum simultaneous number of client connections the server will ever accept.

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **32**.

`org.w3c.jigsaw.http.socket.SocketClientFactory.idleTimeout`

semantics

The timeout that will indicate that a thread is no longer usefull. A minmum number of threads will always remain alive, even though this timeout value is provided.

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **10000**.

`org.w3c.jigsaw.http.socket.SocketClientFactory.maxThreads`

semantics

The maximum number of allowed threads, whatever happens.

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **40**.

Jigsaw Team

$Id: org.w3c.jigsaw.http.socket.SocketConnectionProp.html,v 1.2 1998/03/27 08:20:18 bmahe Exp $

All resources All frames

LoggingProp

This resource class provides access to **Jigsaw** logging properties. It allows you to change the logging properties of **Jigsaw**.

Inherits

The LoggingProp class inerits from the following classes:

- Resource
- PropertySet

Attributes description

The LoggingProp defines the following attributes:

- org.w3c.jigsaw.logger
- org.w3c.jigsaw.logger.errlogname
- org.w3c.jigsaw.logger.logname
- org.w3c.jigsaw.logger.tracelogname
- org.w3c.jigsaw.logger.bufferSize

`org.w3c.jigsaw.logger`

semantics

This property gives the name of the logger class to use. The logger class can be any class that implements the logger interface. Right now, the only implemented logger is the CommonLogger. If undefined, **Jigsaw** will not log any requests, and emit a warning at startup time.

type

This attribute is an editable ClassAttribute

default value

This attribute defaults to **null**.

`org.w3c.jigsaw.logger.errlogname`

semantics

This property should give the name of the error log file. The error log file is used when some abnormal situation is encoutered inside the server. Abnormal, here, means a situation that the server can't cope with (i.e., implementation bugs), rather than just HTTP errors (which are logged in the above log file). This file is also used by the server to trace important configuration actions.

type

This attribute is an editable FileAttribute

default value

This attribute defaults to <*w3c.jigsaw.root*>/logs/errlog

`org.w3c.jigsaw.logger.logname`

semantics

This property should give the name of the log file to which the logging record will be emited. Note that this property is a property of the CommonLogger (and any of its subclasses), rather than a property of the logging system as a whole.

type

This attribute is an editable FileAttribute

default value

This attribute defaults to <*w3c.jigsaw.root*>/logs/log

`org.w3c.jigsaw.logger.tracelogname`

semantics

This property should provide the name of the trace log. The trace log is used to output debugging information when the server is turned into debug mode. To turn the server into debug mode, two properties have to be turned to true (depending on what part of the server you want to debug):

- *org.w3c.jigsaw.trace* Make the core server emit traces.
- *org.w3c.jigsaw.client.debug* Make every clients emit traces.

type

This attribute is an editable FileAttribute

default value

This attribute defaults to <*w3c.jigsaw.root*>/logs/tracelog.

`org.w3c.jigsaw.logger.bufferSize`

semantics

The size of the log buffer to use. This property defines the size of the buffer used before flushing the output log. It is highly recommended to keep this value at least as high as it is. You may still, however, set it to 0 in order to prevent any buffering of log writing.

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **8192**.

Jigsaw Team

$Id: org.w3c.jigsaw.http.LoggingProp.html,v 1.2 1998/03/27 08:19:56 bmahe Exp $

All resources All frames

GeneralProp

This resource provides access to **Jigsaw**'s general properties.

Inherits

The GeneralProp class inerits from the following classes:

- Resource
- PropertySet

Attributes description

The GeneralProp defines the following attributes:

- org.w3c.jigsaw.server
- org.w3c.jigsaw.checkSensitivity
- org.w3c.jigsaw.root
- org.w3c.jigsaw.host
- org.w3c.jigsaw.port
- org.w3c.jigsaw.root.name
- org.w3c.jigsaw.publicMethods
- org.w3c.jigsaw.trace
- org.w3c.jigsaw.docurl
- org.w3c.jigsaw.checkpointer

`org.w3c.jigsaw.server`

semantics

The string that **Jigsaw** will emit as its identification through the Server header.

type

This attribute is an editable StringAttribute

default value

This attribute defaults to Jigsaw/major.minor, where major is the current major release number, and minor is the minor one.

`org.w3c.jigsaw.checkSensitivity`

semantics

Should **Jigsaw** double check case sensitivity when accessing files. On non-broken systems, the underlying file system is case-sensitive and the whole world is very happy. However a few, old operating systems have not incorporated that feature yet. For those system, turning this falg to true will ensure that the URLs served by **Jigsaw** remains case-sensitive.

type

This attribute is an editable BooleanAttribute

default value

Because they are security implications related to not having this flag set when running on a case-insensitive file system, this attribute defaults to **true**. It is highly recommended that you turn it to **false** when possible.

`org.w3c.jigsaw.root`

semantics

This property should provide the top directory of the server. This property, by itself, is only used to get defaults value for other

properties. A typical top directory for a server will contain the following sub directories:

- **config** which will contain the required configuration files and directories, among which:
  - server.props, *the file containing the servers to launch list.*
  - admin-server.props, *the file containing the default setting for the admin sever.*
  - http-server.props, *the file containing the default setting for the http sever.*
  - jigadm.zip *the ServerBrowser configuration file.*
  - indexers *the indexers directory.*
  - icons *the icons for ServerBrowser.*
  - auth *the authentication directory containing the realms.*
  - stores *the directory where resources will be saved.*
- **configadm** which will contains the required configuration directories for jigadmin:
  - auth *the authentication directory containing the realms.*
  - stores *the directory where resources will be saved.*
- **logs** which will contain the log files.
- **WWW** which will contain the actual entities to be exported, along with their description.
- **bin** will optionally contain some scripts to run the binaries associated with Jigsaw.

type

This attribute is an editable FileAttribute

default value

This property defaults to Jigsaw's current directory.

`org.w3c.jigsaw.host`

semantics

This property should provide the full name (including the domain name) of the machine that hosts the server. Don't forget to provide the host's domain-name along with its name, otherwise, all urls dynamically generated by the server will fail when trigered from a machine outside the server's domain.

type

This attribute is an editable StringAttribute

default value

This property defaults to the machine on which the server runs.

`org.w3c.jigsaw.port`

semantics

The port on which the server will listen for client connections. Unless you run the server as root, you won't be able to run the server on a port lesser than 1024. This will be fixed in future releases.

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **8001**.

`org.w3c.jigsaw.root.name`

semantics

The name of the resource to be used as the server's root resource. This name should be a valid resource name within the scope of the space directory resource store (see above). This property is used, for example, to turn the server into a proxy (by giving the name of some ProxyDirectory resource, instead of the default name of the DirectoryResource for the space directory).

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **root**.

`org.w3c.jigsaw.publicMethods`

semantics

The set of public methods supported by that server. This set of methods is being exported through

the HTTP Public header.

type

This attribute is an editable StringArrayAttribute

default value

This attribute defaults to **{GET, HEAD, PUT, POST, LINK, UNLINK, OPTIONS, DELETE}**.

`org.w3c.jigsaw.trace`

semantics

Turn the server in trace mode. This flag wil lmake the server emit various traces into the logger trace file, along with some debugging information to the server's standard output stream. This flag has to be turned on for clients trace flag to be effective.

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **false**.

`org.w3c.jigsaw.docurl`

semantics

The URL path of the Jigsaw's documentation, as served by that server. This can be either a relative URL (it will be absolutized relative to the server's root URL), or an absolute URL.

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

`org.w3c.jigsaw.checkpointer`

semantics

The checkpointer URL, this is the resource to activate to starts CheckPoint. (see CheckPointResource)

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

Jigsaw Team

$Id: org.w3c.jigsaw.http.GeneralProp.html,v 1.2 1998/03/27 08:19:45 bmahe Exp $

All resources All frames

ProxyProp

This class provides editable access to the client-side HTTP API that **Jigsaw** uses to proxy requests.

Inherits

The ProxyProp class inerits from the following classes:

- Resource
- PropertySet

Attributes description

The ProxyProp defines the following attributes:

- org.w3c.www.protocol.http.connections.max
- org.w3c.www.protocol.http.connections.timeout
- proxySet
- proxyHost
- proxyPort
- org.w3c.www.protocol.http.filters

`org.w3c.www.protocol.http.connections.max`

semantics

The maximum number of connections the client side API is allowed to use to external servers.

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **5.**

`org.w3c.www.protocol.http.connections.timeout`

semantics

The timeout on the client socket

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **300000**.

`proxySet`

semantics

Indicates to use another proxy instead of going to origin servers. The proxy to use is defined by proxyHost and proxyPort.

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **null**.

`proxyHost`

semantics

When the proxySet flag is set to **true**, this attribute indicates the internet name of the proxy to use.

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

`proxyPort`

semantics

When the proxySet flag is set to **true**, this attribute indicates the port number on which the intermediate proxy is listening.

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **80**.

`org.w3c.www.protocol.http.filters`

semantics

The list of filters you want to run on the client side HTTP API. A filter is a class then can catch requests before they leave the proxy and get replies has they come back. The currently available filters available for the proxy is:

- w3c.www.protocol.http.cache.CacheFilter *A filter that provides HTTP/1.1 caching.*
- w3c.www.protocol.http.DebugFilter *A filter that will dump outgoing requests and incoming replies to the standard output.*

type

This attribute is an editable StringArrayAttribute

default value

This attribute has no default value. It is highly recommended that you use at least the **w3c.www.protocol.http.cache.CacheFilter** filter to provide caching functionality.

Jigsaw Team

$Id: org.w3c.jigsaw.proxy.ProxyProp.html,v 1.2 1998/03/27 08:21:28 bmahe Exp $

All resources All frames

ProxyFrame

This class implements a full blown caching-proxy module for **Jigsaw**. It relies on the w3c's HTTP client side API to handle both request forwarding and reply caching.

If the handle-ftp attribute is set to **true** this resource will also fullfil FTP requests. It will not cache the result, however. Redirectin FTP traffic to an upward proxy can be done through the following (standard) Java properties (to be set in the config/server.props file):

```
ftpProxyPort=8080

ftpProxyHost=cache-sop.inria.fr

ftpProxySet=true
```

Warning: Do not try to use this resource class in conjunction with the VirtualHostFrame class: HTTP specification doesn't allow to have virtual proxies.

Inherits

The ProxyFrame class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ProtocolFrame
- HTTPFrame
- ForwardFrame

Attributes description

The ProxyFrame defines the following attributes:

- handle-ftp

`handle-ftp`

semantics

Should the proxy also handle FTP urls ? If set to true the proxy will handle requests targeted toward ftp servers.

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **false**.

Jigsaw Team

$Id: org.w3c.jigsaw.proxy.ProxyFrame.html,v 1.3 1998/03/27 08:21:14 bmahe Exp $

All resources All frames

CacheProp

This resource class provides editable access to the cache configuration. It's use is to allow easy customization of Jigsaw when used as a proxy or as a mirror.

Inherits

The CacheProp class inerits from the following classes:

- Resource
- PropertySet

Attributes description

The CacheProp defines the following attributes:

- org.w3c.www.protocol.http.cache.size
- org.w3c.www.protocol.http.cache.debug
- org.w3c.www.protocol.http.cache.shared
- org.w3c.www.protocol.http.cache.connected
- org.w3c.www.protocol.http.cache.garbageCollectionEnabled
- org.w3c.www.protocol.http.cache.fileSizeRatio

`org.w3c.www.protocol.http.cache.size`

semantics

The size of the cache, given as the number of bytes that the cache is allowed to use on the disk.

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **5000000**.

`org.w3c.www.protocol.http.cache.debug`

semantics

Turn the cache filter debug mode on or off.

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **false**.

`org.w3c.www.protocol.http.cache.shared`

semantics

Is this cache shared among multiple users ?

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **true**.

`org.w3c.www.protocol.http.cache.connected`

semantics

Tell the cache wether it is allowed to use the net to answer to any request. This has the same effect as if all requests served by the cache had the only-if-cache cache control directive set.

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **true**.

`org.w3c.www.protocol.http.cache.garbageCollectionEnabled`

semantics

Enables or disables the garbage collector of the cache. This is most usefull when you are planning to use the proxy in disconnected mode: turn **off** this flag while filling in the cache, then disconnect it by using the org.w3c.www.protocol.http.cache.connected property.

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **true**.

`org.w3c.www.protocol.http.cache.fileSizeRatio`

semantics

Indicates the ratio - relative to the cache size - of the number of bytes a single entry in the cache is

allowed to occupy. For example, if set to **0**.1 and the cache size is set to **5000000**, then any document whose length is greater then **500000** will not be cached. This ratio is given as a floating point value between **0** and **1**.

type

This attribute is an editable DoubleAttribute

default value

This attribute defaults to **0.1**.

Jigsaw Team

$Id: org.w3c.jigsaw.proxy.CacheProp.html,v 1.2 1998/03/27 08:20:43 bmahe Exp $

All resources All frames

CvsProp

Gives editable access to Jigsaw cvs properties, when CvsFrame is used somewhere.

Inherits

The CvsProp class inerits from the following classes:

- Resource
- PropertySet

Attributes description

The CvsProp defines the following attributes:

- org.w3c.cvs.path
- org.w3c.cvs.root
- org.w3c.cvs.wrapper

`org.w3c.cvs.path`

semantics

This property should provide the absolute path of the cvs program.

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

`org.w3c.cvs.root`

semantics

This property should give the absolute path of the CVS repository.

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

`org.w3c.cvs.wrapper`

semantics

This property should provide the absolute path of the cvs wrapper program. In order to run cvs in the appropriate directory, the CvsDirectoryResource runs cvs through a shell script. This shell script is provided with Jigsaw distribution (in the w3c.cvs package).

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

Jigsaw Team

$Id: org.w3c.jigsaw.cvs.CvsProp.html,v 1.1 1998/03/27 10:47:35 bmahe Exp $

All resources All frames

CvsFrame

The CvsDirectoryResource allows you to control the CVS status of the files under some directory (which itself, should be under CVS control). This resource allows you to add, remove, update and commit files. It also provides you with the log and the diff with the latest version of any file.

The best way to use it, is through a directory template, that maps any CVS directory to this resource .

Inherits

The CvsFrame class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ProtocolFrame
- HTTPFrame
- PostableFrame

Attributes description

The CvsFrame doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.jigsaw.cvs.CvsFrame.html,v 1.3 1998/03/27 08:15:55 bmahe Exp $

All resources All frames

ServletProps

This resource provides access to **Jigsaw**'s servlet properties.

Inherits

The ServletProps class inerits from the following classes:

- Resource
- PropertySet

Attributes description

The ServletProps defines the following attributes:

- org.w3c.jigsaw.servlet.servlet-log-file
- org.w3c.jigsaw.servlet.servlet-timeout
- org.w3c.jigsaw.servlet.sessions-max-idle-time
- org.w3c.jigsaw.servlet.max-sessions
- org.w3c.jigsaw.servlet.sessions-sweep-delay
- org.w3c.jigsaw.servlet.session.cookie.name
- org.w3c.jigsaw.servlet.session.cookie.path
- org.w3c.jigsaw.servlet.session.cookie.domain
- org.w3c.jigsaw.servlet.session.cookie.comment
- org.w3c.jigsaw.servlet.session.cookie.maxage
- org.w3c.jigsaw.servlet.session.cookie.secure

`org.w3c.jigsaw.servlet.servlet-log-file`

semantics

The file where Jigsaw is supposed to write servlets log.

type

This attribute is an editable FileAttribute

default value

This attribute defaults to **<Jigsaw-dir>/Jigsaw/Jigsaw/logs/servlets.**

`org.w3c.jigsaw.servlet.sessions-max-idle-time`

semantics

Amount of time a session is allowed to go unused before it is invalidated. Value is specified in milliseconds.

type

This attribute is an editable LongAttribute

default value

This attribute defaults to **1800000**.

`org.w3c.jigsaw.servlet.max-sessions`

semantics

Max number of sessions in memory, if the number of sessions exceeds this number the sessions with the bigest idle time will be invalidated. This is checked each time a session is added.

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **1024**.

`org.w3c.jigsaw.servlet.sessions-sweep-delay`

semantics

Time interval when Jigsaw checks for sessions that have gone unused long enough to be invalidated. Value is an integer, specifying the interval in milliseconds.

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **30000**.

`org.w3c.jigsaw.servlet.session.cookie.name`

semantics

The name of the cookie carrying the session ID.

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **JIGSAW-SESSION-ID**.

`org.w3c.jigsaw.servlet.session.cookie.path`

semantics

The path of the cookie carrying the session ID. (optionnal)

type

This attribute is an editable StringAttribute

default value

This attribute defaults to /.

`org.w3c.jigsaw.servlet.session.cookie.domain`

semantics

The domain field of the cookie carrying the session ID. (optionnal)

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

`org.w3c.jigsaw.servlet.session.cookie.comment`

semantics

The comment field of the cookie carrying the session ID. (optionnal)

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **Jigsaw Server Session Tracking Cookie**.

`org.w3c.jigsaw.servlet.session.cookie.maxage`

semantics

The value of the maximum age of the cookie carrying the session ID. Default value is 86400 seconds (24h). No maxage is set if equals to -1.

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **86400**.

`org.w3c.jigsaw.servlet.session.cookie.secure`

semantics

The secure field of the cookie carrying the session ID.

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **false**.

`org.w3c.jigsaw.servlet.servlet-timeout`

semantics

Max idle time before a servlet is destroyed (in milliseconds), the servlet will never be destroyed if equals to -1.

type

This attribute is an editable LongAttribute

default value

This attribute defaults to **null**.

Jigsaw Team

$Id: org.w3c.jigsaw.servlet.ServletProps.html,v 1.3 1999/03/19 15:46:52 bmahe Exp $

All resources All frames

PageCompileProp

Gives editable access to Jigsaw Page Compilation properties, when PageCompileFrame is used somewhere.

Inherits

The PageCompileProp class inerits from the following classes:

- Resource
- PropertySet

Attributes description

The PageCompileProp defines the following attributes:

- org.w3c.jigsaw.pagecompile.dir
- org.w3c.jigsaw.pagecompile.compiler

`org.w3c.jigsaw.pagecompile.dir`

semantics

Specify the directory for storing generated class file.

type

This attribute is an editable FileAttribute

default value

This attribute defaults to **<instdir>/Jigsaw/Jigsaw/compiledPages/** .

`org.w3c.jigsaw.pagecompile.compiler`

semantics

Specify the compiler used to compile generated frames.

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **org.w3c.jigsaw.pagecompile.JDKCompiler**.

Jigsaw Team

$Id: org.w3c.jigsaw.pagecompile.PageCompileProp.html,v 1.1 1998/09/04 09:49:03 bmahe Exp $

All resources All frames

PageCompileFrame

PageCompileFrame allows you to generate HTML pages from HTML/Java pages. Read the Page Compilation documentation.

Inherits

The PageCompileFrame class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ProtocolFrame
- HTTPFrame

Attributes description

The PageCompileFrame doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.jigsaw.pagecompile.PageCompileFrame.html,v 1.1 1998/09/04 09:49:06 bmahe Exp $

All resources All frames

CgiFrame

This resource class runs CGI scripts, conforming to the CGI/1.1 specification. You can use it as an extension template, to get automatic indexing of your CGI scripts, based on their extension.

It also has a special interpreter attribute that makes running scripts on Windows easy to configure, and somwhow more efficient then you could expect.

Inherits

The CgiFrame class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ProtocolFrame
- HTTPFrame

Attributes description

The CgiFrame defines the following attributes:

- interpreter
- command
- noheader
- generates-form
- remote-host
- cgi-debug

`interpreter`

semantics

Some operating systems don't support the nice UNIX feature !# that allows UNIX to launch the appropriate interpreter for the appropriate script. This attribute allows the CgiResource to be configured to use a certain interpreter to interpret the script pointed to by the command attribute.

By using this attribute in the predefined extensions, you can state that on Windows boxes, all .pl files are to be exported by the CgiResource, with default interpreter /Perl/bin/perl.exe. This will allow you to automatically index cgi scripts.

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

`command`

semantics

The command to launch in order to run the CGI script. This should provide the full pat hto the executable script. Each entry of the (String) array gives is passed as an extra argument to the script, the first one being (by convention) the full script's path.

type

This attribute is an editable StringArrayAttribute

default value

This attribute defaults to **null**.

`noheader`

semantics

Will the script emits its one headers, or should **Jigsaw** emit them. Classical CGI scripts will usually not emit any headers, so it is up to their hosting server to emit them. However, in some situation (server push, for example), the script might prefer to emit its own set of headers. When this flag is set to **true** it is up to the script to emit the headers, otherwise, the server will do it.

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **false**.

`generates-form`

semantics

Should the script be used to generate the form to be filled-in ? This resource allows you to put your form in a file, instead of having to launch the script to generate it. For compatibilty, this flags is turned to **false** by default, although, the second alternative (having the form in a separate file) will probably be much (much) more efficient.

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **true**.

`remote-host`

semantics

If turned on, this flag will enable the REMOTE_HOST env var computation.

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **null**.

`cgi-debug`

semantics

Turns debug on for that script. This nifty feature allow you to debug your script, by emitting a document containing the script output, instead of letting the script generates the document. The returned document will be of content type **text/plain**.

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **false**.

Jigsaw Team

$Id: org.w3c.jigsaw.frames.CgiFrame.html,v 1.3 1998/03/27 08:19:05 bmahe Exp $

All resources All frames

PostableFrame

The basic frame class for handling the HTTP POST method.

Inherits

The PostableFrame class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ProtocolFrame
- HTTPFrame

Attributes description

The PostableFrame defines the following attributes:

- override
- convert-get

`override`

semantics

Should this frame override form values when multiple ?

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **false**.

`convert-get`

semantics

Should this frame silently convert GET to POST methods ?

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **true**.

Jigsaw Team

$Id: org.w3c.jigsaw.frames.PostableFrame.html,v 1.3 1998/03/27 08:19:22 bmahe Exp $

All resources All frames

RedirecterFrame

This frame perform an internal redirect.

Inherits

The RedirecterFrame class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ProtocolFrame
- HTTPFrame

Attributes description

The RedirecterFrame defines the following attributes:

- target

`target`

semantics

The internal target URL.

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

Jigsaw Team

$Id: org.w3c.jigsaw.frames.RedirecterFrame.html,v 1.3 1998/03/27 08:19:27 bmahe Exp $

All resources All frames

RelocateFrame

This frame emits a HTTP redirect.

Inherits

The RelocateFrame class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ProtocolFrame
- HTTPFrame

Attributes description

The RelocateFrame defines the following attributes:

- location
- handle-pathinfo
- permanent-redirect
- use-usual-response

`location`

semantics

The relocation location.

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

`handle-pathinfo`

semantics

Should this frame also handle extra path infos ? Example: `http://yourserver.com/foobar` is relocated at `http://yourserver.com/relocated`. If `handle-pathinfo` is set to **true** then a request on `http://youserver.com/foobar/index.html` will return a HTTP redirect on `http://yourserver.com/relocated/index.html`.

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **true**.

`permanent-redirect`

semantics

If use-usual-response is not set, you can select the redirect type with this parameter (permanent or temporary).

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **false**.

`use-usual-response`

semantics

If you want your server to use the classical and deprecated "302 Found" response.

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **true**.

Jigsaw Team

$Id: org.w3c.jigsaw.frames.RelocateFrame.html,v 1.7 1999/09/08 14:58:11 ylafon Exp $

All resources All frames

SeeOtherFrame

Generates a 303 See Other reply on a POST, client should do the redirect with a GET.

Inherits

The SeeOtherFrame class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ProtocolFrame
- HTTPFrame
- PostableFrame

Attributes description

The SeeOtherFrame defines the following attributes:

- target-url

`target-url`

semantics

The redirect URL, another page that contains the response to the request.

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

Jigsaw Team

$Id: org.w3c.jigsaw.frames.SeeOtherFrame.html,v 1.3 1999/09/08 14:58:11 ylafon Exp $

All resources All frames

ServletDirectoryFrame

A DirectoryResource (or subclass) associated with a ServletDirectoryFrame is the only possible container for servlets that uses a ServletContext. That is, all ServletWrapper instances should be children of an instance of that resource.

Inherits

The ServletDirectoryFrame class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ProtocolFrame
- HTTPFrame

Attributes description

The ServletDirectoryFrame defines no specific attribute.

Jigsaw Team
$Id: org.w3c.jigsaw.servlet.ServletDirectoryFrame.html,v 1.3 1998/03/27 08:23:31 bmahe Exp $

All resources All frames

ServletWrapperFrame

Allways attached to a ServletWrapper. Act as the HTTP interface of ServletWrapper.

Inherits

The ServletWrapperFrame class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ProtocolFrame
- HTTPFrame

Attributes description

The ServletWrapperFrame doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.jigsaw.servlet.ServletWrapperFrame.html,v 1.3 1998/03/27 08:23:58 bmahe Exp $

All resources All frames

SSIFrame

The SSI frame implements server-side parsing of HTML documents. Inside an SSIFrame-indexed file, any comment of the form `<!--#commandName param1=val1 param2=val2 ... paramn=valn -->` will be interpreted as a command. Commands are looked up in an instance of the class supplied in the registryClass attribute. This class must be a subclass of the abstract class org.w3c.jigsaw.ssi.commands.CommandRegistry. Commands are implementations of the Command interface or the ControlCommand interface.

If no command registry is specified, the resource will default to org.w3c.jigsaw.ssi.commands.DefaultCommandRegistry, which incorporates the most commonly used commands (including a set of commands analogous to the directives used by the NCSA server SSI module.)

The replies from each of the commands ("partial replies") are merged into a global reply. A Content-Length header will be emitted, provided that each of the commands emits one.

The following variables are always defined initially, independent of the command registry used:

- `Boolean secure`: the value of the secure attribute at the time of the request.
- `Integer maxDepth`: the value of the maxDepth attribute at the time of the request.
- `Integer depth`: the current include nesting depth.
- `CommandRegistry registry`: the current command registry.

Please note that both the CommandRegistry base class and the Command interface are likely to change in future releases.

Known Bugs / Limitations

- In EchoCommand, GMT dates are not formatted using the datefmt config variable (instead, they're always formatted in "Java format").
- Validation using entity tags is not supported (yet).
- There is no support of a customized failure message in DefaultCommandRegistry.
- Commands have no option of determining their own persistency format. This could give better performance for some commands.
- Not all the headers from partial replies are merged. (It doesn't really make sense for some).

Inherits

The SSIFrame class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ProtocolFrame
- HTTPFrame

Attributes description

The SSIFrame defines the following attributes:

- maxDepth
- secure
- registryClass

`maxDepth`

semantics

The maximum depth of recursive document inclusion. Every time a document is included, a counter is increased. If this count gets to be equal to maxDepth, any further inclusion commands will be ignored. (Note that whether or not a command qualifies as an "inclusion command" is completely dependent on the command registry being used). If set to **0**, no recursion limit will be enforced.

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **10**.

`secure`

semantics

If true, only *secure* commands will be permitted. The definition of "secure" is fully dependent on the command registry used. In the case of the DefaultCommandRegistry, all commands except for exec are considered secure.

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **true**.

`registryClass`

semantics

The class that the command registry is to be an instance of. It must be a subclass of org.w3c.jigsaw.ssi.commands.CommandRegistry.

type

This attribute is an editable ClassAttribute

default value

This attribute defaults to **org.w3c.jigsaw.ssi.commands.DefaultCommandRegistry**.

Jigsaw Team

$Id: org.w3c.jigsaw.ssi.SSIFrame.html,v 1.3 1998/03/27 08:24:08 bmahe Exp $

All resources All frames

CvsFileFrame

This frame check cvs before performing a PUT request (If a CVS directory exists). This frame can only be attached to a FileResource.

Inherits

The CvsFileFrame class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ProtocolFrame
- HTTPFrame

Attributes description

The CvsFileFrame doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.jigsaw.cvs.CvsFileFrame.html,v 1.3 1998/03/27 08:15:41 bmahe Exp $

All resources All frames

PutListFrame

Manages a list of last puted documents and allows users to publish them. Allways attached to a PutListResource. This frame act as the HTTP interface of the PutListResource.

Inherits

The PutListFrame class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ProtocolFrame
- HTTPFrame
- PostableFrame

Attributes description

The PutListFrame doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.jigsaw.filters.PutListFrame.html,v 1.3 1999/09/08 14:58:11 ylafon Exp $

All resources All frames

LabelBureauFrame

The HTTP interface of the LabelBureauResource.

Inherits

The LabelBureauFrame class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ProtocolFrame
- HTTPFrame
- PostableFrame

Attributes description

The LabelBureauFrame doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.jigsaw.pics.LabelBureauFrame.html,v 1.1 1998/07/28 12:29:58 benoit Exp $

All resources All frames

ZipFrame

This frame allows yout to browse a fip file. See the tutorial.

Inherits

The ZipFrame class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ProtocolFrame
- HTTPFrame

Attributes description

The ZipFrame doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.jigsaw.zip.ZipFrame.html,v 1.2 1999/09/08 14:58:11 ylafon Exp $

All resources All frames

AccessLimitFilter

The AccessLimitFilter allows you to control and limit the number of simultaneous requests run by the same resource. The limit attribute indicates the maximum number of simultaneous requests that the target resource is willing to accept. The timeout attribute can be used to specify how long a request should wait before being elected to be run by the resource. When this timeout expires, the filter sends back a reply to the browser with an appropriate error message.

Inherits

The AccessLimitFilter class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ResourceFilter

Attributes description

The AccessLimitFilter defines the following attributes:

- limit
- timeout

`limit`

semantics

Maximum number of simultaneous requests to the target resource.

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **1**.

`timeout`

semantics

How long should requests be blocked before an error message is sent back to the requesting client. The given duration should be expressed as a number of milliseconds.

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **60000**.

Jigsaw Team

$Id: org.w3c.jigsaw.filters.AccessLimitFilter.html,v 1.3 1998/03/27 08:16:18 bmahe Exp $

All resources All frames

CacheFilter

...description...

Inherits

The CacheFilter class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ResourceFilter

Attributes description

The CacheFilter defines the following attributes:

- maxSize
- maxEntries
- defaultMaxAge

`maxSize`

semantics

....

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **8192**.

`maxEntries`

semantics

....

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **-1**.

`defaultMaxAge`

semantics

....

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **300**.

Jigsaw Team

$Id: org.w3c.jigsaw.filters.CacheFilter.html,v 1.5 1999/09/08 14:58:11 ylafon Exp $

All resources All frames

CookieFilter

A demo for how to use cookies from Jigsaw.

Inherits

The CookieFilter class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ResourceFilter

Attributes description

The CookieFilter defines the following attributes:

- cookie-maxage

`cookie-maxage`

semantics

The duration of the cookie in seconds.

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **20**.

Jigsaw Team

$Id: org.w3c.jigsaw.filters.CookieFilter.html,v 1.3 1998/03/27 08:16:40 bmahe Exp $

All resources All frames

CounterFilter

The CounterFilter maintains a count of the number of hits to its target resource.

Inherits

The CounterFilter class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ResourceFilter

Attributes description

The CounterFilter defines the following attributes:

- counter

`counter`

semantics

Maximum number of simultaneous requests to the target resource.

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **0**.

Jigsaw Team

$Id: org.w3c.jigsaw.filters.CounterFilter.html,v 1.3 1998/03/27 08:16:51 bmahe Exp $

All resources All frames

DebugFilter

The debug filter is usefull for debugging specific resources of the server. It will print out the incomming request before the target resource process it, and print out the reply the target resource has generated.

Inherits

The DebugFilter class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ResourceFilter

Attributes description

The DebugFilter defines the following attributes:

- onoff

`onoff`

semantics

Turn debugging on or off. The filter will only emit traces when this flag is set ot true.

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **true**.

Jigsaw Team

$Id: org.w3c.jigsaw.filters.DebugFilter.html,v 1.3 1998/03/27 08:17:05 bmahe Exp $

All resources All frames

ErrorFilter

The error filter allows you to redefine on the fly all the error messages flowing out of **Jigsaw**. The general principle is that all error messages comming out of **Jigsaw** are filtered, and intercepted. **Jigsaw** then uses internal requests to query some other resource for generating the exact text of the error message.

Inherits

The ErrorFilter class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ResourceFilter

Attributes description

The ErrorFilter defines the following attributes:

- base-url
- extension

`base-url`

semantics

The base URL of the directory containing the error messages. When this filter is enabled it will issue an internal request to the resource whose URL starts with this base url attribute value, and whose name is the number of the error code (plus an extra extension, see below).

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **/errors**.

`extension`

semantics

The extension of the resources that knows how to generate eror messages. If the base-url attribute is set to /errors, and this attribute is set to html, then the when generating the message for a 407 error, the error filter will use the /errors/407.html resource.

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **html**.

Jigsaw Team

$Id: org.w3c.jigsaw.filters.ErrorFilter.html,v 1.3 1998/03/27 08:17:15 bmahe Exp $

All resources All frames

GZIPFilter

This filter will compress the content of replies using GZIP. Compression is done *on the fly*. This assumes that you're really on a slow link, where you have lots of CPU, but not much bandwidth.

A nifty usage for that filter, is to plug it on top of a ProxyDirectory, in which case it will compress the data when it flies out of the proxy.

Inherits

The GZIPFilter class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ResourceFilter

Attributes description

The GZIPFilter defines the following attributes:

- mime-types

`mime-types`

semantics

The set of MIME types this filter is allowed to compress.

type

This attribute is an editable StringArrayAttribute

default value

This attribute defaults to **null**.

Jigsaw Team

$Id: org.w3c.jigsaw.filters.GZIPFilter.html,v 1.3 1998/03/27 08:17:29 bmahe Exp $

All resources All frames

HeaderFilter

Enforces a specific header value on all replies. Useful for testing.

Inherits

The HeaderFilter class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ResourceFilter

Attributes description

The HeaderFilter defines the following attributes:

- header-name
- header-value
- no-cache
- connection

`header-name`

semantics

The header name to add to replies.

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

`header-value`

semantics

The header value to add to replies.

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

`no-cache`

semantics

Should we use no-cache on that header.

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **false**.

`connection`

semantics

Should we use connection on that header.

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **false**.

Jigsaw Team

$Id: org.w3c.jigsaw.filters.HeaderFilter.html,v 1.3 1998/03/27 08:17:50 bmahe Exp $

All Resources All frames

LogFilter

The log filter purpose is to allow very detail logging of a particular area of your web server. The logs produced by this filter can be as verbose as you want (it allows you to log any request or reply field, check the request-headers and reply-headers attribute).

This filter generates a very verbose log format, consisting of a sequence of *records*. Each record is made of a number of lines of the following format:

property=value

A property , whose name is always in lower case, can be either:

- The special, record delimiter **url** property. That property always indicate the request url, and mark the beginning of the properties logging that request.
- The **request.*** set of properties indicate the value of a request header field. If the log filter is configured to log, say the request referer header, then when it is available, the log filter will emit a line logging its value as **request.referer**.
- The **reply.*** set of properties are used to denote reply header values. For example, when logging the reply **content-location** header, the log filter will emit a line logging its value as **reply.content-location**.

As an example, if you configure the logger to log the Refer , User-Agent and Accept HTTP request headers, along with the Content-Length reply header, the following kind of log will be outputed:

```
...
url=http://www43.inria.fr:8008/Admin/
request.user-agent=Mozilla/3.0 (X11; I; SunOS 5.5 sun4u)
request.accept=image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
reply.content-length=908
url=http://www43.inria.fr:8008/Admin/Properties
request.referer=http://www43.inria.fr:8008/Admin/
request.user-agent=Mozilla/3.0 (X11; I; SunOS 5.5 sun4u)
request.accept=image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
reply.content-length=615
url=http://www43.inria.fr:8008/Admin/Properties/general
request.referer=http://www43.inria.fr:8008/Admin/Properties
request.user-agent=Mozilla/3.0 (X11; I; SunOS 5.5 sun4u)
request.accept=image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
reply.content-length=3829
...
```

This extract is made of tree record, logging a transaction on respectively http://www43.inria.fr:8008/Admin/, http://www43.inria.fr:8008/Admin/Properties and http://www43.inria.fr:8008/Admin/Properties/general.

Inherits

The LogFilter class inherits from the following classes:

- ResourceFilter (aka abstract class)

Attributes description

The HTTPResource defines the following attributes:

- request-headers
- reply-headers
- logfile

`request-headers`

semantics

The list of request HTTP headers to log. All header names are to be entered in **lower** case, one header per line. For each entered header, an appropriate line will be added in all records, provided the requested header exists for the given transaction.

type

This attribute is an editable StringArrayAttribute

default value

This attribute defaults to the resource that loaded it into memory, as explained above.

---

reply-headers

semantics

The list of HTTP reply headers to log. All header names are to be entered in **lower** case, one header per line. For each entered header, an appropriate line will be added to each log record.

type

This attribute is an editable StringArrayAttribute

default value

This attribute defaults to the resource that loaded it into memory, as explained above.

---

logfile

semantics

The name of the file to output the log to.

type

This attribute is an editable FileAttribute

default value

This attribute has no default value. The log filter will not log anything until this is set to a correct file name.

Jigsaw Team

$Id: w3c.jigsaw.contrib.LogFilter.html,v 1.2 1996/12/09 03:17:12 jigsaw Exp $

All resources All frames

ProcessFilter

The ProcessFilter is for post-processing a reply's content. It allows you to run the target's reply content through external filters before it is sent back to the browser.

The external program is given by the command attribute, which can specify any additional arguments for the external process (along with its path, of course).

Inherits

The ProcessFilter class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ResourceFilter

Attributes description

The ProcessFilter defines the following attributes:

- command

`command`

semantics

The command to run when invoking the external program that will do the filtering. This attribute must contain at least the program name as its first String entry, and any additional parameters (as required).

type

This attribute is an editable StringArrayAttribute

default value

This attribute defaults to **null**.

Jigsaw Team

$Id: org.w3c.jigsaw.filters.ProcessFilter.html,v 1.3 1998/03/27 08:18:12 bmahe Exp $

All resources All frames

PutFilter

This filter update the PutListResource. Each time a PUT is performed, the PutFilter add an entry to the PutListResource that manage the list of the last puted documents.

Inherits

The PutFilter class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ResourceFilter

Attributes description

The PutFilter defines the following attributes:

- put-list

`put-list`

semantics

The PutListResource URL (on the same server)

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

Jigsaw Team

$Id: org.w3c.jigsaw.filters.PutFilter.html,v 1.3 1998/03/27 08:18:24 bmahe Exp $

All resources All frames

GrepPutFilter

This filter looks for a forbidden string in all incomming document (via PUT), if this string is found then emit a 403 reply (FORBIDDEN) whith a link on an explaination document.

Inherits

The GrepPutFilter class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ResourceFilter
- PutFilter

Attributes description

The GrepPutFilter defines the following attributes:

- forbidden-string
- redirect-url

`forbidden-string`

semantics

The string to search in the incomming document.

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

`redirect-url`

semantics

The URL pointing to the explaination document.

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

Jigsaw Team

$Id: org.w3c.jigsaw.filters.GrepPutFilter.html,v 1.3 1998/03/27 08:17:39 bmahe Exp $

All resources All frames

PutSizeFilter

The PutSizeFilter allow you to control the size of documents that can be PUT on the filtered resource. The put-size attribute indicates the maximum size in bytes of the accepted documents. The strict attribute is used to know how to handle requests without a Content-Lenght: field.

Inherits

The PutSizeFilter class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ResourceFilter

Attributes description

The PutSizeFilter defines the following attributes:

- put-size
- strict

`put-size`

semantics

Maximum size of a PUT document in bytes.

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **65536**.

`strict`

semantics

If set to true the PUT requests that don't have a valid Content-Length: field are dropped.

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **true**.

Jigsaw Team

$Id: org.w3c.jigsaw.filters.PutSizeFilter.html,v 1.3 1998/03/27 08:18:44 bmahe Exp $

All resources All frames

SimpleCacheFilter

...description...

Inherits

The SimpleCacheFilter class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ResourceFilter

Attributes description

The SimpleCacheFilter defines the following attributes:

- maxSize
- maxEntries
- defaultMaxAge
- flush

`maxSize`

semantics

....

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **8192**.

`maxEntries`

semantics

....

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **-1**.

`defaultMaxAge`

semantics

....

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **300**.

`flush`

semantics

....

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **false**.

Jigsaw Team

$Id: org.w3c.jigsaw.filters.SimpleCacheFilter.html,v 1.5 1999/09/08 14:58:11 ylafon Exp $

All resources All frames

HourLimiterFilter

Filter something which can be unavailable, depending on the date and time. If it is not available, it will give the delay of expected unavailability.

Inherits

The HourLimiterFilter class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ResourceFilter

Attributes description

The HourLimiterFilter defines the following attributes:

- day_repeat
- week_repeat
- month_repeat
- year_repeat
- start
- end

`day_repeat`

semantics

Repeat every day?

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **false**.

`week_repeat`

semantics

Repeat every week?

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **false**.

`month_repeat`

semantics

Repeat every month?

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **false**.

`year_repeat`

semantics

Repeat every year?

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **false**.

`start`

semantics

Beginning of the unavailability period.

type

This attribute is an editable DateAttribute

default value

This attribute defaults to **null**.

`end`

semantics

End of the unavailability period.

type

This attribute is an editable DateAttribute

default value

This attribute defaults to **null**.

Jigsaw Team

$Id: org.w3c.jigsaw.filters.HourLimiterFilter.html,v 1.1 1998/07/22 12:39:31 benoit Exp $

All resources All frames

URISizeLimiterFilter

This filters limit the size of URI. Could be used in a proxy for security reaseons.

Inherits

The URISizeLimiterFilter class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ResourceFilter

Attributes description

The URISizeLimiterFilter defines the following attributes:

- limit

`limit`

semantics

Maximum size of URI.

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **8192**.

Jigsaw Team

$Id: org.w3c.jigsaw.filters.URISizeLimiterFilter.html,v 1.3 1999/09/08 14:58:11 ylafon Exp $

All resources All frames

UseProxyFilter

Restrict access to a proxy, to acces the protected resource, you must go to a specific proxy. It acts as a demontrator for the HTTP/1.1 spec.

Inherits

The UseProxyFilter class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ResourceFilter

Attributes description

The UseProxyFilter defines the following attributes:

- proxyURL

`proxyURL`

semantics

The proxy URL.

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

Jigsaw Team

$Id: org.w3c.jigsaw.filters.UseProxyFilter.html,v 1.3 1999/09/08 14:58:11 ylafon Exp $

All resources All frames

TEFilter

This filter will compress the content of replies using GZIP. Compression is done *on the fly*. This assumes that you're really on a slow link, where you have lots of CPU, but not much bandwidth.
A nifty usage for that filter, is to plug it on top of a org.w3c.jigsaw.proxy.ProxyFrame, in which case it will compress the data when it flies out of the proxy.

Inherits

The TEFilter class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ResourceFilter

Attributes description

The TEFilter defines the following attributes:

- mime-types

`mime-types`

semantics

The set of MIME types we are allowed to compress.

type

This attribute is an editable StringArrayAttribute

default value

This attribute defaults to **null**.

Jigsaw Team
$Id: org.w3c.jigsaw.filters.TEFilter.html,v 1.1 1998/07/28 13:04:28 benoit Exp $

All resources All frames

AclFilter

This authentication filter is compliant with the Acl interface, it must be associated to a specific metadata frame implementing the Acl interface (see AclRealm).

Inherits

The AclFilter class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ResourceFilter

Attributes description

The AclFilter defines the following attributes:

- security-level
- strict-acl-merge-policy
- algorithm
- nonce_ttl
- shared-cachability
- private-cachability
- public-cachability

`security-level`

semantics

0 = Basic Authentication, 1 = Digest Authentication

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **0**.

`strict-acl-merge-policy`

semantics

If true and if the AclFilter is associated to several metadata frames (that are Acls) the access must be granted by all Acls.

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **true**.

`algorithm`

semantics

MD5, SHA (used only by the Digest Authentication)

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

`nonce_ttl`

semantics

Time to leave of Digest Authentication.

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **300**.

`shared-cachability`

semantics

When this flag is set to true, the authentication filter will decorate replies on the way back, marking the reply as being cachable by proxies only if the proxy is willing to revalidate it.

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **false**.

`private-cachability`

semantics

When this flag is set to true, it overrides all other settings, and mark the reply on its way back as

being cachable by any cache (a private or shared cache) prvided that cache revalidates it on each access.

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **false**.

`public-cachability`

semantics

When set to true, the filter will mark all outgoing replies it catches as being cachable, witouht even requiring revalidation.

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **false**.

Jigsaw Team
$Id: org.w3c.jigsaw.acl.AclFilter.html,v 1.1 1999/07/26 14:25:17 bmahe Exp $

All resources All frames

MirrorFrame

This frameclass allows you to mirror any existing site. It works by using the caching module of Jigsaw and by using a different lookup strategy for the cached resource. It will handle redirections too, so that if the origin server emits a redirection to an URL that it holds, then the mirror resource will rewrite the redirection and hand it out as being the target of the relocation.

Warning: because this resource really does what it claims to do, it should always be used in conjunction with the VirtualHostFrame. As **Jigsaw** will mirror a given site, it will no longer export the Admin directory, hence were you not to use the virtual host handling the configuration would no longer be accessible.

Inherits

The MirrorFrame class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ProtocolFrame
- HTTPFrame
- ForwardFrame

Attributes description

The MirrorFrame defines the following attributes:

- mirrors

`mirrors`

semantics

The base URL of the site to mirror. This is typically of the form **http://host:port**, eg http://www.w3.org.

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

Jigsaw Team

$Id: org.w3c.jigsaw.proxy.MirrorFrame.html,v 1.3 1998/03/27 08:21:03 bmahe Exp $

All resources All frames

GcStatFrame

Used???

Inherits

The GcStatFrame class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ProtocolFrame
- HTTPFrame

Attributes description

The GcStatFrame doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.jigsaw.status.GcStatFrame.html,v 1.2 1998/03/27 08:24:19 bmahe Exp $

All resources All frames

StatisticsFrame

The Statistics frame displays at regular interval of time (see the refresh attribute) a bunch of server's statistics. This includes the number of processed requests, the number of emitted bytes and the average, min and max times of request processing.

Inherits

The StatisticsFrame class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ProtocolFrame
- HTTPFrame

Attributes description

The StatisticsFrame defines the following attributes:

- refresh

`refresh`

semantics

At what interval should the thread listing refresh itself. When available, this resource uses the Refresh header to make itself refresh regularly. This attribute gives the period of this refresh in seconds.

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **5**.

Jigsaw Team

$Id: org.w3c.jigsaw.status.StatisticsFrame.html,v 1.2 1998/03/27 08:24:29 bmahe Exp $

All resources All frames

ThreadStatFrame

The ThreadStat frame allows you to monitor the current set of threads running inside the server process. This is useful for debugging purposes. It also gives some feedback on memory usage.

Inherits

The ThreadStatFrame class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ProtocolFrame
- HTTPFrame

Attributes description

The ThreadStatFrame defines the following attributes:

- refresh

`refresh`

semantics

At what interval should the thread listing refresh itself. When available, this resource uses the Refresh header to make itself refresh regularly. This attribute gives the period of this refresh in seconds.

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **5**.

Jigsaw Team

$Id: org.w3c.jigsaw.status.ThreadStatFrame.html,v 1.2 1998/03/27 08:24:40 bmahe Exp $

All resources All frames

DirectoryListerFrame

Specific frame of DirectoryLister.

Inherits

The DirectoryListerFrame class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ProtocolFrame
- HTTPFrame

Attributes description

The DirectoryListerFrame doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.jigsaw.resources.DirectoryListerFrame.html,v 1.2 1998/03/27 08:22:03 bmahe Exp $

All resources All frames

PasswordEditorFrame

This frame provides a mean for web site users to change their own password within a given realm. Allways attached to a PasswordEditor resource.

Warning: for obvious security reasons, when installed, that resource should be protected within the realm it edits.

Inherits

The PasswordEditorFrame class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ProtocolFrame
- HTTPFrame
- PostableFrame

Attributes description

The PasswordEditorFrame defines the following attributes:

- realm

`realm`

semantics

The realm within which user passwords should be edited when that resource is running. If undefined, the resource will emit an error in **Jigsaw** error log

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

Jigsaw Team

$Id: org.w3c.jigsaw.resources.PasswordEditorFrame.html,v 1.2 1998/03/27 08:23:20 bmahe Exp $

All resources All frames

ToolsListerFrame

This frame emit the content of its associated resource parent directory and provide a way to delete the children resources of this directory.

Inherits

The ToolsListerFrame class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ProtocolFrame
- HTTPFrame
- PostableFrame

Attributes description

The ToolsListerFrame doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.jigsaw.cvs.ToolsListerFrame.html,v 1.1 1998/03/27 15:09:23 benoit Exp $

All resources All frames

MapFrame

This frame is the HTTP interface of MapResource.

Inherits

The MapFrame class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ProtocolFrame
- HTTPFrame

Attributes description

The MapFrame doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.jigsaw.map.MapFrame.html,v 1.4 1999/09/08 14:58:11 ylafon Exp $

All resources All frames

AclRealm

This metadata frame is used to access the Jigsaw Realms through the Acl API.

Inherits

The AclRealm class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- MetaDataFrame
- JAcl

Attributes description

The AclRealm defines the following attributes:

- realm
- users
- methods

`realm`

semantics

The realm name

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

`users`

semantics

The list of the names of the users allowed to access the information protected by the authentication filter.

type

This attribute is an editable StringArrayAttribute

default value

This attribute defaults to **null**.

`methods`

semantics

The set of methods to protect against. When undefined, the filter will protected all HTTP method, otherwise it will just protect against the given set of methods. Each method is given by its name (in upper case).

type

This attribute is an editable StringArrayAttribute

default value

This attribute defaults to **null**.

Jigsaw Team

$Id: org.w3c.jigsaw.acl.AclRealm.html,v 1.1 1999/07/26 14:25:17 bmahe Exp $

All resources All frames

ResourceFilter

The ResourceFilter class is the basic class for all resource filters. If you are not yet familiar with filters, check Jigsaw architecture overview. Resource filters can maintain a shadow of their target attributes in order to overide them. This is how the ProcessFilter, for example, can be used to do on-the-fly MIME type conversion: it shadows its target resource content-type attribute value. This is why the generic resource editor displays the ShadowedBy... button when a filter is attached to the resource it edits.

When a resource's attribute is queried, its filters are first queried for a shadow value of the attribute. If one of the filters attached to the resource defines the attribute, its value is returned. Otherwise, the normal resource attribute value is fetched and returned.

Inherits

The ResourceFilter class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame

Attributes description

The ResourceFilter doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.tools.resources.ResourceFilter.html,v 1.3 1998/03/27 08:26:22 bmahe Exp $

All resources All frames

AuthFilter

A filter that implements authentication. Currently this filter handles only Basic Authentication, as defined by the HTTP/1.1 specification.

Inherits

The AuthFilter class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ResourceFilter

Attributes description

The AuthFilter defines the following attributes:

- methods
- realm
- shared-cachability
- private-cachability
- public-cachability

`methods`

semantics

The set of methods to protect against. When undefined, the filter will protected all HTTP method, otherwise it will just protect against the given set of methods. Each method is given by its name (in upper case).

type

This attribute is an editable StringArrayAttribute

default value

This attribute defaults to **null**.

`realm`

semantics

The realm (a.k.a. database) to use to obtain user information. Jigsaw manages a catalog of realms, allowing you to define as many of them as needed. This attribute is a realm identifier, i.e it is the symbolic name under which the realm catalog knows about it. When undefined, the filter will not run any authentication code.

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

`shared-cachability`

semantics

When this flag is set to true, the authentication filter will decorate replies on the way back, marking the reply as being cachable by proxies only if the proxy is willing to revalidate it.

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **false**.

`private-cachability`

semantics

When this flag is set to true, it overrides all other settings, and mark the reply on its way back as being cachable by any cache (a private or shared cache) prvided that cache revalidates it on each access.

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **false**.

`public-cachability`

semantics

When set to true, the filter will mark all outgoing replies it catches as being cachable, witouht even requiring revalidation.

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **false**.

Jigsaw Team

$Id: org.w3c.jigsaw.auth.AuthFilter.html,v 1.3 1998/03/27 08:14:58 bmahe Exp $

Jigsaw Configuration

Jigsaw Home / Documentation Overview

From the 1.0alpha5 release of **Jigsaw**, the best way to configure the server is to use the graphical application named **JigAdmin**. The form-based configuration is still here but is no longer maintained.

This document has the following sections:

How to start JigAdmin

For a quickstart, you should read this little Howto. The main command top start JigAdmin is:

```
java org.w3c.jigadm.Main -root INSTDIR/Jigsaw/Jigsaw
```

(change the '/' in '\' for Windows boxes)

Security considerations

By default, Jigsaw comes with predefined usernames/password (admin/admin) for the administration server, and no protection on the Admin directory of the http server. If you care a little about your site, you must do the following:

1. Change the password of the administration server.
2. Put an authentification filter on the Admin directory of each server managed by the admin server.
    - Create an admin realm, if not already present.
        - Click on the "realm" leaf
        - Add a realm

        - Open the "realm" node
        - Click on the realm node you just created
        - Add one user and change its password (by default, there is no password)
    - Open the "space" node of the server
    - Click on the "Admin" directory
    - Click on the "Add an AuthFilter" icon
    - Set the realm of the filter to the realm you created
3. Save everything
    - Click on each "control" node
    - Click on the save button

If you really care about security, you can add IP filters to the AuthFilters set, you can do it in the attributes of the user entries in the administration realms.

On Unix, the best way is to run Jigsaw as root in a chrooted environment. Read the FAQ entry on chroot.

Resources & Frames

The Resources are the object exported by **Jigsaw** to the outside world. It can be the raw object, like text files or image files, or objects created on the fly, like servlets, cgi scripts, filtered resources. In **Jigsaw-1.0**, the resource was in charge of handling the HTTP requests, so the Resource has all the knowledge needed by HTTP. Since Jigsaw-2.0, the Resources are very basic and contains only the intrinsic knowledge of this resource. For file, you have the size, last modification date... The 2.0 Resources have protocol frames. Those frames are handling the different protocols used to fetch this particular resource.

There are two ways of configuring the resources, by adding directly a specific resource at a specific place, or by letting the indexers create the resource in the server hierarchy. Of course the manual tune can be used along with the indexers. That is the most common way to configure **Jigsaw**.

Indexers

An indexers, placed on a Container, will be in charge of creating its sons resources. It will create Resource of a special kind depending, for example, on the extension of the filename ("html" for an html page, "png" for a PNG image file....). Or place a specific indexer for cgi on a directory named "cgi-bin". In Jigsaw-2.0, the indexer is not only in charge of creating the resource, it has also to put the right protocol frames (and other frames if necessary) on the created resource.

An indexer have two main part:

1. The Directories.
2. The Extensions.

In the Directories, you have to specify how to index directories with a specific name. The default name is

"*default*", in the default indexer the resource created is a DirectoryResource. In the 2.0 version, it creates a DirectoryResource with a default HTTPFrame.

In the extensions, you have to specify how to index files or leaf Resources. The default extension are mapped to FileResource, html, gif, png, txt... In the 2.0 version, an HTTPFrame is added to the Resource.

A tutorial about the setup of indexers is available, it helps understanding how it works.

Filters

The filters alters the resources, by requesting authentification for example. The filters are attached to the resource. In the 2.0 version the filters are atteched to the protocol frame of the resource rather than to the resource itself because many filtering scheme depends on the protocol used, the authentification in HTTP is very specific to HTTP and can't be used in other protocols. The filters are called before and after serving the resource. You may have filters that are called only before the resource is served, like an authentification filter, or after, like a "find and replace" filter.

Deletion and reindexation

In the 1.0 versions of Jigsaw, the reindexation was not present, the only way to force it was to delete the resource. Of course, if there was a specific configuration (e.g, "hand-modified" Resources), it was lost. The 2.0 version solves this problem.

In the 2.0 version of JigAdmin, you will find, close to the "Delete Resource" button, a "Reindex Children" button. By doing this, it will verufy/create/delete all its children resources. It will save your special configurations and update and propagate the other changes.

Jigsaw Team

$Id: configuration.html,v 1.20 1999/03/16 13:39:29 bmahe Exp $

Jigsaw
Frequently Answered Questions

Jigsaw Home / Documentation Overview

What is Jigsaw ?

Using Jigsaw

Technical stuff

Using Servlets

You think your question about **Jigsaw** is worth entering this list, you have a better formulation for some of the answers ? Mail to jigsaw@w3.org !

What's the purpose of Jigsaw ?

Jigsaw is the new W3C reference server. Its main purpose is to demonstrate new protocol features as they are defined (such as HTTP/1.1 or PICS), and to provide the basis for experimentations in the field of server software (such as the provided MUX prototypical implementation).

Where can I get Jigsaw ?

You can download **Jigsaw** distribution file in various formats, and using either ftp or http:

Windows zip files (1.5Mo)

- FTP
- HTTP

UNIX gzip tar file (700Ko)

- FTP
- HTTP

Why is Jigsaw written in Java ?

Java has a number of advantages that fit well with our purposes. It provides portable threads and garbage collection, allows for a very dynamic server architecture. It's ability to move code around may be use in future development to experiment with the mobile code concept.

Jigsaw is written in Java, it must be really slow...

No so true ! Check out the performance evaluation of **Jigsaw**, which indicates that it performs at least as well as the CERN server.

The admin server reply "Method GET not implemented.", how can I use it?

The administration server does not use plain HTTP but a variant of it. The only tool available for now is an application called JigAdmin.

What is this *indexing* thing all about ?

To a normal HTTP web-admin, **Jigsaw**'s configuration process might look really strange...**Jigsaw**'s design emphasis two different processing stage in serving documents:

Indexing stage
: When documents *enter* the Web

Serving stage
: When documents are served to clients

In short the rationale for separating these two stages is to make the *serving stage* as efficient as possible, by having the *indexing stage* prepare as much as possible the work. A document *enter* the WEB space the first time **Jigsaw** serves it; this happen behind the scene most of the time, and consist in creating a *resource* for the document to be served by querying a *resource factory*. The current resource factory can be configured to create various kind of resources based either on the file name extensions, or (in case of a directory) based on its name.

However, once a resource is created, it no longer reflects the settings of the resource factory (see the indexers); in some sense the information gets *compiled* so that at serving time, processing overhead is reduced as much as possible.

How do I tie a resource to a directory not under my server root?
Contributed by *Sirilyan <sirilyan@io.com>*

CERN server implemented this through the Pass directive, which let you map a server path to an absolute path on your file system. Jigsaw uses the org.w3c.jigsaw.resources.PassDirectory resource.

COOKBOOK METHOD: Create a new resource with *Add resources* in JigAdmin at the location you want the new server path to exist. Assign whatever name you desire to the new resource, and assign org.w3c.jigsaw.resources.PassDirectory for the class. Edit this new resource to change the pass-target attribute to the absolute path to the directory you want to serve.

You may want to change the name attribute of the 'index' file in that directory to Overview.html. name and filename are two separate attributes in Jigsaw; this is also a poor man's (or a Win95/NT user's) symlink.

Serving directories outside your server root may be a security risk.

How do I setup CGI scripts ?

They are two ways to setup CGI scripts. The manual way requires that you describe each script to the server. Let's say your script's path relative to the server root is `WWW/cgi-bin/myscript`. You will first have to create an appropriate org.w3c.tools.resources.FileResource with a org.w3c.jigsaw.frames.CgiFrame instance to wrap your script. See this tutorials to know how to create a resource in **Jigsaw**.

Then edit the newly created resource, and setup it's command line (the command line the server will use to run your script). Each line of the text field should represent one argument, the first one being the script full path.

You can also register files of a given extension as scripts, by using a specialized indexer. When required, you can even

specify the interpreter to be run to execute the script (for example, under Windows 95, or NT).

How can I use php3 in Jigsaw ?

This is quite easy, the best way is to do it directly using an indexer. php3 scripts will be handled as CGIs (so see the FAQ entry above).

You need first to compile php3 as a cgi interpreter, then configure the "php3" extension in your indexer using a FileResource and a CgiFrame, like aforementionned. Then, in your CgiFrame, set the "interpreter" field to your php binary, and "Generates Form" to true. *(Note, this will work with Jigsaw 2.0.2 and up)*

How can I use JSP in Jigsaw?

First of all, download the gnu jsp implementation at http://www.xs4all.nl/~vincentp/gnujsp/ and put the gnujsp.jar file in your **CLASSPATH.**

Now, read the installation procedure of gnujsp and install the JSPServlet under your servlet directory, don't forget to setup the parameters. For more details on servlet installation read the servlet documentation.

Now setup the indexer (eg: the default indexer), add the "jsp" extension using a FileResource and a ServletMapperFrame. Then in the ServletMapperFrame, set the "servlet-url" field to the JSPServlet URI (eg: /servlet/JSPServlet). *(Note, this will work with Jigsaw 2.0.2 and up)*

How do I set the default document?

With JigAdmin select the the HTTPFrame (or subclass) associated to the DirectoryResource (or subclass) where you want to modify the default document. Then, set the "index" field to your default document (eg: index.html).

Note: Don't forget to commit and save your changes.

Does Jisaw support SSL?

There was a SSL version of Jigsaw, done at IAIK. As of june 1999 the link to their SSL version of Jigsaw is back up!

What is the default Login Name and Password for JigAdmin?

The default Login Name is: **admin**, the default Password is: **admin**.

This information is also available in the JigAdmin documentation.

How can I extend Jigsaw ?

Jigsaw can be extended in a number of ways. Here are just three possible things you can play with, from the simplest to the complex ones:

- Writing new resource classes and new frame classes: you may want to read the tutorial on this subject before getting any further. You can think of this as being a n efficient replacement for CGI scripts, although this is a much more powerful environment to extend the server.
- Writing new filter classes: if you want to experiment with specific authentication needs, or if you require special logging formats. A tutorial on writing new filters is also available.
- Hacking **Jigsaw**. Right now you can just override any of the **Jigsaw** classes to replace its implementation. In the

future, most of the interesting things that you can do this way will be turned into specific interfaces (as is already the case for logging right now).

Why does Jigsaw implement its own persistence mechanism

Jigsaw implements its own persistence mechanism while RMI already provides a way to serialize objects, why is it so, will it change ? What **Jigsaw** implements in the `w3c.tools.store` package is more then persistence. It provides both a way of serializing objects *and* a way of describing what and how the object will be dumped. The available meta-description of objects (that you can obtain through the `getAttributes` method of resources), is a central part of **Jigsaw** architecture, since it offers the ability to create generic resource editors. This is not likely to disappear.

However, **Jigsaw** persistence mechanism may be merged in the future to the RMI interface, just by providing an implementation of the `readObject` and `writeObject` method through its existing mechanism.

How can I use HotJava on top of Jigsaw's HTTP client API ?

HotJava is Sun's Java based browser. If you want to experience an HTTP/1.1 compliant browser, you can run this browser on top of **Jigsaw**'s HTTP/1.1 compliant HTTP client API. To do so, you need to define the `java.protocol.handler.pkgs` property to `w3c.www.protocol` before launching HotJava. The best way to do so is to edit the HotJava property files.

Jigsaw's HTTP client API defines a number of other properties, if you are planing to use this setting, you should read the HttpManager documentation to get the complete list of available properties. These will allow you to add caching, authentication, proxying and more to HotJava !

Running Jigsaw on port 80 under UNIX

As of release 1.0alpha5, **Jigsaw** can now run on port 80, without running as the *root* user. To implement that you need to install the relevant piece of native code. This C code has been compiled and tested under Solaris, porting it to a different platform/architecture should be pretty easy.

Follow the normal installation procedure, and try to run **Jigsaw** on a port greater than 1024. Once this work, stop **Jigsaw** (through `/Admin/Exit`), Make sure your LD_LIBRARY_PATH variable includes the directory containing `libUnix.so` (this is the `Jigsaw/lib` directory under the standard release).

Select the user and group you want **Jigsaw** to run as. Make sure that user has read/write access to the entire `config` directory. Then, you just need to run **Jigsaw** through that special command line:

`java org.w3c.jigsaw.Main -user` *`user`* `-group` *`group`* `<other-options>`

Where *user* and *group* should provide (resp.) the user and the group you want **Jigsaw** to run as.

Warning: As the underlying UNIX process will change personnality right after acquiring the socket, the form based *restart* button (available from the properties editor) will no longer work, since the process will no longer be able to allocate a socket on port 80.
Under such a setting, the only way to restart **Jigsaw** is to kill it (through `/Admin/Exit`) and restart it manually.

Chrooting Jigsaw under UNIX

As of release 1.0alpha5, **Jigsaw** can now - under UNIX - be chroot'ed. If you try to do so, you are supposed to have some experiences with chroot'ing programs. We will assume that you already have a correct root to run a standard HTTP server and that you have read and understood the *Running Jigsaw on port 80* FAQ entry.

You need to install Java in that root (using whatever prefered way you want). To install **Jigsaw**, I would recommend using

the following directory structure:

/usr/local/jigsaw/classes

Should contain **Jigsaw**'s current distribution *jigsaw.zip* class file

/usr/local/jigsaw/extensions

Is used to add extensions to the server while running, and should be included in your CLASSPATH, *before* you start **Jigsaw**.

/usr/local/jigsaw/lib

Should contain **Jigsaw**'s native code support - ie *libUnix.so* -(to make the appropriate UNIX system calls)

Given that (don't forget to *symlink* the real `/usr/local/jigsaw` to the chroot'ed one), you should be able to reuse that script:

```
 #!/bin/sh
# Jigsaw
# $Id: FAQ.html,v 1.35 1999/09/21 11:39:51 ylafon Exp $
# Jigsaw launcher

# LD_LIBRARY_PATH will be set to work *only* before chroot, by the java script
# We add here, support for PATH as they appear to be after the chroot

CR_LD_LIBRARY_PATH=/usr/local/jigsaw/lib:/usr/local/java/lib/sparc:/usr/lib:/lib
LD_LIBRARY_PATH=$CR_LD_LIBRARY_PATH:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH

# CLASSPATH needs two hacks: one to include jigsaw.zip before chroot, and
# an other one to include everything after chroot

CLASSPATH=/0/w3c/abaird/sbroot/usr/local/jigsaw/classes/jigsaw.zip:/0/w3c/abaird
/sbroot/usr/local/jigsaw/classes/servlet.zip:/0/w3c/abaird/sbroot/usr/local/jigs
aw/classes/plus.zip:/0/w3c/abaird/sbroot/usr/local/jigsaw/extensions
export CLASSPATH
CR_CLASSPATH=/usr/local/jigsaw/classes/jigsaw.zip:/usr/local/jigsaw/classes/serv
let.zip:/usr/local/jigsaw/classes/plus.zip:/usr/local/jigsaw/extensions
CLASSPATH=$CR_CLASSPATH:$CLASSPATH
export CLASSPATH

# Ready to run jigsaw, now:

cd /0/w3c/abaird/sbroot/jigsaw
exec /usr/local/java/bin/java org.w3c.jigsaw.daemon.ServerHandlerManager $* -group n
obody -user nobody -chroot /0/w3c/abaird/sbroot -root /jigsaw
```

Running Jigsaw as a NT service

To be able to run Jigsaw as a service, you may use a tool allowing java programs to run as a service. You just have to make your choice in this list. Remember that the safest way to shut down Jigsaw is to use the administration interface of JigKill

---

Jigsaw Team

$Id: FAQ.html,v 1.32 1999/08/05 15:20:56 bmahe Exp $

Jigsaw
Frame tutorial

Jigsaw Home / Documentation Overview

This tutorial explains you how to write a new frame, by walking through a complete example. It is assumed that you are familiar with Jigsaw architecture, and that you have understand the configuration tutorial.

The frame we will write here will display a message describing its attribute and its associated resource. The tutorial will go through the following steps:

1. writing the frame class,
2. installing and configuring it.

Writing the frame class

Before actually writing a new frame, some decisions must be made about:

1. What will be its super class
2. In what package should it go
3. What attribute should it define
4. What method should it redefine

Picking a super class

Deciding for the super class of your frame is a pretty simple process right now. Here are the rule of thumbs:

- If your frame is supposed to handle forms, then you have to choose PostableFrame as your super class. This will give you the form arguments decoding for free.
- If not, you need to pick a sub-class of HTTPFrame as your super class (see NegotiatedFrame, RelocateFrame, CgiFrame, VirtualHostFrame)

Given these short rules, it should be obvious that for our sample frame, what we want to do is subclass the HTTPFrame. So right now, we can start writing the following piece of code (we will keep in bold the additional code we add as we walk through the example):

```
import java.util.*;

import org.w3c.tools.resources.*;
import org.w3c.jigsaw.frames.*;
import org.w3c.jigsaw.html.*;
import org.w3c.jigsaw.http.* ;
import org.w3c.www.http.* ;
```

```
public class FancyFrame extends HTTPFrame {

}
```

Note that we don't know yet were to put this file until we have selected an appropriate package for our frame.

Selecting a package

There is no particular problem with regard to the package your frame belong to: **Jigsaw** impose no constraint on this. The only thing you should be aware of is your CLASSPATH environment variable. This variable setting is particularly crucial in **Jigsaw** since it may impact its security: you don't want anyone to be able to plug new resource classes in the server !

For our sample frame, we can create a new package, let's call it tutorial, under the Jigsaw classes directory. We want this package to be under the w3c.jigsaw package. We can now create the appropriate directory (`src/classes/w3c/jigsaw/tutorial`), and write in it the following FancyFrame.java file:

```
package org.w3c.jigsaw.tutorials;

import java.util.*;

import org.w3c.tools.resources.*;
import org.w3c.jigsaw.frames.*;
import org.w3c.jigsaw.html.*;
import org.w3c.jigsaw.http.* ;
import org.w3c.www.http.* ;

public class FancyFrame extends HTTPFrame {

}
```

Defining the attributes

The next thing we have to figure out, is the list of attributes for our new frame. The HTTPFrame already defines a number of attributes (see the reference manual). Defining the set of attributes of a frame also defines the way the frame will be configured (since a frame is configured by editing its attribute values). Here, we want to be able to configure the message that will be emitted by the frame.

The message emitted by the frame can be described as an editable StringAttribute, which defaults to Hello.

Now that we now the attribute our frame is to have, we should declare it to the AttributeRegistry. This Registry keeps track of all the attributes of all resource classes. For each class it knows of, it maintains an ordered list of the attribute it defines. The fact that this list is ordered is important, since it allows for fast attribute value access (through a simple indirection in the attribute value array of each frame instance). Attribute declaration should be done at class initialization time, so we introduce a static statement in the class, whose purpose is to declare our attribute:

```
package org.w3c.jigsaw.tutorials;

import java.util.*;

import org.w3c.tools.resources.*;
import org.w3c.jigsaw.frames.*;
```

```
import org.w3c.jigsaw.html.*;
import org.w3c.jigsaw.http.* ;
import org.w3c.www.http.* ;

public class FancyFrame extends HTTPFrame {

    /**
     * Attribute index - Message to display
     */
    protected static int ATTR_MESSAGE = -1 ;

    static {
        Attribute a   = null ;
        Class     cls = null ;

        try {
            cls = Class.forName("org.w3c.jigsaw.tutorials.FancyFrame");
        } catch (Exception ex) {
            ex.printStackTrace() ;
            System.exit(1) ;
        }

        // The message attribute
         a = new StringAttribute("message", "Hello", Attribute.EDITABLE) ;
         ATTR_MESSAGE = AttributeRegistry.registerAttribute(cls, a) ;
    }

    /**
     * Get the message.
     * @return A String instance.
     */
    public String getMessage() {
        return getString(ATTR_MESSAGE, null);
    }
}
```

Redefining some methods

At this point, we have declared the set of attributes that our frame defines, the attribute Registry knows about it, we can now focus on the actual behavior of the frame. The only HTTP method that our frame will redefine is the GET method, which will synthesize a reply on the fly for each specific request. Jigsaw comes with a simple HtmlGenerator class for generating HTML that we want to use for this purpose. Our FancyFrame could be associated with many kinds of resources, FileResource, DirectoryResource, any subclass of FramedResource or FramedResource itself. In this particular case, we want to deal with all these resources, so we have to redefine the followings method of HTTPFrame:

- `protected Reply getFileResource(Request request)` which is called when the associated resource is a FileResource and the request method is GET.
- `protected Reply getDirectoryResource(Request request)` which is called when the associated resource is a DirectoryResource and the request method is GET.

- `protected Reply getOtherResource(Request request)` which is called when the associated resource is not a usual resource (FileResource, DirectoryResource) and the request method is GET.

The actual implementation of these methods is the following:

```
package org.w3c.jigsaw.tutorials;

import java.util.*;

import org.w3c.tools.resources.*;
import org.w3c.jigsaw.frames.*;
import org.w3c.jigsaw.html.*;
import org.w3c.jigsaw.http.* ;
import org.w3c.www.http.* ;

public class FancyFrame extends HTTPFrame {

    /**
     * Attribute index - Message to display
     */
    protected static int ATTR_MESSAGE = -1 ;

    static {
        Attribute a   = null ;
        Class     cls = null ;

        try {
            cls = Class.forName("org.w3c.jigsaw.tutorials.FancyFrame");
        } catch (Exception ex) {
            ex.printStackTrace() ;
            System.exit(1) ;
        }

        // The message attribute
         a = new StringAttribute("message", "Hello", Attribute.EDITABLE) ;
         ATTR_MESSAGE = AttributeRegistry.registerAttribute(cls, a) ;
    }

    /**
     * Get the message.
     * @return A String instance.
     */
    public String getMessage() {
        return getString(ATTR_MESSAGE, null);
    }

    /**
     * Display the Frame message and some attributes of our
     * associated FileResource. This method is called only if
     * our associated resource *is* a FileResource.
     * @param request The request to handle.
     * @return A Reply instance.
```

```
     * @exception ProtocolException if processing the request failed
     * @exception NotAProtocolException if an internal error occurs
     */
    protected Reply getFileResource(Request request)
        throws ProtocolException, NotAProtocolException
    {
        // get our associated FileResource
        FileResource fres = getFileResource();
        // Create the HTML generator, and set titles:
        HtmlGenerator g = new HtmlGenerator("FancyFrame");
        g.append("<h1>FancyFrame output</h1>");
        // emit the message
        g.append("<p>",getMessage(),"</p>");
        // display information about our FileResource
        g.append("<h2> FileResource associated : </h2>");
        g.append("<ul><li>Identifier : ",fres.getIdentifier());
        g.append("<li>File : "+fres.getFile());
        g.append("<li>Last Modified Time : ",
                 new Date(fres.getLastModified()).toString(),
                 "</ul>");
        // now emit the reply
        Reply reply = createDefaultReply(request, HTTP.OK) ;
        reply.setStream(g) ;
        return reply ;
    }

    /**
     * Display the Frame message and some attributes of our
     * associated DirectoryResource. This method is called only if
     * our associated resource *is* a DirectoryResource.
     * @param request The request to handle.
     * @return A Reply instance.
     * @exception ProtocolException if processing the request failed
     * @exception NotAProtocolException if an internal error occurs
     */
    protected Reply getDirectoryResource(Request request)
        throws ProtocolException, NotAProtocolException
    {
        // get our associated DirectoryResource
        DirectoryResource dres = getDirectoryResource();
        // Create the HTML generator, and set titles:
        HtmlGenerator g = new HtmlGenerator("FancyFrame");
        g.append("<h1>FancyFrame output</h1>");
        // emit the message
        g.append("<p>",getMessage(),"</p>");
        // display information about our DirectoryResource
        g.append("<h2> DirectoryResource associated : </h2>");
        g.append("<ul><li>Identifier : ",dres.getIdentifier());
        g.append("<li>Directory : "+dres.getDirectory());
        g.append("<li>Last Modified Time : ",
                 new Date(dres.getLastModified()).toString(),
```

```
                "</ul>");
        // now emit the reply
        Reply reply = createDefaultReply(request, HTTP.OK) ;
        reply.setStream(g) ;
        return reply ;
    }

    /**
     * Display the Frame message and some attributes of our
     * associated Resource. This method is called if the associated
     * resource has been registered with <strong>registerOtherResource</strong>
     * or if it's not a usual resource (FileResource, DirectoryResource)
     * @param request The request to handle.
     * @return A Reply instance.
     * @exception ProtocolException if processing the request failed
     * @exception NotAProtocolException if an internal error occurs
     */
    protected Reply getOtherResource(Request request)
        throws ProtocolException, NotAProtocolException
    {   // get our associated Resource
        FramedResource res = getResource();
        // Create the HTML generator, and set titles:
        HtmlGenerator g = new HtmlGenerator("FancyFrame");
        g.append("<h1>FancyFrame output</h1>");
        // emit the message
        g.append("<p>",getMessage(),"</p>");
        // display information about our Resource
        g.append("<h2> Resource associated : </h2>");
        g.append("<ul><li>Identifier : ",res.getIdentifier());
        g.append("<li>Last Modified Time : ",
                 new Date(res.getLastModified()).toString(),
                 "</ul>");
        // now emit the reply
        Reply reply = createDefaultReply(request, HTTP.OK) ;
        reply.setStream(g) ;
        return reply ;
    }
}
```

Note:

Sometimes we don't need to know what kind of resource is associated with our frame, or we are sure to be associated with a resource which is not a FileResource neither a DirectoryResource. In that case we could redefine the following method like this:

```
    /**
     * register our associated resource as an "other" resource.
     */
    public void registerResource(FramedResource resource) {
        super.registerOtherResource(resource);
    }
```

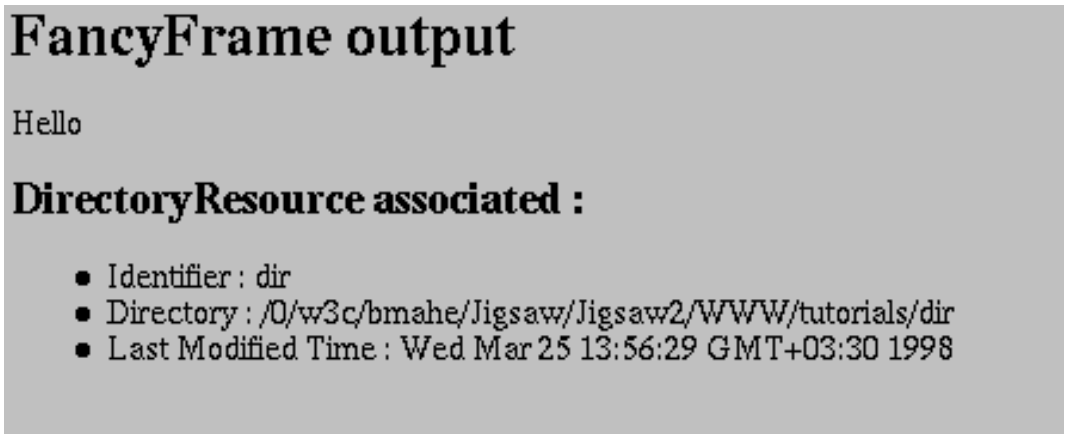
So, we just have to redefine getOtherResource.

Installing the frame

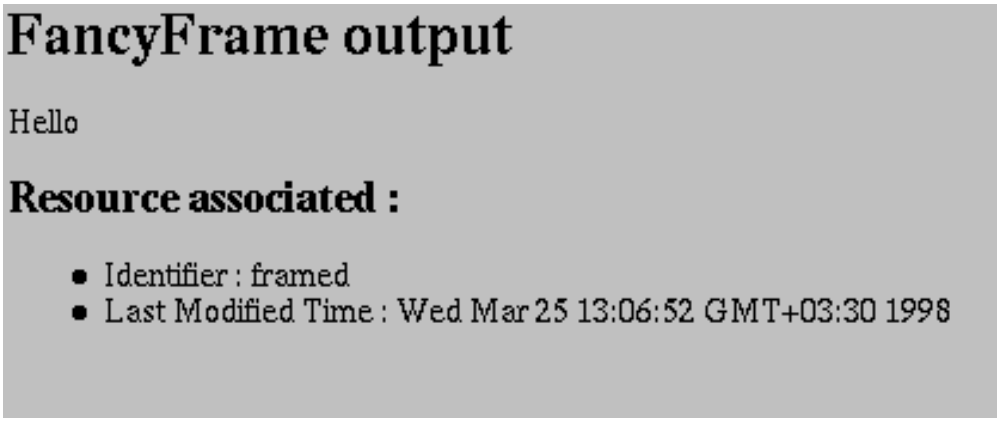
After reading the Resource configuration tutorial you will be able to install the FancyFrame. Here is what I get with my configuration:

FancyFrame associated with a FileResource

FancyFrame associated with a DirectoryResource



FancyFrame associated with a FramedResource



The example we have been walking through is probably one of the simplest one, however, by now, you should be able to read and understand the basic Frame classes provided by Jigsaw. I would recommend reading them in the following order:

1. You can start by going through the code of the HTTPFrame , which is more complex then our FancyFrame. Every frame relative to the HTTP protocol must be a subclass of it.
2. You can then continue by browsing the RelocateFrame, which emit an HTTP redirect reply.
3. If you still have more courage, then try reading the NegotiatedFrame, which manage content negotiation. There is a significant increase in complexity here.
4. Take a look at the Sample code page.

Enjoy !

Jigsaw Team

$Id: writing-frames.html,v 1.9 1999/03/16 13:38:21 bmahe Exp $

Jigsaw
Reference Manual

Jigsaw Home / Documentation Overview

Jigsaw reference manual is made of the following sections:

Server Side Components:

- Resources

  Describe all the resources included in **Jigsaw** release.
- Frames

  Describe all the frames included in **Jigsaw** release.
- Indexers

  Describe all the indexers included in **Jigsaw** release.

Client Side Components:

- Client side components

  Describe all the client side components included in **Jigsaw** release.

Jigsaw Team

$Id: Overview.html,v 1.4 1999/03/30 09:38:54 bmahe Exp $

Resource configuration

Jigsaw Home / Documentation Overview

In this section, the new resource model of Jigsaw 2.0 will be presented, along with a tutorial on how to create and edit a resource in Jigsaw.

What is a resource?

A resource is the basic object served by Jigsaw. It encapsulate a "real" object like a file (text, image or whatever) or a generated content (by a cgi-script or a servlet). This object is accessible within the server according to its URL. To be able to serve this object, Jigsaw needs to know a protocol that can serve this object. All the informations about the resource are stored in "Frames". Those frames may be of different kinds: protocol frames, describing the way to serve this resource for a specific protocol (ex: `HTTPFrame`), or describing the resource itself (ex: a metada frame encoded in RDF). The combination of an HTTPFrame and a basic resource corresponds to the HTTPResource of the Jigsaw 1.0.

Description of a resource

Basically, there are not many things in the resource itself as all the information is stored in the frames. There are two kinds of resource, the containers and the leafs. A basic resource will only have the identifier and last-modified attributes. A container will have also an indexer attribute to select the indexer used to index this the resources contained. There are extra attributes for:

`org.w3c.jigsaw.resources.DirectoryResource`

`extensible`

If true, new resources can be created by the indexer.

`negotiable`

If true, Content-Negotiation is activated, it means that an image named "foo" will match foo.png or foo.gif depending on the browser's preferences.

`org.w3c.tools.resources.FileResource`

`filename`

If set, gives the relative name of the real resource (default file name is the identifier).

Configuration of a resource

To configure a resource, you have to add frames and edit the attributes of this resource. If the resource is a container, you can also add resources to it.

Frames

Using **JigAdmin**, click on the resource in the tree, on the right part of the window, you will have some buttons (usually, two if the resource is a leaf, three if it is a container) located at the top. These button are used to select the helpers. To edit the frames, click on "frames" and the corresponding helper will appear below the button bar.

At the bottom of the window, a new tree browser will appear, containing the resource. if you open it, you will see all the frames attached to it. To add a new frame to the resource, click on the resource (highlight in a pinkish color) then click on "back to add frame menu", or, if you did not open the little tree, add directly the frame by selecting (or typing) the frame class.

To edit the attributes of a frame, just click on the frame, its own attribute helper will show up in the top part of the window.

A filter is now a frame of a protocol frame. If you want to add a CounterFilter to a HTTPFrame:

1. Click on the resource
2. select the frames helper
3. open the resource in the little tree
4. click on the HTTPFrame to edit it
5. click on "back to add frame menu"
6. select `org.w3c.jigsaw.filters.CounterFilter`
7. click on Add Frame

Now you can click on the new filter added and edit its attributes.

Attributes

Using **JigAdmin**, click on the resource in the tree, then select the Attributes helper. You will then have, in the attributes helper, the descrition of attributes on the left hand size, and the editable fields on the right hand side. To understand the meaning of the attributes names for each resource, see the reference documentation. For example, the meaning of the "negotiable" attribute of `org.w3c.jigsaw.resources.DirectoryResource` is:

`negotiable`

semantics

Should the directory resource automatically create resource with NegotiatedFrame? When this flag

is turned to true, the directory resource will automatically create negotiable frames on top of normal resources: each time a new resource is added to the directory, the resource looks up for a resource having the new child name, but possibly different extensions. If this succeeds, either the found resource is already a negotiated resource, in which case the new child is added as one of its variant resource; otherwise (the negotiated resource doesn't exist), the directory resource creates it with only one variant (the new child resource).

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **false**.

There is also a specific tutorial on the edition of the attributes.

Jigsaw Team

$Id: resource.html,v 1.12 1999/09/08 14:58:11 ylafon Exp $

```
// FancyFrame.java
// $Id: FancyFrame.html,v 1.2 1998/03/26 09:52:03 benoit Exp $
// (c) COPYRIGHT MIT and INRIA, 1998.
// Please first read the full copyright statement in file COPYRIGHT.html

package org.w3c.jigsaw.tutorials;

import java.util.*;

import org.w3c.tools.resources.*;
import org.w3c.jigsaw.frames.*;
import org.w3c.jigsaw.html.*;
import org.w3c.jigsaw.http.* ;
import org.w3c.www.http.* ;

/**
 * @version $Revision: 1.2 $
 * @author  Benoît Mahé (bmahe@w3.org)
 */
public class FancyFrame extends HTTPFrame {

    /**
     * Attribute index - Message to display
     */
    protected static int ATTR_MESSAGE = -1 ;

    static {
        Attribute a   = null ;
        Class     cls = null ;

        try {
            cls = Class.forName("org.w3c.jigsaw.tutorials.FancyFrame");
        } catch (Exception ex) {
            ex.printStackTrace() ;
            System.exit(1) ;
        }

        // The message attribute
         a = new StringAttribute("message", "Hello", Attribute.EDITABLE) ;
         ATTR_MESSAGE = AttributeRegistry.registerAttribute(cls, a) ;
    }

    /**
     * Get the message.
     * @return A String instance.
     */
    public String getMessage() {
        return getString(ATTR_MESSAGE, null);
    }
```

```
    /**
     * Display the Frame message and some attributes of our
     * associated FileResource. This method is called only if
     * our associated resource *is* a FileResource.
     * @param request The request to handle.
     * @return A Reply instance.
     * @exception ProtocolException if processing the request failed
     * @exception NotAProtocolException if an internal error occurs
     */
    protected Reply getFileResource(Request request)
        throws ProtocolException, NotAProtocolException
    {
        // get our associated FileResource
        FileResource fres = getFileResource();
        // Create the HTML generator, and set titles:
        HtmlGenerator g = new HtmlGenerator("FancyFrame");
        g.append("<h1>FancyFrame output</h1>");
        // emit the message
        g.append("<p>",getMessage(),"</p>");
        // display information about our FileResource
        g.append("<h2> FileResource associated : </h2>");
        g.append("<ul><li>Identifier : ",fres.getIdentifier());
        g.append("<li>File : "+fres.getFile());
        g.append("<li>Last Modified Time : ",
                 new Date(fres.getLastModified()).toString(),
                 "</ul>");
        // now emit the reply
        Reply reply = createDefaultReply(request, HTTP.OK) ;
        reply.setStream(g) ;
        return reply ;
    }

    /**
     * Display the Frame message and some attributes of our
     * associated DirectoryResource. This method is called only if
     * our associated resource *is* a DirectoryResource.
     * @param request The request to handle.
     * @return A Reply instance.
     * @exception ProtocolException if processing the request failed
     * @exception NotAProtocolException if an internal error occurs
     */
    protected Reply getDirectoryResource(Request request)
        throws ProtocolException, NotAProtocolException
    {
        // get our associated DirectoryResource
        DirectoryResource dres = getDirectoryResource();
        // Create the HTML generator, and set titles:
        HtmlGenerator g = new HtmlGenerator("FancyFrame");
        g.append("<h1>FancyFrame output</h1>");
        // emit the message
        g.append("<p>",getMessage(),"</p>");
```

```
        // display information about our DirectoryResource
        g.append("<h2> DirectoryResource associated : </h2>");
        g.append("<ul><li>Identifier : ",dres.getIdentifier());
        g.append("<li>Directory : "+dres.getDirectory());
        g.append("<li>Last Modified Time : ",
                 new Date(dres.getLastModified()).toString(),
                 "</ul>");
        // now emit the reply
        Reply reply = createDefaultReply(request, HTTP.OK) ;
        reply.setStream(g) ;
        return reply ;
    }

    /**
     * Display the Frame message and some attributes of our
     * associated Resource. This method is called if the associated
     * resource has been registered with <strong>registerOtherResource</strong>
     * or if it's not a usual resource (FileResource, DirectoryResource)
     * @param request The request to handle.
     * @return A Reply instance.
     * @exception ProtocolException if processing the request failed
     * @exception NotAProtocolException if an internal error occurs
     */
    protected Reply getOtherResource(Request request)
        throws ProtocolException, NotAProtocolException
    {   // get our associated Resource
        FramedResource res = getResource();
        // Create the HTML generator, and set titles:
        HtmlGenerator g = new HtmlGenerator("FancyFrame");
        g.append("<h1>FancyFrame output</h1>");
        // emit the message
        g.append("<p>",getMessage(),"</p>");
        // display information about our Resource
        g.append("<h2> Resource associated : </h2>");
        g.append("<ul><li>Identifier : ",res.getIdentifier());
        g.append("<li>Last Modified Time : ",
                 new Date(res.getLastModified()).toString(),
                 "</ul>");
        // now emit the reply
        Reply reply = createDefaultReply(request, HTTP.OK) ;
        reply.setStream(g) ;
        return reply ;

    }

}
```

```
// RelocateFrame.java
// $Id: RelocateFrame.html,v 1.2 1998/03/26 09:52:41 benoit Exp $
// (c) COPYRIGHT MIT and INRIA, 1996.
// Please first read the full copyright statement in file COPYRIGHT.html

package org.w3c.jigsaw.frames;

import java.util.*;
import java.io.* ;
import java.net.*;

import org.w3c.tools.resources.*;
import org.w3c.jigsaw.http.*;
import org.w3c.www.http.*;

import org.w3c.tools.resources.ProtocolException;
import org.w3c.tools.resources.NotAProtocolException;

/**
 * Emit a HTTP redirect.
 */
public class RelocateFrame extends HTTPFrame {

    /**
     * Name of the state to hold the PATH_INFO in the request.
     */
    public final static
        String PATH_INFO =
        "org.w3c.jigsaw.resources.RelocateResource.PathInfo";

    /**
     * Attribute index - The relocation location.
     */
    protected static int ATTR_LOCATION = -1 ;
    /**
     * Attribute index - Should we also handle extra path infos ?
     */
    protected static int ATTR_HANDLE_PATHINFO = -1;

    static {
        Attribute a = null ;
        Class     c = null ;

        try {
            c = Class.forName("org.w3c.jigsaw.frames.RelocateFrame");
        } catch (Exception ex) {
            ex.printStackTrace() ;
            System.exit(1) ;
        }
```

```
        // The location attribute
        a = new StringAttribute("location"
                                , null
                                , Attribute.EDITABLE|Attribute.MANDATORY) ;
        ATTR_LOCATION = AttributeRegistry.registerAttribute(c, a) ;
        // The handle path info attribute
        a = new BooleanAttribute("handle-pathinfo"
                                 , Boolean.TRUE
                                 , Attribute.EDITABLE);
        ATTR_HANDLE_PATHINFO = AttributeRegistry.registerAttribute(c, a);
    }

    /**
     * Get the location for the relocation
     */

    public String getLocation() {
        return (String) getValue(ATTR_LOCATION, null) ;
    }

    public boolean checkHandlePathInfo() {
        return getBoolean(ATTR_HANDLE_PATHINFO, true);
    }

    public void registerResource(FramedResource resource) {
        super.registerOtherResource(resource);
    }

    protected boolean lookupOther(LookupState ls, LookupResult lr)
        throws ProtocolException
    {
        // Perform our super-class lookup strategy:
        if ( super.lookupOther(ls, lr) ) {
            return true;
        } else if ( ! checkHandlePathInfo() ) {
            return false;
        }
        // Compute PATH INFO, store it as a piece of state in the request:
        StringBuffer pathinfo = new StringBuffer();
        while ( ls.hasMoreComponents() ) {
            pathinfo.append('/');
            pathinfo.append(ls.getNextComponent());
        }
        if (ls.hasRequest() ) {
            Request request = (Request) ls.getRequest();
            request.setState(PATH_INFO, pathinfo.toString());
        }
        lr.setTarget(resource.getResourceReference());
        return true;
    }
```

```
/**
 * Emit a redirect.
 * All GET requests are redirected toward the target location.
 * @param client The client issuing the request.
 * @param request The request to handle.
 * @exception ProtocolException If the request couldn't be handled.
 */

protected Reply getOtherResource (Request request)
    throws ProtocolException
{
    String location = getLocation() ;
    if ( location == null ) {
        Reply error = request.makeReply(HTTP.INTERNAL_SERVER_ERROR) ;
        error.setContent("The target RelocateResource doesn't define the"
                         + " relocation location. The server is "
                         + " misconfigured.") ;
        throw new HTTPException(error) ;
    } else {
        Reply  reply    = request.makeReply(HTTP.MOVED_TEMPORARILY) ;
        URL    loc      = null;
        try {
            loc = new URL(getURL(request), location);
            if (checkHandlePathInfo()) {
                String pathinfo = (String) request.getState(PATH_INFO);
                // Given the way pathinfo is computed, it starts with a /
                try {
                    if (pathinfo != null)
                        loc = new URL(loc.toExternalForm()+pathinfo);
                    else
                        resource.getServer().errlog(resource,
                                    "This resource handle Pathinfo "+
                                    "but the request has no "+
                                    "PATH_INFO state.");
                } catch (MalformedURLException ex) {
                    resource.getServer().errlog(resource,
                                    "This resource handle Pathinfo "+
                                    "but the request has an invalid "+
                                    "PATH_INFO state.");
                }
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        }
        reply.setLocation(loc);
        return reply ;
    }
}
```

RelocateFrame.java

```
}
```

```
// NegotiatedFrame.java
// $Id: NegotiatedFrame.html,v 1.3 1998/07/09 10:49:05 benoit Exp $
// (c) COPYRIGHT MIT and INRIA, 1996.
// Please first read the full copyright statement in file COPYRIGHT.html

package org.w3c.jigsaw.frames;

import java.io.*;
import java.util.*;

import org.w3c.tools.resources.*;
import org.w3c.jigsaw.http.* ;
import org.w3c.jigsaw.html.*;
import org.w3c.jigsaw.html.HtmlGenerator ;
import org.w3c.www.mime.* ;
import org.w3c.www.http.*;

import org.w3c.tools.resources.ProtocolException;
import org.w3c.tools.resources.NotAProtocolException;

/**
 * Content negotiation.
 */
public class NegotiatedFrame extends HTTPFrame {

    class VariantState {
        ResourceReference variant = null ;
        double qs      = 0.0 ;
        double qe      = 0.0 ;
        double qc      = 0.0 ;
        double ql      = 0.0 ;
        double q       = 0.0 ;  // quality (mime type one)
        double Q       = 0.0 ;  // the big Q

        public String toString() {
            try {
                Resource res = variant.lock();
                String name = (String) res.getIdentifier() ;
                if ( name == null )
                    name = "<noname>" ;
                return "[" + name
                    + " qs=" + qs
                    + " qe=" + qe
                    + " ql=" + ql
                    + " q =" + q
                    + " Q =" + getQ()
                    +"]" ;
            } catch (InvalidResourceException ex) {
                return "invalid";
            } finally {
```

```
                variant.unlock();
            }
        }

        void setContentEncodingQuality (double qe) {
            this.qe = qe ;
        }

        void setQuality (double q) {
            this.q = q ;
        }

        void setQuality (HttpAccept a) {
            q = a.getQuality() ;
        }

        void setLanguageQuality (double ql) {
            this.ql = ql ;
        }

        void setLanguageQuality (HttpAcceptLanguage l) {
            this.ql = l.getQuality() ;
        }

        double getLanguageQuality () {
            return ql ;
        }

        ResourceReference getResource () {
            return variant ;
        }

        double getQ() {
            return qe * q * qs * ql ;
        }

        VariantState (ResourceReference variant, double qs) {
            this.qs      = qs ;
            this.variant = variant ;
        }
    }

    private static Class httpFrameClass = null;

    static {
        try {
            httpFrameClass = Class.forName("org.w3c.jigsaw.frames.HTTPFrame") ;
        } catch (Exception ex) {
            throw new RuntimeException("No HTTPFrame class found.");
        }
    }
```

```
    /**
     * Turn debugging on/off.
     */
    private static final boolean debug = false;
    /**
     * Minimum quality for a resource to be considered further.
     */
    private static final double REQUIRED_QUALITY = 0.0001 ;
    /**
     * The Vary header field for this resource is always the same.
     */
    protected static HttpTokenList VARY = null;

    /**
     * Attribute index - The set of names of variants.
     */
    protected static int ATTR_VARIANTS = -1 ;

    static {
        // Compute and initialize the Vary header once and for all
        String vary[] = { "Accept",
                          "Accept-Charset",
                          "Accept-Language",
                          "Accept-Encoding" };
        VARY = HttpFactory.makeStringList(vary);
    }

    static {
        Attribute   a = null ;
        Class     cls = null ;
        try {
            cls = Class.forName("org.w3c.jigsaw.frames.NegotiatedFrame") ;
        } catch (Exception ex) {
            ex.printStackTrace() ;
            System.exit(1) ;
        }
        // The names of the varint we negotiate
        a = new StringArrayAttribute("variants"
                                     , null
                                     , Attribute.EDITABLE) ;
        ATTR_VARIANTS = AttributeRegistry.registerAttribute(cls, a) ;
    }

    /**
     * Get the variant names.
     */
    public String[] getVariantNames() {
        return (String[]) getValue(ATTR_VARIANTS, null) ;
    }
```

```
    public void setVariants(String variants[]) {
        setValue(ATTR_VARIANTS, variants);
    }

    /**
     * Get the variant resources.
     * This is somehow broken, it shouldn't allocate the array of variants
     * on each call. However, don't forget that the list of variants can be
     * dynamically edited, this means that if we are to keep a cache of it
     * (as would be the case if we kept the array of variants as instance var)
     * we should also take care of editing of attributes (possible, but I
     * just don't have enough lifes).
     * @return An array of ResourceReference, or <strong>null</strong>.
     * @exception ProtocolException If one of the variants doesn't exist.
     */

    public ResourceReference[] getVariantResources()
        throws ProtocolException
    {
        // Get the variant names:
        String names[] = getVariantNames() ;
        if ( names == null )
            return null ;
        // Look them up in our parent directory:
        ResourceReference variants[] = new ResourceReference[names.length] ;
        ResourceReference r_parent     = resource.getParent() ;
        try {
            DirectoryResource parent = (DirectoryResource) r_parent.lock();
            for (int i = 0 ; i < names.length ; i++) {
                variants[i] = parent.lookup(names[i]) ;
                if (variants[i] == null)
                    throw new HTTPException(names[i]+
                                            ": couldn't be restored.");
            }
        } catch (InvalidResourceException ex) {
            throw new HTTPException("invalid parent for negotiation");
        } finally {
            r_parent.unlock();
        }
        return variants ;
    }

    /**
     * Print the current negotiation state.
     * @param header The header to print first.
     * @param states The current negotiation states.
     */

    protected void printNegotiationState (String header, Vector states) {
        if ( debug ) {
            System.out.println ("------" + header) ;
```

```
            for (int i = 0 ; i < states.size() ; i++) {
                VariantState state = (VariantState) states.elementAt(i) ;
                System.out.println (state) ;
            }
            System.out.println ("-----") ;
        }
    }

    /**
     * Negotiate among content encodings.
     * <p>BUG: This will work only for single encoded variants.
     * @param states The current negotiation states.
     * @param request The request to handle.
     */

    protected boolean negotiateContentEncoding (Vector states,
                                                Request request)
        throws ProtocolException
    {
        if ( ! request.hasAcceptEncoding() ) {
            // All encodings accepted:
            for (int i = 0 ; i < states.size() ; i++) {
                VariantState state = (VariantState) states.elementAt(i) ;
                state.setContentEncodingQuality(1.0) ;
            }
        } else {
            String encodings[] = request.getAcceptEncoding() ;
            for (int i = 0 ; i < states.size() ; i++) {
                VariantState state    = (VariantState) states.elementAt(i) ;
                ResourceReference rr  = state.getResource();
                try {
                    FramedResource resource = (FramedResource)rr.lock() ;
                    HTTPFrame itsframe =
                        (HTTPFrame) resource.getFrame(httpFrameClass);
                    if (itsframe != null) {
                        if ( !itsframe.definesAttribute("content-encoding") ) {
                            state.setContentEncodingQuality (1.0) ;
                        } else {
                            String ve = itsframe.getContentEncoding() ;
                            state.setContentEncodingQuality (0.001) ;
                        encoding_loop:
                            for (int j = 0 ; j < encodings.length ; j++) {
                                if ( ve.equals (encodings[j]) ) {
                                    state.setContentEncodingQuality(1.0) ;
                                    break encoding_loop ;
                                }
                            }
                        }
                    }
                } catch (InvalidResourceException ex) {
```

```
            } finally {
                rr.unlock();
            }
        }
        // FIXME We should check here against unlegible variants as now
    }
    return false ;
}

/**
 * Negotiate on charsets.
 * <p>BUG: Not implemented yet.
 * @param states The current states of negotiation.
 * @param request The request to handle.
 */

protected boolean negotiateCharsetQuality (Vector states
                                           , Request request) {
    return false ;
}

/**
 * Negotiate among language qualities.
 * <p>BUG: This will only work for variants that have one language tag.
 * @param states The current states of negotiation.
 * @param request The request to handle.
 */

protected boolean negotiateLanguageQuality (Vector states
                                            , Request request)
    throws ProtocolException
{
    if ( ! request.hasAcceptLanguage() ) {
        for (int i = 0 ; i < states.size() ; i++) {
            VariantState state = (VariantState) states.elementAt(i) ;
            state.setLanguageQuality (1.0) ;
        }
    } else {
        HttpAcceptLanguage languages[] = request.getAcceptLanguage() ;
        boolean  varyLang    = false ;
        for (int i = 0 ; i < states.size() ; i++) {
            VariantState state    = (VariantState) states.elementAt(i) ;
            ResourceReference rr  = state.getResource();
            try {
                FramedResource resource = (FramedResource)rr.lock() ;
                HTTPFrame itsframe =
                    (HTTPFrame) resource.getFrame(httpFrameClass);
                if (itsframe != null) {
                    if ( !itsframe.definesAttribute("content-language") ) {
                        state.setLanguageQuality (-1.0) ;
                    } else {
```

```
                        varyLang = true ;
                        String lang = itsframe.getContentLanguage() ;
                        int jidx    = -1 ;
                        for (int j = 0 ; j < languages.length ; j++) {
                            if ( languages[j].getLanguage().equals(lang) )
                                jidx = j ;
                        }
                        if ( jidx < 0 )
                            state.setLanguageQuality(0.001) ;
                        else
                            state.setLanguageQuality (languages[jidx]) ;
                    }
                }
            } catch (InvalidResourceException ex) {
                //FIXME
            } finally {
                rr.unlock();
            }
        }
        if ( varyLang ) {
            for (int i = 0 ; i < states.size() ; i++) {
                VariantState s = (VariantState) states.elementAt(i);
                if ( s.getLanguageQuality() < 0 )
                    s.setLanguageQuality (0.5) ;
            }
        } else {
            for (int i = 0 ; i < states.size() ; i++) {
                VariantState s = (VariantState) states.elementAt(i) ;
                s.setLanguageQuality (1.0) ;
            }
        }
    }
    return false ;
}

/**
 * Negotiate among content types.
 * @param states The current states of negotiation.
 * @param request The request to handle.
 */

protected boolean negotiateContentType (Vector states,
                                        Request request)
    throws ProtocolException
{
    if ( ! request.hasAccept() ) {
        // All variants get a quality of 1.0
        for (int i = 0 ; i < states.size() ; i++) {
            VariantState state = (VariantState) states.elementAt(i) ;
            state.setQuality (1.0) ;
        }
```

```
        } else {
            // The browser has given some preferences:
            HttpAccept accepts[] = request.getAccept() ;
            for (int i = 0 ; i < states.size() ; i++ ) {
                VariantState state = (VariantState) states.elementAt(i) ;
                // Get the most specific match for this variant:
                ResourceReference rr = state.getResource();
                try {
                    FramedResource resource = (FramedResource)rr.lock() ;
                    HTTPFrame itsframe =
                        (HTTPFrame) resource.getFrame(httpFrameClass);
                    if (itsframe != null) {
                        MimeType vt = itsframe.getContentType();
                        int jmatch = -1 ;
                        int jidx   = -1 ;
                        for (int j = 0 ; j < accepts.length ; j++) {
                            int match = vt.match (accepts[j].getMimeType()) ;
                            if ( match > jmatch ) {
                                jmatch = match ;
                                jidx   = j ;
                            }
                        }
                        if ( jidx < 0 )
                            state.setQuality (0.0) ;
                        else
                            state.setQuality(accepts[jidx]) ;
                    }
                } catch (InvalidResourceException ex) {
                    //FIXME
                } finally {
                    rr.unlock();
                }
            }
        }
        return false ;
    }

    /**
     * Negotiate among the various variants for the Resource.
     * We made our best efforts to be as compliant as possible to the HTTP/1.0
     * content negotiation algorithm.
     */
    protected ResourceReference negotiate (Request request)
        throws ProtocolException
    {
        // Check for zero or one variant:
        ResourceReference variants[] = getVariantResources() ;
        if ( variants.length < 2 ) {
            if ( variants.length == 0 ) {
                Reply reply = request.makeReply(HTTP.NOT_ACCEPTABLE) ;
                reply.setContent ("<p>No acceptable variants.") ;
```

```
            throw new HTTPException (reply) ;
        } else {
            return variants[0] ;
        }
    }
    // Build a vector of variant negociation states, one per variants:
    Vector states = new Vector (variants.length) ;
    for (int i = 0 ; i < variants.length ; i++) {
        double qs = 1.0 ;
        try {
            FramedResource resource = (FramedResource)variants[i].lock() ;
            HTTPFrame itsframe =
                (HTTPFrame) resource.getFrame(httpFrameClass);
            if (itsframe != null) {
                if ( itsframe.definesAttribute ("quality") )
                    qs = itsframe.getQuality() ;
                if ( qs > REQUIRED_QUALITY )
                    states.addElement(new VariantState (variants[i], qs)) ;
            }
        } catch (InvalidResourceException ex) {
            //FIXME
        } finally {
            variants[i].unlock();
        }
    }
    // Content-encoding negociation:
    if ( debug )
        printNegotiationState ("init:", states) ;
    if ( negotiateContentEncoding (states, request) )
        // Remains a single acceptable variant:
        return ((VariantState) states.elementAt(0)).getResource() ;
    if ( debug )
        printNegotiationState ("encoding:", states) ;
    // Charset quality negociation:
    if ( negotiateCharsetQuality (states, request) )
        // Remains a single acceptable variant:
        return ((VariantState) states.elementAt(0)).getResource() ;
    if ( debug )
        printNegotiationState ("charset:", states) ;
    // Language quality negociation:
    if ( negotiateLanguageQuality (states, request) )
        // Remains a single acceptable variant:
        return ((VariantState) states.elementAt(0)).getResource() ;
    if ( debug )
        printNegotiationState ("language:", states) ;
    // Content-type negociation:
    if ( negotiateContentType (states, request) )
        // Remains a single acceptable variant:
        return ((VariantState) states.elementAt(0)).getResource() ;
    if ( debug )
        printNegotiationState ("type:", states) ;
```

```
        // If we reached this point, this means that multiple variants are
        // acceptable at this point. Keep the one that have the best quality.
        if ( debug )
            printNegotiationState ("before Q selection:", states) ;
        double qmax = REQUIRED_QUALITY ;
        for (int i = 0 ; i < states.size() ; ) {
            VariantState state = (VariantState) states.elementAt(i) ;
            if ( state.getQ() > qmax ) {
                for (int j = i ; j > 0 ; j--)
                    states.removeElementAt(0) ;
                qmax = state.getQ() ;
                i = 1 ;
            } else {
                states.removeElementAt(i) ;
            }
        }
        if ( debug )
            printNegotiationState ("After Q selection:", states) ;
        if ( qmax == REQUIRED_QUALITY ) {
            Reply reply = request.makeReply(HTTP.NOT_ACCEPTABLE) ;
            reply.setContent ("<p>No acceptable variant.") ;
            throw new HTTPException (reply) ;
        } else if ( states.size() == 1 ) {
            return ((VariantState) states.elementAt(0)).getResource() ;
        } else {
            // Respond with multiple choice (for the time being, there should
            // be a parameter to decide what to do.
            Reply reply = request.makeReply(HTTP.MULTIPLE_CHOICE) ;
            HtmlGenerator g = new HtmlGenerator ("Multiple choice for "+
                                                 resource.getIdentifier()) ;
            g.append ("<ul>") ;
            for (int i = 0 ; i < states.size() ; i++) {
                VariantState state = (VariantState) states.elementAt(i) ;
                String name = null;
                ResourceReference rr = state.getResource();
                try {
                    name = rr.lock().getIdentifier();
                    g.append ("<li>"
                              + "<a href=\"" + name + "\">" + name + "</a>"
                              + " Q= " + state.getQ()) ;
                } catch (InvalidResourceException ex) {
                    //FIXME
                } finally {
                    rr.unlock();
                }
            }
            reply.setStream (g) ;
            throw new HTTPException (reply) ;
        }
    }
```

```
    public void registerResource(FramedResource resource) {
        super.registerOtherResource(resource);
    }

    /**
     * Perform an HTTP request.
     * Negotiate among the variants, the best variant according to the request
     * fields, and make this elect3d variant serve the request.
     * @param request The request to handle.
     * @exception ProtocolException If negotiating among the resource variants
     *    failed.
     */

    public ReplyInterface perform(RequestInterface req)
        throws ProtocolException, NotAProtocolException
    {
        ReplyInterface repi = performFrames(req);
        if (repi != null)
            return repi;

        if (! checkRequest(req))
            return null;

        Request request = (Request) req;
        // Run content negotiation now:
        ResourceReference selected = negotiate(request) ;
        // This should never happen: either the negotiation succeed, or the
        // negotiate method should return an error.
        if ( selected == null ) {
            Reply error = request.makeReply(HTTP.INTERNAL_SERVER_ERROR) ;
            error.setContent("Error negotiating among resource's variants.");
            throw new HTTPException(error) ;
        }
        // FIXME content neg should be done at lookup time
        // FIXME enhencing the reply should be done at outgoingfilter
        // Get the original variant reply, and add its location as a header:
        try {
            FramedResource resource = (FramedResource) selected.lock();
            Reply reply = (Reply)resource.perform(request) ;
            reply.setHeaderValue(reply.H_VARY, VARY);
            HTTPFrame itsframe =
                (HTTPFrame) resource.getFrame(httpFrameClass);
            if (itsframe != null) {
                reply.setContentLocation(
                                itsframe.getURL(request).toExternalForm()) ;
                return reply;
            }
            Reply error = request.makeReply(HTTP.INTERNAL_SERVER_ERROR) ;
            error.setContent("Error negotiating : "+
                             "selected resource has no HTTPFrame");
            throw new HTTPException(error) ;
```

```
        } catch (InvalidResourceException ex) {
            Reply error = request.makeReply(HTTP.INTERNAL_SERVER_ERROR) ;
            error.setContent("Error negotiating : Invalid selected resource");
            throw new HTTPException(error) ;
        } finally {
            selected.unlock();
        }
    }

}
```

Jigsaw
Sample code

Jigsaw Home / Documentation Overview

This documents has the following sections:

Standard frames

- ResourceFrame

  The generic frame class.
- ProtocolFrame

  The generic protocol frame.
- HTTPFrame

  The basic frame for all HTTP accessible resources.
- CgiFrame

  A frame that allows you to run CGI/1.1 compliant scripts. This is, of course, not the best, nor the recommended way of extending Jigsaw.
- NegotiatedFrame

  A frame that handles negotiation among a given set of variant resources.
- PostableFrame

  The basic frame class for handling the HTTP POST method.
- RedirecterFrame

  A frame that handle internal redirection.
- RelocateFrame

  A frame that handle HTTP redirection.
- VirtualHostFrame

  A top level frame that will handle virtual hosts as described in HTTP/1.1 specification.

Filter frames

- ResourceFilter

  The top filter class.
- GenericAuthFilter

  This filter provides several ways of protecting part of your information space, using BasicAuthentication.
- AccessLimitFilter

  Limit the number of simultaneous accesses to a resource.
- CounterFilter

  Count the number of traversals or hits of its target.
- DebugFilter

  Print incoming request and outgoing replies.
- ErrorFilter

  The error filter allows you to redefine on the fly all error messages emitted by **Jigsaw** by using internal redirections: all errors are then emitted by some other resource (which can be any of the Jigsaw supported resources).
- GZIPFilter

  This filter will compress "on the fly" the content of replies using GZIP.
- HeaderFilter

  Enforces a specific header value on all replies.
- LogFilter

  The log filter allows you to get very detailed logging of transactions for a particular sub-space of your web server.
- ProcessFilter

  A filter that will process a reply's content through any external filter program.
- PutFilter

  This filter update the PutListResource.
- GrepPutFilter

  This PutFilter allows you to control the content of puted documents.
- PutSizeFilter

  This filter allows you to limit the size of puted documents

Standard resources

- Resource

  The generic resource class in **Jigsaw**.
- FramedResource

  This resource can be associated with frames. This is the base resource used.
- AbstractContainer

  This is the generic resource container class.
- ContainerResource

  This resource can manage and store its children.
- ExternalContainer

  This container can have a specific and external store.
- DirectoryResource

  The directory resource.
- PassDirectory

  A directory resource that emulates the CERN-server PASS rule. (May manage content negotiation)
- VirtualHostResource

  Just a base for the VirtualHostFrame. It handles the URI lookup.
- FileResource

  The basic file resource.

Jigsaw Team

$Id: Overview.html,v 1.19 1999/03/16 13:38:29 bmahe Exp $

Jigsaw
Resource tutorial

Jigsaw Home / Documentation Overview

This tutorial explains you how to write a new resource, by walking through a complete example. It is assumed that you are familiar with Jigsaw architecture, and that you have understand the configuration tutorial.

The resource we will write here will be the PassDirectory. The tutorial will go through the following steps:

1. writing the resource class,
2. installing and configuring it.

Writing the resource class

Before actually writing a new resource, some decisions must be made about:

1. What will be its super class
2. In what package should it go
3. What attribute should it define
4. What method should it redefine

Picking a super class

Deciding for the super class of your resource is a pretty simple process right now. Here are the rule of thumbs:

- If your resource is supposed to wrap a file, then you have to choose FileResource as your super class.
- If your resource is to wrap a directory, then you have to choose org.w3c.tools.resource.DirectoryResource (or org.w3c.jigsaw.resources.DirectoryResource if you need content negotiation)
- If you need to manage children then you probably want to sub-class the ContainerResource.
- In any other case, you need to pick FramedResource as your super class.

Given these short rules, it should be obvious that for our sample resource, what we want to do is subclass the DirectoryResource. So right now, we can start writing the following piece of code (we will keep in bold the additional code we add as we walk through the example):
Note that we don't know yet were to put this file until we have selected an appropriate package for our resource.

```
import java.util.*;
import java.io.*;
import org.w3c.tools.resources.*;

public class PassDirectory extends org.w3c.jigsaw.resources.DirectoryResource {
```

```
}
```

Selecting a package

There is no particular problem with regard to the package your resource belong to: **Jigsaw** impose no constraint on this. The only thing you should be aware of is your CLASSPATH environment variable. This variable setting is particularly crucial in **Jigsaw** since it may impact its security: you don't want anyone to be able to plug new resource classes in the server !

For our sample resource, we don't need to create a new package, let's use `org.w3c.jigsaw.resources`. We can write in it the following PassDirectory.java file:

```
package org.w3c.jigsaw.resources ;

import java.util.*;
import java.io.*;
import org.w3c.tools.resources.*;

public class PassDirectory extends org.w3c.jigsaw.resources.DirectoryResource {

}
```

Defining the attributes

The next thing we have to figure out, is the list of attributes for our new frame. The DirectoryResource already defines a number of attributes (see the reference manual). Defining the set of attributes of a resource also defines the way the resource will be configured (since a resource is configured by editing its attribute values). Here, we want to be able to configure the target directory that will be wrapped by the resource.

The directory wrapped by the resource can be described as an editable FileAttribute, which has no defaults value.

Now that we now the attribute our resource is to have, we should declare it to the AttributeRegistry. This Registry keeps track of all the attributes of all resource classes. For each class it knows of, it maintains an ordered list of the attribute it defines. The fact that this list is ordered is important, since it allows for fast attribute value access (through a simple indirection in the attribute value array of each resource instance). Attribute declaration should be done at class initialization time, so we introduce a static statement in the class, whose purpose is to declare our attribute:

```
package org.w3c.jigsaw.resources ;

import java.util.*;
import java.io.*;
import org.w3c.tools.resources.*;

public class PassDirectory extends org.w3c.jigsaw.resources.DirectoryResource {

    /**
     * Attribute index - The target physical directory of this resource.
     */
    protected static int ATTR_PASSTARGET = -1 ;

    static {
        Attribute a   = null ;
```

```
        Class     cls = null ;

        // Get a pointer to our class.
        try {
            cls = Class.forName("org.w3c.jigsaw.resources.PassDirectory") ;
        } catch (Exception ex) {
            ex.printStackTrace() ;
            System.exit(1) ;
        }
        // The directory attribute.
        a = new FileAttribute("pass-target"
                              , null
                              , Attribute.EDITABLE);
        ATTR_PASSTARGET = AttributeRegistry.registerAttribute(cls, a) ;
    }
}
```

Redefining some methods

At this point, we have declared the set of attributes that our resource defines, the attribute Registry knows about it, we can now focus on the actual behavior of the resource. The only difference between PassDirectory and DirectoryResource is that PassDirectory wraps an external directory instead of the inherited one.
In this case, we have to redefine the followings method of DirectoryResource:

- `public void setValue(int idx, Object value)`
- `public File getDirectory()`
- `public void initialize(Object values[])`

The actual implementation of these methods is the following:

```
package org.w3c.jigsaw.resources ;

import java.util.*;
import java.io.*;
import org.w3c.tools.resources.*;

public class PassDirectory extends org.w3c.jigsaw.resources.DirectoryResource {

    /**
     * Attribute index - The target physical directory of this resource.
     */
    protected static int ATTR_PASSTARGET = -1 ;

    static {
        Attribute a   = null ;
        Class     cls = null ;

        // Get a pointer to our class.
        try {
            cls = Class.forName("org.w3c.jigsaw.resources.PassDirectory") ;
        } catch (Exception ex) {
            ex.printStackTrace() ;
```

```
            System.exit(1) ;
        }
        // The directory attribute.
        a = new FileAttribute("pass-target"
                              , null
                              , Attribute.EDITABLE);
        ATTR_PASSTARGET = AttributeRegistry.registerAttribute(cls, a) ;
    }

    /**
     * Catch side-effects on pass-target, to absolutize it.
     * @param idx The attribute to set.
     * @param value The new value.
     */

    public void setValue(int idx, Object value) {
        super.setValue(idx, value);
        if ( idx == ATTR_PASSTARGET ) {
            File file = (File) value;
            if ( ! file.isAbsolute() ) {
                // Make it absolute, relative to the server space.
                File abs = new File(getServer().getRootDirectory()
                                    , file.toString());
                values[ATTR_PASSTARGET] = abs;
                values[ATTR_DIRECTORY]  = abs;
            }
        }
    }

    /**
     * The getDirectory method now returns the pass-directory.
     * @return The pass target location.
     */

    public File getDirectory() {
        return (File) getValue(ATTR_PASSTARGET, null) ;
    }

    /**
     * Make the directory attribute default to the target location.
     * This is required for classes that rely on the directory attribute to
     * compute their own attributes.
     * @param values The values we should initialized from.
     */

    public void initialize(Object values[]) {
        super.initialize(values);
        File target = getDirectory();
        if ( target != null )
            setValue(ATTR_DIRECTORY, target);
    }
```

```
}
```

Installing the resource

After reading the Resource configuration tutorial you will be able to install the PassDirectory.

The example we have been walking through is probably one of the simplest one, however, by now, you shouldn't have to write a new resource class but a new frame class, see the internal design of Jigsaw.

Enjoy !

Jigsaw Team

$Id: writing-resources.html,v 1.3 1999/03/16 13:38:26 bmahe Exp $

Jigsaw
Resource's filter tutorial

Jigsaw Home / Documentation Overview

This tutorial assumes that your are familiar with **Jigsaw** architecture, and that you have read both the configuration and the resource tutorials. It will explain you the last major concept of **Jigsaw** you should be aware of: resource filters.

Note: In Jigsaw2.0, filters must be attached to frames, NOT resources!

As was said in the architectural overview of **Jigsaw**, each HTTP request ends up performed by some target resource and frame instance. For the sake of simplicity, we didn't mentioned at that time, that frame classes provided with the **Jigsaw** server inherits from the FramedResource class. All instances of this class are able to manage a set of frames that can be filters: these filters, which are to be instances of sub-classes of ResourceFilter, are called-back twice during request processing:

- During lookup: before the actual target of the request has been selected. The `ingoingFilter` method of the filter gets called with the request as a parameter.
- After the request has been processed by the target resource. The `outgoingFilter` of the filter gets called with the request and the reply as parameters.

A resource filter is itself a resource: this means, in particular, that it is described - like all other resources - by a Java class, and a set of attributes. It also means that it can be pickled/unpickled, etc.

The remaining of this tutorial will walk through the code of the CounterFilter. This filter just counts the number of hits to its target resource. Before going any further, you should again make sure you have understand the resource tutorial. This tutorial has two sections: the first one will describe our filter's code, and the second one will explain how to attach it to a frame of the target resource.

The CounterFilter class

Its my hope that you will be amazed by how easy this filter is to code. As for writing new resource classes, the first thing we need to is to define the set of attributes that our filter will handle. In the case of a simple counter filter, the only attribute we need is simply the counter itself. Assuming we want our class to be in the `org.w3c.jigsaw.filters` package, we start by writing the following piece of code:

```
// CounterFilter.java
// $Id: writing-filters.html,v 1.2 1999/03/16 13:38:19 bmahe Exp $
// (c) COPYRIGHT MIT and INRIA, 1996.
// Please first read the full copyright statement in file COPYRIGHT.html

package org.w3c.jigsaw.filters;
```

```
import org.w3c.tools.resources.*;
import org.w3c.jigsaw.http.*;
import org.w3c.jigsaw.resources.*;

/**
 * Count the number of hits to the target.
 * This resource maintains the number of hits to some target resource, as
 * one of its persistent attribute.
 * It will decorate the request on the way in with a fake field
 * <code>org.w3c.jigsaw.filters.CounterFilter.count</code>, that will
 * hold the current hit counts for the target resource to use.
 */

public class CounterFilter extends ResourceFilter {
    /**
     * The name of the piece if state that receives the hit count value.
     * To get to the hit-count, use the <code>getState</code> method of
     * Request, with the following key.
     */
    public static final
        String STATE_COUNT = "org.w3c.jigsaw.filters.CounterFilter.count";

    /**
     * Attribute index - The counter attribute.
     */
    protected static int ATTR_COUNTER = -1 ;

    static {
        Attribute a   = null ;
        Class     cls = null ;

        try {
            cls = Class.forName("org.w3c.jigsaw.filters.CounterFilter") ;
        } catch (Exception ex) {
            ex.printStackTrace() ;
            System.exit(1) ;
        }
        // Declare the counter attribute
        a = new IntegerAttribute("counter"
                                 , new Integer(0)
                                 , Attribute.EDITABLE) ;
        ATTR_COUNTER = AttributeRegistry.registerAttribute(cls, a) ;
    }
}
```

This code does the following: it starts by declaring our resource filter class, and specify (through the `static` statement) its counter attribute, which is an editable integer attribute, whose default value is **0**. We register this attribute to the AttributeRegistry, and get back the *token* for accessing the attribute. If some of this seems unfamiliar, then refer to the resource tutorial.

We can now implement the `ingoingFilter` method of our filter, which is as simple as you might expect:

```
// CounterFilter.java
// $Id: writing-filters.html,v 1.2 1999/03/16 13:38:19 bmahe Exp $
// (c) COPYRIGHT MIT and INRIA, 1996.
// Please first read the full copyright statement in file COPYRIGHT.html

package org.w3c.jigsaw.filters;

import org.w3c.tools.resources.*;
import org.w3c.jigsaw.http.*;
import org.w3c.jigsaw.resources.*;

/**
 * Count the number of hits to the target.
 * This resource maintains the number of hits to some target resource, as
 * one of its persistent attribute.
 * It will decorate the request on the way in with a fake field
 * <code>org.w3c.jigsaw.filters.CounterFilter.count</code>, that will
 * hold the current hit counts for the target resource to use.
 */

public class CounterFilter extends ResourceFilter {
    /**
     * The name of the piece if state that receives the hit count value.
     * To get to the hit-count, use the <code>getState</code> method of
     * Request, with the following key.
     */
    public static final
        String STATE_COUNT = "org.w3c.jigsaw.filters.CounterFilter.count";

    /**
     * Attribute index - The counter attribute.
     */
    protected static int ATTR_COUNTER = -1 ;

    static {
        Attribute a   = null ;
        Class     cls = null ;

        try {
            cls = Class.forName("org.w3c.jigsaw.filters.CounterFilter") ;
        } catch (Exception ex) {
            ex.printStackTrace() ;
            System.exit(1) ;
        }
        // Declare the counter attribute
        a = new IntegerAttribute("counter"
                                 , new Integer(0)
```

```
                                    , Attribute.EDITABLE) ;
        ATTR_COUNTER = AttributeRegistry.registerAttribute(cls, a) ;
    }

    /**
     * We count all accesses, even the one that failed.
     * We also define the
     * <code>org.w3c.jigsaw.filters.CounterFilter.count</code>
     * request state as the number of hits on that resource (stored as
     * an Integer instance).
     * @param request The request being processed.
     * @return Always <strong>null</strong>.
     */

    public synchronized ReplyInterface ingoingFilter(RequestInterface req) {
        Request request = (Request) req;
        int i = getInt (ATTR_COUNTER, 0) + 1;
        setInt(ATTR_COUNTER, i) ;
        if(! request.hasState(STATE_COUNT))
            request.setState(STATE_COUNT, new Integer(i)) ;
        return null;
    }

}
```

That's all ! However, this needs a bit more explanations. First of all, you might be surprised that we didn't provide an implementation for the `outgoingFilter` method. There is one reasons for this: our super-class provide an empty `outgoingFilter` method, this solves the compiler problem.

Note that the counter value will be made persistent across servers invocation by the **Jigsaw** runtime. To be clear, this means that you can shutdown the server, and restart it: the count will keep up to date. Now, enough coding, let's play with our filter, the next section explains how to attach an instance of the filter to some target resource.

Installing the filter

First, let's decide on some resource we want to count access to. For the sake of simplicity, let's say we want to count the number of accesses to the `/User` directory resource. As you might have guess, the first thing we want to do is to launch the admin tool. Then setting up a filter to the `/User` directory resource involves the followings steps:

1. Select the "`User`" node in the tree.
2. Click on the "Frames" button.
3. Expand the "`User`" node in the little tree (you should see a HTTPFrame or any subclass).
4. Select the HTTPFrame node.
5. Click on "Back to Add Frame menu"
6. Then select the filter's class (ie `org.w3c.jigsaw.filters.CounterFilter`)
7. Click on "Add Frame"

Now your filter is added, but we could need to customize it:

1. Click on the "`Frames`" button of the "`User`" node. (previous steps 1,2,3)
2. Expand the little tree (right part of the window)
3. Select the "`CounterFilter`" node
4. Change the attribute you want. In this case, you don't need to change anything. (Don't forget to commit your changes)

The actual count is **0**, and we don't care about the filter's name.

Now, we can access `/User` (point your browser to `/User`). Reload the filter's editor...the value is now **1**.

Further reading

If you want to understand better the concept of filters, then you can look at the available filters, here is a path (by increasing complexity):

- Start by reading the code of the DebugFilter. This one has an `outgoingFilter` method, and doesn't represent a significant step in complexity.
- You can then try to read the code of the AccessLimitFilter. It doesn't use more features then the one above, but the synchronized methods it relies on are significantly more complex then the previous to examples.
- Now, if you are courageous enough, you can go to the GenericAuthFilter. Once you have understand this one, then you can carry on with the following lists of possible filters (left as an exercise to the reader)
- Try to write a filter that will log the referer field of the requests (this can be pretty easy if done in the naive way, and can become much complex if you want to use a something more efficient).

Enjoy !

Jigsaw Team

$Id: writing-filters.html,v 1.2 1999/03/16 13:38:19 bmahe Exp $

```
// DebugFilter.java
// $Id: DebugFilter.html,v 1.1 1998/03/30 08:30:50 benoit Exp $
// (c) COPYRIGHT MIT and INRIA, 1996.
// Please first read the full copyright statement in file COPYRIGHT.html

package org.w3c.jigsaw.filters;

import java.io.*;

import org.w3c.tools.resources.*;
import org.w3c.jigsaw.http.*;
import org.w3c.jigsaw.resources.*;

/**
 * Print incoming request and outgoing replies.
 */


public class DebugFilter extends ResourceFilter {
    /**
     * Attribute index - The on/off toggle.
     */
    protected static int ATTR_ONOFF = -1 ;

    static {
        Attribute a   = null ;
        Class     cls = null ;
        try {
            cls = Class.forName("org.w3c.jigsaw.filters.DebugFilter");
        } catch (Exception ex) {
            ex.printStackTrace() ;
            System.exit(1) ;
        }
        // The onoff toggle
        a = new BooleanAttribute("onoff"
                                 , Boolean.TRUE
                                 , Attribute.EDITABLE) ;
        ATTR_ONOFF = AttributeRegistry.registerAttribute(cls, a) ;
    }

    /**
     * Get the onoff toggle value.
     */

    public boolean getOnOffFlag() {
```

```
        return getBoolean(ATTR_ONOFF, true) ;
    }

    /**
     * The ingoing filter - fearly easy !
     * @param request The incomming request.
     * @return Always <strong>null</strong>.
     */

    public ReplyInterface ingoingFilter(RequestInterface req) {
        Request request = (Request) req;
        if ( getOnOffFlag() )
            request.dump(System.out);
        return null;
    }

    /**
     * The outgoing filter - As easy as the ingoing filter.
     * @param request The original request.
     * @param reply The target's reply.
     * @exception HTTPException If processing failed.
     */

    public ReplyInterface outgoingFilter(RequestInterface req,
                                         ReplyInterface rep)
    {
        Reply reply = (Reply) rep;
        if ( getOnOffFlag() )
            reply.dump(System.out);
        return null ;
    }

}
```

```
// AccessLimitFilter.java
// $Id: AccessLimitFilter.html,v 1.1 1998/03/30 08:30:39 benoit Exp $
// (c) COPYRIGHT MIT and INRIA, 1996.
// Please first read the full copyright statement in file COPYRIGHT.html

package org.w3c.jigsaw.filters ;

import java.io.* ;
import java.util.Hashtable ;

import org.w3c.tools.resources.*;
import org.w3c.www.http.*;
import org.w3c.jigsaw.http.* ;
import org.w3c.jigsaw.resources.* ;

/**
 * This filters limit the simultaneous accesses to its target resource.
 */

public class AccessLimitFilter extends ResourceFilter {
    /**
     * Attribute index - The maximum allowed simultaneous accesses.
     */
    protected static int ATTR_LIMIT = -1 ;
    /**
     * Attribute index - The time to wait for the lock (if limit reached)
     */
    protected static int ATTR_TIMEOUT = -1 ;

    static {
        Attribute a   = null ;
        Class     cls = null ;

        try {
            cls = Class.forName("org.w3c.jigsaw.filters.AccessLimitFilter");
        } catch (Exception ex) {
            ex.printStackTrace() ;
            System.exit(1) ;
        }
        // The limit attribute
        a = new IntegerAttribute("limit"
                                 , new Integer(1)
                                 , Attribute.EDITABLE) ;
        ATTR_LIMIT = AttributeRegistry.registerAttribute(cls, a) ;
        // The timeout attribute
        a = new IntegerAttribute("timeout"
                                 , new Integer(60000)
                                 , Attribute.EDITABLE) ;
        ATTR_TIMEOUT = AttributeRegistry.registerAttribute(cls, a) ;
    }
```

```
    /**
     * Current number of requets that have reached the target.
     */

    protected int count   = 0 ;


    /**
     * Get the maximum number of allowed simultaneous accesses.
     */

    public int getLimit() {
        return getInt(ATTR_LIMIT, 1) ;
    }

    /**
     * Get the timeout before we send back an error.
     * A client will wait only for this duration before being thrown an
     * error.
     */

    public int getTimeout() {
        return getInt(ATTR_TIMEOUT, -1);
    }


    /**
     * Count number of hits to the page, block when limit exceeded.
     * This filter maintains the actual number of hits to its target. When this
     * number exceeds the predefined limit, it blocks the caller until some
     * othre thread exits the target.
     * @param request The request to be handled.
     */

    public synchronized
        ReplyInterface ingoingFilter (RequestInterface req)
        throws HTTPException
    {
        Request request = (Request) req;
        int limit   = getLimit() ;
        int timeout = getTimeout() ;

        if ( limit < 0 )
            return null;
        while ( count >= limit ) {
            if ( timeout > 0 ) {
                try {
                    wait((long) timeout, 0) ;
                } catch (InterruptedException ex) {
                }
```

```
                if ( count >= limit ) {
                    String msg = "Simultaneous number of access to this page "
                        + "is limited to " + limit + " you was not able to "
                        + "get in." ;
                    Reply error = request.makeReply(HTTP.SERVICE_UNAVAILABLE);
                    error.setContent (msg) ;
                    throw new HTTPException (error) ;
                }
            } else {
                try {
                    wait() ;
                } catch (InterruptedException ex) {
                }
            }
        }
        count++ ;
        return null;
    }

    /**
     * Notify that someone has exit the target entity.
     * This method decrements the actual number of hits to the filter's
     * target, and notify any awaiting threads that they can now enter
     * safely.
     * @param request The request being handled.
     * @param reply The reply to be emited.
     */

    public synchronized
        ReplyInterface outgoingFilter (RequestInterface req,
                                       ReplyInterface rep)
        throws HTTPException
    {
        count-- ;
        notifyAll() ;
        return null ;
    }


}
```

```
// GenericAuthFilter.java
// $Id: GenericAuthFilter.html,v 1.1 1998/03/30 08:30:19 benoit Exp $
// (c) COPYRIGHT MIT and INRIA, 1996.
// Please first read the full copyright statement in file COPYRIGHT.html

package org.w3c.jigsaw.auth;

import java.io.*;
import java.util.*;
import java.net.*;

import org.w3c.tools.resources.*;
import org.w3c.jigsaw.http.*;
import org.w3c.jigsaw.html.*;
import org.w3c.tools.codec.* ;
import org.w3c.www.http.*;

import org.w3c.tools.resources.ProtocolException;

/**
 * A generic authentication filter.
 * This filter will use both IP and basic authentication to try to authenticate
 * incomming request. It should not be use for big user's database (typically
 * the ones that have more than 1000 entries).
 */

class BasicAuthContextException extends Exception {

    BasicAuthContextException (String msg) {
        super (msg) ;
    }
}

class BasicAuthContext {
    String user     = null ;
    String password = null ;
    String cookie   = null ;

    public String toString() {
        return user+":"+password;
    }

    BasicAuthContext (Request request)
        throws BasicAuthContextException, ProtocolException
    {
        HttpCredential credential = null;

        credential = (request.isProxy()
                      ? request.getProxyAuthorization()
                      : request.getAuthorization());
```

```
        if ( ! credential.getScheme().equalsIgnoreCase("Basic") ) {
            String msg = ("Invalid authentication scheme \""
                          + credential.getScheme()
                          + " expecting \"Basic\"");
            throw new BasicAuthContextException (msg) ;
        }
        // Decode the credentials:
        String decoded = null ;
        this.cookie    = credential.getAuthParameter("cookie");
        try {
            Base64Decoder b  = new Base64Decoder (cookie) ;
            decoded          = b.processString() ;
        } catch (Base64FormatException e) {
            String msg = "Invalid BASE64 encoding of credentials." ;
            throw new BasicAuthContextException (msg) ;
        }
        // Get user and password:
        int icolon = decoded.indexOf (':') ;
        if ( (icolon > 0) && (icolon+1 < decoded.length()) ) {
            // ok, parse was find, check user:
            this.user     = decoded.substring (0, icolon) ;
            this.password = decoded.substring (icolon+1) ;
        } else {
            String msg = "Invalid credentials syntax in " + decoded ;
            throw new BasicAuthContextException (msg) ;
        }
    }
}

/**
 * GenericAuthFilter provides for both IP and basic authentication.
 * This is really a first implementation. It looses on several points:
 * <ul>
 * <li>AuthUser instances, being a subclass of resource dump their classes
 * along with their attributes, although here we know that they will all
 * be instances of AuthUser.
 * <li>The way the ipmatcher is maintained doesn't make much sense.
 * <li>The way groups are handled is no good.
 * <li>The SimpleResourceStore is not an adequat store for the user database,
 * it should rather use the jdbmResourceStore (not written yet).
 * </ul>
 * However, this provides for the basic functionnalities.
 */

public class GenericAuthFilter extends AuthFilter {
    /**
     * Attribute index - The list of allowed users.
     */
    protected static int ATTR_ALLOWED_USERS = -1 ;
    /**
     * Attribute index - The list of allowed groups.
```

```
     */
    protected static int ATTR_ALLOWED_GROUPS = -1 ;

    static {
        Attribute   a = null ;
        Class       c = null ;
        try {
            c = Class.forName("org.w3c.jigsaw.auth.GenericAuthFilter");
        } catch (Exception ex) {
            ex.printStackTrace() ;
            System.exit(1) ;
        }
        // The list of allowed users
        a = new StringArrayAttribute("users"
                                     , null
                                     , Attribute.EDITABLE) ;
        ATTR_ALLOWED_USERS = AttributeRegistry.registerAttribute(c, a) ;
        // The list of allowed groups:
        a = new StringArrayAttribute("groups"
                                     , null
                                     , Attribute.EDITABLE);
        ATTR_ALLOWED_GROUPS = AttributeRegistry.registerAttribute(c, a) ;
    }

    /**
     * The IPMatcher to match IP templates to user records.
     */
    protected IPMatcher ipmatcher = null ;
    /**
     * The catalog of realms that make our scope.
     */
    protected RealmsCatalog catalog = null ;
    /**
     * Our associated realm.
     */
    protected ResourceReference rr_realm = null ;
    /**
     * The nam of the realm we cache in <code>realm</code>.
     */
    protected String loaded_realm = null ;

    /**
     * The challenge to issue to any client for Basic Authentication.
     */
    protected HttpChallenge challenge = null;

    /**
     * Get a pointer to our realm, and initialize our ipmatcher.
     */

    protected synchronized void acquireRealm() {
```

```
        // Get our catalog:
        if ( catalog == null ) {
            httpd server = (httpd)
              ((FramedResource) getTargetResource()).getServer() ;
            catalog = server.getRealmsCatalog() ;
        }
        // Check that our realm name is valid:
        String name = getRealm() ;
        if ( name == null )
            return ;
        if ((rr_realm != null) && name.equals(loaded_realm))
            return ;
        // Load the realm and create the ipmtacher object
        rr_realm = catalog.loadRealm(name) ;
        if (rr_realm != null) {
          try {
            AuthRealm realm = (AuthRealm) rr_realm.lock();
            Enumeration enum = realm.enumerateUserNames() ;
            if (enum.hasMoreElements())
              ipmatcher       = new IPMatcher() ;
            while (enum.hasMoreElements()) {
              String   uname = (String) enum.nextElement() ;
              ResourceReference rr_user = realm.loadUser(uname) ;
              try {
                AuthUser user  = (AuthUser) rr_user.lock();
                short    ips[][] = user.getIPTemplates() ;
                if ( ips != null ) {
                  for (int i = 0 ; i < ips.length ; i++)
                    ipmatcher.add(ips[i], rr_user) ;
                }
              } catch (InvalidResourceException ex) {
                System.out.println("Invalid user reference : "+uname);
              } finally {
                rr_user.unlock();
              }
            }
          } catch (InvalidResourceException ex) {

          } finally {
            rr_realm.unlock();
          }
        }
    }

    /**
     * Check that our realm does exist.
     * Otherwise we are probably being initialized, and we don't authenticate
     * yet.
     * @return A boolean <strong>true</strong> if realm can be initialized.
     */
```

```
    protected synchronized boolean checkRealm() {
        acquireRealm() ;
        return (ipmatcher != null) ;
    }

    /**
     * Get the list of allowed users.
     */

    public String[] getAllowedUsers() {
        return (String[]) getValue(ATTR_ALLOWED_USERS, null) ;
    }

    /**
     * Get the list of allowed groups.
     */

    public String[] getAllowedGroups() {
        return (String[]) getValue(ATTR_ALLOWED_GROUPS, null) ;
    }

    /**
     * Lookup a user by its IP address.
     * @param ipaddr The IP address to look for.
     * @return An AuthUser instance or <strong>null</strong>.
     */

    public synchronized ResourceReference lookupUser (InetAddress ipaddr) {
        if ( ipmatcher == null )
            acquireRealm() ;
        return (ResourceReference) ipmatcher.lookup(ipaddr.getAddress()) ;
    }

    /**
     * Lookup a user by its name.
     * @param name The user's name.
     * @return An AuthUser instance, or <strong>null</strong>.
     */

    public synchronized ResourceReference lookupUser (String name) {
        if ( rr_realm == null )
            acquireRealm() ;
        try {
          AuthRealm realm = (AuthRealm) rr_realm.lock();
          return realm.loadUser(name) ;
        } catch (InvalidResourceException ex) {
          return null;
        } finally {
          rr_realm.unlock();
        }
    }
```

```
  /**
   * Check the given Basic context against our database.
   * @param ctxt The basic auth context to check.
   * @return A AuthUser instance if check succeeded, <strong>null</strong>
   *    otherwise.
   */

  protected ResourceReference checkBasicAuth(BasicAuthContext ctxt) {
    ResourceReference rr_user = (ResourceReference)lookupUser(ctxt.user) ;
    if (rr_user != null) {
      try {
        AuthUser user = (AuthUser) rr_user.lock();
        // This user doesn't even exists !
        if ( user == null )
          return null ;
        // If it has a password check it
        if ( ! user.definesAttribute("password") ) {
          return  rr_user;
        } else {
          return user.getPassword().equals(ctxt.password) ? rr_user : null ;
        }
      } catch (InvalidResourceException ex) {
        return null;
      } finally {
        rr_user.unlock();
      }
    }
    return null;
  }

  /**
   * Is this user allowed in the realm ?
   * First check in the list of allowed users (if any), than in the list
   * of allowed groups (if any). If no allowed users or allowed groups
   * are defined, than simply check for the existence of this user.
   * @return A boolean <strong>true</strong> if access allowed.
   */

  protected boolean checkUser(AuthUser user) {
      String allowed_users[] = getAllowedUsers() ;
      // Check in the list of allowed users:
      if ( allowed_users != null ) {
          for (int i = 0 ; i < allowed_users.length ; i++) {
              if (allowed_users[i].equals(user.getName()))
                  return true ;
          }
      }
      // Check in the list of allowed groups:
      String allowed_groups[] = getAllowedGroups() ;
      if ( allowed_groups != null ) {
```

```
            String ugroups[] = user.getGroups() ;
            if ( ugroups != null ) {
                for (int i = 0 ; i < ugroups.length ; i++) {
                    for (int j = 0 ; j < allowed_groups.length ; j++) {
                        if ( allowed_groups[j].equals(ugroups[i]) )
                            return true ;
                    }
                }
            }
        }
        // If no users or groups specified, return true
        if ((allowed_users == null) && (allowed_groups == null))
            return true ;
        return false ;
    }

    /**
     * Catch set value on the realm, to maintain cached values.
     */

    public void setValue(int idx, Object value) {
        super.setValue(idx, value);
        if ( idx == ATTR_REALM ) {
            // Initialize the filter challenge:
            challenge = HttpFactory.makeChallenge("Basic");
            challenge.setAuthParameter("realm", getRealm());
        }
    }

    /**
     * Authenticate the given request.
     * We first check for valid authentication information. If no
     * authentication is provided, than we try to map the IP address to some
     * of the ones we know about. If the IP address is not found, we challenge
     * the client for a password.
     * <p>If the IP address is found, than either our user entry requires an
     * extra password step (in wich case we challenge it), or simple IP
     * based authentication is enough, so we allow the request.
     * @param request The request to be authentified.
     */

    public void authenticate (Request request)
        throws ProtocolException
    {
        // Are we being edited ?
        if ( ! checkRealm() )
            return ;
        // Internal requests always allowed:
        Client client = request.getClient() ;
        if ( client == null )
            return ;
```

```
        // Check for User by IP address:
        boolean ipchecked = false ;
        ResourceReference rr_user = lookupUser(client.getInetAddress());
        if (rr_user != null) {
          try {
            AuthUser user = (AuthUser) rr_user.lock();
            if ( user != null ) {
              ipchecked = true ;
              // Good the user exists, does it need more authentication ?
              if ( ! user.definesAttribute("password") && checkUser(user)) {
                request.setState(STATE_AUTHUSER, user.getName()) ;
                request.setState(STATE_AUTHTYPE, "ip");
                return ;
              }
            }
          } catch (InvalidResourceException ex) {
            //FIXME
          } finally {
            rr_user.unlock();
          }
        }
        // Check authentication according to auth method:
        if ((request.hasAuthorization() && ! request.isProxy())
            || (request.isProxy() && request.hasProxyAuthorization())) {
            BasicAuthContext ctxt = null ;
            try {
                ctxt = new BasicAuthContext(request);
            } catch (BasicAuthContextException ex) {
                ctxt = null;
            }
            // Is that user allowed ?
            if ( ctxt != null ) {
              rr_user = checkBasicAuth(ctxt) ;
              if (rr_user != null) {
                try {
                  AuthUser user = (AuthUser) rr_user.lock();
                  if ((user != null) && checkUser(user)) {
                    // Check that if IP auth was required, it succeeded:
                    boolean iprequired = user.definesAttribute("ipaddress") ;
                    if ( ( ! iprequired) || ipchecked ) {
                      // Set the request fields, and continue:
                      request.setState(STATE_AUTHUSER, ctxt.user);
                      request.setState(STATE_AUTHTYPE, "Basic") ;
                      return ;
                    }
                  }
                } catch (InvalidResourceException ex) {
                  //FIXME
                } finally {
                  rr_user.unlock();
                }
```

```
            }
        }
    }

    // Every possible scheme has failed for this request, emit an error
    Reply e = null;
    if ( request.isProxy() ) {
        e = request.makeReply(HTTP.PROXY_AUTH_REQUIRED);
        e.setProxyAuthenticate(challenge);
    } else {
        e = request.makeReply(HTTP.UNAUTHORIZED);
        e.setWWWAuthenticate (challenge);
    }
    HtmlGenerator g = new HtmlGenerator("Unauthorised");
    g.append ("<h1>Unauthorised access</h1>"
              + "<p>You are denied access to this resource.");
    e.setStream(g);
    throw new HTTPException (e);
}

/**
 * Initialize the filter.
 */

public void initialize(Object values[]) {
    super.initialize(values) ;
    if ( getRealm() != null ) {
        // Initialize the filter challenge:
        challenge = HttpFactory.makeChallenge("Basic");
        challenge.setAuthParameter("realm", getRealm());
    }
}

}
```

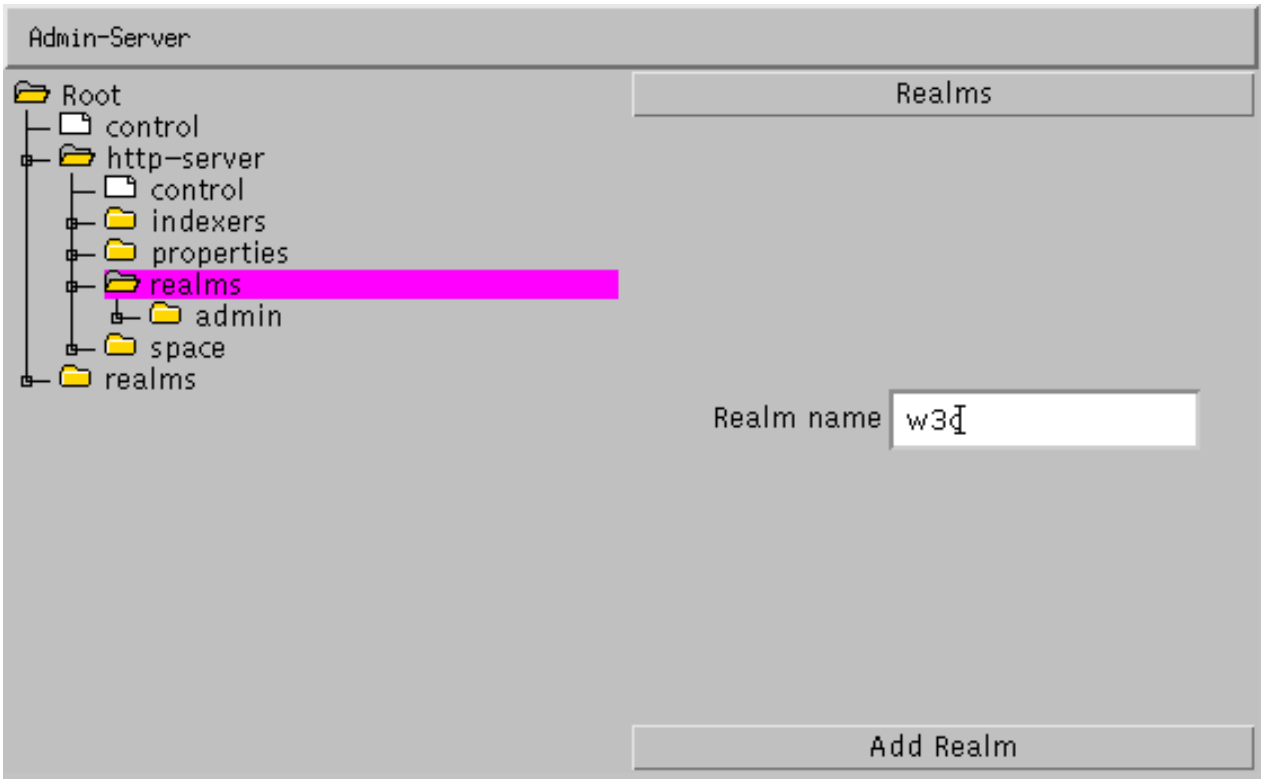
Authentication in Jigsaw

Jigsaw Home / Documentation Overview

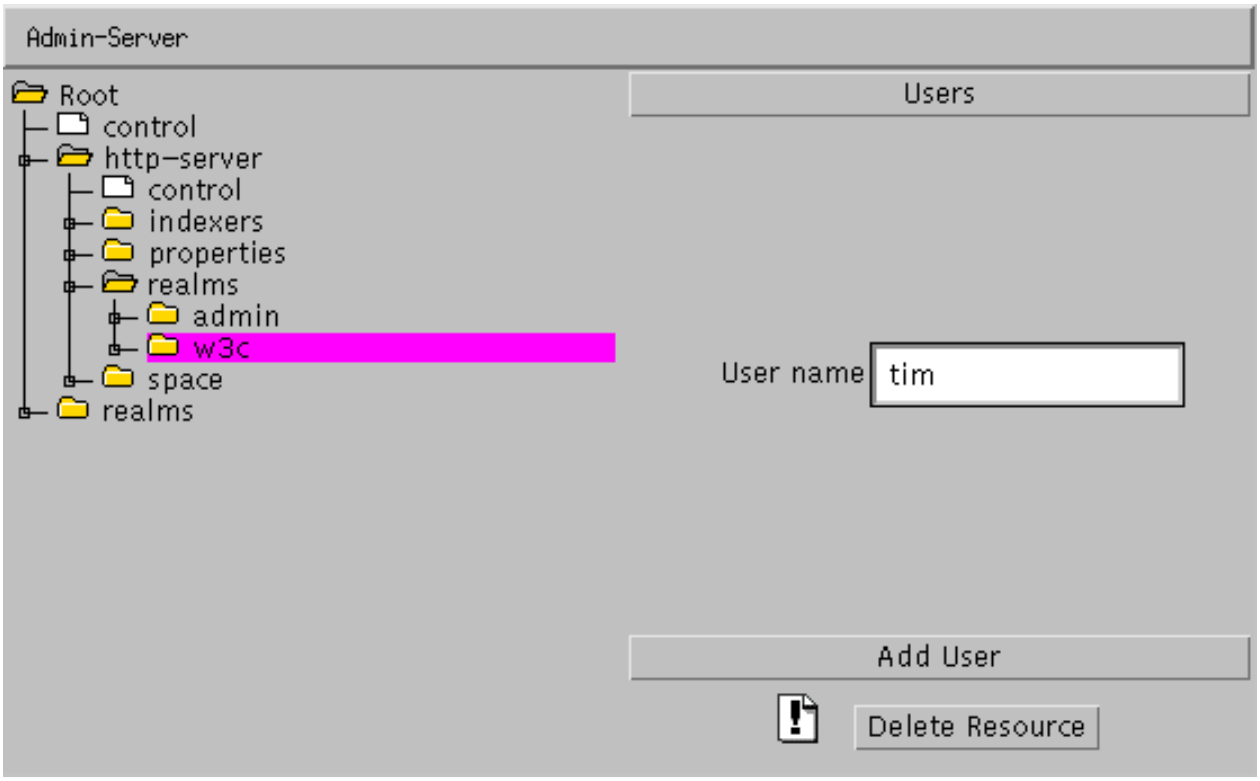
Setting up authentication requires you to run through the following stages:

1) Open a JigAdmin window (see JigAdmin documentation)

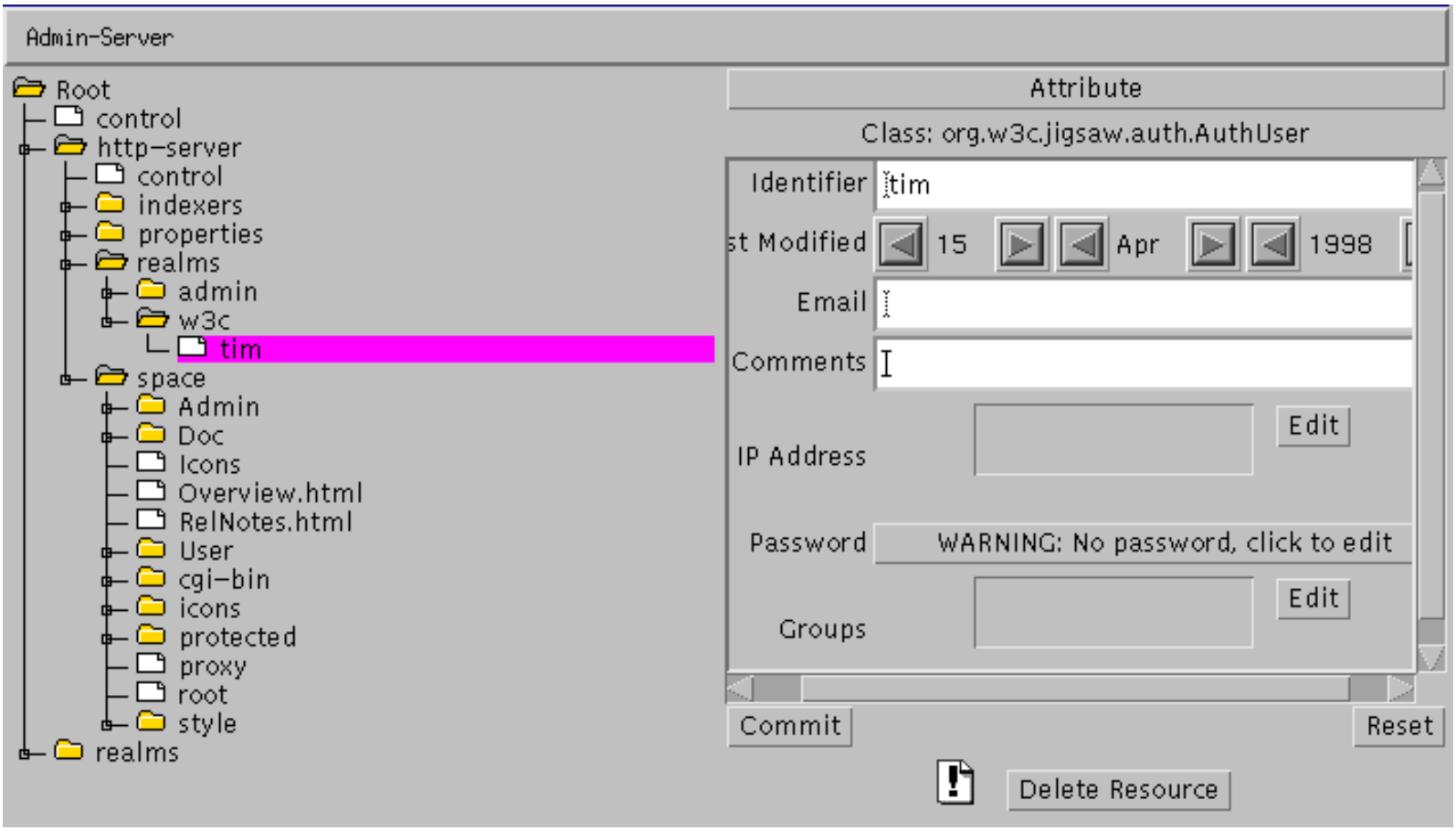
2) Create an authentication realm, if required. You can of course reuse authentication realms to protect different areas of the server. Select the "`realms`" node under "`http-server`", type the name of your new realm (ie "`w3c`") in the "`Realm name`" field and click on the "`Add Realm`" button.



3) Add some user for the new realm. Select the new "`w3c`" node under "`realms`", type the user name (ie "`tim`") in the appropiate field and click on the "`Add User`" button.

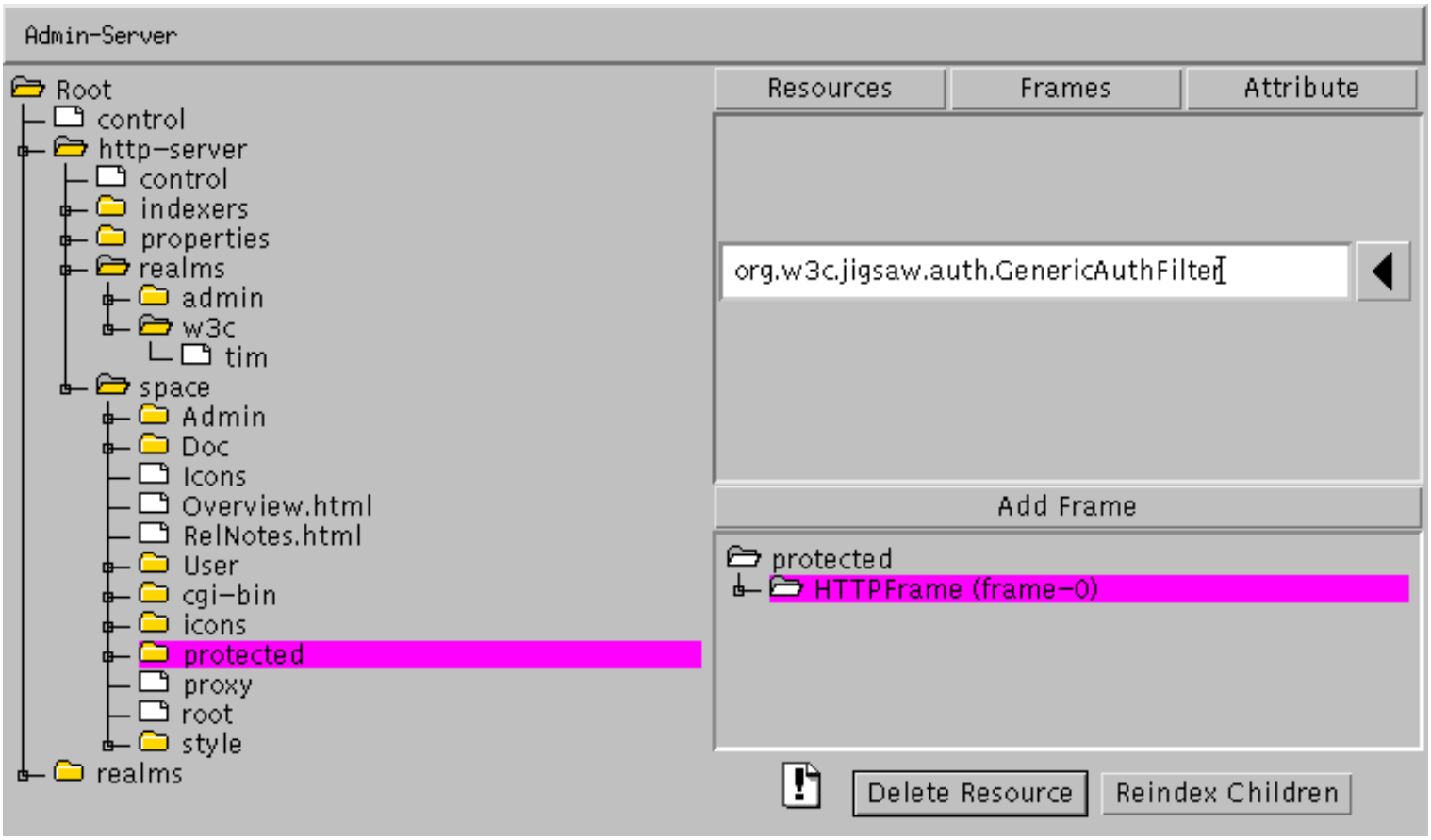


4) Setup the user attributes. The user attributes are describe in the AuthUser reference page. Don't forget to commit your changes.

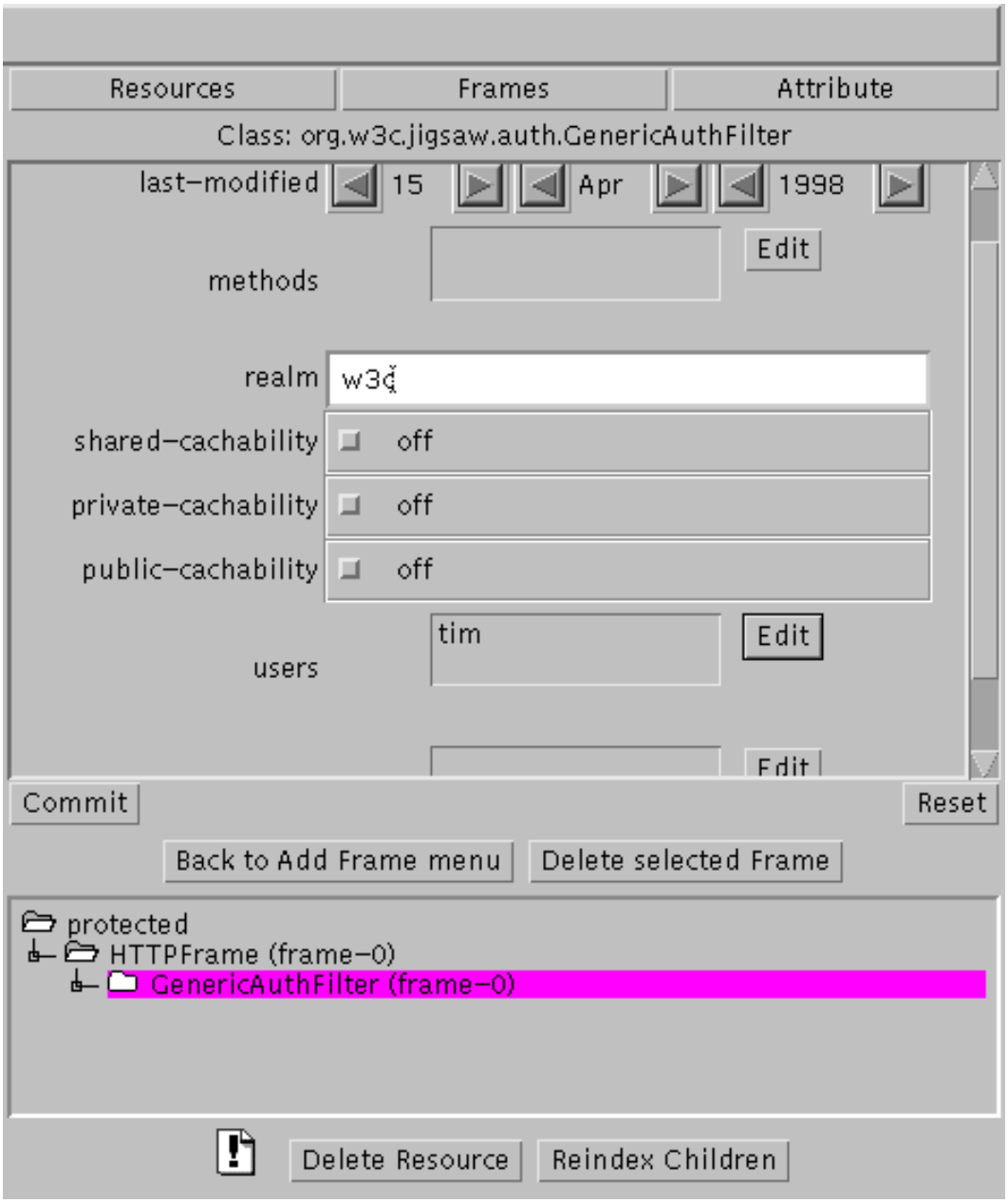


5) Setup an authentication filter on the appropriate frame. The target of the filter can be either a directory resource (in which case the filter will protect all the area defined by the sub-resources of this directory resource), or any other resource (hence allowing you to protect a single resource). Actually, the filter must be attached to the frame of the resource to filter. In this example we will setup a filter on a DirectoryResource.

1. Select the resource you want to filter
2. Click on the "`Frames`" button on top of the right part of the window
3. Expand the resource node (in the little tree on the right)
4. Select the frame of this resource (should be HTTPFrame)
5. Click on the "`Back to Add  Frame menu`" button
6. Select "`org.w3c.jigsaw.auth.GenericAuthFilter`" as the filter class
7. Click on the "AddFrame" button.



6) Setup the authentication filter parameters. The GenericAuthFilter attributes are described in the reference page. Type "`w3c`" in the "`realm`" field and "`tim`" in the "`users`" field. Don't forget to commit your changes.



That's it, your resource has now a restricted access!

Jigsaw Team

$Id: authentication.html,v 1.3 1999/03/16 13:39:21 bmahe Exp $

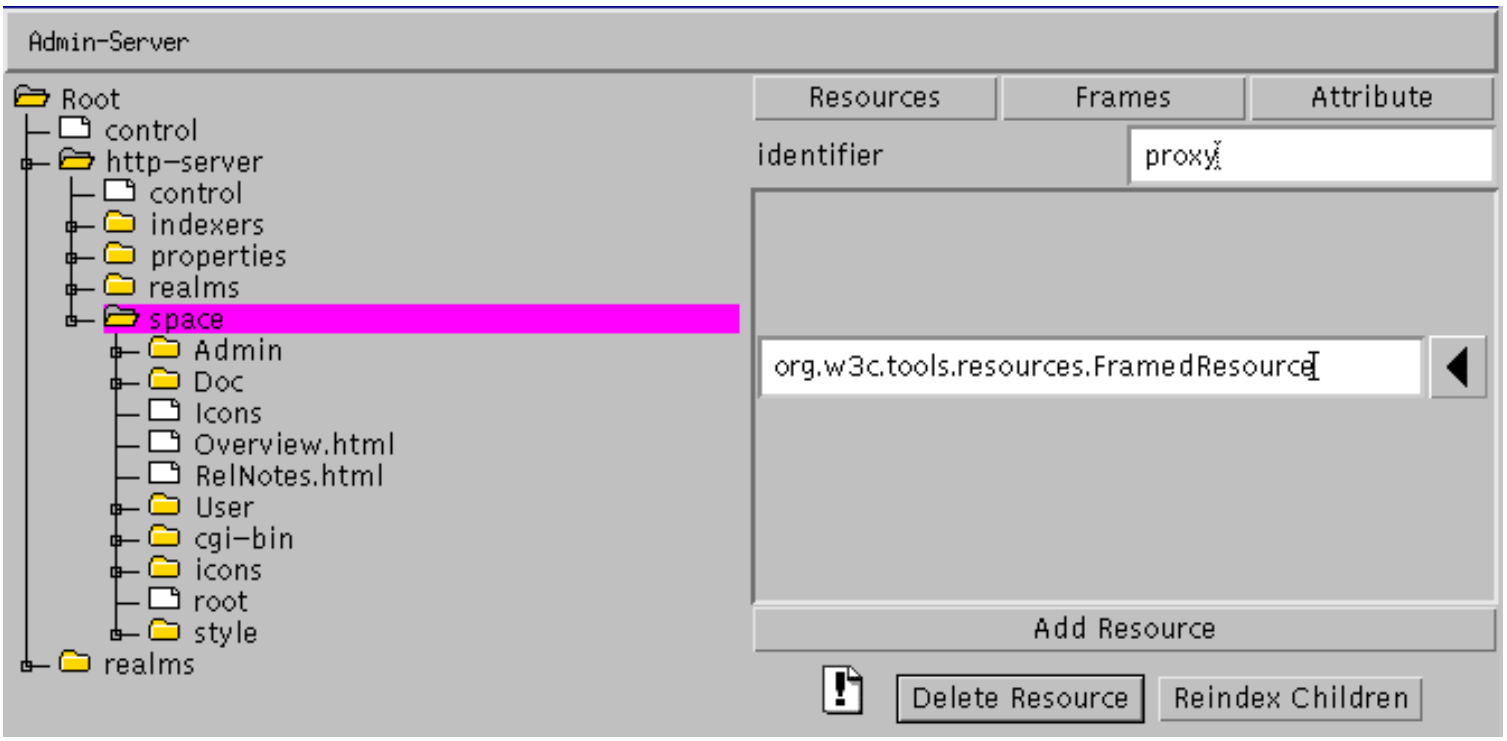
Setting up Jigsaw as a proxy

Jigsaw Home / Documentation Overview

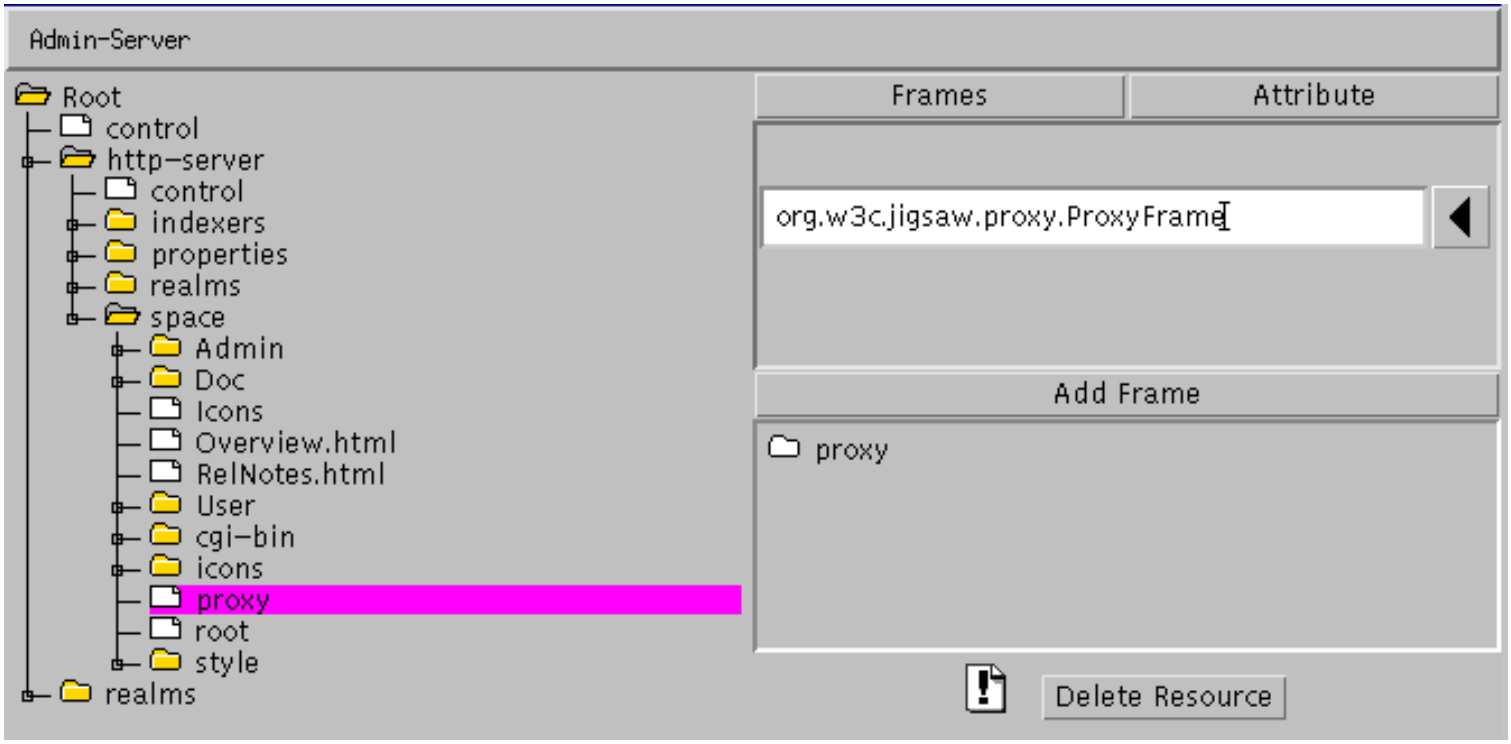
Running Jigsaw as a proxy involves the following steps:

1) Open a JigAdmin window (see JigAdmin documentation)

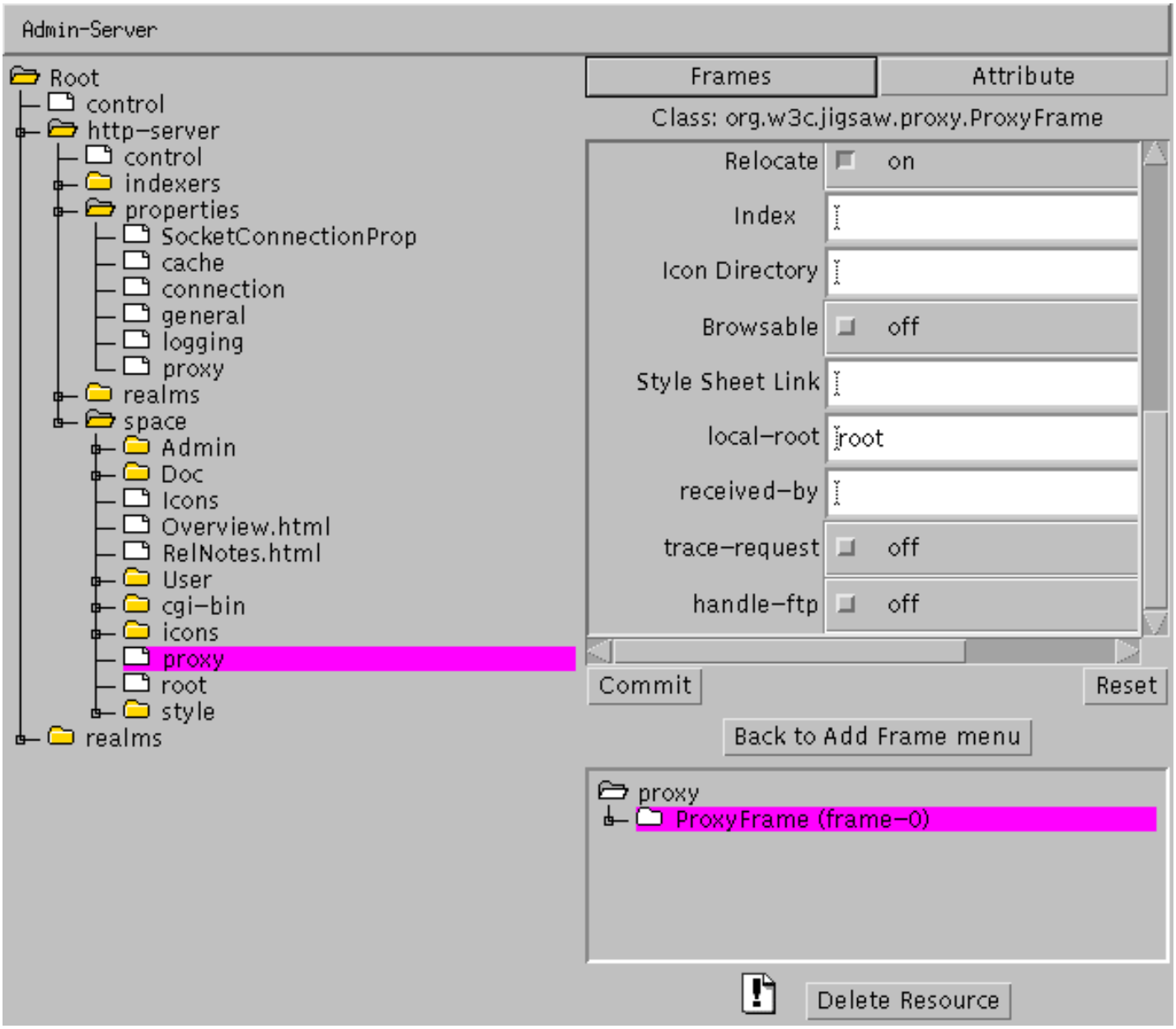
2) Create an instance of the org.w3c.tools.resources.FramedResource resource in the root resource store, by clicking on the "`space`" node and selecting FramedResource as the resource class. Type the resource name in the identifier field (ie proxy), and click on "`Add Resource`".



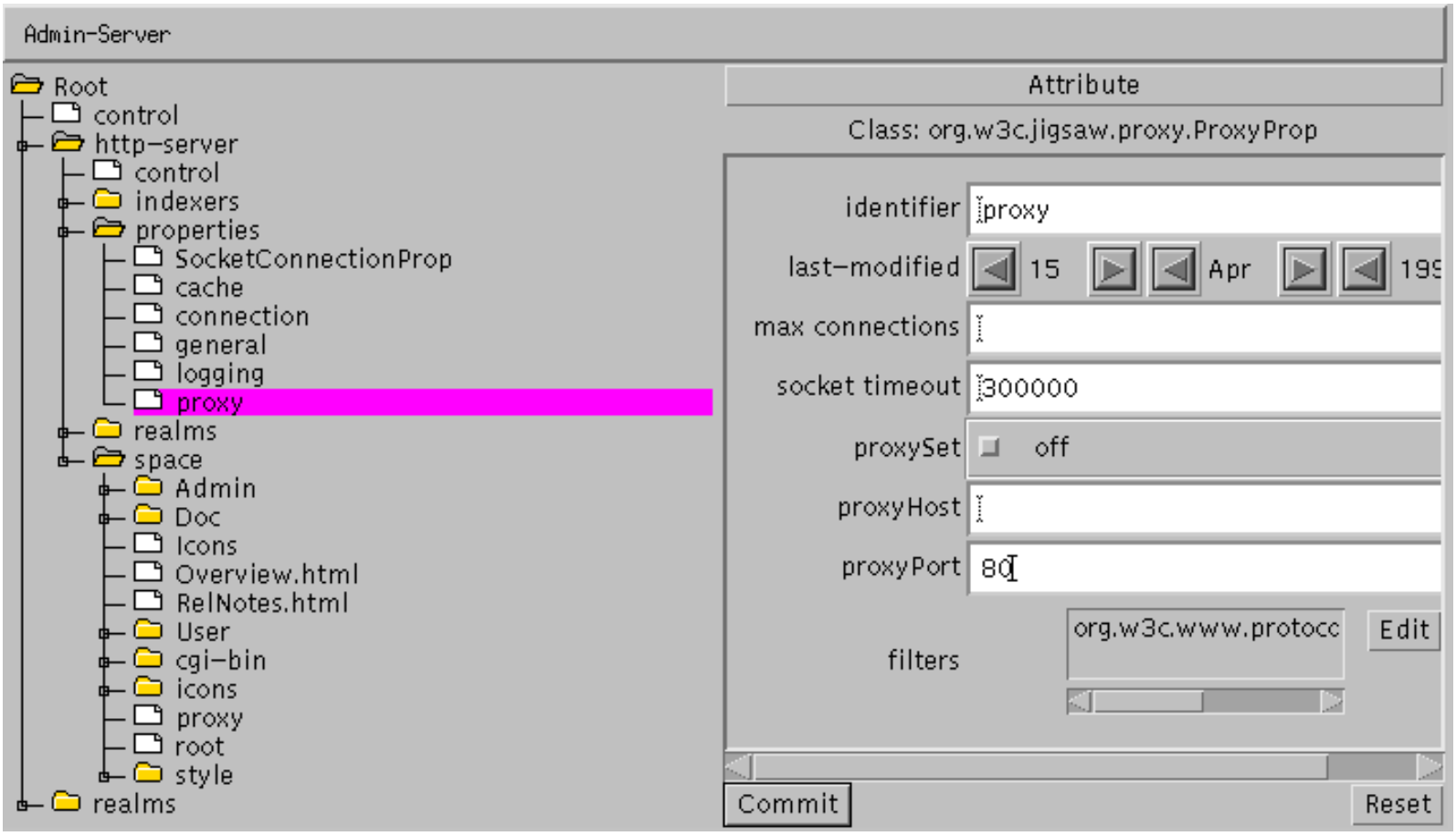
3) Add a ProxyFrame to your new resource. Click on the new "`proxy`" node, select "`org.w3c.jigsaw.proxy.ProxyFrame`" as the frame class to add and click on "`Add Frame`".



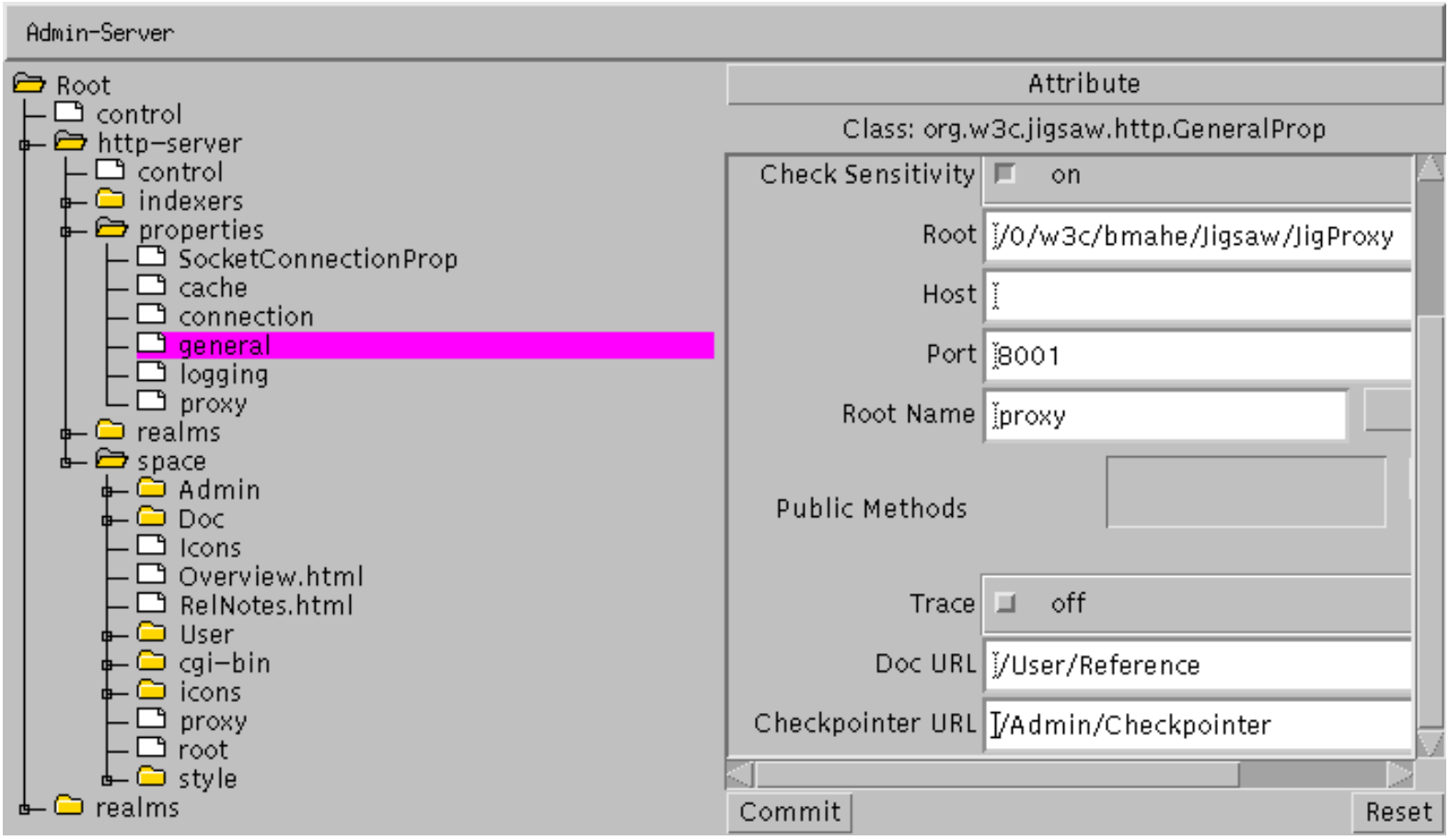
4) Edit the ProxyFrame attributes by selecting the "`proxy`" node in the left tree and expanding the other "`proxy`" node in the rignt tree. Select the "`ProxyFrame`" node and edit the attributes above. Type the old root resource name in the "`local-root`" field (ie root) and click on the "`Commit`" button. Attributes are described in the reference page of ProxyFrame.



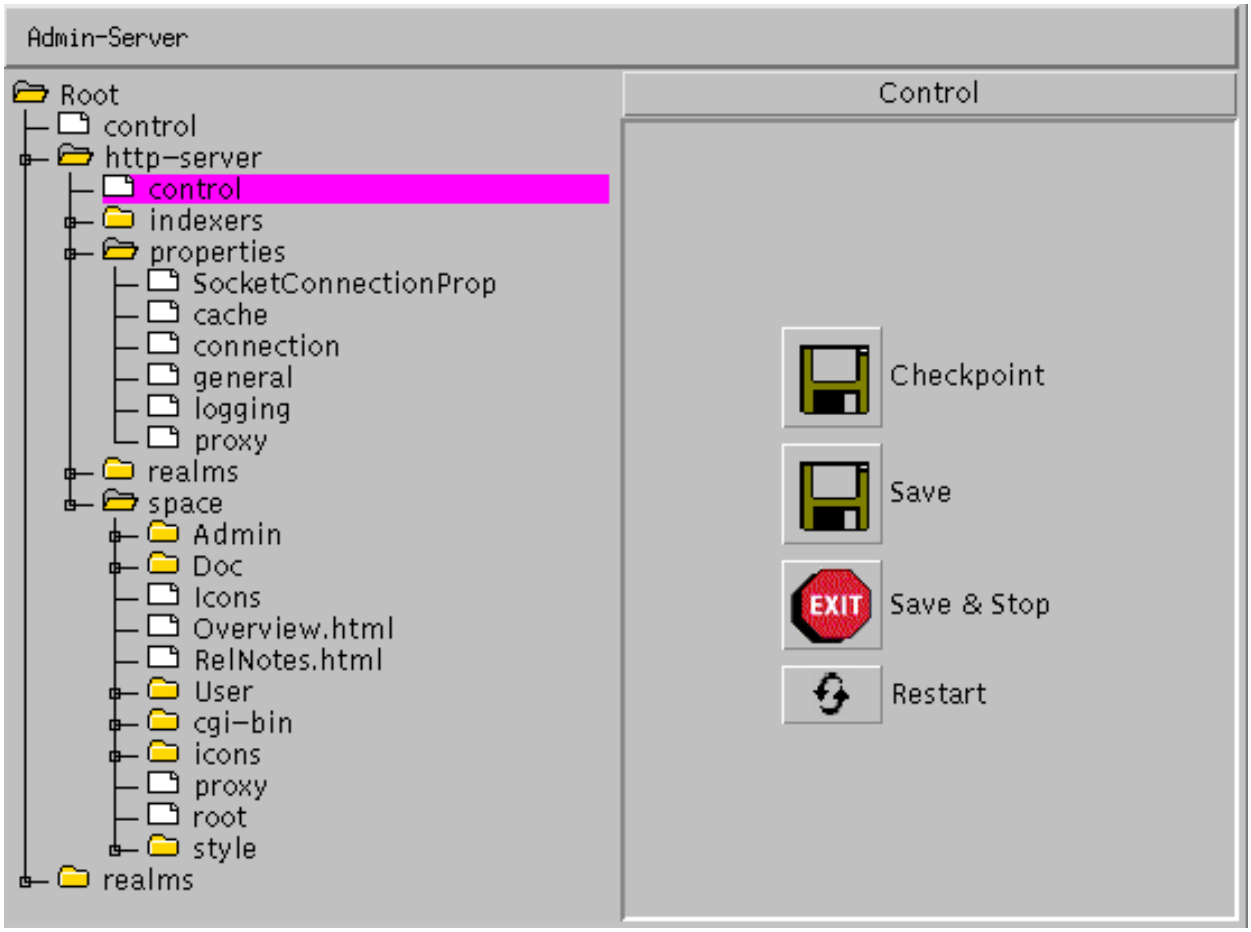
5) The proxy properties. Attributes are described in the reference page of ProxyProp.



6) Edit the general properties. Set the "`Root Name`" property to *proxy* and click on the "`Commit`" button.



7) Save the configuration, **Jigsaw** is now running as a proxy. Reconfigure your browser to run it...and have fun !



Jigsaw Team

$Id: proxy.html,v 1.2 1999/03/16 13:39:47 bmahe Exp $

Virtual Hosting in Jigsaw

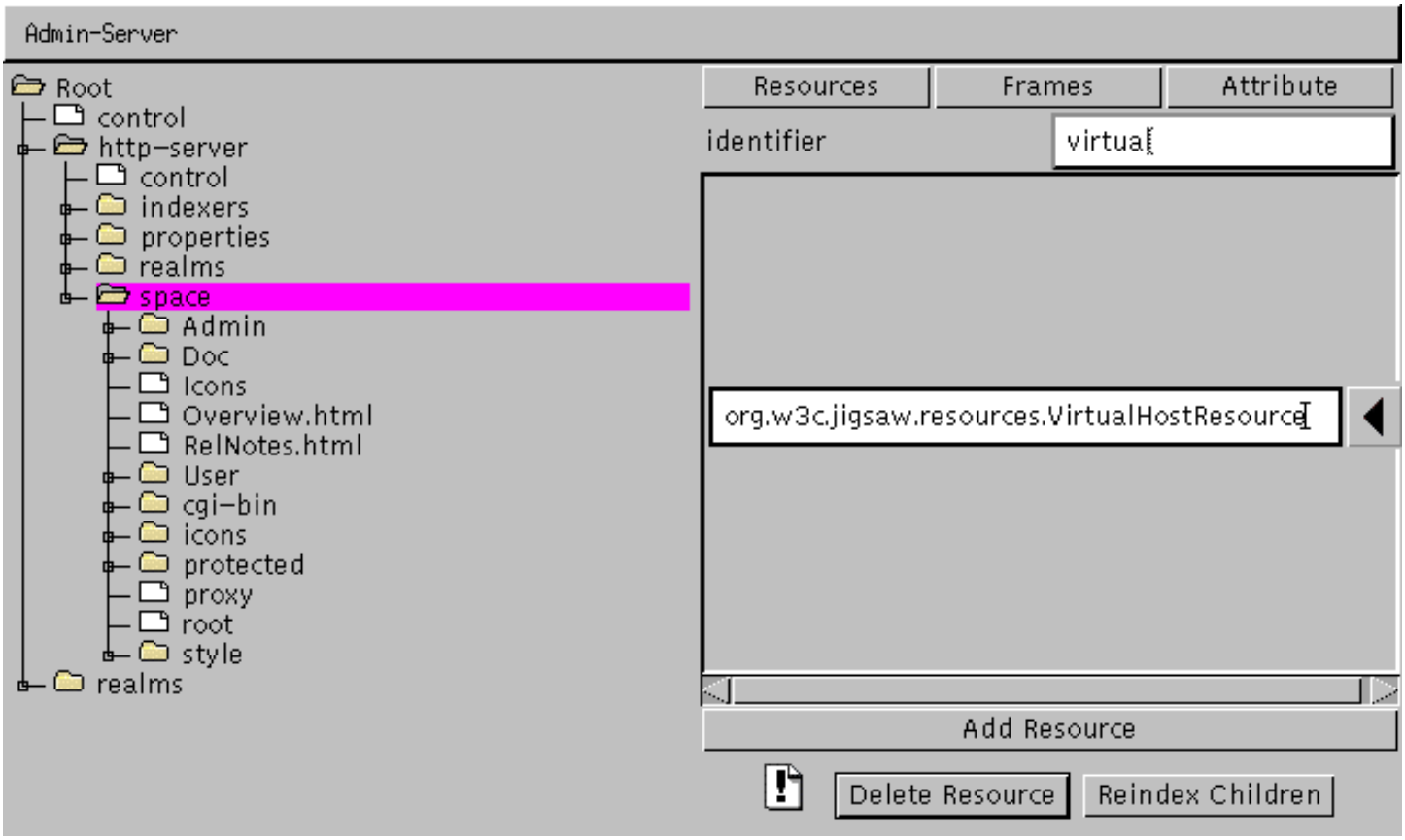
Jigsaw Home / Documentation Overview

Running Jigsaw with virtual hosting involves the following steps:

1) Open a JigAdmin window (see JigAdmin documentation)

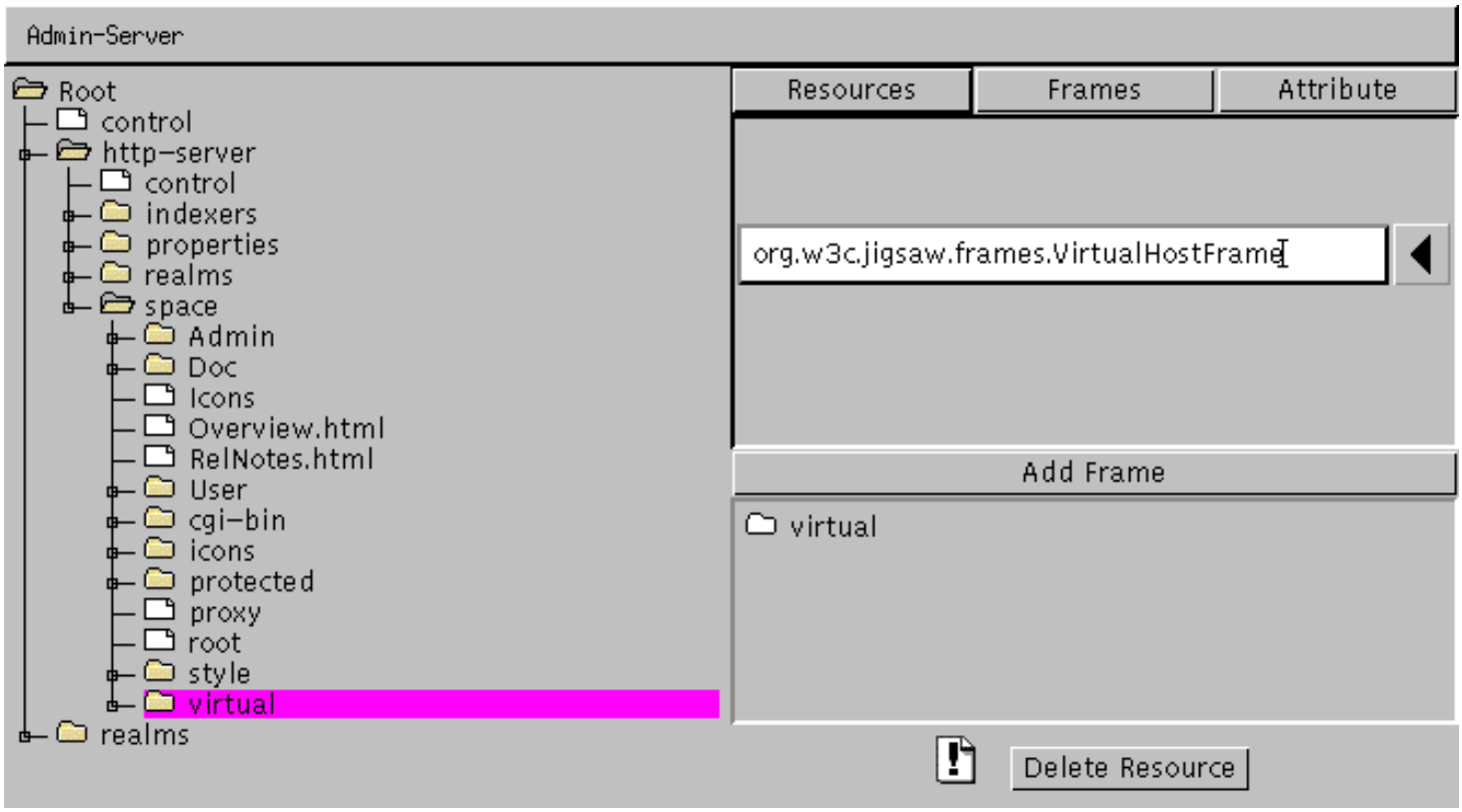
2) Create the virtual resource.

1. Select the "space" node.
2. Click on the "Resources" button on top of the right part of the window.
3. Type the name of the virtual resource in the Identifier field (ie "virtual").
4. Select the "**`org.w3c.tools.resources.VirtualHostResource`**" class.
5. Click on the "Add Resource" button.

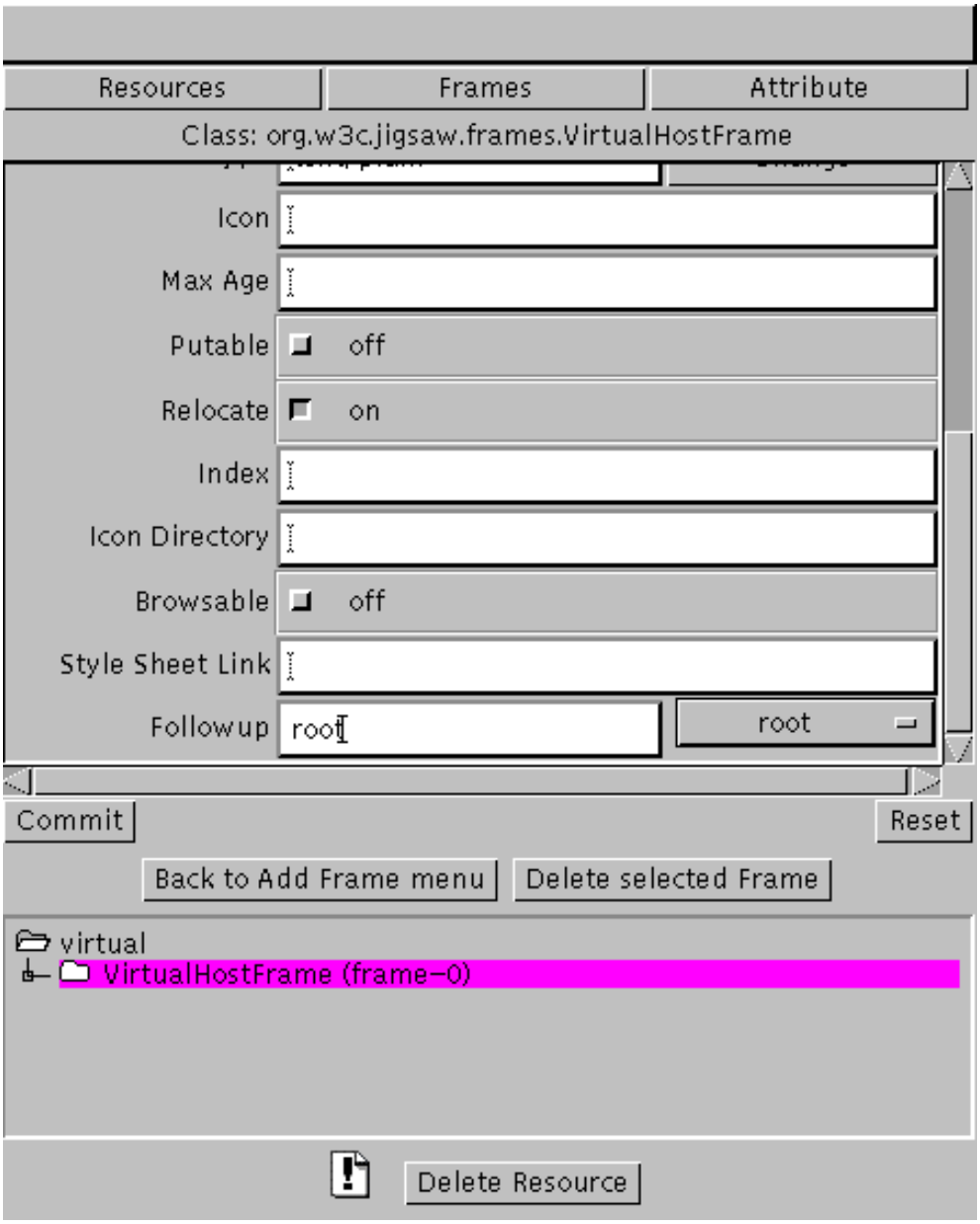


3) Attach a VirtualHostFrame to the new "virtual" resource.

1. Select the new "virtual" node.
2. Click on the "Frames" button (near the "Resources" button).
3. Select the "**`org.w3c.jigsaw.frames.VirtualHostFrame`**" class.
4. Click on the "Add Frame" button.

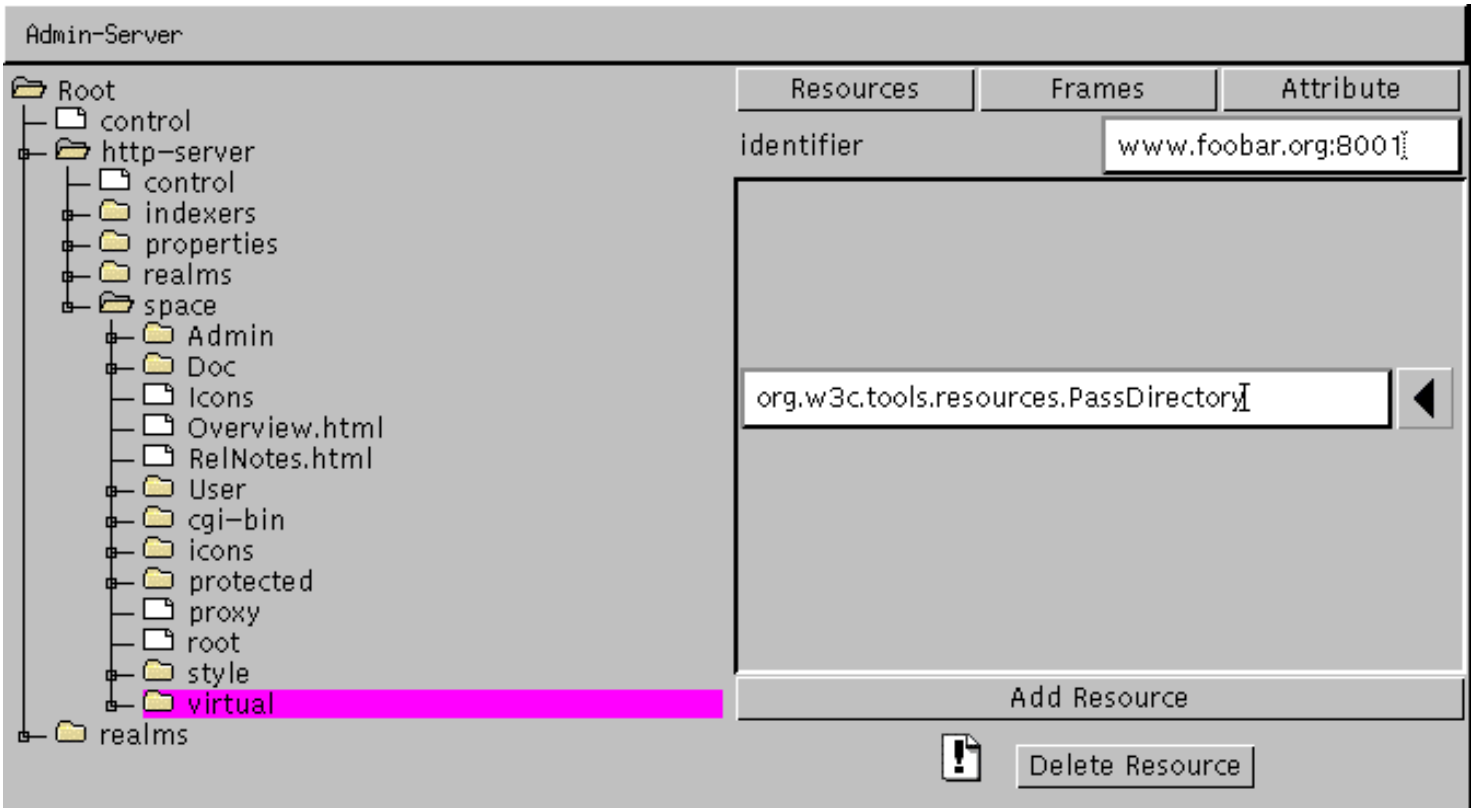


4) Modify the VirtualHostFrame attributes. See the VirtualHostFrame reference page that describe these attributes. Type the old root resource name (ie "root") in the "Followup" field. Don't forget to commit your changes by clicking on the "Commit" button.



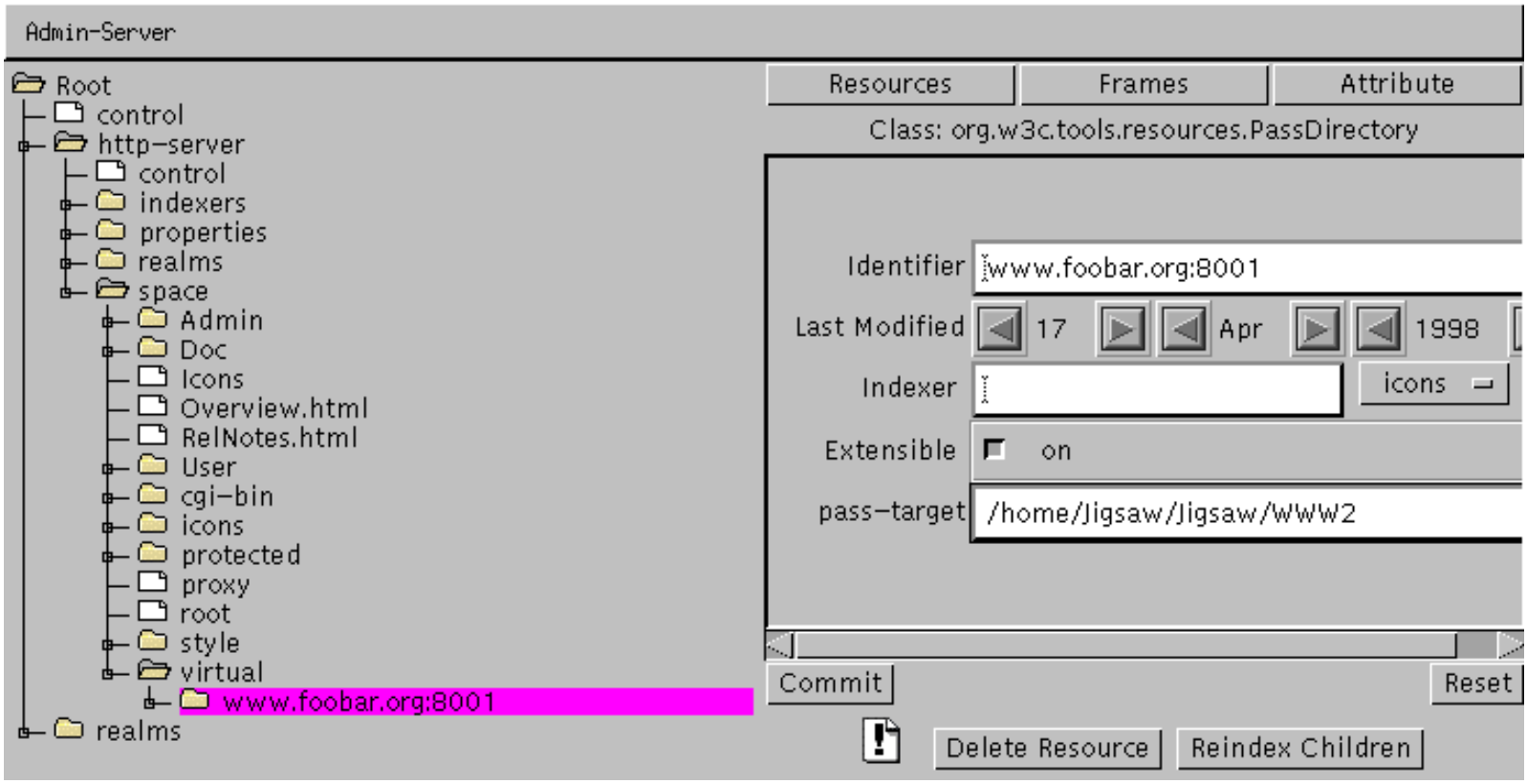
5) Add you virtual host. You can add as many virtual host as you want, but in this example we will add only one virtual host that will be a PassDirectory (see here for more details).

1. Select the "virtual" node.
2. Click on the "Resources" button.
3. Type the name of the virtual host with the port number in the "Identifier" field (ie www.foobar.org:8001)
4. Select the "o**`org.w3c.tools.resources.PassDirectory`**" class.
5. Click on the "Add Resource" button.



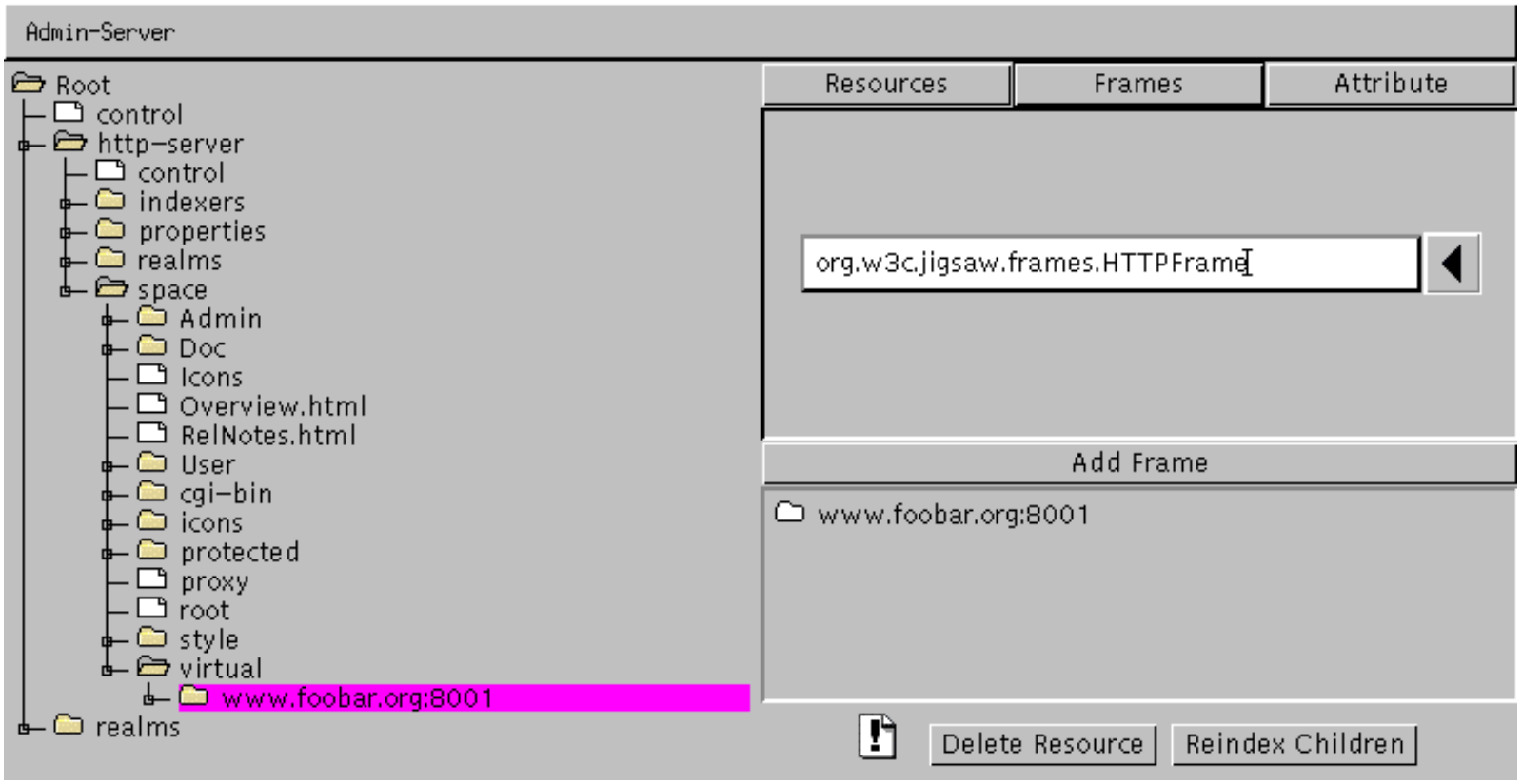
6) Configure your virtual host.

1. Expand the "virtual" node.
2. Select the child node "www.foobar.org:8001".
3. Click on the "Attribute" button.
4. Type, in the "pass-target" field, the path of the directory (ie /home/Jigsaw/Jigsaw/WWW2) that will be the root directory of your virtual host.
5. Commit your changes by clicking on the "Commit" button.

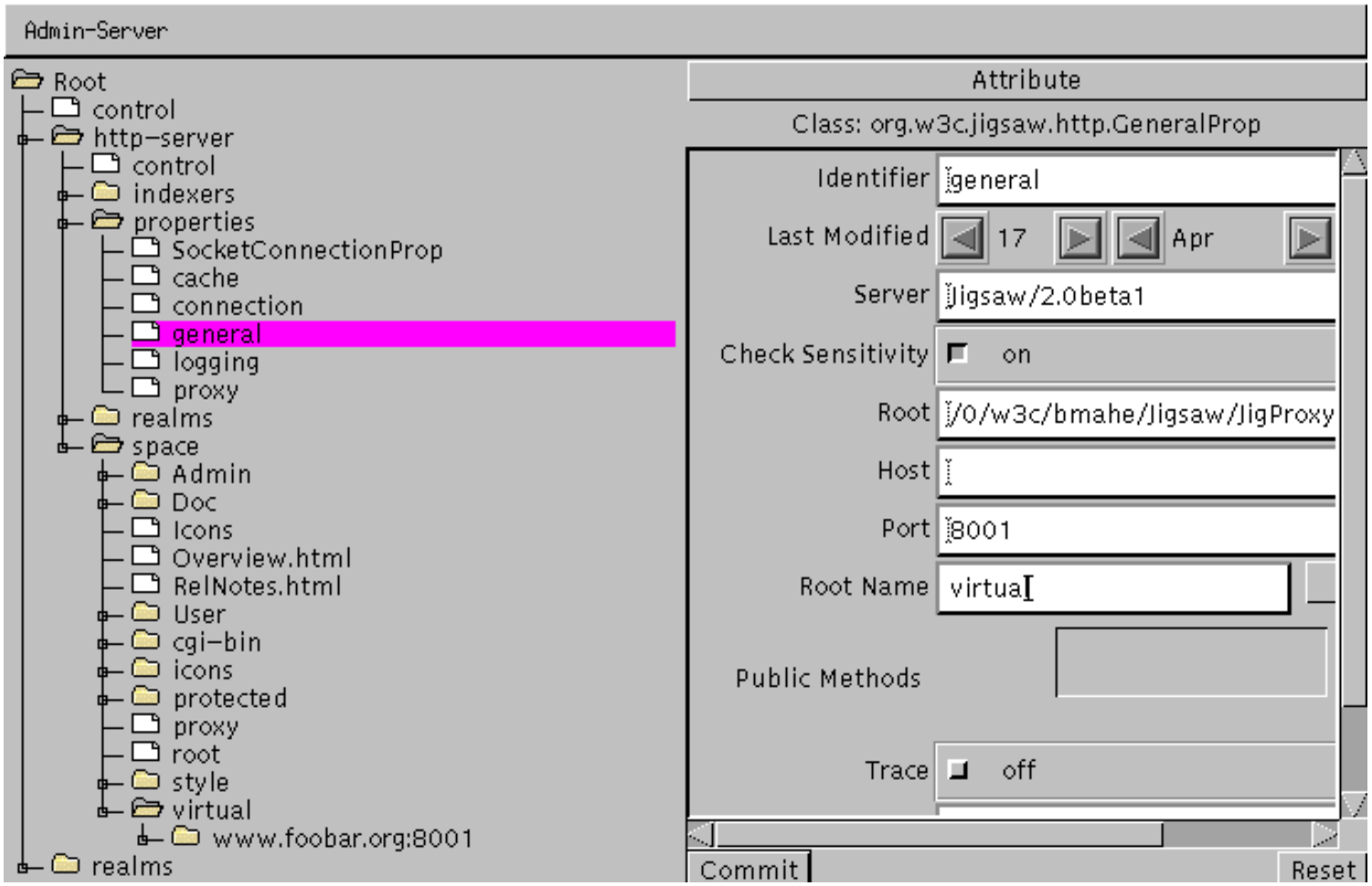


7) Add a HTTPFrame to your new PassDirectory.

1. Select the "www.foobar.org:8001" node.
2. Click on the "Frames" button.
3. Select the "**`org.w3c.jigsaw.frames.HTTPFrame`**" class.
4. Click on the "Add Frames" button.



8) Edit the general properties. Set the "Root Name" property to "virtual" and click on the "Commit" button.



9) Save the configuration, Jigsaw has a new virtual host! Now incoming request for www.foobar.org on port 8001 will be on the /virtual/www.foobar.org:8001 resources, the others requests will be dispatched to the "root" resource.

Jigsaw Team

$Id: virtual-hosting.html,v 1.5 1999/03/16 13:39:56 bmahe Exp $

Jigsaw
How to add a new Mime Type

Jigsaw Home / Documentation Overview

The association between mime types and file extensions is managed by the Indexers in **Jigsaw**. For example, the html extension is associated to a FileResource with an HTTPFrame, the content-type of this HTTPFrame is text/html.

So, if you want to add a new extension/mime-type association, you just have to add the new extension in the default indexer under the "extensions" node.

Example: You want to add the **xml** extension. Add a FileResource associated to an HTTPFrame to the "extensions" node of the "default" indexer. Set the HTTPFrame content-type to **text/xml**. Now files with xml extension (test.xml for example) will be indexed like that.

For more details on indexers management, read the JigAdmin 2.0 documentation or the JigAdmin 1.0 documentation about indexers.

Jigsaw Team
$Id: mimetype.html,v 1.1 1999/03/24 10:07:32 null Exp $

Jigsaw
How to compile Jigsaw

Jigsaw Home / Documentation Overview

Jigsaw is delivered with a set of makefiles, so you can use the make tool to compile **Jigsaw**.

On UNIX machine

According to your shell (for example **bash**), just set the following environment variable:

MAKEDIR = `<instdir>/Jigsaw/src/makefiles`

export **MAKEDIR**

Note: Since 2.0.2, the **MAKEDIR** variable is no more used, so don't take care of it.

Now you have to update your **CLASSPATH** to compile **Jigsaw** and use the new compiled classes.

CLASSPATH = `<instdir>/Jigsaw/src/classes/:...`

instead of

CLASSPATH = `<instdir>/Jigsaw/classes/jigsaw.jar:...`

then

export **CLASSPATH**

Then you can use make in any directory under `<instdir>/Jigsaw/src/classes`.

On Windows Machine

You can use the make tool from the GNU tools for Windows.

Then you just have to set the same environment variable (with **bash**):

MAKEDIR = `<instdir>\Jigsaw\src\makefiles`

export **MAKEDIR**

Note: Since 2.0.2, the **MAKEDIR** variable is no more used, so don't take care of it.

And your **CLASSPATH** should become:

CLASSPATH = `<instdir>\Jigsaw\src\classes\:...`

instead of

CLASSPATH = `<instdir>\Jigsaw\classes\jigsaw.jar:...`

then

export **CLASSPATH**

Then you can use make in any directory under `<instdir>\Jigsaw\src\classes`, provided you have "make" installed.

Jigsaw Team

$Id: compile.html,v 1.10 1999/03/26 15:34:04 bmahe Exp $

Servlets and Jigsaw

Jigsaw Home / Documentation Overview

Jigsaw is compatible with the Servlet Specification version 2.1.

What are servlets ?

Servlets are server-side extensions programmed against the *Servlet API*. This interface is philosophically equivalent to the old CGI interface, but is both more powerful and extremely more efficient. More information is available at jeeves.javasoft.com.

NOTE: As of 2.0beta2, **Jisgaw** is compatible with the **JSDK2.1** including session tracking (using cookies or URL rewriting).

How do I install a servlet ?

IMPORTANT: Before setting up servlets in **Jigsaw**, you must get the **Servlet Development Kit** available at Javasoft and update your **CLASSPATH** to use those classes (add **jsdk.jar** in your CLASSPATH).

Automatic installation

Since 2.0beta3, by default you just have to put the servlet class file in the `<instdir>/Jigsaw/Jigsaw/WWW/servlet/` directory that is already configured for servlets. For example, if you put **SessionServlet.class** in `<instdir>/Jigsaw/Jigsaw/WWW/servlet/`, this servlet will be reachable at `http://your-server-host/servlet/SessionServlet`.

Note: with some zip tools, the `servlet` directory may have been removed (because it is empty), please check it and (if it's necessary) create it manually before accessing the `http://your-server-host/servlet/` URL.

Manual installation

First of all, you have to choose where you want to place your servlets in your file system; this will usually be a single directory, typically something like *servlet* under your server's **WWW** directory. Once you have created the file system directory, create a org.w3c.jigsaw.resources.DirectoryResource with a org.w3c.jigsaw.servlet.ServletDirectoryFrame frame to export it (if the directory has already been indexed, you have to remove the existing org.w3c.jigsaw.frames.HTTPFrame). That frame will act as a *context* for all the servlets under it

(Jigsaw can handle multiple servlet contexts within a single server).

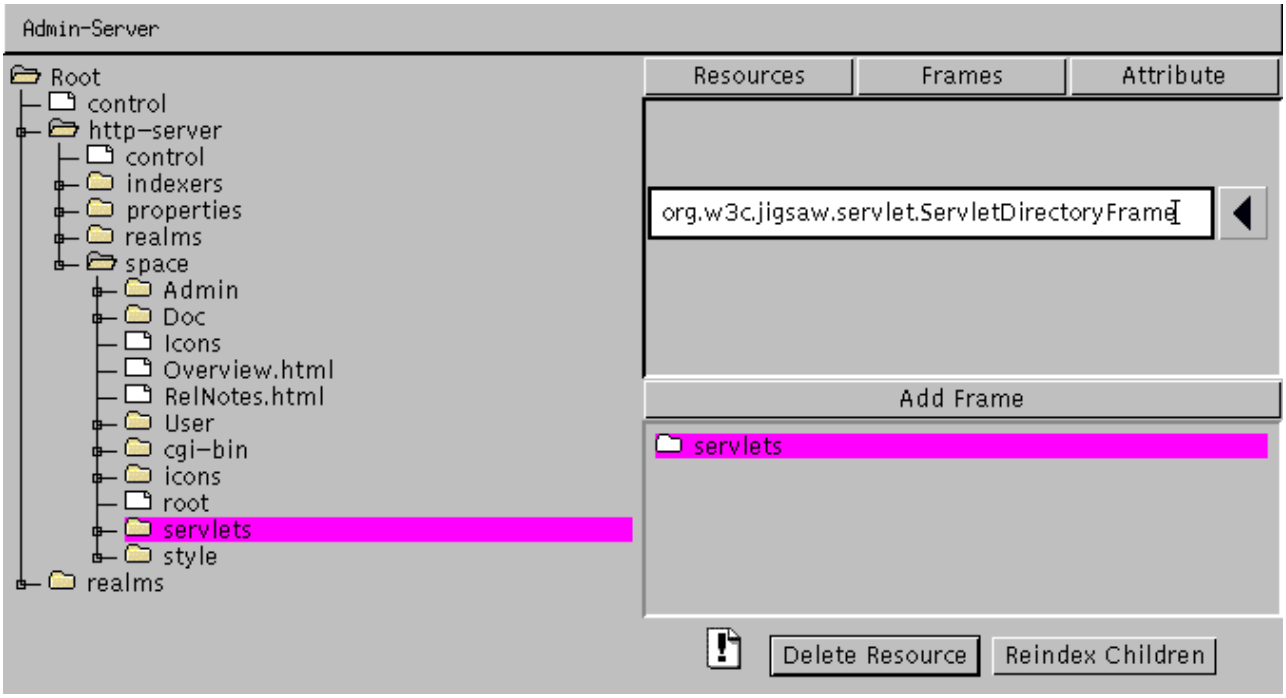
Then, you can setup your servlet manually, or use the servlet indexer (`servlet-indexer`). The servlet indexer will automaticaly create the ServletWrapper and configure it. If you decide to use it, set `servlet-indexer` as the indexer of the DirectoryResource `servlet`.

But you still can do it manually. Put your servlet into the `servlet` directory. Each servlet is managed by a org.w3c.jigsaw.servlet.ServletWrapper, to add a servlet you just have to create a ServletWrapper and in the *servlet-class* field put the class name of the servlet witch must be in the directory relative to the DirectoryResource or in the **CLASSPATH**.

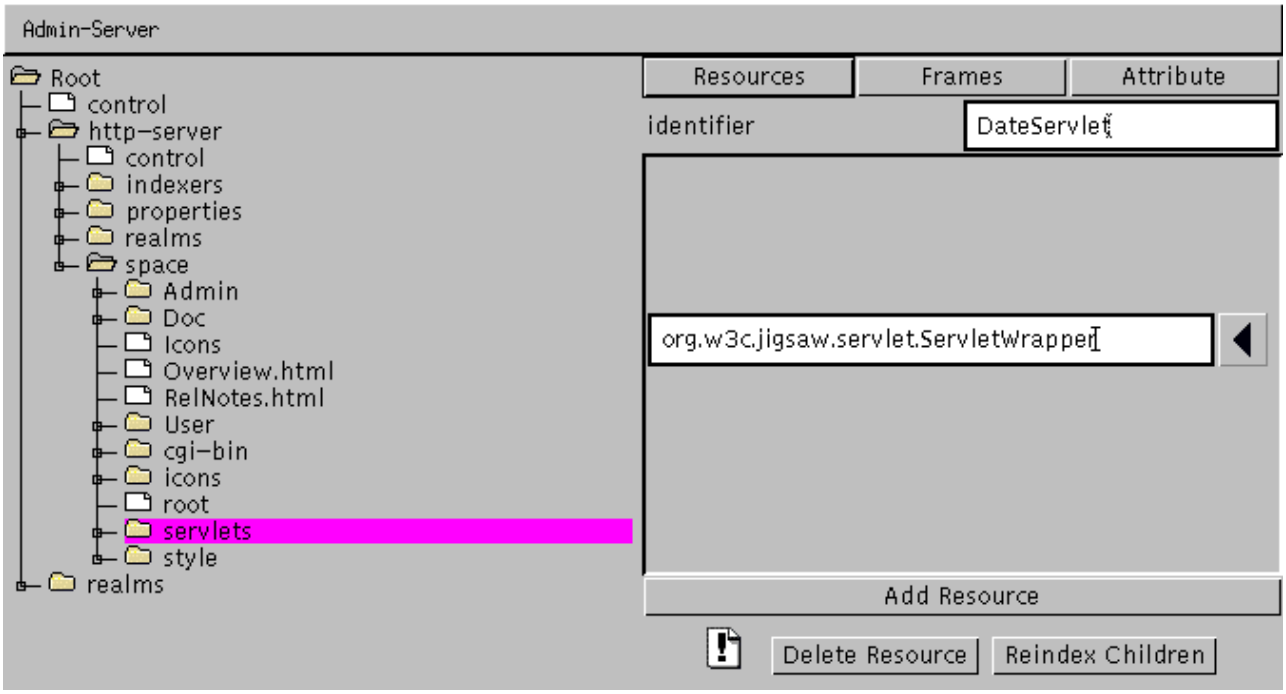
Example :

You want to install the DateServlet in the WWW/servlet directory:

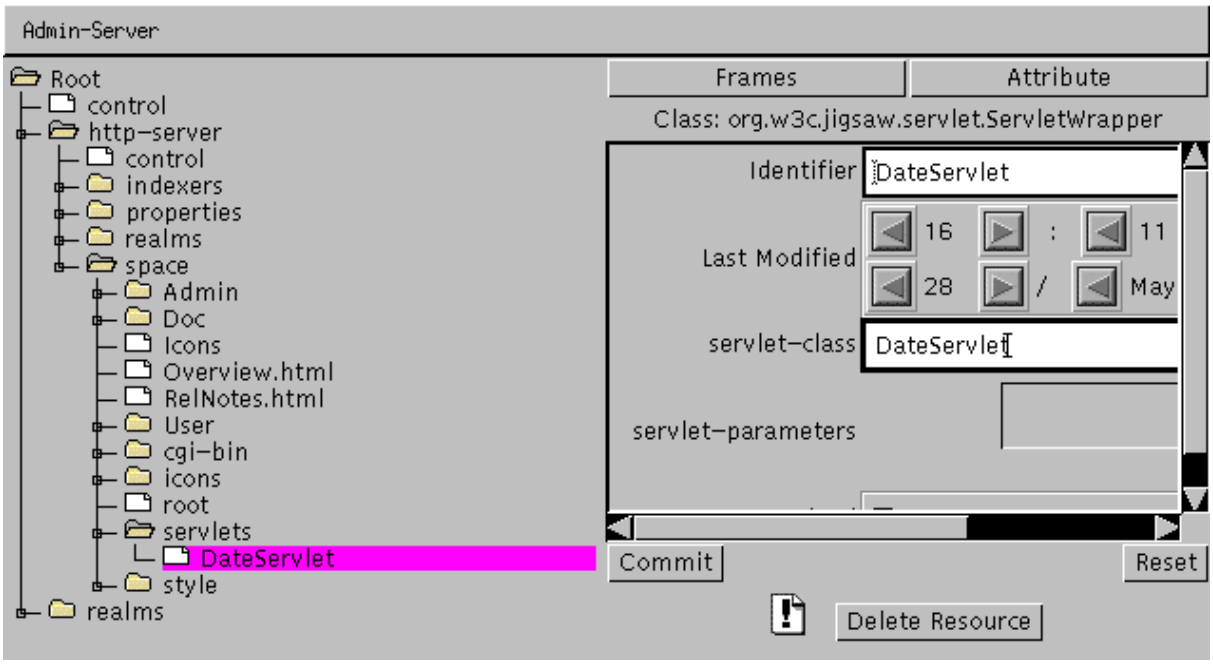
1. Create the directory *servlet* under **WWW**
2. Start the JigAdmin program and unfold all the nodes down to the *space* node.
3. Select the *space* node, and activate the *Resources* editor helper. Enter the DirectoryResource identifier (ie *servlet*) and the class of the resource (ie *org.w3c.jigsaw.resources.DirectoryResource*). This should be done automatically by the indexer.
4. Add a *org.w3c.jigsaw.servlet.ServletDirectoryFrame* to your new DirectoryResource (and eventually remove the old HTTPFrame).



5. Put the servlet class (ie *DateServlet.class*) in WWW/servlet (or in the CLASSPATH).
6. Create a ServletWrapper in the DirectoryResource and call it DateServlet for example.



7. Set the servlet-class field of the ServletWrapper to DateServlet



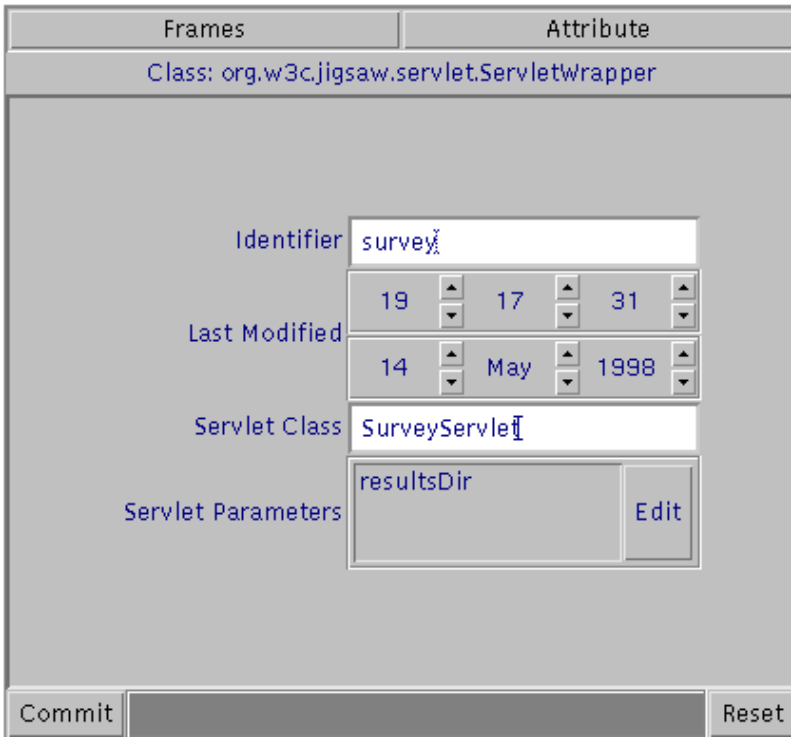
8. Commit your changes

Now DateServlet is reachable at *http://your-server-host/servlet/DateServlet.*

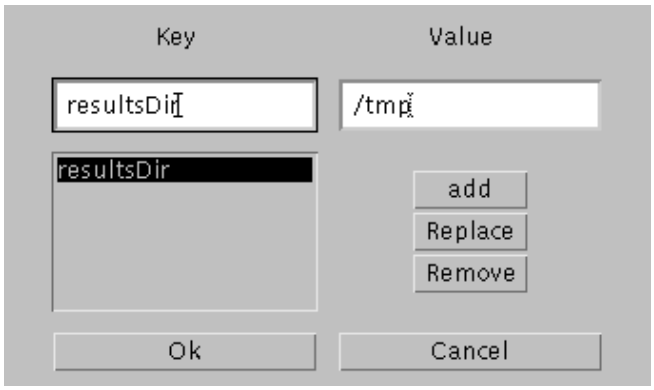
How can I setup servlet parameters?

Some servlets needs parameters at initializing time, so we may need to specify those parameters. Here we describe how to setup the `resultsDir` parameter of the Survey servlet. This servlet needs a directory to write some output, this is the `resultsDir.` In this example we take `/tmp` as the `resultsDir`.

1. Select the servlet wrapper of the SurveyServlet and click on the Edit button (Servlet Parameters).



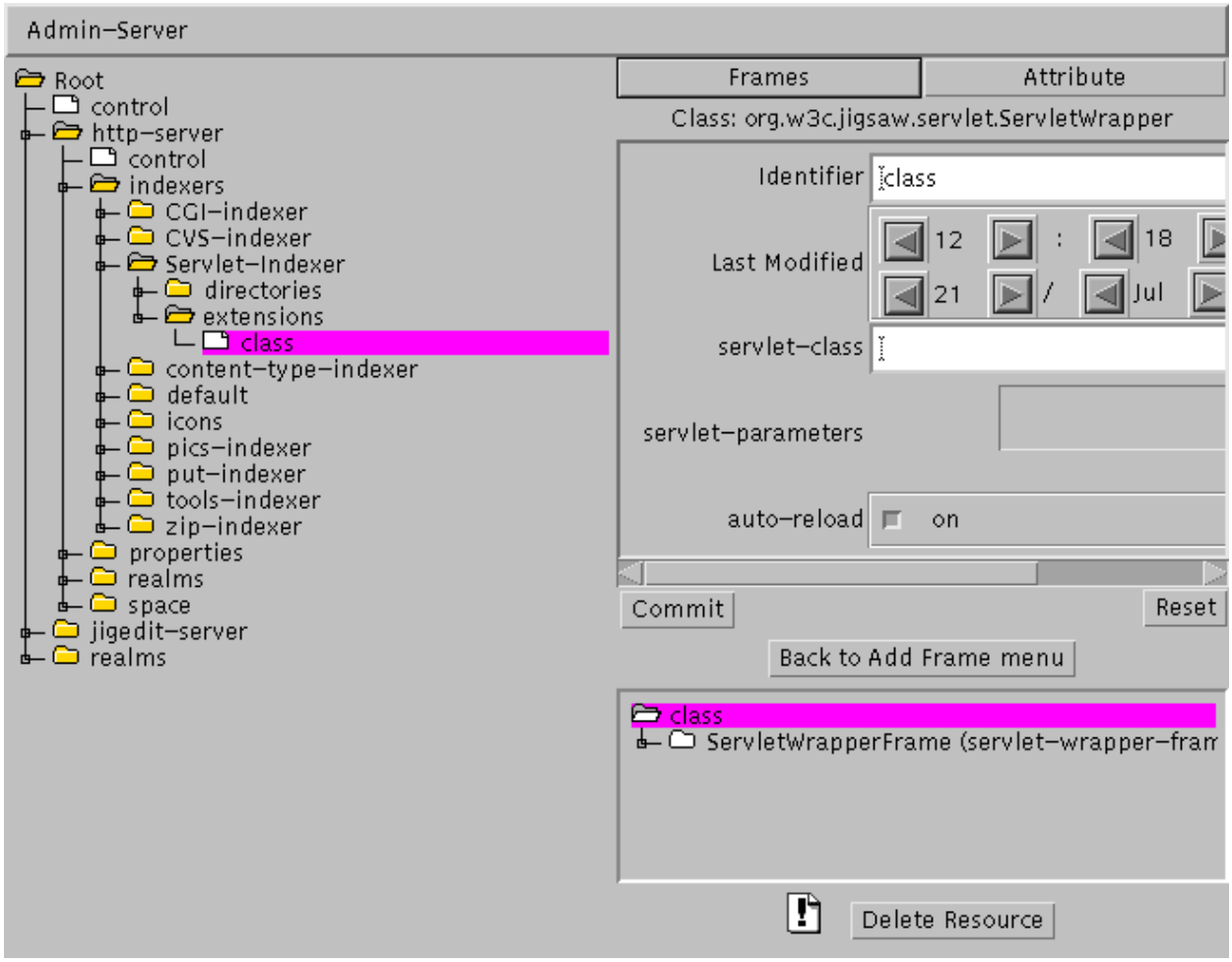
2. Enter the parameter name in the `Key` field and the parameter value in the `Value` field.



3. Click on add (or hit enter). Then click on Ok.
4. Commit your changes and save your configuration.

How do I configure a servlet indexer? (Since 2.0 beta3).

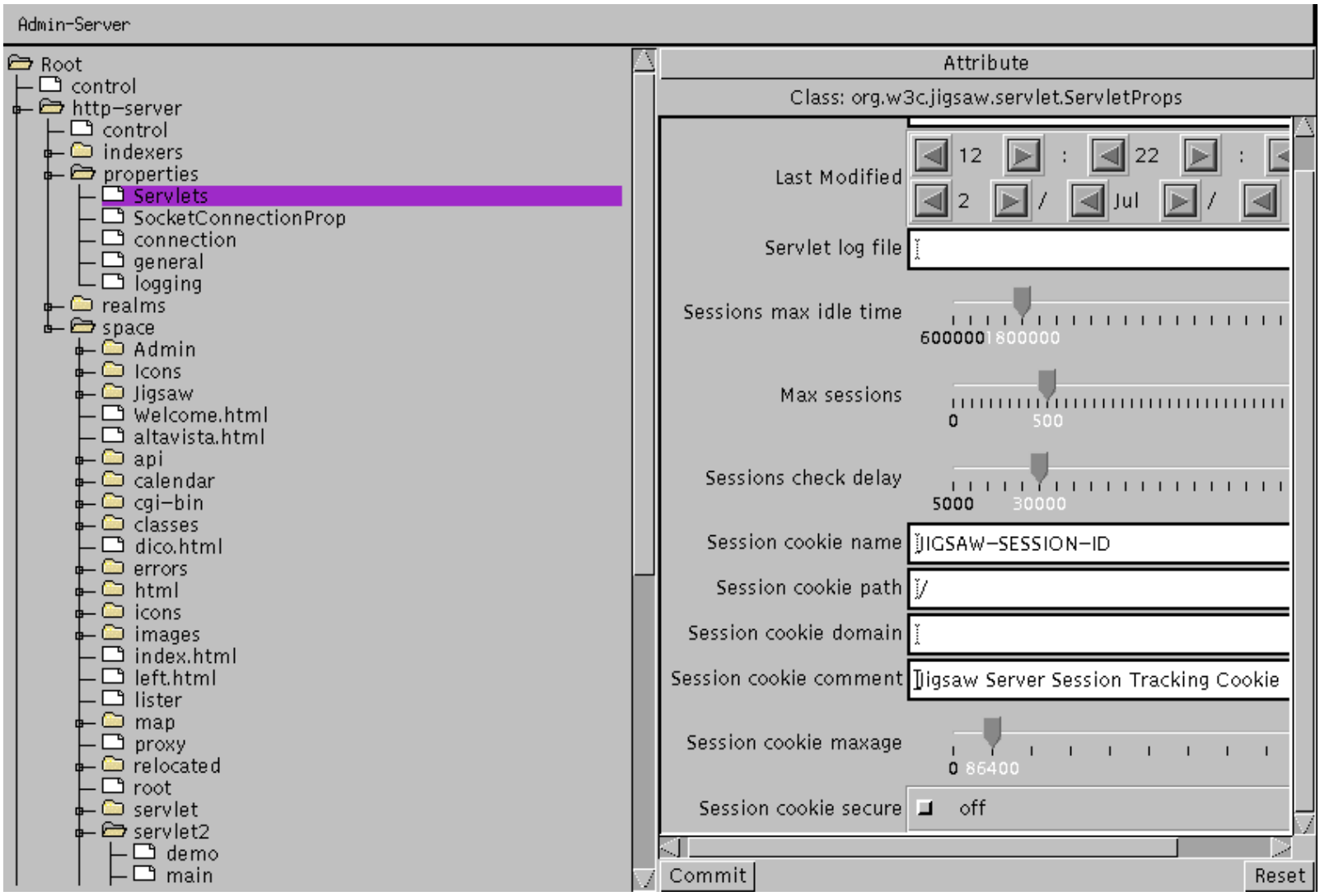
Now you can setup a special indexer for servlets: **`org.w3c.jigsaw.servlet.ServletIndexer`**. This indexer perfoms some servlet specific actions before indexing the class file in a ServletWrapper. It verify that the class file is really a servlet class file, then it put the filename (without the "class" extension) as the identifier and the entire filename as the servlet class. So a class file that is not a servlet class file will NOT be indexed by this indexer but it could be indexed (in a FileResource for example) by one of its super indexer.
See the Indexer Configuration documentation for more details on indexers.



Just add a ServletWrapper called "class" in the extensions node of your ServletIndexer, modify the field you want in the ServletDirectoryFrame (ie: Title, Icon), commit and save your modifications. Now you just have to set your indexer as the indexer of your servlet directory and your servlets will automaticaly be indexed.

How can I setup the servlets properties?

In the last version of **Jigsaw**, you have many properties to configure.



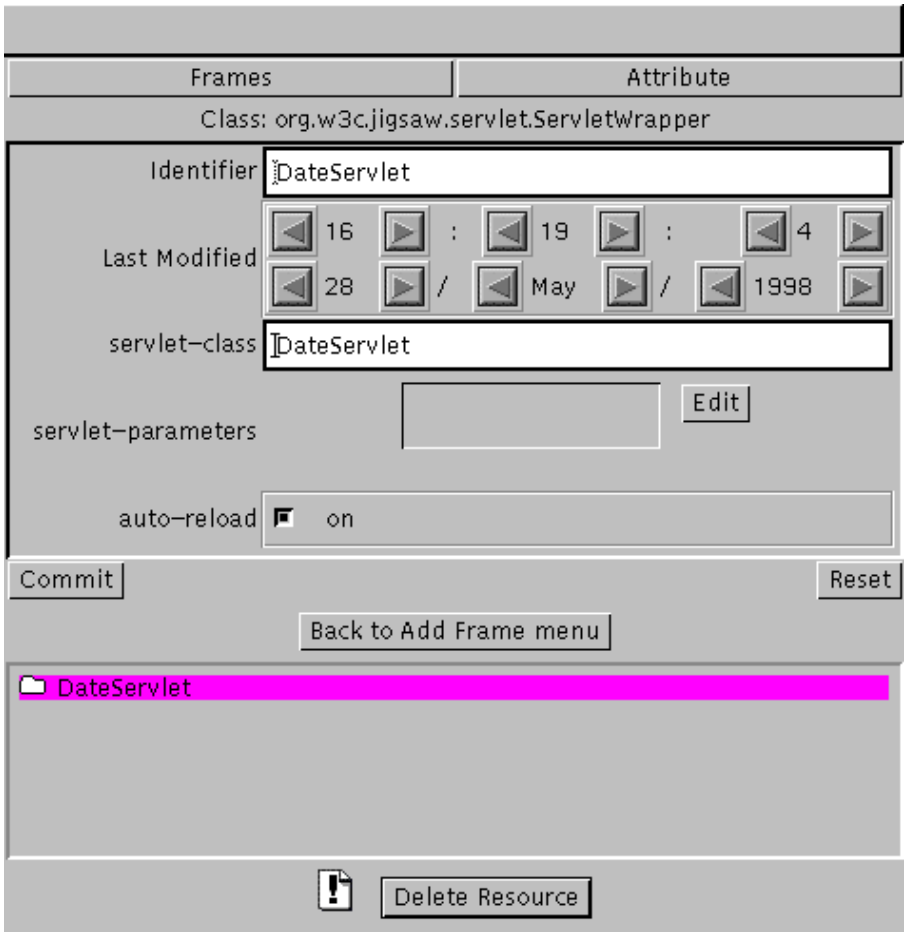
1. **Servlet log file** : You can specify the log file to use for servlets. The default log file is `<instdir>/Jigsaw/Jigsaw/logs/servlets`.
2. **Sessions max idle time** : Amount of time a session is allowed to go unused before it is invalidated. Value is specified in milliseconds.
3. **Max sessions** : Max number of sessions in memory, if the number of sessions exceeds this number the sessions with the bigest idle time will be invalidated. This is checked each time a session is added.
4. **Sessions check delay** : Time interval when **Jigsaw** checks for sessions that have gone unused long enough to be invalidated. Value is an integer, specifying the interval in milliseconds.
5. **Session cookie name** : The name of the cookie carrying the session ID.
6. **Session cookie path** : The path of the cookie carrying the session ID. (optionnal)
7. **Session cookie domain** : The domain field of the cookie carrying the session ID. (optionnal)
8. **Session cookie comment** : The comment field of the cookie carrying the session ID. (optionnal)
9. **Session cookie maxage** : The value of the maximum age of the cookie carrying the session ID. Default value is 86400 seconds (24h).
10. **Session cookie secure** : The secure field of the cookie carrying the session ID.

How does Jigsaw load local servlet classes ?

Jigsaw use a local ClassLoader to load servlet classes from the servlet Directory. If a servlet class is modified when **Jigsaw** is running, the ClassLoader load automatically the new class. It's a very useful feature for servlets developers.

In the two first versions of **Jigsaw2.0** (beta1 and beta2) this feature can be disabled because in some case the `auto-reload` feature could create some problems. In those versions there is a `auto-reload` flag in ServletWrapper.

Now this problem has been resolved and we don't need the `auto-reload` flag anymore. But the `auto-reload` feature works only for the servlet classes located in the servlet directory. Servlets in the **CLASSPATH** are loaded by the system ClassLoader and their modified classes are not reloaded while the server is running.



Note: For very good reasons, when a servlet class is reloaded all sessions are invalidated.

How can I load remote servlets ?

You can use remote servlets by using a RemoteServletWrapper instead of the ServletWrapper.

Example : Add the remote servlet *http://www.servlet.com/RemoteServlet.class* to jigsaw.

1. Create the RemoteServletWrapper in the ServletDirectory.
2. Set its "servlet-base" field to *http://www.servlet.com/*
3. Set its "servlet-class" field to "*RemoteServlet*" (without .class)
4. Commit changes.

Note : the url *http://www.servlet.com/RemoteServlet.class* must be a Java compliant class file.

I need to compile Jigsaw with servlets, how should I do?

NOTE: The default version of **Jigsaw** has been compiled with servlet support, to use it dowload the **JSDK** available at Javasoft and update your **CLASSPATH** to use those classes.

Be sure to have a recent version of the servlet classes. Modify the Makefile in src/classes/org/w3c/jigsaw and add the servlet package at the end of the PACKAGES list, like this:

```
    tutorials \
```

```
zip \
servlet
```

Then, uncomment the line 146 of org/w3c/jigsaw/ssi/commands/DefaultCommandRegistry.java like this:

```
new org.w3c.jigsaw.ssi.servlets.ServletCommand()
```

and adds the servlets package in the Makefile of the org.w3c.jigsaw.ssi package, like this:

```
PACKAGES = \
commands\
jdbc \
servlets
```

Now, you can recompile jigsaw.

Jigsaw Team

$Id: servlets.html,v 1.57 1999/04/01 08:07:58 bmahe Exp $

All resources All frames

ServletMapperFrame

This frame is used to map a file to a servlet, for example jsp file are mapped to the JSPServlet.This frame update the path info (and the path translated) and perform an internal redirect to the servlet.

Inherits

The ServletMapperFrame class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ProtocolFrame
- HTTPFrame

Attributes description

The ServletMapperFrame defines the following attributes:

- servlet-url

`servlet-url`

semantics

The servlet URI where the request will be redirected.(eg /servlet/JSPServlet)

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

Jigsaw Team

$Id: org.w3c.jigsaw.servlet.ServletMapperFrame.html,v 1.2 1999/09/08 14:58:11 ylafon Exp $

Jigsaw
How to use JigKill

Jigsaw Home / Documentation Overview

This tool is used to save and/or kill **Jigsaw**

Usage

java org.w3c.jigsaw.admin.JigKill [opions] admin-server-url

Options

- -u username *User name (default to "admin")*
- -p password *Password (required)*
- --username *Same as -u*
- --password *Same as -p*
- --save *Save configuration of all servers*
- --stop *Stop all servers*
- --ping *Check if servers are reachable*

Example usage

Be sure to have jigsaw.jar, sax.jar and xp.jar in your **CLASSPATH**. Read the installation page.

- Save and exit the servers:
  java org.w3c.jigsaw.admin.JigKill -u admin -p admin http://your-server-host:8009/
  OR (using explicit options)
  java org.w3c.jigsaw.admin.JigKill -u admin -p admin --save --stop http://your-server-host:8009/
- Stop the server (as fast as possible, e.g. for system shutdown):
  java org.w3c.jigsaw.admin.JigKill -u admin -p admin --stop http://your-server-host:8009/
- Save the current configuration:
  java org.w3c.jigsaw.admin.JigKill -u admin -p admin --save http://your-server-host:8009/

- Check if a server is still alive:
  java org.w3c.jigsaw.admin.JigKill -u admin -p admin --ping http://your-server-host:8009/

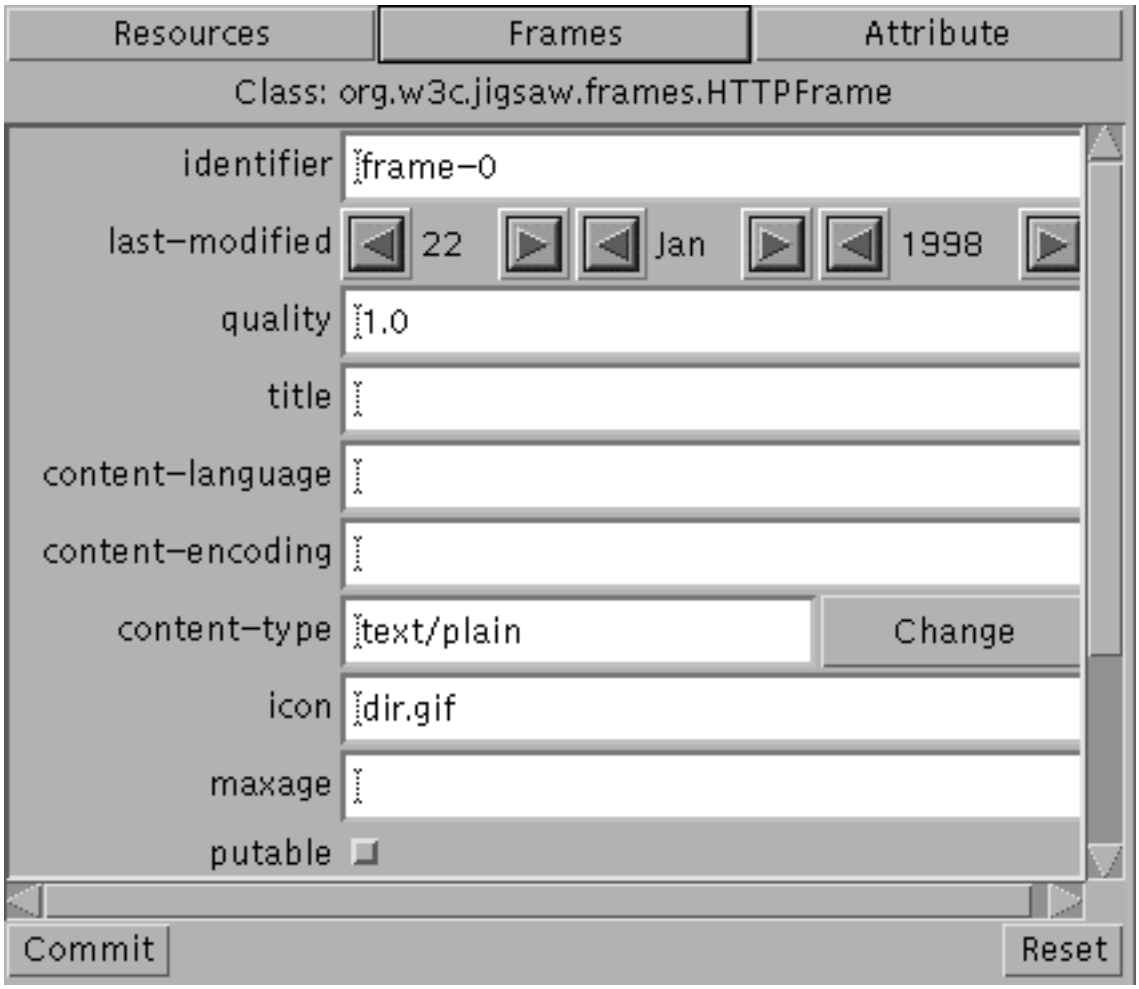
Thanks to Roland Mainz for his contribution on JigKill.

Jigsaw Team
$Id: jigkill.html,v 1.3 1999/09/09 09:34:06 bmahe Exp $

Configuration of Attributes

Jigsaw Home / Documentation Overview

Resources, frames and filters have a set of attributes describing them. Some of them are automatically computed, like the size of a file, some can be set by the user. The size of a file will be calculated automatically while the content-type can be set manually (for example, you can set a specific .foo file to be `text/html`, but you cannot say that a 1234 bytes file has a size of 2345 bytes). For each editable attribute, **JigAdmin** will use a specific indexer. More than that, the editor used will depend on the resource edited.



In the previous screenshot you can see:

- A text field, to modify the identifier name (use that with great care).
- A set of buttons to modify the date.
- A textfield or a slider to modify the quality

- some other textfield (the default editor for most attributes).
- a popup on the "change" button for the content type, allowing you to select your content type from a menu.
- some other text fields
- a radio button to change the "putable" attribute (boolean value).

Once all the changes has been done in the interface, you must tell the server that you change things. To do so, press the "Commit" button located bottor left of the attributes helper. If you made some modification, and you want the original values back (ie: the last values before a commit), press the "reset" button located bottom right of the helper.

The changes are not done incrementally, to allow you the "reset", and to avoid doing unnecessary requests to the admin server, and save bandwith (usually you will do that locally, but that is not a valid reason to waste bytes on the wire :)).

Jigsaw Team

$Id: attributes.html,v 1.7 1999/03/16 13:39:19 bmahe Exp $

```
// CgiFrame.java
// $Id: CgiFrame.html,v 1.2 1998/07/09 10:49:14 benoit Exp $
// (c) COPYRIGHT MIT and INRIA, 1996.
// Please first read the full copyright statement in file COPYRIGHT.html

package org.w3c.jigsaw.frames ;

import java.io.* ;
import java.util.*;
import java.net.*;

import org.w3c.tools.resources.*;
import org.w3c.tools.resources.ProtocolException;
import org.w3c.www.mime.* ;
import org.w3c.jigsaw.http.* ;
import org.w3c.www.http.*;
import org.w3c.jigsaw.auth.AuthFilter;

/**
 * Parsing the CGI output - The CGIHeaderHolder, to hold CGI headers.
 */
class CGIHeaderHolder implements MimeHeaderHolder {
    // Status and Location deserve special treatments:
    String status   = null;
    String location = null;
    // Anyway, he is going to pay for using CGI
    Hashtable headers = null;
    // The MIME parse we are attached to:
    MimeParser parser = null;

    /**
     * The parsing is now about to start, take any appropriate action.
     * This hook can return a <strong>true</strong> boolean value to enforce
     * the MIME parser into transparent mode (eg the parser will <em>not</em>
     * try to parse any headers.
     * <p>This hack is primarily defined for HTTP/0.9 support, it might
     * also be usefull for other hacks.
     * @param parser The Mime parser.
     * @return A boolean <strong>true</strong> if the MimeParser shouldn't
     * continue the parsing, <strong>false</strong> otherwise.
     */

    public boolean notifyBeginParsing(MimeParser parser)
        throws IOException
    {
        return false;
    }

    /**
     * All the headers have been parsed, take any appropriate actions.
```

```
     * @param parser The Mime parser.
     */

    public void notifyEndParsing(MimeParser parser)
        throws IOException
    {
        return ;
    }

    /**
     * A new header has been emited by the script.
     * If the script is not an NPH, then it <strong>must</strong> at least
     * emit one header, so we are safe here, although people may not be safe
     * against the spec.
     * @param name The header name.
     * @param buf The header bytes.
     * @param off The begining of the value bytes  in above buffer.
     * @param len The length of the value bytes  in above buffer.
     */

    public void notifyHeader(String name, byte buf[], int off, int len)
        throws MimeParserException
    {
        if ( name.equalsIgnoreCase("status") ) {
            status = new String(buf, 0, off, len);
        } else if ( name.equalsIgnoreCase("location") ) {
            location = new String(buf, 0, off, len);
        } else {
            String extraval =  new String(buf, 0, off, len);
            if ( headers == null ) {
                headers = new Hashtable(11);
            } else {
                String val = (String) headers.get(name.toLowerCase());
                if ( val != null )
                    extraval = val + "," + extraval;
            }
            headers.put(name.toLowerCase(), extraval);
        }
    }

    /**
     * Get the status emited by the script.
     */

    public String getStatus() {
        return status;
    }

    /**
     * Get the location header value emited by the script.
     */
```

```
    public String getLocation() {
        return location;
    }

    /**
     * Get any header value (except status and location).
     * @param name The name of the header to fetch.
     * @return The string value of requested header, or <strong>null</strong>
     * if header was not defined.
     */

    public String getValue(String name) {
        return (headers == null) ? null : (String) headers.get(name);
    }

    /**
     * Enumerate the headers defined by the holder.
     * @return A enumeration of header names, or <strong>null</strong> if no
     * header is defined.
     */

    public Enumeration enumerateHeaders() {
        if ( headers == null )
            return null;
        return headers.keys();
    }

    /**
     * Get the remaining output of the stream.
     * This should be called only once header parsing is done.
     */

    public InputStream getInputStream() {
        return parser.getInputStream();
    }

    CGIHeaderHolder(MimeParser parser) {
        this.parser = parser;
    }

}

/**
 * Parsing the CGI output - Always create a CGIHeaderHolder.
 */

class CGIHeaderHolderFactory implements MimeParserFactory {

    /**
     * Create a new header holder to hold the parser's result.
```

```
     * @param parser The parser that has something to parse.
     * @return A MimeParserHandler compliant object.
     */

    public MimeHeaderHolder createHeaderHolder(MimeParser parser) {
        return new CGIHeaderHolder(parser);
    }

    CGIHeaderHolderFactory() {
    }

}

/**
 * A simple process feeder class.
 */

class ProcessFeeder extends Thread {
    Process      proc    = null ;
    OutputStream out     = null ;
    InputStream  in      = null ;
    int          count = -1 ;

    public void run () {
        try {
            byte buffer[] = new byte[4096] ;
            int  got      = -1 ;

            // Send the data to the target process:
            if ( count >= 0 ) {
                while ( (count > 0) && ((got = in.read(buffer)) > 0) ) {
                    out.write (buffer, 0, got) ;
                    count -= got ;
                }
            } else {
                while ( (got = in.read(buffer)) > 0 ) {
                    out.write (buffer, 0, got) ;
                }
            }
        } catch (Exception e) {
            System.out.println ("ProcessFeeder: caught exception !") ;
            e.printStackTrace() ;
        } finally {
            // Clean up the process:
            try { out.flush() ; } catch (IOException ex) {}
            try { out.close() ; } catch (IOException ex) {}
            try { proc.waitFor() ; } catch (Exception ex) {}
        }
    }

    ProcessFeeder (Process proc, InputStream in) {
```

```
        this (proc, in, -1) ;
    }

    ProcessFeeder (Process proc, InputStream in, int count) {
        this.proc   = proc ;
        this.out    = proc.getOutputStream() ;
        this.in     = in ;
        this.count  = count ;
    }
}

/**
 * Handle CGI scripts.
 */
public class CgiFrame extends HTTPFrame {

    private final static
        String STATE_EXTRA_PATH = "org.w3c.jigsaw.frames.CgiFrame.extraPath";

    /**
     * Attribute index - The interpreter to use, if any.
     */
    protected static int ATTR_INTERPRETER = -1;
    /**
     * Attribute index - The array of string that makes the command to run.
     */
    protected static int ATTR_COMMAND = -1 ;
    /**
     * Attribute index - Does the script takes care of its headers ?
     */
    protected static int ATTR_NOHEADER = -1 ;
    /**
     * Attribute index - Does the script generates the form on GET ?
     */
    protected static int ATTR_GENERATES_FORM = -1 ;
    /**
     * Attribute index - Do DNS, to fill in REMOTE_HOST env var.
     */
    protected static int ATTR_REMOTE_HOST = -1;
    /**
     * Attribute index - Turn the script in debug mode.
     */
    protected static int ATTR_CGI_DEBUG = -1;

    static {
        Attribute a   = null ;
        Class     cls = null ;
        try {
            cls = Class.forName("org.w3c.jigsaw.frames.CgiFrame") ;
        } catch (Exception ex) {
            ex.printStackTrace() ;
```

```
            System.exit(1) ;
        }
        // The interpreter attribute:
        a = new StringAttribute("interpreter"
                                , null
                                , Attribute.EDITABLE);
        ATTR_INTERPRETER = AttributeRegistry.registerAttribute(cls, a);
        // The command attribute:
        a = new StringArrayAttribute("command"
                                     , null
                                     , Attribute.MANDATORY|Attribute.EDITABLE);
        ATTR_COMMAND = AttributeRegistry.registerAttribute(cls, a) ;
        // The noheader attribute:
        a = new BooleanAttribute("noheader"
                                 , Boolean.FALSE
                                 , Attribute.EDITABLE) ;
        ATTR_NOHEADER = AttributeRegistry.registerAttribute(cls, a) ;
        // The generates form attribute
        a = new BooleanAttribute("generates-form"
                                 , Boolean.TRUE
                                 , Attribute.EDITABLE) ;
        ATTR_GENERATES_FORM = AttributeRegistry.registerAttribute(cls, a);
        // Registerr the DODNS attribute.
        a = new BooleanAttribute("remote-host"
                                 , null
                                 , Attribute.EDITABLE);
        ATTR_REMOTE_HOST = AttributeRegistry.registerAttribute(cls, a);
        // Register the debug mode flag:
        a = new BooleanAttribute("cgi-debug"
                                 , Boolean.FALSE
                                 , Attribute.EDITABLE);
        ATTR_CGI_DEBUG = AttributeRegistry.registerAttribute(cls, a);
    }

    /**
     * Get the interpreter to use to execute the script.
     * This is most usefull for operating systems that don't have a
     * <code>!#</code> convention ala UNIX.
     * @return The interpreter to run the script.
     */

    public String getInterpreter() {
        return getString(ATTR_INTERPRETER, null);
    }

    /**
     * Get the command string array.
     */

    public String[] getCommand() {
        return (String[]) getValue(ATTR_COMMAND, null) ;
```

```
    }

    /**
     * Get the noheader flag.
     * @return The boolean value of the noheader flag.
     */

    public boolean checkNoheaderFlag() {
        return getBoolean(ATTR_NOHEADER, false) ;
    }

    /**
     * Get the generates form flag.
     * @return The boolean value of the generates form flag.
     */

    public boolean checkGeneratesFormFlag() {
        return getBoolean(ATTR_GENERATES_FORM, true) ;
    }

    /**
     * Get the remote host attribute value.
     * If turned on, this flag will enable the REMOTE_HOST env var computation.
     * @return A boolean.
     */

    public boolean checkRemoteHost() {
        return getBoolean(ATTR_REMOTE_HOST, false);
    }

    /**
     * Get the CGI debug flag.
     * @return The boolean value of the CGI debug flag.
     */

    public boolean checkCgiDebug() {
        return getBoolean(ATTR_CGI_DEBUG, false);
    }

    /**
     * Turn the given header name into it's env var canonical name.
     * This guy is crazy enough to run CGI scripts, he can pay for that
     * overhead.
     * @param name The header name.
     * @return A String giving the official env variable name for that header.
     */

    public String getEnvName(String name) {
        int          sl = name.length();
        StringBuffer sb = new StringBuffer(5+sl);
        sb.append("HTTP_");
```

```
        for (int i = 0 ; i < sl ; i++) {
            char ch = name.charAt(i);
            sb.append((ch == '-') ? '_' : Character.toUpperCase(ch));
        }
        return sb.toString();
    }

    /**
     * Handle the CGI script output.
     * This methods handles the CGI script output. Depending on the
     * value of the <strong>noheader</strong> attribute it either:
     * <ul>
     * <li>Sends back the script output directly,</li>
     * <li>Parses the script output, looking for a status header or a
     * location header, or a content-length header, or any combination
     * of those three.
     * </ul>
     * @param process The underlying CGI process.
     * @param request The processed request.
     * @exception ProtocolException If an HTTP error should be sent back
     * to the client.
     */

    protected Reply handleCGIOutput (Process process, Request request)
        throws ProtocolException
    {
        // No header script don't deserve attention:
        if ( checkNoheaderFlag() ) {
            Reply reply = request.makeReply(HTTP.NOHEADER) ;
            reply.setStream (process.getInputStream()) ;
            return reply ;
        }
        // Check for debugging mode:
        if ( checkCgiDebug() ) {
            Reply reply = request.makeReply(HTTP.OK);
            reply.setContentType(MimeType.TEXT_PLAIN);
            reply.setStream(process.getInputStream());
            return reply;
        }
        // We MUST parse at least one header:
        MimeParser p = new MimeParser(process.getInputStream(),
                                      new CGIHeaderHolderFactory());
        Reply           reply  = null ;
        try {
            CGIHeaderHolder h = (CGIHeaderHolder) p.parse();
            // Check for a status code:
            String svalue   = h.getStatus();
            String location = h.getLocation();
            if ( svalue != null ) {
                int status = -1;
                try {
```

```
                status = Integer.parseInt(svalue);
            } catch (Exception ex) {
                // This script has emited an invalid status line:
                String msg = ("Emited an invalid status line ["+
                              svalue + "].");
                getServer().errlog(this, msg);
                // Throw an HTTPException:
                reply = request.makeReply(HTTP.INTERNAL_SERVER_ERROR);
                reply.setContent("CGI script emited invalid status.");
                throw new HTTPException(reply);
            }
            reply = request.makeReply(status);
        } else {
            // No status code available, any location header ?
            reply = request.makeReply((location == null)
                                      ? HTTP.OK
                                      : HTTP.FOUND);
        }
        // Set up the location header if needed:
        if ( location != null ) {
            try {
                reply.setLocation(new URL(getURL(request), location));
            } catch (MalformedURLException ex) {
                // This should really not happen:
                getServer().errlog(this, "unable to create location url "+
                                   location+
                                   " in base "+getURL(request));
            }
        }
        // And then, the remaining headers:
        Enumeration e = h.enumerateHeaders();
        if ( e != null ) {
            while ( e.hasMoreElements() ) {
                String hname = (String) e.nextElement();
                reply.setValue(hname, (String) h.getValue(hname));
            }
        }
        reply.setStream(p.getInputStream()) ;
    } catch (IOException ex) {
        ex.printStackTrace();
    } catch (MimeParserException ex) {
        // This script has generated invalid output:
        String msg = (getURL(request)
                      +": emited invalid output ["+
                      ex.getMessage() +"]");
        getServer().errlog(this, msg);
        // Throw an HTTPException:
        Reply error = request.makeReply(HTTP.INTERNAL_SERVER_ERROR) ;
        error.setContent("CGI error: unable to parse script headers.") ;
        throw new HTTPException (error) ;
    }
```

```
        return reply ;
    }

    /**
     * Add an enviornment binding to the given vector.
     * @param name The name of the enviornment variable.
     * @param val Its value.
     * @param into The vector to which accumulate bindings.
     */

    private void addEnv (String name, String val, Vector into) {
        into.addElement (name+"="+val) ;
    }

    /**
     * Prepare the command to run for this CGI script, and run it.
     * @param request The request to handle.
     * @return The running CGI process object.
     * @exception HTTPException If we weren't able to build the command or
     *    the environment.
     */

    protected Process makeCgiCommand (Request request)
        throws ProtocolException, IOException
    {
        // Check the command attribute first:
        String      query     = null;
        String      command[] = getCommand() ;
        if ( command == null ) {
            Reply error = request.makeReply(HTTP.INTERNAL_SERVER_ERROR) ;
            error.setContent("CgiResource mis-configured: it doesn't have a "
                             + " command attribute");
            throw new HTTPException(error);
        }
        // Ok:
        Vector      env       = new Vector(32) ;
        httpd       server    = request.getClient().getServer() ;
        InetAddress sadr      = server.getInetAddress() ;
        // Specified environment variables:
        // We do not handle the following variables:
        // - PATH_TRANSLATED: I don't understand it
        // - REMOTE_IDENT: would require usage of IDENT protocol.
        // Authentification type, if any:
        String svalue = (String) request.getState(AuthFilter.STATE_AUTHTYPE);
        if ( svalue != null )
            addEnv("AUTH_TYPE", svalue, env);
        // Content length, if available:
        svalue = request.getValue("content-length");
        if ( svalue != null )
            addEnv("CONTENT_LENGTH", svalue, env);
        // Content type, if available:
```

```
        svalue = request.getValue("content-type");
        if ( svalue != null )
            addEnv("CONTENT_TYPE", svalue, env);
        // The gateway interface, hopefully 1.1 !
        addEnv ("GATEWAY_INTERFACE", "CGI/1.1", env) ;
        // The PATH_INFO, which I am afraid I still don't understand:
        svalue = (String) request.getState(STATE_EXTRA_PATH);
        if ( svalue == null )
            addEnv ("PATH_INFO", "/", env) ;
        else
            addEnv ("PATH_INFO", svalue, env) ;
        // The query string:
        query = request.getQueryString();
        if ( query != null )
            addEnv("QUERY_STRING", query, env) ;
        // The remote client IP address:
        svalue = request.getClient().getInetAddress().toString();
        addEnv ("REMOTE_ADDR", svalue, env);
        // Authentified user:
        svalue = (String) request.getState(AuthFilter.STATE_AUTHUSER);
        if ( svalue != null )
            addEnv("REMOTE_USER", svalue, env);
        // Remote host name, if allowed:
        if ( checkRemoteHost() ) {
            String host = request.getClient().getInetAddress().getHostName();
            addEnv("REMOTE_HOST", host, env);
        }
        // The request method:
        addEnv ("REQUEST_METHOD", request.getMethod(), env) ;
        // The script name :
        addEnv("SCRIPT_NAME", getURLPath(), env);
        // Server name:
        addEnv ("SERVER_NAME", getServer().getHost(), env) ;
        // Server port:
        svalue = Integer.toString(getServer().getLocalPort());
        addEnv ("SERVER_PORT", svalue, env);
        // Server protocol:
        addEnv ("SERVER_PROTOCOL", request.getVersion(), env) ;
        // Server software:
        addEnv ("SERVER_SOFTWARE", server.getSoftware(), env) ;
        // All other request fields, yeah, let's lose even more time:
        Enumeration e = request.enumerateHeaderDescriptions(false);
        while ( e.hasMoreElements() ) {
            HeaderDescription d = (HeaderDescription) e.nextElement();
            addEnv(getEnvName(d.getName())
                   , request.getHeaderValue(d).toString()
                   , env);
        }
        // Command line:
        if ( query != null ) {
            String querycmd[] = new String[command.length+1] ;
```

```
            System.arraycopy(command, 0, querycmd, 0, command.length) ;
            querycmd[command.length] = query ;
            command = querycmd ;
        }
        String aenv[] = new String[env.size()] ;
        env.copyInto (aenv) ;
        // Run the process:
        if ( getInterpreter() != null ) {
            String run[] = new String[command.length+1];
            run[0] = getInterpreter();
            System.arraycopy(command, 0, run, 1, command.length);
            return Runtime.getRuntime().exec (run, aenv) ;
        } else {
            return Runtime.getRuntime().exec (command, aenv) ;
        }
    }

    /**
     * Lookup sub-resources.
     * Accumulate the remaning path in some special state of the request.
     * <p>This allows us to implement the <code>PATH_INFO</code>
     * CGI variable properly.
     * @param ls Current lookup state.
     * @param lr Lookup result under construction.
     * @return A boolean <strong>true</strong> if lookup should continue,
     * <strong>false</strong> otherwise.
     */

    public boolean lookup(LookupState ls, LookupResult lr)
        throws ProtocolException
    {
        // Get the extra path information:
        String extraPath = ls.getRemainingPath(true);
        if ((extraPath == null) || extraPath.equals(""))
            extraPath = "/";
        // Keep this path info into the request, if possible:
        Request request = (Request) ls.getRequest();
        if ( request != null )
            request.setState(STATE_EXTRA_PATH, extraPath);
        lr.setTarget(getResource().getResourceReference());
        return super.lookup(ls, lr);
    }

    /**
     * GET method implementation.
     * this method is splitted into two cases:
     * <p>If the resource is able to generates its form, than run the script
     * to emit the form. Otherwsie, use our super class (FileResource) ability
     * to send the file that contains the form.
     * <p>Note that there is no need to feed the underlying process with
     * data in the GET case.
```

```
    * @param request The request to handle.
    * @exception ProtocolException If processing the request failed.
    */
   public Reply get(Request request)
       throws ProtocolException, NotAProtocolException
  {
      if ( ! checkGeneratesFormFlag() )
          return super.get (request) ;
      Process       process = null ;
      try {
          process = makeCgiCommand (request) ;
      } catch (IOException e) {
          Reply error = request.makeReply(HTTP.NOT_FOUND) ;
          error.setContent("The resource's script wasn't found.") ;
          throw new HTTPException (error) ;
      }
      return handleCGIOutput (process, request) ;
  }


   /**
    * Handle the POST method according to CGI/1.1 specification.
    * The request body is sent back to the launched CGI script, as is, and
    * the script output is handled by the handleCGIOutput method.
    * @param request The request to process.
    * @exception ProtocolException If the processing failed.
    */
   public Reply post(Request request)
       throws ProtocolException, NotAProtocolException
   {
       Process       process = null ;
       // Launch the CGI process:
       try {
           process = makeCgiCommand(request) ;
       } catch (IOException ex) {
           // The process wasn't executable, emit a errlog message:
           String msg = ("The process "+
                         getCommand()[0] +" couldn't be executed ["+
                         ex.getMessage() + "]");
           getServer().errlog(this, msg);
           // Throw an internal server error:
           Reply error = request.makeReply(HTTP.INTERNAL_SERVER_ERROR) ;
           error.setContent("CGI script is misconfigured.");
           throw new HTTPException (error) ;
       }
       // Now feed the process:
       try {
           // Send the 100 status code:
           Client client = request.getClient();
           if ( client != null )
               client.sendContinue();
```

```
            InputStream in = request.getInputStream();
            if ( in == null ) {
                // There was no input to that CCI, close process stream
                process.getOutputStream().close();
            } else {
                // Some input to feed the process with:
                (new ProcessFeeder(process, in)).start();
            }
        } catch (IOException ex) {
            // This is most probably a bad request:
            Reply error = request.makeReply(HTTP.BAD_REQUEST);
            error.setContent("The request didn't have a valid input.");
            throw new HTTPException(error);
        }
        return handleCGIOutput(process, request);
    }

    /**
     * At register time, if no command, use a suitable default.
     * THis method will set the command to the identifier, if it is not
     * provided.
     * @param values Default attribute values.
     */

    public void registerResource(FramedResource resource) {
        super.registerResource(resource);
        //if no command is specified look for a file resource attached
        //and get its File absolute path if available.
        if (getCommand() == null) {
            if (getFileResource() != null) {
                if (getFileResource().getFile() != null) {
                    String cmd[] = new String[1];
                    cmd[0] = getFileResource().getFile().getAbsolutePath();
                    setValue(ATTR_COMMAND, cmd);
                }
            }
        }
    }
}
```

```
// PostableFrame.java
// $Id: PostableFrame.html,v 1.2 1998/07/09 10:48:24 benoit Exp $
// (c) COPYRIGHT MIT and INRIA, 1996.
// Please first read the full copyright statement in file COPYRIGHT.html

package org.w3c.jigsaw.frames ;

import java.io.* ;
import java.util.*;

import org.w3c.tools.resources.*;
import org.w3c.jigsaw.forms.*;
import org.w3c.www.mime.* ;
import org.w3c.www.http.* ;
import org.w3c.jigsaw.http.* ;
import org.w3c.jigsaw.html.HtmlGenerator ;

/**
 * Handle POST.
 */
public class PostableFrame extends HTTPFrame {
    private static HttpTokenList _post_allowed = null;
    private static HttpTokenList _put_allowed = null;
    static {
        String post_allowed[] = { "GET", "HEAD", "OPTIONS", "POST", "TRACE" } ;
        _post_allowed = HttpFactory.makeStringList(post_allowed);
        String put_allowed[] = { "GET", "HEAD", "OPTIONS", "PUT",
                                 "POST", "TRACE" } ;
        _put_allowed = HttpFactory.makeStringList(put_allowed);
    }

    private static MimeType type = MimeType.APPLICATION_X_WWW_FORM_URLENCODED ;
    /**
     * Attribute index - Should we override form values when multiple ?
     */
    protected static int ATTR_OVERIDE = -1 ;
    /**
     * Attribute index - Should we silently convert GET to POST methods ?
     */
    protected static int ATTR_CONVERT_GET = -1 ;

    static {
        Attribute a   = null ;
        Class     cls = null ;
        try {
            cls = Class.forName("org.w3c.jigsaw.frames.PostableFrame") ;
        } catch (Exception ex) {
            ex.printStackTrace() ;
            System.exit(1) ;
        }
```

```
        // The override attribute:
        a = new BooleanAttribute("override",
                                 Boolean.FALSE,
                                 Attribute.EDITABLE);
        ATTR_OVERIDE = AttributeRegistry.registerAttribute(cls, a) ;
        // The convert get attribute:
        a = new BooleanAttribute("convert-get",
                                 Boolean.TRUE,
                                 Attribute.EDITABLE) ;
        ATTR_CONVERT_GET = AttributeRegistry.registerAttribute(cls, a) ;
    }

    /**
     * Get the 'convert GET to POST' flag.
     */

    public boolean getConvertGetFlag() {
        return getBoolean(ATTR_CONVERT_GET, false) ;
    }

    /**
     * Get the 'override multiple form field value' flag.
     */

    public boolean getOverrideFlag() {
        return getBoolean(ATTR_OVERIDE, true) ;
    }


    /**
     * Catch setValue, to maintain cached header values correctness.
     * @param idx The index of the attribute to be set.
     * @param value The new value for the attribute.
     */

    public synchronized void setValue(int idx, Object value) {
        super.setValue(idx, value);
        if (idx == ATTR_PUTABLE) {
            if (value == Boolean.TRUE)
                allowed = _put_allowed;
            else
                allowed = _post_allowed;
        }
    }


    /**
     * Get this resource body.
     * If we are allowed to convert GET requests to POST, than we first
     * check to see if there is some search string in the request, and continue
     * with normal POST request processing.
```

```
     * <p>If there is no search string, or if we are not allowed to convert
     * GETs to POSTs, than we just invoke our <code>super</code> method,
     * which will perform the appropriate job.
     * @param request The request to handle.
     * @exception ProtocolException If request couldn't be processed.
     */
    public Reply get (Request request)
        throws ProtocolException, NotAProtocolException
    {
        // Check if we should handle it (is it a POST disguised in GET ?)
        if ((! getConvertGetFlag()) || ( ! request.hasState("query")))
            return super.get (request) ;
        // Get the request entity, and decode it:
        String      query = request.getQueryString() ;
        InputStream in    = new StringBufferInputStream(query) ;
        URLDecoder  d     = new URLDecoder (in, getOverrideFlag()) ;
        try {
            d.parse () ;
        } catch (URLDecoderException e) {
            Reply error = request.makeReply(HTTP.BAD_REQUEST) ;
            error.setContent("Invalid request:unable to decode form data.");
            throw new HTTPException (error) ;
        } catch (IOException e) {
            Reply error = request.makeReply(HTTP.BAD_REQUEST) ;
            error.setContent("Invalid request: unable to read form data.");
            throw new HTTPException (error) ;
        }
        return handle (request, d) ;
    }

    public Reply post (Request request)
        throws ProtocolException, NotAProtocolException
    {
        // Check that we are dealing with an application/x-www-form-urlencoded:
        if ((! request.hasContentType())
            || (type.match(request.getContentType()) < 0) ) {
            Reply error = request.makeReply(HTTP.UNSUPPORTED_MEDIA_TYPE) ;
            error.setContent("Invalid request content type.");
            throw new HTTPException (error) ;
        }
        // Get and decode the request entity:
        URLDecoder dec = null;
        try {
            InputStream in = request.getInputStream() ;
            // Notify the client that we are willing to continue processing:
            Client client = request.getClient();
            if ( client != null )
                client.sendContinue();
            dec = new URLDecoder (in, getOverrideFlag()) ;
            dec.parse () ;
        } catch (URLDecoderException e) {
```

```
            Reply error = request.makeReply(HTTP.BAD_REQUEST) ;
            error.setContent("Invalid request: unable to decode form data.") ;
            throw new HTTPException (error) ;
        } catch (IOException ex) {
            Reply error = request.makeReply(HTTP.BAD_REQUEST) ;
            error.setContent("Invalid request: unable to read form data.") ;
            throw new ClientException(request.getClient(), ex) ;
        }
        // Handle the stuff:
        return handle (request, dec) ;
    }

    /**
     * Handle the form submission, after posted data parsing.
     * <p>This method ought to be abstract, but for reasonable reason, it
     * will just dump (parsed) the form content back to the client, so that it
     * can be used for debugging.
     * @param request The request proper.
     * @param data The parsed data content.
     * @exception ProtocolException If form data processing failed.
     * @see org.w3c.jigsaw.forms.URLDecoder
     */

    public Reply handle (Request request, URLDecoder data)
        throws ProtocolException
    {
        // Now we just dump back the variables we got:
        Enumeration   e = data.keys() ;
        HtmlGenerator g = new HtmlGenerator ("Form decoded values") ;
        g.append ("<p>List of variables and values:</p><ul>") ;
        while ( e.hasMoreElements () ) {
            String name = (String) e.nextElement() ;
            g.append ("<li><em>"+
                      name+"</em> = <b>"+
                      data.getValue(name)+
                      "</b></li>");
        }
        g.append ("</ul>") ;
        Reply reply = request.makeReply(HTTP.OK) ;
        reply.setStream (g) ;
        return reply ;
    }
}
```

```
// RedirecterFrame.java
// $Id: RedirecterFrame.html,v 1.1 1998/03/30 08:31:37 benoit Exp $
// (c) COPYRIGHT MIT and INRIA, 1996.
// Please first read the full copyright statement in file COPYRIGHT.html

package org.w3c.jigsaw.frames;

import java.util.*;
import java.io.* ;
import java.net.*;

import org.w3c.tools.resources.*;
import org.w3c.jigsaw.http.*;
import org.w3c.www.http.*;

import org.w3c.tools.resources.ProtocolException;
import org.w3c.tools.resources.NotAProtocolException;

/**
 * Perform an internal redirect.
 */
public class RedirecterFrame extends HTTPFrame {
    /**
     * Attributes index - The index for the target attribute.
     */
    protected static int ATTR_TARGET = -1 ;

    static {
        Attribute a   = null ;
        Class     cls = null ;
        // Get a pointer to our class:
        try {
            cls = Class.forName("org.w3c.jigsaw.frames.RedirecterFrame") ;
        } catch (Exception ex) {
            ex.printStackTrace() ;
            System.exit(1) ;
        }
        a = new StringAttribute("target"
                                , null
                                , Attribute.EDITABLE);
        ATTR_TARGET = AttributeRegistry.registerAttribute(cls, a) ;
    }

    protected String getTarget() {
        return (String) getValue(ATTR_TARGET, null);
    }
```

```
    public ReplyInterface perform(RequestInterface req)
        throws ProtocolException, NotAProtocolException
    {
        Reply        reply  = (Reply) performFrames(req);
        if (reply != null)
            return reply;
        Request request = (Request) req;
        httpd        server = (httpd) getServer();
        request.setReferer(getURLPath());
        try {
            request.setURL( new URL(server.getURL(), getTarget()));
        } catch (MalformedURLException ex) {
            Reply error = request.makeReply(HTTP.INTERNAL_SERVER_ERROR);
            error.setContent("<html><head><title>Server Error</title>"+
                             "</head><body><h1>Server misconfigured</h1>"+
                             "<p>The resource <b>"+getIdentifier()+"</b>"+
                             "has an invalid target attribute : <p><b>"+
                             getTarget()+"</b></body></html>");
            throw new HTTPException (error);
        }
        return server.perform(request);
    }
}
```

```
// VirtualHostFrame.java
// $Id: VirtualHostFrame.html,v 1.2 1998/07/09 10:48:53 benoit Exp $
// (c) COPYRIGHT MIT and INRIA, 1998.
// Please first read the full copyright statement in file COPYRIGHT.html

package org.w3c.jigsaw.frames;

import org.w3c.tools.resources.*;
import org.w3c.jigsaw.http.* ;
import org.w3c.www.http.* ;

/**
 * For Virtual Hosting.
 */
public class VirtualHostFrame extends HTTPFrame {

    /**
     * Attribute index - The default root (for unknown hosts)
     */
    protected static int ATTR_FOLLOWUP = -1;

    static {
        Class     c = null;
        Attribute a = null;

        try {
            c = Class.forName("org.w3c.jigsaw.frames.VirtualHostFrame");
        } catch (Exception ex) {
            ex.printStackTrace();
            System.exit(1);
        }
        // Register our default root:
        a = new StringAttribute("followup"
                                , null
                                , Attribute.EDITABLE);
        ATTR_FOLLOWUP = AttributeRegistry.registerAttribute(c, a);
    }

    protected ResourceReference followup = null;

    /**
     * Get the name of the resource used as a followup.
     * @return A String giving the name of the resource to be used as the
     * default.
     */

    public String getFollowup() {
```

```
        return getString(ATTR_FOLLOWUP, null);
    }

    public void registerResource(FramedResource resource) {
        super.registerOtherResource(resource);
    }

    /**
     * Lookup the followup resource.
     * @return The loaded resource for the current followup.
     */

    public synchronized ResourceReference lookupFollowup() {
        if ( followup == null ) {
            String name  = getFollowup();
            if ( name != null )
                followup = getServer().loadRoot(name);
            if ( followup == null ) {
                getServer().errlog(getIdentifier()
                                   + "[" + getClass().getName() + "]: "
                                   + "unable to restore \"" + name + "\" "
                                   + " from root store.");
            }
        }
        return followup;
    }

    protected boolean lookupOther(LookupState ls, LookupResult lr)
        throws ProtocolException
    {
        // Try to lookup on the host header:
        ResourceReference vrroot = null;
        ContainerResource root = null;

        root = (ContainerResource)getResource();
        Request r = (Request)ls.getRequest();
        if ( r != null ) {
            String  host = r.getHost();
            if ( host != null ) {
                vrroot = root.lookup(host.toLowerCase());
            }
        }
        if ( vrroot == null )
            vrroot  = lookupFollowup();
        // Check for what we got:
        if (vrroot == null)
            return super.lookupOther(ls, lr);
        try {
```

```
            lr.setTarget(vrroot);
            FramedResource resource = (FramedResource) vrroot.lock();
            boolean done =
              (resource != null ) ? resource.lookup(ls, lr) : false;
            if (! done)
              lr.setTarget(null);
              // because the vroot eats the lookup state components
              // we have to return true.
              // Should not be continued by the caller.
            return true;
        } catch (InvalidResourceException ex) {
            return false;
        } finally {
            vrroot.unlock();
        }
    }
}
```

```
// CounterFilter.java
// $Id: CounterFilter.html,v 1.1 1998/03/30 08:31:32 benoit Exp $
// (c) COPYRIGHT MIT and INRIA, 1996.
// Please first read the full copyright statement in file COPYRIGHT.html

package org.w3c.jigsaw.filters;

import org.w3c.tools.resources.*;
import org.w3c.jigsaw.http.*;
import org.w3c.jigsaw.resources.*;

/**
 * Count the number of hits to the target.
 * This resource maintains the number of hits to some target resource, as
 * one of its persistent attribute.
 * It will decorate the request on the way in with a fake field
 * <code>org.w3c.jigsaw.filters.CounterFilter.count</code>, that will
 * hold the current hit counts for the target resource to use.
 */

public class CounterFilter extends ResourceFilter {
    /**
     * The name of the piece if state that receives the hit count value.
     * To get to the hit-count, use the <code>getState</code> method of
     * Request, with the following key.
     */
    public static final
        String STATE_COUNT = "org.w3c.jigsaw.filters.CounterFilter.count";

    /**
     * Attribute index - The counter attribute.
     */
    protected static int ATTR_COUNTER = -1 ;

    static {
        Attribute a   = null ;
        Class     cls = null ;

        try {
            cls = Class.forName("org.w3c.jigsaw.filters.CounterFilter") ;
        } catch (Exception ex) {
            ex.printStackTrace() ;
            System.exit(1) ;
        }
        // Declare the counter attribute
        a = new IntegerAttribute("counter"
                                 , new Integer(0)
                                 , Attribute.EDITABLE) ;
```

```
        ATTR_COUNTER = AttributeRegistry.registerAttribute(cls, a) ;
    }

    /**
     * We count all accesses, even the one that failed.
     * We also define the
     * <code>org.w3c.jigsaw.filters.CounterFilter.count</code>
     * request state as the number of hits on that resource (stored as
     * an Integer instance).
     * @param request The request being processed.
     * @return Always <strong>null</strong>.
     */

    public synchronized ReplyInterface ingoingFilter(RequestInterface req) {
        Request request = (Request) req;
        int i = getInt (ATTR_COUNTER, 0) + 1;
        setInt(ATTR_COUNTER, i) ;
        if(! request.hasState(STATE_COUNT))
            request.setState(STATE_COUNT, new Integer(i)) ;
        return null;
    }

}
```

```
// ErrorFilter.java
// $Id: ErrorFilter.html,v 1.1 1998/03/30 08:31:52 benoit Exp $
// (c) COPYRIGHT MIT and INRIA, 1996.
// Please first read the full copyright statement in file COPYRIGHT.html


package org.w3c.jigsaw.filters;

import org.w3c.tools.resources.*;
import org.w3c.jigsaw.http.*;
import org.w3c.jigsaw.resources.*;
import org.w3c.www.http.*;

/**
 * The ErroFilter class allows you to customize and enhance error messages.
 * This filter will catch all errors on their way back to the client, and
 * use internal requests to provide a nice customizable error message.
 * <p>You can use any resources (including server side includes, content
 * negotiated resources, etc) to deliver error messages.
 */

public class ErrorFilter extends ResourceFilter {
    /**
     * A request state, to avoid looping on errors about errors.
     */
    protected static final String ERRED = "org.w3c.jigsaw.filters.ErrorFilter";

    /**
     * Attribute index - The base URL for error messages.
     */
    protected static int ATTR_BASEURL = -1;
    /**
     * Attribute index - The common extension for error messages (can be null).
     */
    protected static int ATTR_EXTENSION = -1;

    static {
        Attribute a   = null;
        Class     cls = null;

        try {
            cls = Class.forName("org.w3c.jigsaw.filters.ErrorFilter");
        } catch (Exception ex) {
            ex.printStackTrace();
            System.exit(1);
        }
        // Declare the error base URL attribute:
        a = new StringAttribute("base-url"
                                , "/errors"
                                , Attribute.EDITABLE);
```

```
        ATTR_BASEURL = AttributeRegistry.registerAttribute(cls, a);
        // Declare the extension attribute:
        a = new StringAttribute("extension"
                                , "html"
                                , Attribute.EDITABLE);
        ATTR_EXTENSION = AttributeRegistry.registerAttribute(cls, a);
    }

    /**
     * Get the base URL describing the error directory.
     * @return The base URL.
     */

    public String getBaseURL() {
        return (String) getValue(ATTR_BASEURL, null);
    }

    /**
     * Get the value of the extension attribute.
     * @return A String, for the common extension to error messages, or
     * <strong>null</strong> if undefined.
     */

    public String getExtension() {
        return (String) getValue(ATTR_EXTENSION, null);
    }

    /**
     * Compute the path for the given status code.
     * @return A path leading to the customizable error message for the
     * given status code.
     */

    public String getErrorResource(int status) {
        String ext = getExtension() ;
        if ( ext != null ) {
            return getBaseURL()+"/"+Integer.toString(status)+"."+ext;
        } else {
            return getBaseURL()+"/"+Integer.toString(status);
        }
    }

    /**
     * This one just makes sure the outgoing filter gets called.
     * @param request The original request to be handled.
     */

    public ReplyInterface ingoingFilter(RequestInterface request)
        throws ProtocolException
    {
        return null;
```

```
    }

    /**
     * Re-compute error message.
     * This filter uses internal redirection to get the error message body.
     * In case of failure, the original reply is returned, otherwise, a new
     * reply is gotten from the appropriate error resource, and is returned.
     * @param request The request that has been handled.
     * @param reply The reply, as emited by the original resource.
     * @return A new error reply, having the same status code, and
     * authentication information then the given reply, but enhanced
     * with the error resource body.
     */

    public ReplyInterface outgoingFilter(RequestInterface req,
                                         ReplyInterface rep)
        throws ProtocolException
    {
        Request request = (Request) req;
        Reply   reply   = (Reply) rep;
        // Filter valid replies:
        int status = reply.getStatus();
        switch (status/100) {
        case 1:
        case 2:
        case 3:
        case 10:
            return null;
        }
        // Filter replies that are already taken care of:
        if ( request.hasState(ERRED) )
            return null;
        // Hack error replies:
        Request ereq  = (Request) request.getClone();
        Reply   erep  = null;
        try {
            ereq.setState(ERRED, Boolean.TRUE);
            ereq.setURLPath(getErrorResource(status));
            ereq.setMethod("GET");
            erep = (Reply) getServer().perform(ereq);
            // Hack back the original reply into the new reply:
            // - Put back the status
            HeaderValue v = null;
            erep.setStatus(reply.getStatus());
            // - Put back the authenticate informations
            v = reply.getHeaderValue(reply.H_WWW_AUTHENTICATE);
            erep.setHeaderValue(reply.H_WWW_AUTHENTICATE, v);
            // - Put back the proxy authenticate informations
            v = reply.getHeaderValue(reply.H_PROXY_AUTHENTICATE);
            erep.setHeaderValue(reply.H_PROXY_AUTHENTICATE, v);
        } catch (Exception ex) {
```

```
            return reply;
        }
        return erep;
    }

    /**
     * We do catch exceptions, just in case we can customize the error.
     * @param request The request tha triggered the exception.
     * @param ex The exception.
     * @param filters Remaining filters to be called.
     * @param idx Current filter index within above array.
     */

    public ReplyInterface exceptionFilter(RequestInterface request,
                                          ProtocolException ex,
                                          FilterInterface filters[],
                                          int idx)
    {
        Reply reply = (Reply) ex.getReply();
        if ( reply != null ) {
            try {
                return outgoingFilter(request, reply, filters, idx);
            } catch (ProtocolException exx) {
            }
        }
        return null;
    }

    public void initialize(Object values[]) {
        super.initialize(values);
    }

}
```

```
// GZIPFilter.java
// $Id: GZIPFilter.html,v 1.1 1998/03/30 08:30:27 benoit Exp $
// (c) COPYRIGHT MIT and INRIA, 1996.
// Please first read the full copyright statement in file COPYRIGHT.html

package org.w3c.jigsaw.filters;

import java.io.*;
import java.util.zip.*;

import org.w3c.tools.resources.*;
import org.w3c.www.mime.*;
import org.w3c.jigsaw.http.*;
import org.w3c.jigsaw.resources.*;

class GZIPDataMover extends Thread {
    InputStream in = null;
    OutputStream out = null;

    public void run() {
        try {
            byte buf[] = new byte[1024];
            int  got   = -1;
            while ((got = in.read(buf)) > 0)
                out.write(buf, 0, got);
        } catch (IOException ex) {
            ex.printStackTrace();
        } finally {
            try { in.close(); } catch (Exception ex) {};
            try { out.close(); } catch (Exception ex) {} ;
        }
    }

    GZIPDataMover(InputStream in, OutputStream out) {
        this.in  = in;
        this.out = out;
        setName("GZIPDataMover");
        start();
    }
}

/**
 * This filter will compress the content of replies using GZIP.
 * Compression is done <em>on the fly</em>. This assumes that you're really
 * on a slow link, where you have lots of CPU, but not much bandwidth.
 * <p>A nifty usage for that filter, is to plug it on top of a
 * <code>org.w3c.jigsaw.proxy.ProxyDirectory</code>, in which case it
```

```
 * will compress the data when it flies out of the proxy.
 */

public class GZIPFilter extends ResourceFilter {
    /**
     * Attribute index - List of MIME type that we can compress
     */
    protected static int ATTR_MIME_TYPES = -1;

    static {
        Class     c = null;
        Attribute a = null;
        try {
            c = Class.forName("org.w3c.jigsaw.filters.GZIPFilter");
        } catch (Exception ex) {
            ex.printStackTrace();
            System.exit(1);
        }
        // Register the MIME types attribute:
        a = new StringArrayAttribute("mime-types"
                                     , null
                                     , Attribute.EDITABLE);
        ATTR_MIME_TYPES = AttributeRegistry.registerAttribute(c, a);
    }

    /**
     * The set of MIME types we are allowed to compress.
     */
    protected MimeType types[] = null;

    /**
     * Catch the setting of mime types to compress.
     * @param idx The attribute being set.
     * @param val The new attribute value.
     */

    public void setValue(int idx, Object value) {
        super.setValue(idx, value);
        if ( idx == ATTR_MIME_TYPES ) {
            synchronized (this) {
                types = null;
            }
        }
    }


    /**
     * Get the set of MIME types to match:
```

```
     * @return An array of MimeType instances.
     */

    public synchronized MimeType[] getMimeTypes() {
        if ( types == null ) {
            String strtypes[] = (String[]) getValue(ATTR_MIME_TYPES, null);
            if ( strtypes == null )
                return null;
            types = new MimeType[strtypes.length];
            for (int i = 0 ; i < types.length ; i++) {
                try {
                    types[i] = new MimeType(strtypes[i]);
                } catch (Exception ex) {
                    types[i] = null;
                }
            }
        }
        return types;
    }

    public ReplyInterface outgoingFilter(RequestInterface req,
                                         ReplyInterface rep)
        throws ProtocolException
    {
      Request request = (Request) req;
      Reply   reply   = (Reply) rep;
        // Anything to compress ?
        if ( ! reply.hasStream() )
            return null;
        // Match possible mime types:
        MimeType t[]     = getMimeTypes();
        boolean  matched = false;
        if ( t != null ) {
            for (int i = 0 ; i < t.length ; i++) {
                if ( t[i] == null )
                    continue;
                if ( t[i].match(reply.getContentType()) > 0 ) {
                    matched = true;
                    break;
                }
            }
        }
        if ( ! matched )
            return null;
        // Compress:
        try {
            PipedOutputStream pout = new PipedOutputStream();
            PipedInputStream  pin  = new PipedInputStream(pout);
```

```
            new GZIPDataMover(reply.openStream()
                              , new GZIPOutputStream(pout));
            reply.addContentEncoding("gzip");
            reply.setContentLength(-1);
            reply.setStream(pin);
        } catch (Exception ex) {
            ex.printStackTrace();
        }
        return null;
    }


}
```

```
// HeaderFilter.java
// $Id: HeaderFilter.html,v 1.1 1998/03/30 08:32:06 benoit Exp $
// (c) COPYRIGHT MIT and INRIA, 1996.
// Please first read the full copyright statement in file COPYRIGHT.html

package org.w3c.jigsaw.filters;

import java.net.*;

import org.w3c.tools.resources.*;
import org.w3c.tools.resources.ProtocolException;
import org.w3c.www.http.*;
import org.w3c.jigsaw.http.*;

/**
 * Enforces a specific header value on all replies.
 * Usefull for testing.
 */

public class HeaderFilter extends ResourceFilter {
    /**
     * Attribute index - The header name to add to replies.
     */
    protected static int ATTR_HEADER_NAME = -1;
    /**
     * Attribute index - The header value.
     */
    protected static int ATTR_HEADER_VALUE = -1;
    /**
     * Attribute index - SHould we use no-cache on that header.
     */
    protected static int ATTR_NOCACHE = -1;
    /**
     * Attribute index - Should we use connection on that header.
     */
    protected static int ATTR_CONNECTION = -1;

    static {
        Class c = null;
        Attribute a = null;
        try {
            c = Class.forName("org.w3c.jigsaw.filters.HeaderFilter");
        } catch (Exception ex) {
            ex.printStackTrace();
            System.exit(1);
```

```
        }
        // Register the header name attribute:
        a = new StringAttribute("header-name"
                                , null
                                , Attribute.EDITABLE);
        ATTR_HEADER_NAME = AttributeRegistry.registerAttribute(c, a);
        // Register the header value attribute.
        a = new StringAttribute("header-value"
                                , null
                                , Attribute.EDITABLE);
        ATTR_HEADER_VALUE = AttributeRegistry.registerAttribute(c, a);
        // Register the nocache attribute.
        a = new BooleanAttribute("no-cache"
                                 , Boolean.FALSE
                                 , Attribute.EDITABLE);
        ATTR_NOCACHE = AttributeRegistry.registerAttribute(c, a);
        // Register the connection attribute.
        a = new BooleanAttribute("connection"
                                 , Boolean.FALSE
                                 , Attribute.EDITABLE);
        ATTR_CONNECTION = AttributeRegistry.registerAttribute(c, a);
    }


    /**
     * Get the header to set, if any.
     * @return A String encoded header name, or <strong>null</strong>.
     */

    public String getHeaderName() {
        String value = getString(ATTR_HEADER_NAME, null);
        if ( value != null )
            return value.toLowerCase();
        return null;
    }

    /**
     * Get the header value to set, if any.
     * @return A String encoded value for the header to set, or <strong>
     * null</strong>.
     */

    public String getHeaderValue() {
        return getString(ATTR_HEADER_VALUE, null);
    }
```

```
    /**
     * Should we add this header's name to the <code>no-cache</code>
     * directive.
     * @return A boolean.
     */

    public boolean checkNoCache() {
        return getBoolean(ATTR_NOCACHE, false);
    }

    /**
     * Should we add this header to the connection header.
     * @return A boolean.
     */

    public boolean checkConnection() {
        return getBoolean(ATTR_CONNECTION, false);
    }

    public ReplyInterface ingoingFilter(RequestInterface request)
        throws ProtocolException
    {
        return null;
    }

    /**
     * The outgoing filter decorates the reply appropriately.
     * @param request The original request.
     * @param reply The originial reply.
     * @return Always <strong>null</strong>.
     */

    public ReplyInterface outgoingFilter(RequestInterface req,
                                         ReplyInterface rep)
        throws ProtocolException
    {
        Reply reply = (Reply) rep;
        String hname = getHeaderName();
        if ( hname != null ) {
            String hvalue = getHeaderValue();
            if ( hvalue == null ) {
                reply.removeHeader(hname);
            } else {
                reply.setValue(hname, hvalue);
            }
            if ( checkNoCache() )
```

```
                reply.addNoCache(hname);
            if ( checkConnection() )
                reply.addConnection(hname);
        }
        return null;
    }
}
```

```
// LogFilter.java
// $Id: LogFilter.html,v 1.1 1998/03/30 08:31:59 benoit Exp $
// (c) COPYRIGHT MIT and INRIA, 1996.
// Please first read the full copyright statement in file COPYRIGHT.html

package org.w3c.jigsaw.filters;

import java.io.*;
import java.net.*;
import java.util.*;

import org.w3c.tools.resources.*;
import org.w3c.tools.resources.ProtocolException;
import org.w3c.www.http.*;
import org.w3c.jigsaw.http.*;

/**
 * This filter provides a very flexible logger.
 * It is not designed as a logger, in order to be plugable only on
 * a sub-tree of the URL space (a logger would log all site accesses).
 * It provides as much details as you want, and uses a very simple format:
 * each log entry (or <em>record</em> is made of several lines having
 * the folowing format:
 * <pre>variable=value</pre>
 * A record starts with the special <code>url</code> variable value which
 * provides the requested URL. The for each header that is to be logged,
 * a variable is added in the record, prefixed by its <em>scope</em>. The
 * scope can be either:
 * <dl><dt>request<dd> to specify a request header,
 * <dt>reply<dd> to specify a reply header,
 * <dt>server<dd> to specify global server samples.
 * </dl>
 * As an example, if you configure that filter to log the request's referer
 * and the reply content length, a sample record will look like:
 * <pre>
 * url=http://www.w3.org/pub/WWW/Jigsaw/
 * request.referer=http://www.w3.org/pub/WWW
 * reply.content-length=10
 * </pre>
 */

public class LogFilter extends ResourceFilter {
    /**
     * Name of the state that when set on the request will prevent logging.
     */
    public static final
        String DONT_LOG = "org.w3c.jigsaw.filters.LogFilter.nolog";

    /**
     * Attribute index - The HTTP request headers to dump
```

```
     */
    protected static int ATTR_REQUEST_HEADERS = -1;
    /**
     * Attribute index - The HTTP reply headers to dump
     */
    protected static int ATTR_REPLY_HEADERS = -1;
    /**
     * Attribute index - The log file to use to emit log record.
     */
    protected static int ATTR_LOGFILE = -1;

    static {
        Class     c = null;
        Attribute a = null;

        try {
            c = Class.forName("org.w3c.jigsaw.filters.LogFilter");
        } catch (Exception ex) {
            ex.printStackTrace();
            System.exit(1);
        }
        // Register the request headers to be logged:
        a = new StringArrayAttribute("request-headers"
                                     , null
                                     , Attribute.EDITABLE);
        ATTR_REQUEST_HEADERS = AttributeRegistry.registerAttribute(c, a);
        // Register the reply headers to be logged:
        a = new StringArrayAttribute("reply-headers"
                                     , null
                                     , Attribute.EDITABLE);
        ATTR_REPLY_HEADERS = AttributeRegistry.registerAttribute(c, a);
        // Register the log file name:
        a = new FileAttribute("logfile"
                              , null
                              , Attribute.EDITABLE);
        ATTR_LOGFILE = AttributeRegistry.registerAttribute(c, a);
    }

    /**
     * Compiled index of the request headers to dump.
     */
    protected HeaderDescription reqheaders[] = null;
    /**
     * Compiled index of the reply headers to dump.
     */
    protected HeaderDescription repheaders[] = null;
    /**
     * Open log descriptor, to write to the log.
     */
    protected RandomAccessFile log = null;
```

```
    /**
     * Compile the given set of header names into header indexes.
     * @param kind An instance of the class whose headers are to be dumped.
     * @param headers The name of headers to compile.
     * @return An array of header description, which will allow fast fetch
     * of header values.
     */

    protected HeaderDescription[] compileHeaders(HttpMessage kind
                                                 , String headers[]) {
        Enumeration       e   = kind.enumerateHeaderDescriptions(true);
        HeaderDescription r[] = new HeaderDescription[headers.length];
        while ( e.hasMoreElements() ) {
            HeaderDescription d = (HeaderDescription) e.nextElement();
            if ( d == null )
                continue;
            for (int i = 0 ; i < headers.length ; i++) {
                if (headers[i].equals(d.getName())) {
                    r[i] = d;
                    break;
                }
            }
        }
        return r;
    }

    /**
     * Write the given string to the log file.
     * @param record The string to write.
     */

    protected synchronized void writelog(String record) {
        try {
            if ( log != null )
                log.writeBytes(record);
        } catch (IOException ex) {
            FramedResource target = (FramedResource) getTargetResource();
            if (target != null) {
                String msg = ("IO error while writing to log file \""+
                              ex.getMessage() + "\".");
                target.getServer().errlog(this, msg);
            }
        }
    }

    /**
     * Open the log stream, and make it available through <code>log</code>.
     * If opening the stream failed, an appropriate error message is emitted
     * and <code>log</code> remains set to <strong>null</strong>. If a log
     * stream was already opened, it is first closed.
     */
```

```
    protected synchronized void openLog() {
        // Close first if needed:
        if ( log != null ) {
            try {
                log.close();
            } catch (IOException ex) {
            }
            log = null;
        }
        // Try opening the log:
        File logfile = getLogfile();
        try {
            if (logfile != null) {
                log = new RandomAccessFile(logfile, "rw");
                log.seek(log.length());
            }
        } catch (Exception ex) {
        }
        // If failed, emit error message:
        if ( log == null ) {
            FramedResource target = (FramedResource) getTargetResource();
            if (target != null) {
                String msg = ("unable to open log file \""+
                              logfile
                              + "\".");
                target.getServer().errlog(this, msg);
            }
        }
    }

    /**
     * Get the log file.
     * @return A File instance, or <strong>null</strong> if not set.
     */

    public File getLogfile() {
        return (File) getValue(ATTR_LOGFILE, null);
    }

    /**
     * Get the list of request headers to dump.
     * @return An array of String containing the name of headers to dump, or
     * <strong>null</strong> if undefined.
     */

    public String[] getRequestHeaders() {
        return (String[]) getValue(ATTR_REQUEST_HEADERS, null);
    }

    /**
```

```
     * Get the list of reply headers to dump.
     * @return An array of String containing the name of headers to dump,
     * or <strong>null</strong> if undefined.
     */

    public String[] getReplyHeaders() {
        return (String[]) getValue(ATTR_REPLY_HEADERS, null);
    }

    /**
     * Traop setValue calls.
     * We maintain a compiled version of both the <code>request-headers</code>
     * and the <code>reply-headers</code> attributes, make sure they stay in
     * sync even when modified.
     */

    public void setValue(int idx, Object value) {
        super.setValue(idx, value);
        if ( idx == ATTR_REQUEST_HEADERS ) {
            synchronized(this) {
                reqheaders = compileHeaders(new HttpRequestMessage()
                                            , (String[]) value);
            }
        } else if (idx == ATTR_REPLY_HEADERS ) {
            synchronized(this) {
                repheaders = compileHeaders(new HttpReplyMessage()
                                            , (String[]) value);
            }
        } else if (idx == ATTR_LOGFILE ) {
            openLog();
        }
    }

    /**
     * Log the given request/reply transaction.
     * Dump a record for the given transaction.
     * @param request The request to log.
     * @param reply It's associated reply.
     */

    protected void log(Request request, Reply reply) {
        StringBuffer sb = new StringBuffer();
        // Start a new log record:
        sb.append("url=");
        sb.append(request.getURL().toString());
        sb.append('\n');
        // Then log all request headers:
        if ( reqheaders != null ) {
            for (int i = 0 ; i < reqheaders.length ; i++) {
                HeaderDescription d = reqheaders[i];
                if ( d == null )
```

```
                continue;
            HeaderValue v = request.getHeaderValue(d);
            if ( v != null ) {
                sb.append("request.");
                sb.append(d.getName());
                sb.append('=');
                sb.append(v.toString());
                sb.append('\n');
            }
        }
    }
    // The log all reply headers:
    if ( repheaders != null ) {
        for (int i = 0 ; i < repheaders.length ; i++) {
            HeaderDescription d = repheaders[i];
            if ( d == null )
                continue;
            HeaderValue v = reply.getHeaderValue(d);
            if ( v != null ) {
                sb.append("reply.");
                sb.append(d.getName());
                sb.append('=');
                sb.append(v.toString());
                sb.append('\n');
            }
        }
    }
    writelog(sb.toString());
}

/**
 * Log the request.
 * @param request The request that has been handled.
 * @param reply It's associated reply.
 */

public ReplyInterface outgoingFilter(RequestInterface req,
                                     ReplyInterface rep)
    throws ProtocolException
{
    Reply   reply   = (Reply) rep;
    Request request = (Request) req;
    if ( ! reply.hasState(DONT_LOG) )
        log(request, reply);
    return null;
}

/**
 * Initialize the filter.
 */
```

```
    public void initialize(Object values[]) {
        super.initialize(values);
        // Open the log file, ready for working:
        if ( log == null )
            openLog();
        // Compile request and reply headers to dump:
        if ( reqheaders == null ) {
            String headers[] = getRequestHeaders();
            if ( headers != null )
                reqheaders = compileHeaders(new HttpRequestMessage(), headers);
        }
        if ( repheaders == null ) {
            String headers[] = getReplyHeaders();
            if ( headers != null )
                repheaders = compileHeaders(new HttpReplyMessage(), headers);
        }
    }

}
```

```
// ProcessFilter.java
// $Id: ProcessFilter.html,v 1.1 1998/03/30 08:30:56 benoit Exp $
// (c) COPYRIGHT MIT and INRIA, 1996.
// Please first read the full copyright statement in file COPYRIGHT.html

package org.w3c.jigsaw.filters ;

import java.io.* ;
import java.util.*;

import org.w3c.tools.resources.*;
import org.w3c.www.http.*;
import org.w3c.jigsaw.http.* ;
import org.w3c.jigsaw.resources.*;

class ProcessFeeder extends Thread {
    Process      proc    = null ;
    OutputStream out     = null ;
    InputStream  in      = null ;
    int          count = -1 ;

    public void run () {
        try {
            byte buffer[] = new byte[4096] ;
            int  got      = -1 ;
            int emitted   = 0;

            // Send the data to the target process:
            if ( count >= 0 ) {
                while ( (count > 0) && ((got = in.read(buffer)) > 0) ) {
                    out.write (buffer, 0, got) ;
                    count   -= got ;
                    emitted += got;
                }
            } else {
                while ( (got = in.read(buffer)) > 0 ) {
                    out.write (buffer, 0, got) ;
                    emitted += got;
                }
            }
            // Clean up the process:
            out.flush() ;
            out.close() ;
            proc.waitFor() ;
        } catch (Exception e) {
            System.out.println ("ProcessFeeder: caught exception !") ;
            e.printStackTrace() ;
        }
    }
```

```
    ProcessFeeder (Process proc, InputStream in) {
        this (proc, in, -1) ;
    }

    ProcessFeeder (Process proc, InputStream in, int count) {
        this.proc   = proc ;
        this.out    = proc.getOutputStream() ;
        this.in     = in ;
        this.count  = count ;
    }
}

/**
 * This filter process a normal entity body through any process.
 * <p>Although, you would probably include the filtered resource inside a
 * NegotiatedResource to make sure the target browser accepts this
 * content-encoding.
 */

public class ProcessFilter extends ResourceFilter {
    /**
     * Attribute index - The command we should pass the stream through.
     */
    protected static int ATTR_COMMAND = -1 ;

    static {
        Attribute a   = null ;
        Class     cls = null ;
        try {
            cls = Class.forName("org.w3c.jigsaw.filters.ProcessFilter") ;
        } catch (Exception ex) {
            ex.printStackTrace() ;
            System.exit(1) ;
        }
        // The command attribute
        a = new StringArrayAttribute("command"
                                     , null
                                     , Attribute.EDITABLE|Attribute.MANDATORY);
        ATTR_COMMAND = AttributeRegistry.registerAttribute(cls, a) ;
    }

    /**
     * A pointer to our runtime object.
     */
    protected Runtime runtime  = null ;

    /**
     * Get the command we should process the reply stream through.
     */
```

```
    public String[] getCommand() {
        return (String[]) getValue(ATTR_COMMAND, null) ;
    }

    /**
     * Process the request output through the provided process filter.
     */

    public ReplyInterface outgoingFilter (RequestInterface req,
                                          ReplyInterface rep)
        throws ProtocolException
    {
        Request request = (Request) req;
        Reply   reply   = (Reply) rep;
        Process       process   = null ;
        ProcessFeeder feeder    = null ;
        String        command[] = getCommand() ;

        // Some sanity checks:
        if (reply.getStatus() != HTTP.OK)
            return null;
        InputStream in = reply.openStream() ;
        if ( in == null )
            return null;
        if ( command == null ) {
            Reply error = request.makeReply(HTTP.INTERNAL_SERVER_ERROR) ;
            error.setContent("The process filter of this resource is not "
                             + " configured properly (it has no command).");
            throw new HTTPException(error);
        }
        // Okay, run the reply stream through the process:
        try {
            process = runtime.exec (command) ;
        } catch (IOException e) {
            Reply error = request.makeReply(HTTP.NOT_FOUND) ;
            error.setContent("The filter's process command "+
                             command[0]+
                             " wasn't found: "+e.getMessage()) ;
            throw new HTTPException (error);
        }
        if ( reply.hasContentLength() ) {
            feeder = new ProcessFeeder (process
                                        , in
                                        , reply.getContentLength()) ;
        } else {
            feeder = new ProcessFeeder (process, in) ;
        }
        feeder.start() ;
        reply.setContentLength(-1) ;
        reply.setStream(process.getInputStream());
        return reply ;
```

```
    }

    /**
     * Initialize a process filter.
     * Just get a pointer to the runtime object.
     * @param values The default attribute values.
     */

    public void initialize(Object values[]) {
        super.initialize(values);
        this.runtime = Runtime.getRuntime();
    }

}
```

```
// PutFilter.java
// $Id: PutFilter.html,v 1.1 1998/03/30 08:30:33 benoit Exp $
// (c) COPYRIGHT MIT and INRIA, 1996.
// Please first read the full copyright statement in file COPYRIGHT.html

package org.w3c.jigsaw.filters;

import java.io.*;

import org.w3c.tools.resources.*;
import org.w3c.www.http.*;
import org.w3c.jigsaw.http.*;
import org.w3c.jigsaw.resources.*;

public class PutFilter extends ResourceFilter {
    /**
     * Attribute index - The companion PutList resource's URL.
     */
    protected static int ATTR_PUTLIST = -1;

    static {
        Class     c = null;
        Attribute a = null;

        try {
            c = Class.forName("org.w3c.jigsaw.filters.PutFilter");
        } catch (Exception ex) {
            ex.printStackTrace();
            System.exit(1);
        }
        // Register the PutList URL attribute:
        a = new StringAttribute("put-list"
                                , null
                                , Attribute.EDITABLE|Attribute.MANDATORY);
        ATTR_PUTLIST = AttributeRegistry.registerAttribute(c, a);
    }

    /**
     * Resolve the companion PutList URL attribute into a resource:
     */

    private ResourceReference list = null;
    protected synchronized ResourceReference resolvePutListResource() {
        // Prepare for lookup:
        ResourceReference rr_root = null;
        rr_root = ((httpd) getServer()).getRootReference();
        FramedResource root = null;
```

```
        root = ((httpd) getServer()).getRoot();
        String       u  = getPutListURL();
        if ( u == null )
            return null;
        // Do the lookup:
        ResourceReference r_target = null;
        try {
            LookupState  ls = new LookupState(u);
            LookupResult lr = new LookupResult(rr_root);
            root.lookup(ls, lr);
            r_target = lr.getTarget();
        } catch (Exception ex) {
            r_target = null;
        }
        if (r_target != null) {
          try {
            Resource target = r_target.lock();
            if (! (target instanceof PutListResource) )
              r_target = null;
            else
              list = r_target;
          } catch (InvalidResourceException ex) {
            // continue
          } finally {
            r_target.unlock();
          }
        }
        return r_target;
    }

    /**
     * Get our companion PutListResource's URL.
     * @return The URL encoded as a String, or <strong>null</strong> if
     * undefined.
     */

    public String getPutListURL() {
        return getString(ATTR_PUTLIST, null);
    }

    /**
     * Catch PUTLIST assignments.
     * @param idx The attribute being updated.
     * @param value It's new value.
     */

    public void setValue(int idx, Object value) {
        super.setValue(idx, value);
```

```
        if ( idx == ATTR_PUTLIST ) {
            synchronized(this) {
                list = null;
            }
        }
    }

    /**
     * Nothing done in the ingoingFilter.
     * We wait until the outgoigFilter.
     * @param request The request that is about to be processsed.
     */

    public ReplyInterface ingoingFilter(RequestInterface request) {
        return null;
    }

    /**
     * Catch successfull PUTs, and keep track of them.
     * @param request The original request.
     * @param reply The original reply.
     * @return Always <strong>null</strong>.
     */

    public ReplyInterface outgoingFilter(RequestInterface req,
                                         ReplyInterface rep)
 {
        Request request = (Request) req;
        Reply   reply   = (Reply) rep;
        // Is this a successfull PUT request ?
        if (request.getMethod().equals("PUT")
            && ((reply.getStatus()/100) == 2)) {
            // Cool, keep track of the modified file:
          ResourceReference rr   = null;
          PutListResource   l    = null;
          boolean           done = false;
          synchronized (this) {
            rr = resolvePutListResource();
            if (rr != null) {
              try {
                l = (PutListResource) rr.lock();
                if ( l != null ) {
                  l.registerRequest(request);
                  done = true;
                }
              } catch (InvalidResourceException ex) {
                done = false;
              } finally {
```

```
                rr.unlock();
              }
            }
          }
          // Make sure we did something:
          if ( done ) {
            httpd s = (httpd) getServer();
            s.errlog(getClass().getName()+
                     ": unable to resolve companion PutListResource at "+
                     getPutListURL());
          }
        }
        return null;
    }


    public void initialize(Object values[]) {
        super.initialize(values);
    }

}
```

```
// PutFilter.java
// $Id: GrepPutFilter.html,v 1.1 1998/03/30 08:31:14 benoit Exp $
// (c) COPYRIGHT MIT and INRIA, 1996.
// Please first read the full copyright statement in file COPYRIGHT.html

package org.w3c.jigsaw.filters;

import java.io.*;

import org.w3c.tools.resources.*;
import org.w3c.www.http.*;
import org.w3c.jigsaw.http.*;
import org.w3c.jigsaw.frames.*;
import org.w3c.jigsaw.resources.*;

public class GrepPutFilter extends PutFilter {

    /**
     * Attribute index - The string to grep.
     */
    protected static int ATTR_FORBIDSTRING = -1;

    /**
     * Attribute index - The url to redirect.
     */
    protected static int ATTR_REDIRECT = -1;

    static {
        Class     c = null;
        Attribute a = null;

        try {
            c = Class.forName("org.w3c.jigsaw.filters.GrepPutFilter");
        } catch (Exception ex) {
            ex.printStackTrace();
            System.exit(1);
        }
        a = new StringAttribute("forbidden-string"
                                , null
                                , Attribute.EDITABLE|Attribute.MANDATORY);
        ATTR_FORBIDSTRING = AttributeRegistry.registerAttribute(c, a);

        a = new StringAttribute("redirect-url"
                                , null
                                , Attribute.EDITABLE|Attribute.MANDATORY);
        ATTR_REDIRECT = AttributeRegistry.registerAttribute(c, a);
    }
```

```
    protected String getForbiddenString() {
        return (String) getValue(ATTR_FORBIDSTRING, null);
    }

    protected String getRedirectURL() {
        return (String) getValue(ATTR_REDIRECT, null);
    }

    /**
     * Search the forbidden string in the body, if found return
     * an ACCES FORBIDDEN Reply.
     * @param request The request that is about to be processsed.
     */

    public ReplyInterface ingoingFilter(RequestInterface req) {
        Request request = (Request) req;
        String expect = request.getExpect();
        if (expect != null) {
            if (expect.startsWith("100")) { // expect 100?
                Client client = request.getClient();
                if (client != null) {
                    try {
                        client.sendContinue();
                    } catch (java.io.IOException ex) {
                        return null;
                    }
                }
            }
        }
        if(request.getMethod().equals("PUT")) {
            InputStream in = null;
            try {
                in = request.getInputStream();
                if ( in == null ) {
                    return null;
                }
            } catch (IOException ex) {
                return null;
            }
            // verify that the target resource is putable
            ResourceReference rr = request.getTargetResource();
            if (rr != null) {
                try {
                    FramedResource target = (FramedResource) rr.lock();
                    HTTPFrame frame = null;
                    try {
                        frame = (HTTPFrame) target.getFrame(
                           Class.forName("org.w3c.jigsaw.frames.HTTPFrame"));
                    } catch (ClassNotFoundException cex) {
                        cex.printStackTrace();
```

```
                    //big big problem ...
                }
                if (frame == null) // can't be putable
                    return null;
                // now we can verify if the target resource is putable
                if (! frame.getPutableFlag()) {
                    return null;
                }
            } catch (InvalidResourceException ex) {
                ex.printStackTrace();
                // problem ...
            } finally {
                rr.unlock();
            }
        }

        if (searchForbiddenString(in)) {
            Reply error = request.makeReply(HTTP.FORBIDDEN);
            error.setContent ("<p>the string \""+getForbiddenString()+
                              "\" is forbidden.</p><br> click "+
                              "<A HREF=\""+getRedirectURL()+"\">here</A>"+
                              " for explaination.");
            return error;
        }
        return null;
    } else
        return null;
}

protected boolean searchForbiddenString(InputStream in) {
    try {
        int len = in.available();
        in.mark(len);

        int c;
        byte forbid [] = getForbiddenString().getBytes();
        int lenforbid = forbid.length;
        int forbididx = 0;

        while ((c = in.read()) != -1) {
            if (forbid[forbididx] == (byte) c) {
                if (++forbididx == forbid.length) {
                    in.reset();
                    return true;
                }
            } else
                forbididx = 0;
        }
        in.reset();
        return false;
```

```
        } catch (IOException ex) {
            return false;
        }
    }

}
```

```
// PutSizeFilter.java
// $Id: PutSizeFilter.html,v 1.1 1998/03/30 08:30:45 benoit Exp $
// (c) COPYRIGHT MIT and INRIA, 1997.
// Please first read the full copyright statement in file COPYRIGHT.html

package org.w3c.jigsaw.filters;

import java.io.*;

import org.w3c.tools.resources.*;
import org.w3c.www.http.*;
import org.w3c.jigsaw.http.*;
import org.w3c.jigsaw.resources.*;

public class PutSizeFilter extends ResourceFilter {
    /**
     * Attribute index - The maximum size of the put document
     */

    protected static int ATTR_PUTSIZE = -1;
    protected static int ATTR_STRICT  = -1;

    static {
        Class     c = null;
        Attribute a = null;

        try {
            c = Class.forName("org.w3c.jigsaw.filters.PutSizeFilter");
        } catch (Exception ex) {
            ex.printStackTrace();
            System.exit(1);
        }
        // Register the PutList URL attribute:
        a = new IntegerAttribute("put-size"
                                 , new Integer(65536)
                                 , Attribute.EDITABLE|Attribute.MANDATORY);
        ATTR_PUTSIZE = AttributeRegistry.registerAttribute(c, a);
        a = new BooleanAttribute("strict"
                                 , new Boolean(true)
                                 , Attribute.EDITABLE|Attribute.MANDATORY);
        ATTR_STRICT = AttributeRegistry.registerAttribute(c, a);
    }

    private Reply notifyFailure(Request request) {
        Reply er = null;
        if (request.getExpect() != null)
            er = request.makeReply(HTTP.EXPECTATION_FAILED);
        else
            er = request.makeReply(HTTP.UNAUTHORIZED);
```

```
        er.setContent("<P>You are not allowed to PUT documents more than " +
                      getInt(ATTR_PUTSIZE, -1) + "bytes long</P>");
        return er;
    }

    public ReplyInterface ingoingFilter(RequestInterface req) {
        Request request = (Request) req;
        if(request.getMethod().equals("PUT")) {
            if(getBoolean(ATTR_STRICT, true) && !request.hasContentLength())
                return notifyFailure(request);
            if(request.getContentLength() > getInt(ATTR_PUTSIZE, -1))
                return notifyFailure(request);
        }
        return null;
    }

    public ReplyInterface outgoingFilter(RequestInterface req,
                                         ReplyInterface rep) {
        return null;
    }
}
```

```
// PassDirectory.java
// $Id: PassDirectory.html,v 1.1 1998/04/06 12:00:07 benoit Exp $
// (c) COPYRIGHT MIT and INRIA, 1996.
// Please first read the full copyright statement in file COPYRIGHT.html

package org.w3c.jigsaw.resources ;

import java.util.*;
import java.io.*;

import org.w3c.tools.resources.*;

public class PassDirectory extends org.w3c.jigsaw.resources.DirectoryResource {

    /**
     * Attribute index - The target physicall directory of this resource.
     */
    protected static int ATTR_PASSTARGET = -1 ;

    static {
        Attribute a   = null ;
        Class     cls = null ;

        // Get a pointer to our class.
        try {
            cls = Class.forName("org.w3c.jigsaw.resources.PassDirectory") ;
        } catch (Exception ex) {
            ex.printStackTrace() ;
            System.exit(1) ;
        }
        // The directory attribute.
        a = new FileAttribute("pass-target"
                              , null
                              , Attribute.EDITABLE);
        ATTR_PASSTARGET = AttributeRegistry.registerAttribute(cls, a) ;
    }

    /**
     * Catch side-effects on pass-target, to absolutize it.
     * @param idx The attribute to set.
     * @param value The new value.
     */

    public void setValue(int idx, Object value) {
        super.setValue(idx, value);
        if ( idx == ATTR_PASSTARGET ) {
            File file = (File) value;
            if ( ! file.isAbsolute() ) {
                // Make it absolute, relative to the server space.
                File abs = new File(getServer().getRootDirectory()
```

```
                                    , file.toString());
                values[ATTR_PASSTARGET] = abs;
                values[ATTR_DIRECTORY]  = abs;
            }
        }
    }

    /**
     * The getDirectory method now returns the pass-directory.
     * @return The pass target location.
     */

    public File getDirectory() {
        return (File) getValue(ATTR_PASSTARGET, null) ;
    }

    /**
     * Make the directory attribute default to the target location.
     * This is required for classes that rely on the directory attribute to
     * compute their own attributes.
     * @param values The values we should initialized from.
     */

    public void initialize(Object values[]) {
        super.initialize(values);
        File target = getDirectory();
        if ( target != null )
            setValue(ATTR_DIRECTORY, target);
    }


}
```

```
// VirtualHostResource.java
// $Id: VirtualHostResource.html,v 1.1 1998/07/09 09:28:57 benoit Exp $
// (c) COPYRIGHT MIT and INRIA, 1998.
// Please first read the full copyright statement in file COPYRIGHT.html

package org.w3c.jigsaw.resources;

import java.lang.*;
import java.util.*;
import org.w3c.tools.resources.* ;

public class VirtualHostResource extends ContainerResource {

    /**
     * Update default child attributes.
     * A parent can often pass default attribute values to its children,
     * such as a pointer to itself (the <em>parent</em> attribute).
     * <p>This is the method to overide when you want your container
     * to provide these kinds of attributes. By default this method will set
     * the following attributes:
     * <dl><dt>name<dd>The name of the child (it's identifier) -
     * String instance.
     * <dt>parent<dd>The parent of the child (ie ourself here) -
     * a ContainerResource instance.
     * <dt>url<dd>If a <em>identifier</em> attribute is defined, that
     * attribute is set to the full URL path of the children.
     * </dl>
     */

    protected ResourceContext updateDefaultChildAttributes(Hashtable attrs) {
        ResourceContext context = super.updateDefaultChildAttributes(attrs);
        if (context == null) {
            context = new ResourceContext(getContext());
            attrs.put("context", context) ;
        }
        attrs.put("url", "/");
        return context;
    }
}
```

Authentication in Jigsaw.

Jigsaw Home / Documentation Overview

This section will provide you with a basic explanation of authentication in **Jigsaw**.

Filters are attached to specific frames in order to filter accesses to their resource. These filters are called once at lookup time, and once at reply time. On the way in (lookup time), they allow you to manipulate the request before the target resource handles it, and on the way out, they allow you to manipulate the target's reply before it is emitted back to the browser.

Although Jigsaw provides a number of filters, we will focus here on the authentication filter, that authenticate requests before they are handled by their appropriate target resources. The GenericAuthFilter is currently the only available authentication filter.

The GenericAuthFilter needs an authentication realm. An authentication realm is a database that will contain the description of a set of users, along with their passwords and/or IP adresses.

Each user defines a set of attributes, email, comments, ipadress, password. The email address is currently unused (but it might be used in the future for email notification). The comments field is used only for informational purposes. The ipaddress field allows you to state from which machine the user is allowed to connect. This field is not mandatory: if left blank, only the password will be used for authentication (be warned that the password authentication scheme used by HTTP is very weak, you should always specify both a password and some IP addresses). If you decide to fill in the ipaddress field, you can enter multiple addresses for the same user (one per line). You can use * in the ip address field, meaning that any user connecting from the given set of IP addresses is to be authentified as the realm user.

Now you should read the tutorial to setup authentication.

Jigsaw Team

$Id: AuthInJigsaw.html,v 1.4 1999/03/16 13:39:11 bmahe Exp $

All resources All frames

AuthUser

This resource describe a user in a **Jigsaw** realm.

Inherits

The AuthUser class inerits from the following classes:

- Resource

Attributes description

The AuthUser defines the following attributes:

- email
- comments
- ipaddress
- password
- groups

`email`

semantics

The email adress of the user. (optionnal)

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

`comments`

semantics

Any comment. (optionnal)

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

`ipaddress`

semantics

Access restricted to request comming from hosts having an IP adress matching one of these (* is accepted, ie "`138.*.*.*`"). This is a list of IP adresses. (optionnal)

type

This attribute is an editable IPTemplatesAttribute

default value

This attribute defaults to **null**.

`password`

semantics

The password needed to accept the incomming request. (mandatory)

type

This attribute is an editable PasswordAttribute

default value

This attribute defaults to **null**.

`groups`

semantics

unused. (optionnal)

type

This attribute is an editable StringArrayAttribute

default value

This attribute defaults to **null**.

Jigsaw Team

$Id: org.w3c.jigsaw.auth.AuthUser.html,v 1.3 1999/09/08 14:58:11 ylafon Exp $

Server Side Include commands

Jigsaw Home / Documentation Overview

These commands should be written in a .shtml file that is indexed as a FileResource associated with a SSIFrame. The commands are separated in two parts:

A complete example.

Commands

ConfigCommand

Used to set the **sizefmt** and **datefmt** variables, which control the output of file sizes and dates.

Parameters:

- **datefmt** = the date format
- **sizefmt** = the size format

Example:

```
<!--#config sizefmt="bytes" -->
```

CookieCommand

Cookies access from server side includes.

Parameters:

- **get** = name of the cookie to get
- **alt** = alternative value
- **if** = name of the cookie (just to test if the cookie exists)
- **then** = value

Examples:

```
<!--#cookie get="C1" alt="hello" -->,
display the value of the cookie "C1" or "hello" if the cookie
is not in the incomming request.

<!--#cookie if="C1" then="hello" alt="bye"-->,
display "hello" if the cookie "C1" is in the request. If not display "bye"
```

CountCommand

This command inserts the number of recorded accesses to this resource, as reported by CounterFilter.

Parameters: **none**

Example:

```
<!--#hitcount -->
```

EchoCommand

The **echo** SSI command. As extensions, it has the parameters "reqstate" (for echoing **Jigsaw** request states) and "reqheader" (for echoing request header).Also, it can take the flag "here", whose presence means that the variable is to be interpreted at the deepest request level (in the case of chained internal requests), instead of doing so at the top (external request) level. It inserts the value of a variable in the document.

Parameters:

- **var** = a SSI variable
- **reqstate** = a Jigsaw request state
- **reqheader** = a request header
- **here** = a flag

Examples:

```
<!--#echo var="DOCUMENT_URI" -->
display the document uri

<!--#echo reqheader="referer"  -->
display the referer header of the request
```

```
<!--#echo reqstate="pathinfo"  -->
display the request state "pahtinfo"
```

ExecCommand

the SSI **exec** command. It inserts the output from a CGI script or a shell command in the document. Note that in the Jigsaw architecture CGI scripts are just another resource class, so that no distinction is made between executing a CGI script or including a file.

Parameters:

- **cmd** = the command to execute

Example:

```
<!--#exec cmd="ls -lsa" -->
display the result of the ls command
```

FLastModCommand

The standard **lastmod** SSI command.

Parameters: **none**

Example:

```
<!--#flastmod-->
```

FSizeCommand

The SSI **fsize** command. It inserts the size of the unparsed file in the document, according to the current value of the variable **sizefmt**.See configCommand.

Parameters: **none**

Example:

```
<!--#fsize -->
```

IncludeCommand

The SSI **include** command. (CGI scripts *can* be included, simply by providing a so-called virtual path to a resource with a CgiFrame).

Parameters:

- **file** = the file to include
- **virtual** = a virtual path
- **ifheader** = a request header
- **else** = a file

Examples:

```
<!--#include file="included.html" -->
include the file "included.html" in the current file
```

```
<!--#include ifheader="Referer" file="included.html" else="included2.html" -->
if the request has a Referer header then include "included.html"
else include "included2.html"
```

jdbcCommand

The SSI **jdbc** command allows you to query a SQL database via JDBC. Combinated with some Control Commands, it allows you to display the content of a database easyly.

Parameters:

- **select** = the SQL request
- **url** = the database URL
- **driver** = the JDBC driver class
- **user** = the username
- **password** = the password
- **name** = the result name
- **column** = the column number
- **next** = a flag

Example:

```
<!--#jdbc select="SELECT * FROM services" name="result"
    driver="com.imaginary.sql.msql.MsqlDriver"
    url="jdbc:msql://www43.inria.fr:4333/services" -->
this is the setup of the command.

<!--#jdbc name="result" next="true" -->
this command move the pointer to the next line of the result set.

<!--#jdbc name="result" column="1" -->
display the first column of the current line.

<!--#jdbc name="result" column="2" -->
display the second column of the current line.

<!--#jdbc name="result" column="3" -->
display the third column of the current line.
```

ServletCommand

The SSI **servlet** command. Servlet can be executed simply by providing a url path to a servlet class.

Parameters:

- **name** = the command identifier
- **code** = the servlet URL
- **param** = a parameter name
- **value** = a parameter value

Example:

```
<!--#servlet name="Snoop" param="p1" value="v1" -->
<!--#servlet name="Snoop" param="p2" value="v2" -->
<!--#servlet name="Snoop" param="p3" value="v3" -->
<!--#servlet name="Snoop" code="/servlet/snoop" -->
```

Control Commands

CounterCommand

The SSI **counter** command. Used to do things like cpt = cpt + 1.

Parameters:

- **name** = the counter identifier
- **init** = the init value
- **incr** = the value to add to the counter
- **value** = a flag

Example:

```
<!--#cpt name="cpt1" init="0" -->
Initialisation: cpt1 = 0

<!--#cpt name="cpt1" incr="1" -->
cpt1 = cpt1 + 1

<!--#cpt name="cpt1" value="true" -->
display the current value of cpt1
```

ElseCommand

Parameters:

- **name =** the *if* command identifier

Example:

```
<!--#else name="if2" -->
```

EndifCommand

Parameters:

- **name** = the *if* command identifier

Example:

```
<!--#endif name="if2" -->
```

EndloopCommand

Parameters:

- **name** = the *loop* command identifier

Example:

```
<!--#endloop name="loop2" -->
```

ExitloopCommand

Parameters:

- **name** = the *loop* command identifier
- **command** = a command name (jdbs, cpt)
- **var** = a command identifier
- **equals** = a string value

Example:

```
<!--#exitloop name="loop2" command="cpt" var="cpt1" equals="4" -->
```

IfCommand

Parameters:

- **name** = the command identifier
- **command** = a command name (jdbc, cpt)
- **var** = a command identifier
- **equals** = a string value

Example:

```
<!--#if name="if2" command="cpt" var="cpt1" equals="2" -->
```

LoopCommand

Parameters:

- **name** = the command identifier

Example:

```
<!--#loop name="loop2" -->
```

Example

The following example, display the four first columns of the three first lines of the users database.

```
<!--#jdbc name="result2" select="SELECT * FROM users"
    user="bmahe" password=""
    url="jdbc:msql://www43.inria.fr:4333/users"
    driver="com.imaginary.sql.msql.MsqlDriver"  -->

<table border=2>
  <!--#cpt name="cpt1" init="0" -->
    <tr><td><b>Name </td><td><b>Login</td>
```

```
            <td><b>Email</td><td><b>Age  </td></tr>
        <!--#loop name="loop2" -->

        <!--#jdbc name="result2" next="true" -->

        <tr>
          <td>
            <!--#jdbc name="result2" column="1" -->
          </td><td>
            <!--#jdbc name="result2" column="2" -->
          </td><td>
            <!--#jdbc name="result2" column="3" -->
          </td><td>
            <!--#jdbc name="result2" column="4" -->
          </td>
        </tr>

        <!--#cpt name="cpt1" incr="1" -->
        <!--#exitloop name="loop2" command="cpt" var="cpt1" equals="3" -->
        <!--#endloop name="loop2" -->

    </table>

    counter value : <!--#cpt name="cpt1" value="true" -->
```

The result could be:

Name	Login	Email	Age
Smith	ssmith	ssmith@foobar.org	25
lafrim	ylafrim	lafrim@foobar.org	25
Teole	pteole	teole@foobar.org	27

counter value : 3

Jigsaw Team

$Id: SSI.html,v 1.38 1999/03/16 13:39:13 bmahe Exp $

All resources All frames

SampleResourceIndexer

The SampleResourceIndexer is the basic Indexer class. Read the indexer documentation for more details on indexers.

Inherits

The SampleResourceIndexer class inerits from the following classes:

- Resource

Attributes description

The SampleResourceIndexer defines the following attributes:

- super-indexer

`super-indexer`

semantics

If an indexer instance doesn't know how to index something, it will ask its super indexer.

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

Jigsaw Team

$Id: org.w3c.tools.resources.indexer.SampleResourceIndexer.html,v 1.3 1999/09/08 14:58:11 ylafon Exp $

All resources All frames

GhostResourceIndexer

deprecated

Inherits

The GhostResourceIndexer class inerits from the following classes:

- Resource
- SampleResourceIndexer

Attributes description

The GhostResourceIndexer doesn't defines any attribute.

Jigsaw Team

$Id: org.w3c.tools.resources.indexer.GhostResourceIndexer.html,v 1.3 1999/09/08 14:58:11 ylafon Exp $

All resources All frames

ContentTypeIndexer

This indexer perform indexation based on the Content-Type Header of incomming PUT requests. Read the indexer documentation for more details.

Inherits

The ContentTypeIndexer class inerits from the following classes:

- Resource
- SampleResourceIndexer

Attributes description

The ContentTypeIndexer doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.jigsaw.indexer.ContentTypeIndexer.html,v 1.3 1999/09/08 14:58:11 ylafon Exp $

All resources All frames

ZipIndexer

Used only for zip browzing.

Inherits

The ZipIndexer class inerits from the following classes:

- Resource
- SampleResourceIndexer

Attributes description

The ZipIndexer doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.jigsaw.zip.ZipIndexer.html,v 1.4 1999/09/08 14:58:11 ylafon Exp $

All resources All frames

ServletIndexer

Indexer dedicated to servlets, read the servlet documentation for more details.

Inherits

The ServletIndexer class inerits from the following classes:

- Resource
- SampleResourceIndexer

Attributes description

The ServletIndexer doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.jigsaw.servlet.ServletIndexer.html,v 1.3 1999/09/08 14:58:11 ylafon Exp $

All Resources All frames

HttpManager

The HttpManager is the class that handles HTTP requests, and gets back HTTP replies. It a highly configurable engine, both by extending it in Java, and by a rich set of supported properties.

Properties

The HttpManager uses the following properties (when defined):

- org.w3c.www.protocol.http.filters
- org.w3c.www.protocol.http.cacheControl.maxStale
- org.w3c.www.protocol.http.cacheControl.minFresh
- org.w3c.www.protocol.http.cacheControl.onlyIfCached
- org.w3c.www.protocol.http.userAgent
- org.w3c.www.protocol.http.accept
- org.w3c.www.protocol.http.acceptLanguage
- org.w3c.www.protocol.http.acceptEncoding
- proxySet
- proxyHost
- proxyPort

`org.w3c.www.protocol.http.filters`

semantics

List the filters to apply globally to all requests. The **Jigsaw** HTTP client API comes with the notion of filters. Filters can be registered either globally (as is the case when you use this property), or locally (when using the Java API). *Local* filters are filters that are applied only to a given sub-domain of URLs. To understand the difference between global and local filters, consider the case of authentication.
Authentication is handled by a global filter, that will intercept any reply requiring credential information. When such a reply is intercepted, the global AuthFilter prompts the user for a user name and password. It then retries the request and if it succeed, it installs a local filter on the sub-domain of protected URLs.
The currently available filters are:

org.w3c.www.protocol.http.DebugFilter

Will print all outgoing requests to the standard output, along with all incoming replies. Very useful for debugging, as the name says.

org.w3c.www.protocol.http.cookies.CookieFilter

The cookie filter. This filter manage cookies, search and send cookies relative to each url requested.

org.w3c.www.protocol.http.cache.CacheFilter

A full caching module for HTTP. This filter will maintain a cache of least recently accessed resources, and will use it to serve requests when possible, thus avoiding network access.

org.w3c.www.protocol.http.auth.AuthFilter

The authentication filter. Handles only Basic authentication for the time being, but will soon be improved to handle Digest authentication too.

default value

This property has no default value. A typical setting of that property looks like:
`org.w3c.www.protocol.http.filters=org.w3c.www.protocol.http.DebugFilter|org.w3c.www.protocol.http.auth.AuthFilter|org.w3c.www.protocol.http.cache.CacheFilter`

`org.w3c.www.protocol.http.cacheControl.maxStale`

semantics

This property controls cache usage. It allows the HTTP client side API to advertise the maximum staleness of cached documents it is willing to accept. This property should be set to a number of seconds, any cache (either local, or proxy) will be able to return stale responses when this property is set to a value greater than **`0`**.

default value

This property has no default value.

`org.w3c.www.protocol.http.cacheControl.minFresh`

semantics

This property controls cache usage. It indicates to the HTTP client side API to advertise a minimum freshness value for cached documents. This property should be set to a number of seconds. Any cache (either local or remote) will be allowed to use cache entries, only if the document will remain fresh for the given number of seconds.

default value

This property has no default value.

`org.w3c.www.protocol.http.cacheControl.onlyIfCached`

semantics

This property control cache usage. It indicates to the HTTP client side API to returns *only* documents if they are available in the cache. This is very useful if you are planning to do some browsing in *disconnected* mode. Note: a nifty thing to do would be to write a robot to fill in the cache, and then turn your browser or proxy into disconnected mode by setting this property (be sure we will have an HTTP robot available in Java real soon ;-)
This property is boolean: setting the property to any value will enable the feature.

default value

This property has no default value.

`org.w3c.www.protocol.http.userAgent`

semantics

This property indicates the `User-Agent` header that the HTTP client API should advertise.

default value

This property defaults to **Jigsaw/1.0a5**.

`org.w3c.www.protocol.http.accept`

semantics

This property indicates the media types the client is willing to deal with. Its value is mapped directly to the `Accept` HTTP header.

default value

This property defaults to ***/***.

`org.w3c.www.protocol.http.acceptLanguage`

semantics

This property indicates what languages the client is willing to receive. Its value is mapped directly to the `Accept-Language` HTTP header.

default value

This property has no default value.

`org.w3c.www.protocol.http.acceptEncoding`

semantics

This property indicates what encodings the client is willing to deal with. Its value is mapped directly to the `Accept-Encoding` HTTP header.

default value

This property has no default value.

`proxySet`

semantics

Should the client API set itself in proxy mode. When you set this property, you turn the whole client API to use a proxy. See the `proxyHost` and `proxyPort` properties.
This is a boolean property, setting it to any value will enable the feature.

default value

This property has no default value.

`proxyHost`

semantics

The host name of the machine running the proxy to connect to for handling the HTTP protocol.

default value

This property chas no default value.

`proxyPort`

semantics

The port number of the HTTP proxy to connect to for handling the HTTP protocol.

default value

This property has no default value.

Jigsaw Team

$Id: w3c.www.protocol.http.HttpManager.html,v 1.1 1996/09/07 23:45:12 abaird Exp $

Cookie Filter

The Cookie Filter provides client side cookie support for the HTTP protocol.

Properties

The CookieFilter defines the following properties:

- org.w3c.www.protocol.http.cookie.file

`org.w3c.www.protocol.http.cookie.file`

semantics

The absolute path of the file used to store cookies.

type

A String propery

default value

"cookie" in the current directory

Jigsaw Team

$Id: w3c.www.protocol.http.cookies.CookieFilter.html,v 1.2 1997/07/31 08:25:49 ylafon Exp $

Cache Filter

The Cache Filter provides client side caching support for the HTTP protocol. The cache is as much as possible HTTP/1.1 compliant and can be used either by standalone browsers,or by a proxy (the typical case when you use **Jigsaw**).

When this filter is used throughout **Jigsaw** proxy, it is best (ie read *recommended*) to use the appropriate resources to configure it (instead of setting manually the properties). In that case, the CacheProp resource.

Properties

The CacheFilter defines the following properties:

- org.w3c.www.protocol.http.cache.size
- org.w3c.www.protocol.http.cache.debug
- org.w3c.www.protocol.http.cache.shared
- org.w3c.www.protocol.http.cache.directory
- org.w3c.www.protocol.http.cache.connected
- org.w3c.www.protocol.http.cache.garbageCollectionEnabled
- org.w3c.www.protocol.http.cache.fileSizeRatio

`org.w3c.www.protocol.http.cache.size`

semantics

The size of the cache, expressed in bytes.

type

An integer property

default value

This property defaults to **5000000** bytes, ie 5Mb

`org.w3c.www.protocol.http.cache.debug`

semantics

Will make the CacheFIlter emit some (hopefully) usefull traces.

type

A boolean property

default value

This property defaults to **false**.

`org.w3c.www.protocol.http.cache.shared`

semantics

Is this cached shared among several users ? Some of HTTP/1.1 caching semantics depends on wether the cache is shared or not, this flag will change the caching policy accordingly.

type

A boolean property.

default value

This property defaults to **false**.

`org.w3c.www.protocol.http.cache.directory`

semantics

The directory that the cache should use to store cached content. If not provided the directory defaults to the `.web_cache` under the current user's home directory. Note that when used through **Jigsaw** the cache directory will automatically be set to `config/cache` where `config` is the main **Jigsaw** configuration repository.

type

A File property (should provide the path of a directory in the file system)

default value

See above.

`org.w3c.www.protocol.http.cache.connected`

semantics

Should the cache consider itself connected to the Internet ? The cache filter can be used in *disconnected* mode, in which case it will only look for document in the cache, and if not found, will emit the appropriate HTTP/1.1 error reply.
You may experience funny results when disconnecting the cache from the Internet, most of them are the results of content providers not providing the appropriate informations (and some time, even intentionnaly).

type

A boolean property

default value

This property defaults to **true**.

`org.w3c.www.protocol.http.cache.garbageCollectionEnabled`

semantics

Should the cache try to run its garbage collector ? This flag is intended to be used when the user's is planning to disconnect the cache. By setting the flag to **false**, he/she can safely browse the web and everything he/she visits will enter the cache (for latter consumption) and stay there.
An interesting project, of course, would be to write a robot to fill the cache automatically (stay tuned, or write it !)

type

A boolean property

default value

This property defaults to **true**.

`org.w3c.www.protocol.http.cache.fileSizeRatio`

semantics

The ratio to the total cache size that a single entry in the cache is allowed to occupy.

type

A double property (a ratio between **0** and **1** indicating how much of the total cache space a single cached entry is able to occupy).

default value

This property defaults to **0.1** (which means that if you are using the default cache size of 5Mb, only file smaller than 500Kb are candidate to enter the cache)

Jigsaw Team
$Id: w3c.www.protocol.http.cache.CacheFilter.html,v 1.2 1997/07/31 08:25:42 ylafon Exp $

ICP Filter

The ICP client side filter add support for the Internet Cache Protocol to the **Jigsaw** HTTP client side implementation. This support can be used both by browsers (such as HotJava, check the FAQ), or by proxies (the typical case).

In brief, ICP allows several proxies to cooperate by exchanging informations about what they currently have in their local cache. **Jigsaw**'s ICP implementation should be compatible with other ICP capable proxies (such as Squid), even though it doesn't make use of all the features of the underlying protocol (in fact, it implements what can be implemented while still remaining compliant with the HTTP/1.1 protocol specification).

Warning: **Jigsaw** doesn't come with the appropriate support for editing the ICP filter properties from the `/Admin/Properties` resource; this means that you will have to manually edit your **Jigsaw** configuration file in order to set the properties relevant to that filter. This file is usually named `config/http-server.props` if not, you probably know already what we are talking about.

ICP Configuration File

This filter uses a configuration file, to know about the proxies it cooperates with. The syntax for this rule file is defined by the following BNF:

```
rule-file := *(rules)
rules     := (comment|host-description)
comment   := '#' <chars up EOL>
host-description := host SPACES port spaces url EOL
host      := <valid Internet host name>
port      := <valid UDP port number>
url       := <URL locating the proxy to use for that host/port>
SPACES    := *(' '|'\t')
EOL       := '\r' | '\r\n' | '\n'
```

A sample ICP configuration file looks like:

```
# Sample ICP configuration file
# -----------------------------
# First neighbour:
icp.host1.com 10345 http://icp.host1.com:8080/
# Second neighbour:
icp.host2.com 10345 http://icp.host2.com:8080/
```

After parsing such a configuration file, the ICP filter, will register both hosts as *neighbours*. When a request comes in for some document, the filter sends a UDP packet to both hosts (it knows about their namee, and the port they are listening on through the first two fields of the configuration file). It then waits for some positive answer, if such an answer arrives, it finally looks it up (matching the UDP sender address with one of the above declared hosts), and use the appropriate URL as a proxy address.

To setup a nice cache hierarchy, one can use a conjunction of the ProxyDispatcher filter, the CacheFilter and the ICP filter. The starting point for setting up such a config, is to set the HttpManager filters property to:

```
org.w3c.www.protocol.http.filters=org.w3c.www.protocol.http.proxy.ProxyDispatcher \
     |org.w3c.www.protocol.http.cache.CacheFilter \
     |org.w3c.www.protocol.http.icp.ICPFilter
```

(the \ character indicates that the line is splitted for best reading, they should not appear in the property value - which should be written as a single line). Note that in this setting the order in which you declare the filters is indeed important.

Properties

The ICPFilter defines the following properties:

- org.w3c.www.protocol.http.icp.debug
- org.w3c.www.protocol.http.icp.config
- org.w3c.www.protocol.http.icp.port
- org.w3c.www.protocol.http.icp.timeout
- org.w3c.www.protocol.http.icp.disable-cache

`org.w3c.www.protocol.http.icp.debug`

semantics

When set to **true**, this property will make the ICP filter emit some (hopefully) interesting traces.

type

A boolean property.

default value

This property defaults to **false**.

`org.w3c.www.protocol.http.icp.config`

semantics

The name of the *rule file* for ICP

type

A File property (it's value should provide a path to an existing file)

default value

This property has no default value, and **must** be set for the ICP filter to be activated.

`org.w3c.www.protocol.http.icp.port`

semantics

The UDP port number that filter should use to communicate with cooperating proxies. This port number is the one the filter will be listening at for requests from other (cooperating proxies).

type

An integer property (it's value should be a valid UDP port number)

default value

This property has no default value and **must** be set for the ICP filter to be activated.

`org.w3c.www.protocol.http.icp.timeout`

semantics

The number of milliseconds the filter should wait for replies from cooperating proxies, before estimating that none of

them is available to answer its query.

type

An integer property (it's value should be a valid number of milliseconds)

default value

This property defaults to **500** ms.

`org.w3c.www.protocol.http.icp.disable-cache`

semantics

Disable the caching of documents retreived through a neighbour proxy. If you're making a set of *local* proxies cooperate, this flag will ensure that no two close proxy will cache the same page.

type

A boolean property.

default value

This property defaults to **true**

Jigsaw Team

$Id: org.w3c.www.protocol.http.icp.ICPFilter.html,v 1.5 1999/09/08 14:58:11 ylafon Exp $

ProxyDispatcher

The ProxyDispatcher is a filter that allows some rule to be applied to some given request before the HTTP client side API emits out a request. The set of rules can be extended in **Java**, check below for the currently defined rules.

Warning: When configuring that filter along with **Jigsaw**'s proxy module, you will need to manually edit **Jigsaw**'s property file (usually found at `config/http-server.props`, otherwise, you know what we are talking about).

ProxyDispatcher Rules

The basic syntax for the ProxyDispatcher *rule file* is captured by the following BNF:

```
rule-file := (record)*
record    := comment | rule
comment   := '#' <any chars up to EOL>
rule      := rule-lhs SPACES rule-rhs
rule-lhs  := token | default
default   := 'default'
rule-rhs  := forbid | direct | proxy | authorization | proxyauth
forbid    := 'forbid'
direct    := 'direct'
proxy     := 'proxy' SPACES url
authorization := 'authorization' SPACES user SPACES password
proxyauth := 'proxyauth' SPACES user SPACES password SPACES url
user      := token
password  := token
EOL       := '\r' | '\r\n' | 'n'
SPACES    := (' '|'\t')+
```

A sample ProxyDispatcher rule file looks like:

```
# Sample ProxyDispatcher rule file
# --------------------------------

# Make all access to US through us.proxy.com
edu proxy http://us.proxy.com:8080/
org proxy http://us.proxy.com:8080/

# Accesses to french site are direct (no proxy)
fr  direct
```

```
# Accesses to 18.59.*.* network are direct
18.59 direct

# Accesses to the protected site gets decorated with auth infos:
www.protected.com authorization joe-user joe-password

# Forbid accesses to some sites

www.evilsite.com forbid

# Access this site through myproxy.com with proxy authentication
www.somesite.org proxyauth joe-user joe-password http://myproxy.com:8008/

# force all other request to go through world.proxy.org
DEFAULT proxy http://world.proxy.org:8080/
```

The rule matching algorithm matches the host name part of urls, or the numeric part, if the address is numeric, no name resolution. The matching algorithm tries to find the best match, starting with the most significant part of the URL (in www.foo.com, com is the most significant part, in 18.23.0.22, 18 is the most significant part) and then walking toward the best match, hence host names are implicitly "terminated" by * if you will. In the above example, any access to `www.foo.fr/x/y` would be handled by:

1. Reverting the host name components: `fr foo www`
2. Looking for a match on fr (found)
3. Looking for a match on `fr foo` (not found)

In that case the rule found at step 2 is the most specific, and gets applied.

This examples is self explanatory, and illustrates all the rules currently handled by the filter. When used in conjunction with the ICP filter, you can get a very powerful caching hierarchy.

Note also that the underlying implementation of the rule matching algorithm allows a large number of rules which can lead to a big static routing table.

Properties

The ProxyDispatcher defines the following properties:

- org.w3c.www.protocol.http.proxy.rules
- org.w3c.www.protocol.http.proxy.debug

`org.w3c.www.protocol.http.proxy.rules`

semantics

The location of the rules for the ProxyDispatcher filter. The *rule file* expresses a rule to be applied when fetching a document, see the rule syntax for more informations.

type

This property can be either a full URL or a filename.

default value

This property has no default value, and **must** be set for the filter to be activated.

`org.w3c.www.protocol.http.proxy.debug`

semantics

Should the filter emit debugging traces ? When set to **true** this will make the filter tells which rule it applies on which fetched URL.

type

A boolean property

default value

This property defaults to **false**.

---

ylafon

$Id: w3c.www.protocol.http.proxy.ProxyDispatcher.html,v 1.4 1997/09/22 09:02:23 ylafon Exp $

All Resources All frames

DefaultCommandRegistry

This class provides the most general and commonly-used SSI commands. Compatibility with the NCSA-style directive set has been maintained as much as it made sense to, and new functionality adequate to Jigsaw has been added.

The following summary of the commands only discuss differences from the NCSA directive set. Please refer to the NCSA server-side includes tutorial for comparison.

The full set of commands of the DefaultCommandRegistry is:

`config`

The `errmsg` tag is not implemented.

`include`

The `file` and `virtual` tags are handled in the same way. Both originate an internal request to the URL given as the value of the tag. There is no provision for including a file that is not indexed by Jigsaw. This command can be used to include the content of *any* resource. This includes the SSIResource.

In addition, the following tags are admissible:

`ifheader`

Its value is interpreted as a header name. It causes the resource to be included only if that header was defined in the original (client) request.

`else`

Used in conjunction with `ifheader`, it specifies a URL to be included in case the header is *not* defined.

`echo`

In addition to the `var` tag, which has the NCSA behavior, the following tags are admissible:

`reqstate`

Its value is interpreted as a Jigsaw request state, and is expanded as the value of the state. For instance, the command
`<!--#echo reqstate="org.w3c.jigsaw.filters.CounterFilter.count"-->`
will print the current hit-count, assuming a CounterFilter exists for the resource.

`reqheader`

Its value is interpreted as a header in the request, and is expanded as the value of the header.

`here` If this tag is present, command is expanded as interpreted relative to the innermost internal request. By default, it is interpreted relative to the original (client) request.

`fsize`

Behaves like its NCSA counterpart, except that it also recognizes the tag `here`. If present, this tag indicates to include the file size of the innermost included file. Normally, it includes the file size of the topmost SSI-parsed file requested by the client. It honors the `sizefmt` variable, as set by `config`.

`flastmod`

In addition to NCSA behavior, it honors the `here` tag, which indicates to include the time stamp of the innermost included file.

`exec`

It accepts *only* the `cmd` tag. Given that the `include` command can include CgiResources, the `cgi` tag is superfluous.

If the SSIResource `secure` attribute is set, this command will be inoperative.

`params`

This command expands to an HTML unordered list of the parameters that it was called with. Provided mainly for instructional purposes.

`count`

Expands to the access count reported by the CounterFilter. (This may or may not mean the access count of the document, depending on the way the CounterFilter is set up)

Antonio Ramírez

$Id: org.w3c.jigsaw.ssi.commands.DefaultCommandRegistry.html,v 1.1 1998/03/30 16:10:38 bmahe Exp $

Page Compilation in Jigsaw.

Jigsaw Home / Documentation Overview

- What is Page Compilation?
- Example
- Tutorials

What is Page Compilation?

Page Compilation allows you to generate HTML pages from HTML/Java pages. **Jigsaw** has a special frame (PageCompileFrame) that translate the HTML/Java file in a **Jigsaw** frame source file, compile it and runs the new compiled frame. The PageCompileFrame can be used like any other frame. Read the Resource Configuration page for more details on **Jigsaw** frames and resources.

Example:

The following HTML/Java page

```
<html>
 <head>
     <title>Page Compilation Test</title>
 </head>
 <body>
     <h1>Page Compilation Test</h1>
     <java>
       out.println ("<p>Page Compilation test");
     </java>
     <h1>Display a list</h1>
     <ul>
       <java>
         for (int i = 1; i < 10; i++) {
           out.println ("<li> item " + i);
         }
       </java>
     </ul>
 </body>
```

```
</html>
```

will be translated in this HTML page:

Page Compilation Test

Page Compilation test

Display a list

- item 1
- item 2
- item 3
- item 4
- item 5
- item 6
- item 7
- item 8
- item 9

Tutorials

- Configuring Page Compilation
- Writing your own HTML/Java pages
- List of <java> tags, available variables, and how to use them.

Jigsaw Team

$Id: Overview.html,v 1.6 1999/03/16 13:40:27 bmahe Exp $

Pics setup

Jigsaw Home / Documentation Overview

This page describes how to setup the **Jigsaw** server for PICS support.

1. Using the server as a label bureau
2. Using the server to serve labels with document

Sample implementation configuration

We describe here the setting for the sample implementation. These objects use a simple database, whose implementation relies on the underlying file system:. To each sample label bureau, is associated a *home directory*. To look for labels on a given URL *U* rated by service whose identifier is *SURL* it parses it into the following components:

- The service host name (shost),
- The service port number (sport),
- The optional service path (spath),
- The target URL protocol (uproto)
- The target URL host name (uhost),
- The target URL port number (uport),
- The target URL path (upath).

Out of these components, it builds a file name:

For generic labels

<bureau-home-directory>/*<shost>*/*<sport>*/*<spath>*/*<uproto>*/*<uhost>*/*<uport>*/*<upath>*.gen

For specific labels

<bureau-home-directory>/*<shost>*/*<sport>*/*<spath>*/*<uproto>*/*<uhost>*/*<uport>*/*<upath>*

If the *port* equals 80 (the normal http port), it is omited. The label bureau directory is a parameter of the sample label bureau implementation. As an example, if the label bureau directory is `www/labels` and we are looking for generic labels for `http://www.w3.org/pub/WWW` by the service `http://www.rating.com`, the sample implementation will check for a file named `www/labels//http/www.rating.com/http/www.w3.org/pub/WWW.gen`. If such a file exists, it should contain a label representation, in a format decribed below.

The rest of this document answers the following questions:

- How do I set up my server to serve labels with documents ?
- How do I set up my server as a label bureau ?
- How can I edit labels, what is the label file format ?
- Sample setting: going through a full example.

Serving labels with documents

Jigsaw uses the PICSFilter to implement this part of the PICS specification . This filter will process each outgoing reply by adding to it the appropriate PICS headers. It takes one parameter, named `bureau` which will be passed to the label bureau factory when the server needs to access the label bureau. In the implementation of label bureau, this should be a string giving the absolute path of the label bureau database.

Upon each request crossing the `labeled` directory, the PICS filter will examine both the request, and the original reply (as built by the target resource). If PICS labels are requested, it will query the label bureau (whose home directory is given through the `bureau` parameter), and add to the reply the additional PICS headers.

Label bureau

Jigsaw can also be used as a label bureau. This is fearly easy to setup: the label bureau itself is implemented as a specific resource, which will decode requests, make the appropriate queries to its bureau implementation, and send back the automatically generated PICS document.

To register a label bureau resource within your server space, follow the normal configuration steps (see the Resource Configuration guide for more informations).

Editing labels

Editing labels, right now, is done by editing the label files, as described in the first section of this document. These files have a very simple format:

```
lines     = line lines | <empty>
line      = <attribute> '=' <value>
attribute = <Any char except '='>
value     = <Any char except EOL>
```

The following attributes are mandatory:

For all labels:

generic
: Wether the label is generic or not

ratings
: The actual ratings for the labeled document

For generic labels:

for
: The full URL of the labeled document

Here is the list of all attributes:

```
by                = quotedname
generic           = boolean
for               = quotedURL
```

```
on                = quoted-ISO-date
signature-RSA-MD5 = "base64-string"
until             = quoted-ISO-date
exp               = quoted-ISO-date
at                = quoted-ISO-date
MIC-md5           = "base64-string"
md5               = "base64-string"
comment           = quotedname
complete-label    = quotedURL
full              = quotedURL
extension         = '(' mand/opt quotedURL data* ')'
 mand/opt :: 'optional' | 'mandatory'
 data :: quoted-ISO-date | quotedURL | number | quotedname | '(' data* ')'
ratings           = pics-ratings
```

Read this for more informations about Pics labels.

Sample setting

This section describes a full example of PICS setting. We will use the following terms:

root
: The server root directory (not to be confused with its space directory, which will usually be *root*/WWW).

host
: The host the server is running on, as defined by your command line, or by the org.w3c.jigsaw.host property

port
: The port the server is listening to, as defined either by your command line, or by the org.w3c.jigsaw.port property.

When ever you encounter these italized tokens, you should replace them with your own value.

Setting up a label bureau database

We will first set-up a label bureau. For this we need to assign a directory to it, let's use *root*/labels for this purpose. We create this directory, empty:

`create the directory root/labels`

Than we need to define some service. Let's say we want to create the `www.rating.com` service, which will use the http protocol. Its identifier URL will be `http://www.rating.com`, so we need to create the following directories:

`create the directory root/labels/http/www.rating.com`

Now, lets make a label for `http://www.w3.org/PICS` by this service. This label will be generic (so it will apply to anything below this url), and will be locate in the following file:

`root/labels/http/www.rating.com/http/www.w3.org/pub/WWW/PICS.gen`

What should we put in this file ? This depends on the category the rating service defines. For the sake of simplicity, lets say that it defines only one category, namely the minimal age the surfer should be. Note that we

need not be concerned by the service description here, as this is the role of the maintainer of this service (ie `www.rating.com`), here we just want to *distribute* labels for this service. Given all this, we can write our label file PICS.gen:

```
create the directory root/labels/http/www.rating.com/http/www.w3.org/pub/WWW/

cd root/labels/http/www.rating.com/http/www.w3.org/pub/WWW/
```

And write the following in PICS.gen

```
on="1995.2.29T14:10+0300"
by="bmahe@w3.org"
for="http://www.w3.org/pub/WWW/PICS"
generic=true
ratings=(age 10)
```

The set of attributes for this label can include any of the attributes defined by the PICS labels specification. Our `ratings` attribute here, state that the reader should be at least 10 to be able the PICS specification (this will prevent kids from understanding PICS, so that they can't hack it ;-). All these attributes will be send *as is* (in the appropriate syntax, though) to any clients requesting the `http://www.w3.org/pub/WWW/PICS` generic label.

Setting up a labeled space (serving labels with documents)

Here, we focus on providing labels (by our `www.rating.com` service) for part of our exported space. Let's say we export a `labeled-space` in our top directory *root*/WWW, that contains stuff to be labeled. To handle PICS in this space, we will need to set it up through the PICS filter. Use Jigadmin to hook this filter on `labeled-space`.

Our `labeled-space` is all set, we now need to install some documents in it, and label them. Let's say we have a `hello.html` document in this directory that we want to label with a specific label. The label will conform to our fake `www.rating.com` service, we want the reader to be at least 1 year:

```
create root/labels/http/www.rating.com/http/host/port/labeled-space

cd root/labels/http/www.rating.com/http/host/port/labeled-space
```

And write the following in hello.html

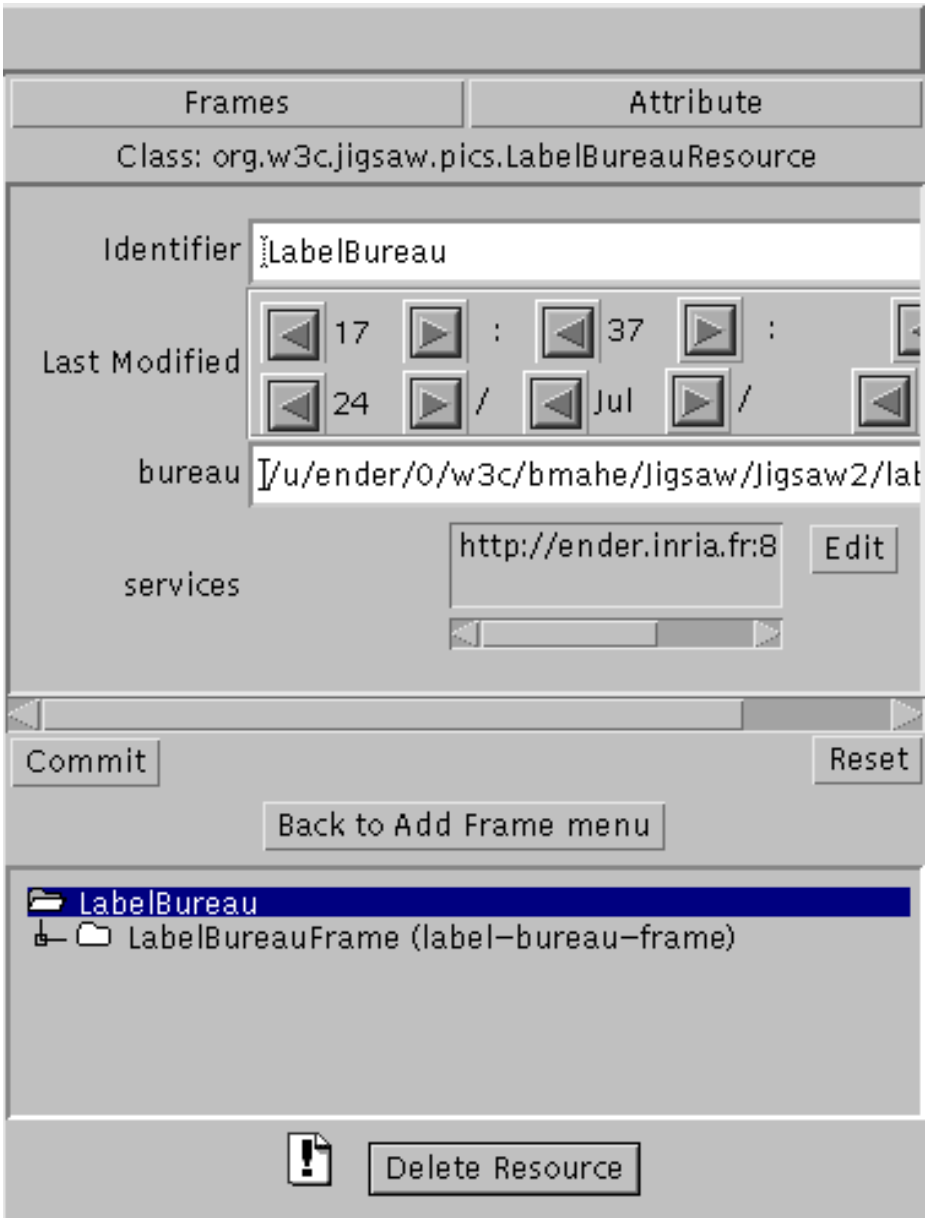
```
generic=false
for="http://host:port/labeled-space/hello.html"
by="bmahe@w3.org"
ratings=(age 1)
```

We can now ask for the labeled document, and we will get the appropriate labels.

Serving labels

Now let's turn **Jigsaw** into a label bureau. You will first need to define the label bureau database, as stated above. The label bureau is implemented as a special resource whose class is org.w3c.jigsaw.pics.LabelBureauResource. The first thing you will need to do is to hook up an instance of this resource in the appropriate place of your

exported space. You can do this with JigAdmin by adding a LabelBureau call it **LabelBureau** (for example). Then go and edit its `bureau` parameter, you're all set.



- **bureau**: The absolute path of the database (described here)
- **services**: List of the services available in this bureau. (This should not be necessary, but some browsers doesn't send the service url in their PICS requests)

Now, the `http://host:port/LabelBureau` will handle the incoming queries for label.

Jigsaw Team

$Id: pics.html,v 1.16 1999/03/16 13:39:44 bmahe Exp $

Browsing zip files with Jigsaw

Jigsaw Home / Documentation Overview

- Introduction
- Configuring the zip indexer
- Configuring the zip file

Introduction

Yes, **Jigsaw** allows you to browse the content of any zip file, that could be done with a set of resources and frames and by configuring a special indexer. Actually, you need to create a ZipDirectoryResource associated to a ZipFrame. This `ZipDirectoryResource` must be associated with a special indexer named `zip-indexer` (org.w3c.jigsaw.zip.ZipIndexer) in this sample configuration.

Configuring the zip indexer

First, read the indexer documentation (if your not aware of that kind of **Jigsaw** feature).

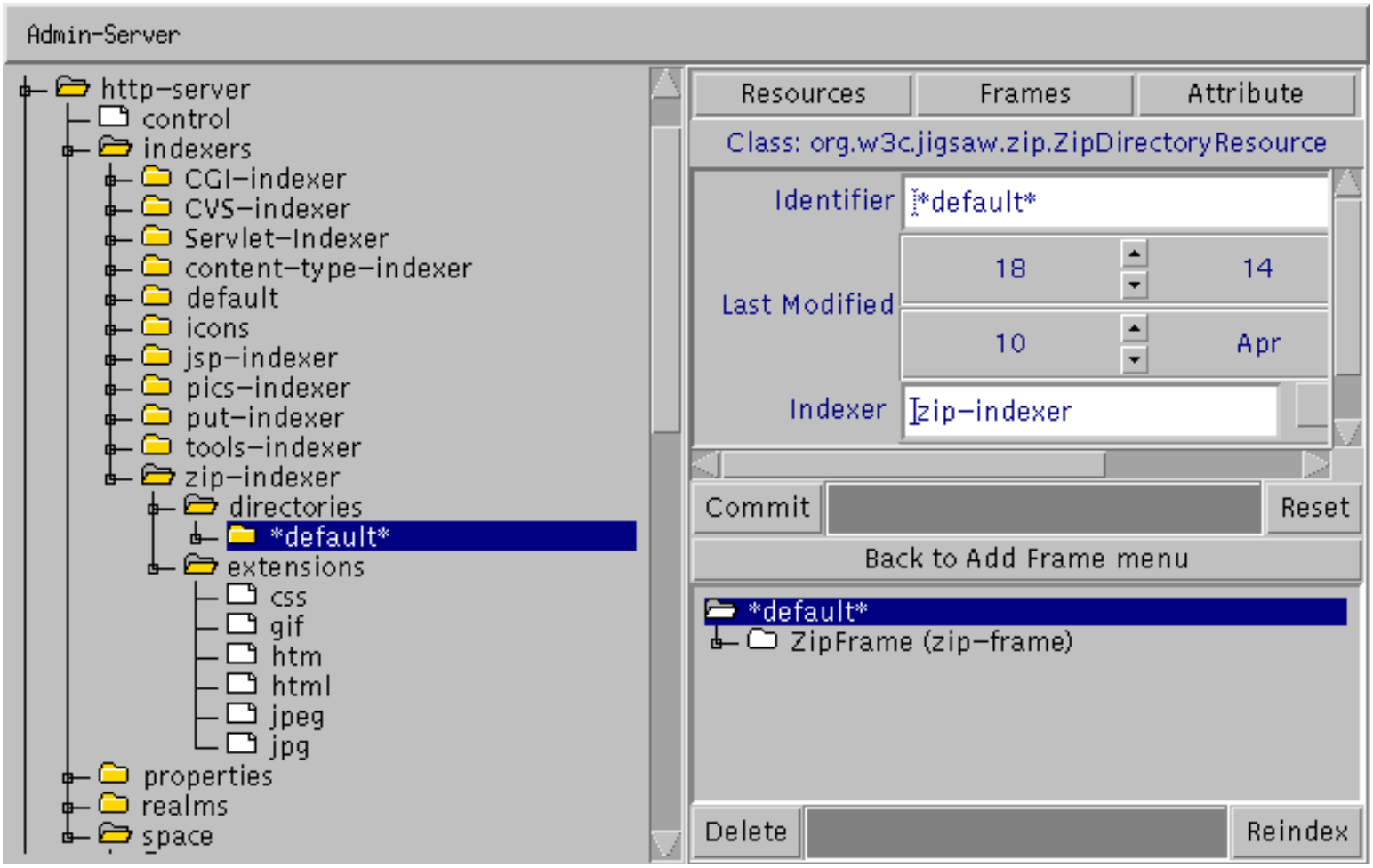
In order to be able to browse a zip file, you need to define the way **Jigsaw** will index the files (or directories) found in this zip file. This could be done by creating a specific indexer. The **ZipIndexer** (`org.w3c.jigsaw.zip.ZipIndexer`) must be configured like any other indexer, except that usuals resources classes must be replaced like that:

- `FileResource => ZipFileResource`
- `DirectoryResource => ZipDirectoryResource`
- `HTTPFrame => ZipFrame`

So, create your `zip-indexer` like in the followings screen shots, save your configuration and go to the following step.

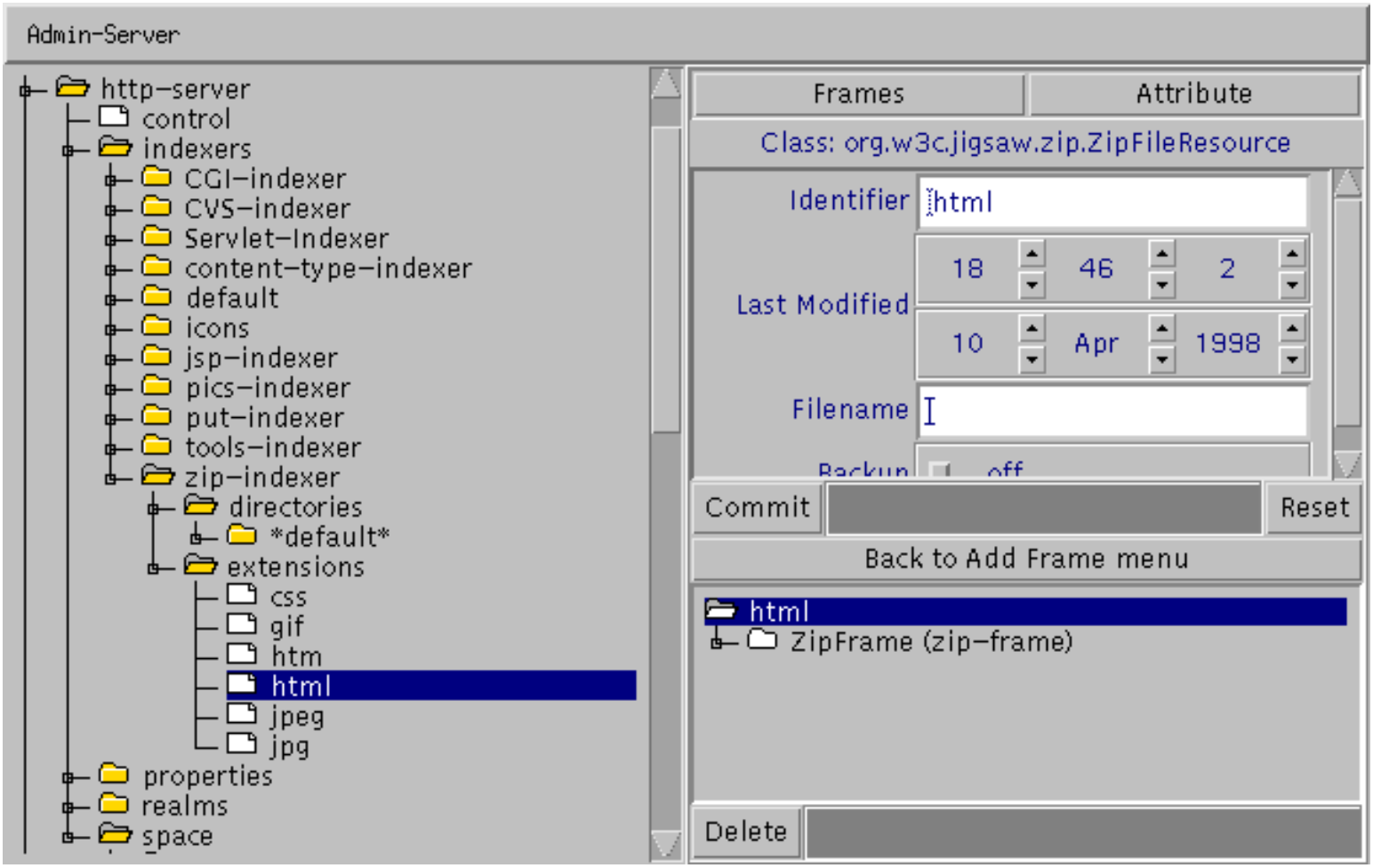
How to index a directory?

A directory found into a zip file should be indexed in a `ZipDirectoryResource` associated to a `ZipFrame`. And its indexer MUST be the `zip-indexer`.



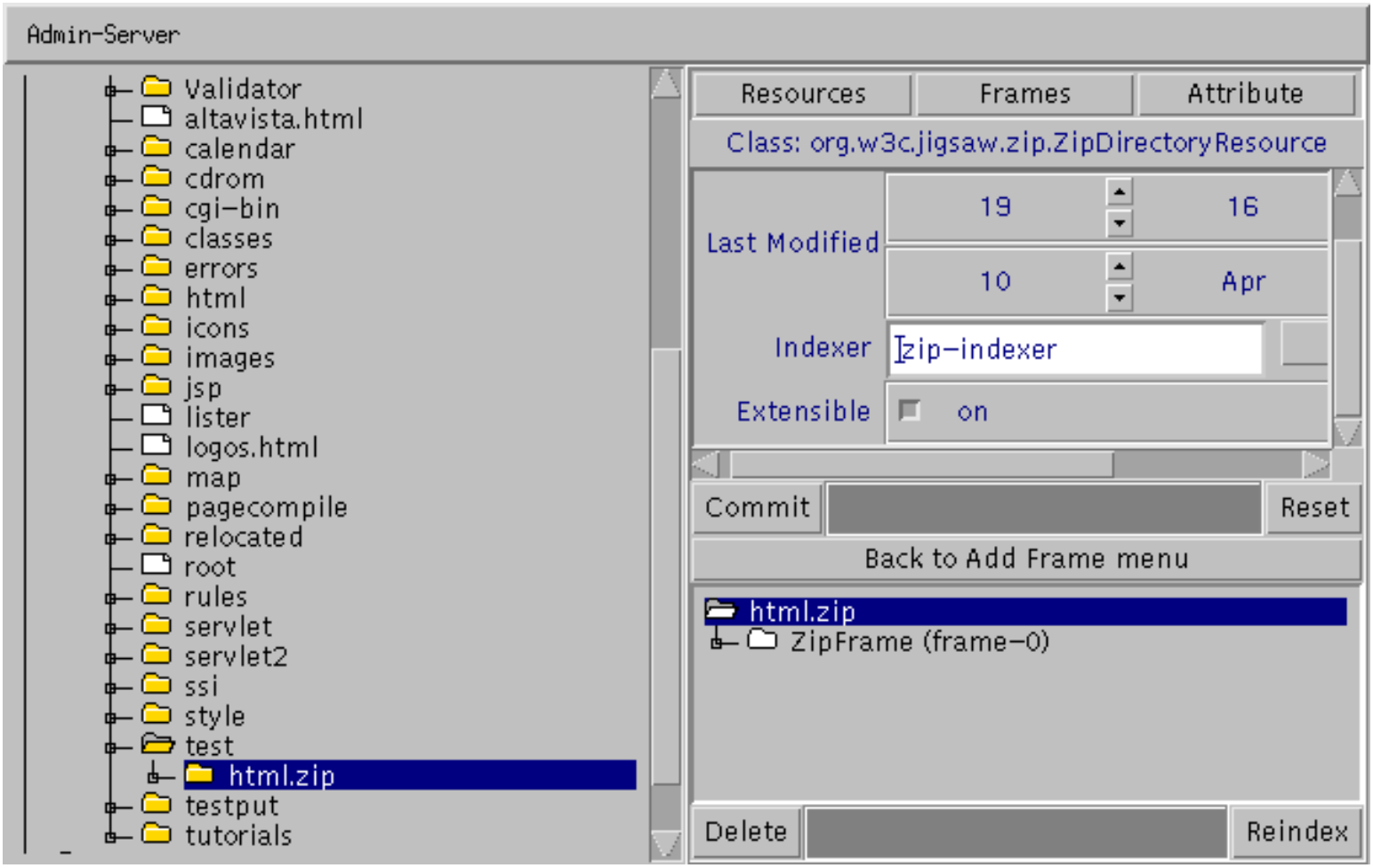
How to index an html file?

An html file must be indexed in a `ZipFileResource` associated to a `ZipFrame`, with a `Content-Type` set to `text/html`.



Configuring the zip file

Your browsable zip file must be indexed (manually or by another indexer) in a `ZipDirectoryResource` associated to a `ZipFrame`. This resource needs your new `zip-indexer`, so edit its attributes and set its indexer to `zip-indexer`. Commit your changes, save the configuration and browse your zip!



This screen shot show you a sample configuration. The file **html.zip** located at `<instdir>/Jigsaw/Jigsaw/WWW/test/html.zip` is reachable (and browsable!) at `http://your-server-host/test/html.zip/` .

Jigsaw Team

$Id: browsezip.html,v 1.15 1999/03/16 13:39:23 bmahe Exp $

All resources All frames

ForwardFrame

This resource is tha base class for the ProxyFrame and the MirrorFrame classes. It uses the w3c HTTP client side API to forward requests to either some other server, or some proxy.

Before configuring the caching part of this resource, you should read the **Jigsaw**'s client API reference manual, which defines all the properties supported.

Inherits

The ForwardFrame class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ProtocolFrame
- HTTPFrame

Attributes description

The ForwardFrame defines the following attributes:

- local-root
- received-by
- trace-request

`local-root`

semantics

The follow up root resource, when the received request is targeted explicitly to the server. This followup root resource, is given by its name, and should already exist in the root resource store.

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

`received-by`

semantics

The String that the proxy will emit as the received-by part of the Via header.

type

This attribute is an editable StringAttribute

default value

This attribute defaults to **null**.

`trace-request`

semantics

Should this frame try to trace how the request has been processed

type

This attribute is an editable BooleanAttribute

default value

This attribute defaults to **null**.

Jigsaw Team

$Id: org.w3c.jigsaw.proxy.ForwardFrame.html,v 1.3 1998/03/27 08:20:53 bmahe Exp $

All resourcesAll frames

MetaDataFrame

Frames used to store metadata (eg: Acl)

Inherits

The MetaDataFrame class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame

Attributes description

The MetaDataFrame doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.tools.resources.MetaDataFrame.html,v 1.1 1999/07/26 14:38:58 bmahe Exp $

All resources All frames

JAcl

If you want to write a new Acl for AclFilter, you must inherits this class.

Inherits

The JAcl class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- MetaDataFrame

Attributes description

The JAcl doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.jigsaw.acl.JAcl.html,v 1.1 1999/07/26 14:25:17 bmahe Exp $

Multiple Choices

The document name you requested (`/Jigsaw/Doc/Reference/org.w3c.jigsaw.pics.PICSFilter.html`) could not be found on this server. However, we found documents with names similar to the one you requested.

Available documents:

- /Jigsaw/Doc/Reference/org.w3c.jigsaw.auth.AuthFilter.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.auth.GenericAuthFilter.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.config.PropertySet.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.cvs.AutoLookupDirectory.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.cvs.CvsFileFrame.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.cvs.CvsFrame.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.cvs.CvsProp.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.cvs.ToolsLister.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.filters.AccessLimitFilter.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.filters.CacheFilter.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.filters.CookieFilter.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.filters.CounterFilter.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.filters.DebugFilter.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.filters.ErrorFilter.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.filters.GZIPFilter.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.filters.GrepPutFilter.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.filters.HeaderFilter.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.filters.LogFilter.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.filters.ProcessFilter.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.filters.PutFilter.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.filters.PutListResource.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.filters.PutSizeFilter.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.filters.SimpleCacheFilter.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.frames.CgiFrame.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.frames.HTTPFrame.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.frames.NegotiatedFrame.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.frames.PostableFrame.html (common basename)

- /Jigsaw/Doc/Reference/org.w3c.jigsaw.frames.RedirecterFrame.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.frames.RelocateFrame.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.frames.VirtualHostFrame.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.http.GeneralProp.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.http.LoggingProp.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.http.UnixProp.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.http.socket.SocketConnectionProp.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.map.MapResource.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.proxy.CacheProp.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.proxy.ForwardFrame.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.proxy.MirrorFrame.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.proxy.ProxyFrame.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.proxy.ProxyProp.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.resources.CheckpointResource.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.resources.DirectoryLister.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.resources.DirectoryListerFrame.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.resources.DirectoryResource.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.resources.HttpDirectoryResource.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.resources.HttpFileResource.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.resources.HttpPassDirectory.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.resources.PassDirectory.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.resources.PasswordEditor.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.resources.PasswordEditorFrame.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.servlet.ServletDirectoryFrame.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.servlet.ServletWrapper.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.servlet.ServletWrapperFrame.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.ssi.SSIFrame.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.status.GcStatFrame.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.status.StatisticsFrame.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.status.ThreadStatFrame.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.tools.resources.AbstractContainer.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.tools.resources.ContainerResource.html (common basename)

- /Jigsaw/Doc/Reference/org.w3c.tools.resources.DirectoryResource.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.tools.resources.FileResource.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.tools.resources.FramedResource.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.tools.resources.PassDirectory.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.tools.resources.ProtocolFrame.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.tools.resources.Resource.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.tools.resources.ResourceFilter.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.tools.resources.ResourceFrame.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.www.protocol.http.HttpManager.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.www.protocol.http.cache.CacheFilter.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.www.protocol.http.cookies.CookieFilter.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.www.protocol.http.icp.ICPFilter.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.www.protocol.http.proxy.ProxyDispatcher.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.cvs.ToolsListerFrame.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.filters.PutListFrame.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.http.ConnectionProp.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.map.MapFrame.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.ssi.commands.DefaultCommandRegistry.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.auth.AuthUser.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.servlet.ServletProps.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.resources.VirtualHostResource.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.filters.UseProxyFilter.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.frames.SeeOtherFrame.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.filters.URISizeLimiterFilter.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.filters.HourLimiterFilter.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.pics.LabelBureauResource.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.filters.TEFilter.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.pics.LabelBureauFrame.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.pagecompile.PageCompileProp.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.pagecompile.PageCompileFrame.html (common basename)

- /Jigsaw/Doc/Reference/org.w3c.jigsaw.zip.ZipDirectoryResource.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.zip.ZipFileResource.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.zip.ZipFrame.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.servlet.RemoteServletWrapper.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.indexer.ContentTypeIndexer.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.servlet.ServletIndexer.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.zip.ZipIndexer.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.tools.resources.ExternalContainer.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.tools.resources.indexer.GhostResourceIndexer.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.tools.resources.indexer.IndexersCatalog.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.tools.resources.indexer.SampleResourceIndexer.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.auth.DigestAuthFilter.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.servlet.ServletMapperFrame.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.tools.resources.MetaDataFrame.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.acl.AclRealm.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.acl.JAcl.html (common basename)
- /Jigsaw/Doc/Reference/org.w3c.jigsaw.acl.AclFilter.html (common basename)

Please consider informing the owner of the referring page about the broken link.

All resources All frames

PropertySet

The resource provides access to a set of properties.

Inherits

The PropertySet class inerits from the following classes:

- Resource

Attributes description

The PropertySet doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.jigsaw.config.PropertySet.html,v 1.2 1998/03/27 08:15:21 bmahe Exp $

All resources All frames

ExternalContainer

...description...

Inherits

The ExternalContainer class inerits from the following classes:

- Resource
- FramedResource
- AbstractContainer
- ContainerResource

Attributes description

The ExternalContainer doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.tools.resources.ExternalContainer.html,v 1.2 1999/09/08 14:58:11 ylafon Exp $

All resources All frames

IndexersCatalog

...description...

Inherits

The IndexersCatalog class inerits from the following classes:

- Resource
- FramedResource
- AbstractContainer
- ContainerResource
- ExternalContainer

Attributes description

The IndexersCatalog doesn't defines any attribute.

Jigsaw Team
$Id: org.w3c.tools.resources.indexer.IndexersCatalog.html,v 1.2 1999/09/08 14:58:11 ylafon Exp $

All resources All frames

DigestAuthFilter

The DigestAuthFilter provides digest authentication mechanism, as described in draft-ietf-http-authentication. It is just a test implementation, and only the password authentication is done, unlike GenericAuthFilter which provides also IP authentication. A new interface will be around soon.

The filter is configured to allow only some users or groups of users to access the information it protects. User are stored in a realm database. If no users are specified, all the users of the realm are allowed.

You should also note that if the filter is set on the protocol frame of a Container, it will protect also all its childs recursively.

Inherits

The DigestAuthFilter class inerits from the following classes:

- Resource
- FramedResource
- ResourceFrame
- ResourceFilter
- AuthFilter

Attributes description

The DigestAuthFilter defines the following attributes:

- users
- groups
- algorithm
- nonce_ttl

`users`

semantics

The list of the users in this realm, that are allowed to access the information protected by this filter

type

This attribute is an editable StringArrayAttribute

default value

This attribute defaults to **null**.

`groups`

semantics

The list of groups allowed to access informations protected by this filter.

type

This attribute is an editable StringArrayAttribute

default value

This attribute defaults to **null**.

`algorithm`

semantics

The algorithm used to encode/decode information, could be SHA, MD5 or some others.

type

This attribute is an editable StringAttribute

default value

This attribute defaults to "MD5".

`nonce_ttl`

semantics

The Time To Live of the nonce (somewhere linked to the challenge). To avoid attack by repetition, you should set it to a low value, but if the value is too low, it may generate too many round trips between the client and the server. The unit is one second.

type

This attribute is an editable IntegerAttribute

default value

This attribute defaults to **300**.

Jigsaw Team

$Id: org.w3c.jigsaw.auth.DigestAuthFilter.html,v 1.1 1999/03/24 09:29:40 bmahe Exp $

Page Compilation in Jigsaw.

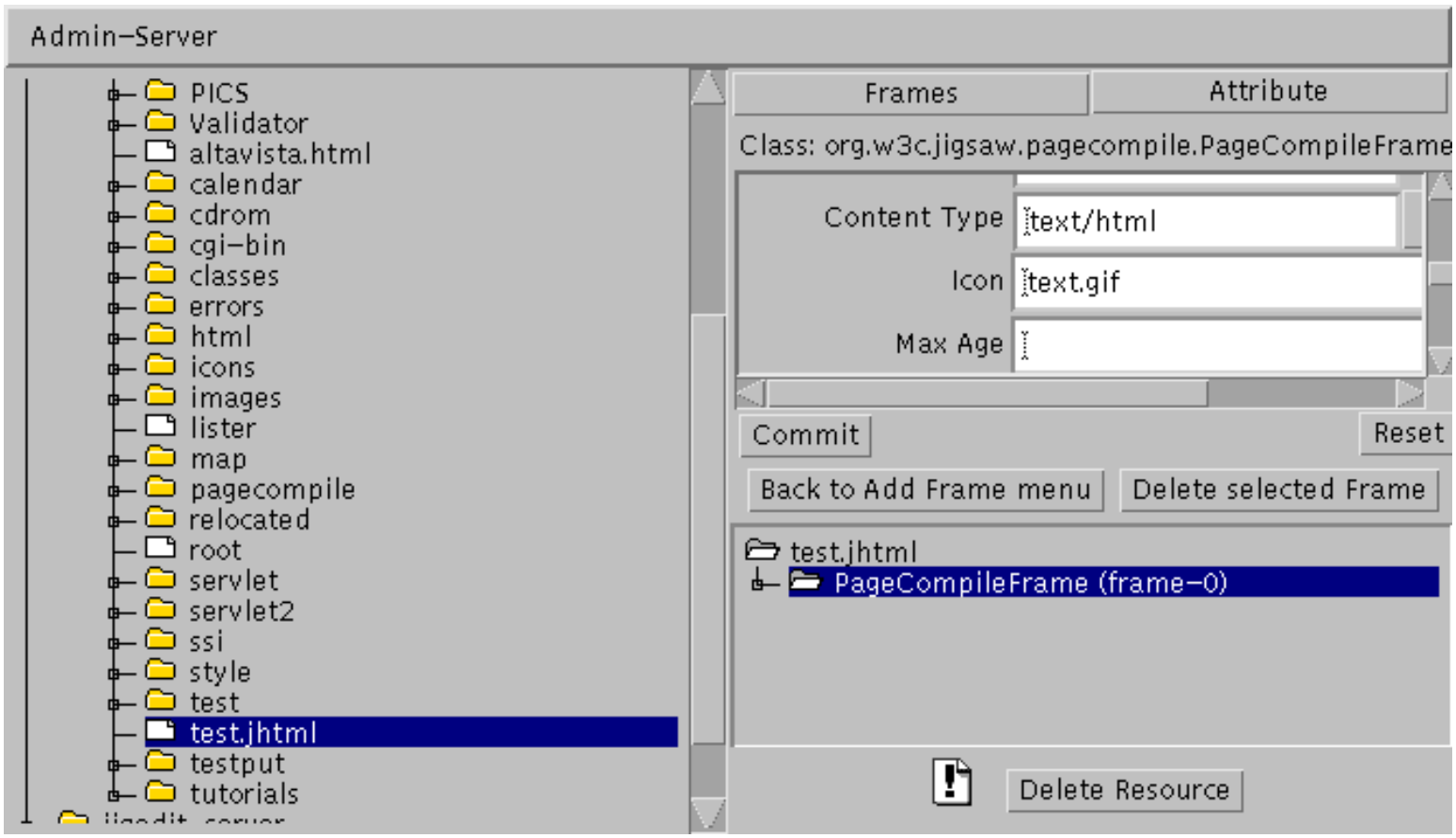
Jigsaw Home / Documentation Overview / Page Compilation Overview

- Configuring a HTML/Java page
- Properties setting
- Note on CLASSPATH

Configuring a HTML/Java page

By default, all files with a **jhtml** extensions are indexed in a FileResource associated to a PageCompileFrame. But if you want to create a HTML/Java page (test.jhtml for example) by hand you should go through the following steps:

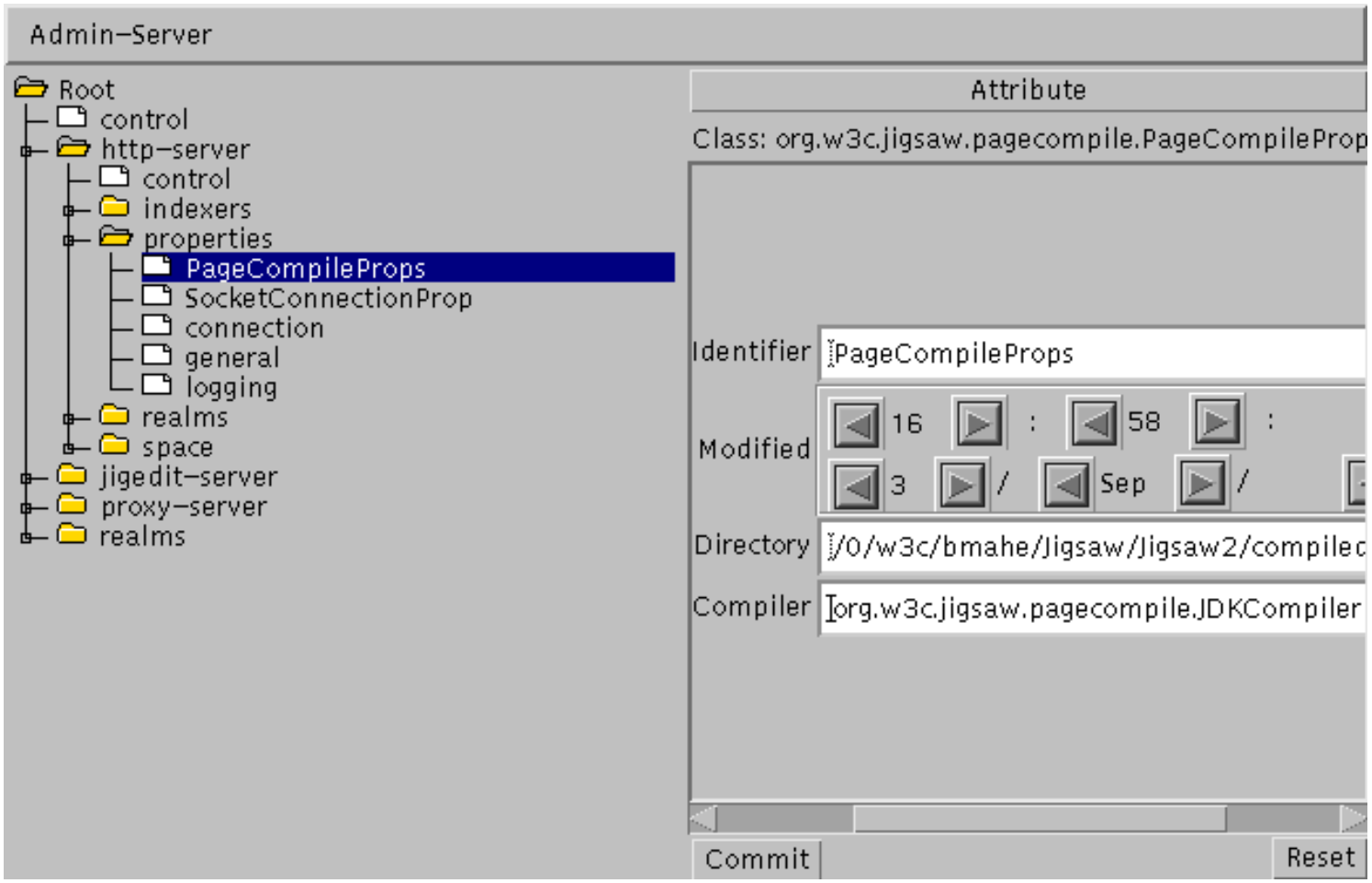
1. Write your HTML/Java page (test.jthml) in `<instdir>/Jigsaw/WWW` (for example).
2. With JigAdmin, create a FileResource called test.jhtml.
3. Select the node corresponding to test.jhtml.
4. Add a `org.w3c.jigsaw.pagecompile.PageCompileFrame` to the test.jhtml FileResource.



Properties

You may need to modify some properties in order to use Page Compilation.

1. **Compiled Pages Directory**: Specify the directory for storing generated class file, by default the Compiled Pages Directory is set to `<instdir>/Jigsaw/Jigsaw/compiledPages/`. **THIS DIRECTORY MUST BE IN YOUR CLASSPATH**, in some cases **Jigsaw** needs to load the generated frames with the system ClassLoader.
2. **Compiler**: Specify the compiler used to compile generated frames, it's not necessary to change the default if you run Jigsaw with the JDK. But in some case you may need to use another compiler, then you will need to write a new compiler class that implements the PageCompiler interface.



Note on CLASSPATH

The `compiledPages` directory must be in the `CLASSPATH` (see Properties), and JDK1.2 users must add the `tools.jar` file from the JDK (`lib/tools.jar`) in the `CLASSPATH` too because the default PageCompiler needs the `sun.tools.javac.Main` class which is no more in classes.zip.

Jigsaw Team

$Id: configuration.html,v 1.12 1999/09/08 14:58:11 ylafon Exp $

Page Compilation in Jigsaw.

Writing your first page

Please notice that what is discussed below is not part of any W3C specification! It is server-side only.

First, choose a place to put your page, for example `<instdir>/Jigsaw/Jigsaw/WWW/MyPage.jhtml`.

Then put the following in MyPage.jhtml.

```
<html>
  <head><title>My Page</title></head>
  <body>
    <h1>My Page</h1>

    <java>
      int age = 42;
    </java>

    <h2>A Print</h2>
    <java>
      out.println ("age : "+age);
    </java>

    <h2>Another Print</h2>
    age : <java type=print>age</java>

    <h2>A Test</h2>
    <java>
      if (age > 100) {
      out.println("more than 100");
      } else {
      out.println("less than 100");
      }
    </java>

    <h2>Another Test</h2>
    <java> if (age > 100) {</java>
    more than 100
    <java> } else { </java>
    less than 100
    <java> } </java>

    <h2>A Loop</h2>
    <java>
      for (int i = 0; i < 80; i++) {
      out.println("*");
      }
    </java>

    <h2>Another Loop</h2>
    <java>for (int i = 0; i < 80; i++) {</java>
    *
```

```
    <java>}</java>

  </body>
</html>
```

The first time you will access this page, the PageCompileFrame will generate a Frame. By default the frame generated extends GeneratedFrame, but it can be a subclass of GeneratedFrame, see the documentation for <java type=extends>. The source code of this frame will looks like:

```
public class MyPage extends org.w3c.jigsaw.pagecompile.GeneratedFrame  {

    protected void get(org.w3c.jigsaw.http.Request request,
                       org.w3c.jigsaw.http.Reply reply,
                       org.w3c.jigsaw.pagecompile.PageCompileOutputStream out)
        throws java.io.IOException
    {
        org.w3c.jigsaw.pagecompile.PageCompileFile _file =
            new
org.w3c.jigsaw.pagecompile.PageCompileFile("/0/w3c/bmahe/Jigsaw/Jigsaw2/WWW/MyPage.jhtml");
        _file.writeBytes(0,79,out);

        int age = 42;

        _file.writeBytes(118,144,out);
        out.println ("age : "+age);

        _file.writeBytes(198,240,out);
        out.println(String.valueOf(age));

        _file.writeBytes(268,297,out);

        if (age > 100) {
            out.println("more than 100");
        } else {
            out.println("less than 100");
        }

        _file.writeBytes(434,469,out);

        if (age > 100) {
            _file.writeBytes(500,522,out);
        } else {
            _file.writeBytes(546,568,out);
        }

        _file.writeBytes(585,614,out);

        for (int i = 0; i < 80; i++) {
            out.println("*");
        }

        _file.writeBytes(702,737,out);

        for (int i = 0; i < 80; i++) {
            _file.writeBytes(781,791,out);
        }

        _file.writeBytes(806,830,out);
    }
}
```

And the page displayed on your browser will looks like:

My Page

A Print

age : 42

Another Print

age : 42

A Test

less than 100

Another Test

less than 100

A Loop

****

Another Loop

****

Handling the POST method

The GeneratedFrame inherits from PostableFrame, so it's really easy to handle the `POST` method with Page Compilation. The following example get the user name from the client and display it. Actually, you just need to override the post method of GeneratedFrame. If you want `GET` requests with query string to be converted in `POST` requests, override the method getConvertGetFlag like in the example.

```
<html>
  <java type=import>
    import java.io.IOException;

    import org.w3c.jigsaw.http.*;
    import org.w3c.jigsaw.pagecompile.*;
    import org.w3c.jigsaw.forms.URLDecoder;
  </java>
  <java type=class>
    protected String NAME_P = "name";

    //Get the 'convert GET to POST' flag.
    public boolean getConvertGetFlag() {
        return true;
    }

    //handle the post method:
    public void post (Request request,
                      Reply reply,
                      URLDecoder data,
                      PageCompileOutputStream out)
        throws IOException
    {
        String name = data.getValue(NAME_P);
```

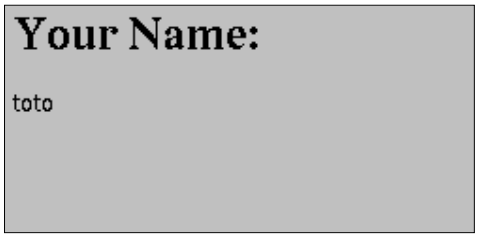
```
        out.println("<h1>Your Name:</h1>");
        out.println(name);
    }
  </java>
  <head>
    <title>Post test</title>
  </head>
  <body>
    <h1>Post test</h1>
    Enter your name:
    <form method="POST">
      <input type="text" name="<java type=print>NAME_P</java>">
      <input type="submit">
    </form>
  </body>
</html>
```

So, the generated java file will looks like:

```
package pagecompile;

import java.io.IOException;

import org.w3c.jigsaw.pagecompile.*;
import org.w3c.jigsaw.http.*;
import org.w3c.jigsaw.forms.URLDecoder;

public class post extends org.w3c.jigsaw.pagecompile.GeneratedFrame  {


    protected String NAME_P = "name";

    //Get the 'convert GET to POST' flag.
    public boolean getConvertGetFlag() {
        return true;
    }

    //handle the post method:
    public void post (Request request,
                      Reply reply,
                      URLDecoder data,
                      PageCompileOutputStream out)
        throws IOException
    {
        String name = data.getValue(NAME_P);
        out.println("<h1>Your Name:</h1>");
        out.println(name);
    }


    protected void get(org.w3c.jigsaw.http.Request request,
                       org.w3c.jigsaw.http.Reply reply,
                       org.w3c.jigsaw.pagecompile.PageCompileOutputStream out)
        throws java.io.IOException
    {
        org.w3c.jigsaw.pagecompile.PageCompileFile _file =
            new
org.w3c.jigsaw.pagecompile.PageCompileFile("/0/w3c/bmahe/Jigsaw/Jigsaw2/WWW/pagecompile/post.jhtml");
        _file.writeBytes(0,6,out);
        _file.writeBytes(73,73,out);
        _file.writeBytes(507,634,out);
        out.print(String.valueOf(NAME_P));
        _file.writeBytes(665,715,out);
    }
}
```

And the page displayed on your browser will looks like:

After a click on "Submit Query":



NOTE: If you don't override the `post` method, the generated frame will return a "`Method POST not allowed`" reply on every `POST` request.

Debugging

The generated java file could be incorrect, for example the following examples emit a **compilation error** (missing '}').

```
<html>
  <head><title>Compilation Error</title></head>
  <body>
    <h1>Compilation error</h1>

    <h2>A Loop</h2>
    <java>
      for (int i = 0; i < 80; i++) {
      out.println("*");
    </java>

  </body>
</html>
```

Then Jigsaw will send the following reply to your browser when you will try to access your page.

```
/0/w3c/bmahe/Jigsaw/Jigsaw2/compiledPages/CompileError.java:16: '}' expected.
}
 ^
1 error
```

Your generated page could also produce some **runtime errors**, for example the following page will produce a ArrayIndexOutOfBoundsException:

```
<html>
  <head><title>Runtime Error</title></head>
  <body>
    <h1>Runtime error</h1>

    <java>
      int tab[] = {1,2,3,4,5};
    </java>

    <h2>A Loop</h2>
    <java>
      for (int i = 0; i < 6; i++) {
        out.print(tab[i]);
      }
```

```
    </java>

  </body>
</html>
```

And the reply sent will be:

```
The generated frame at

http://ender.w3.org/RuntimeError.jhtml

reported this exception :

5

Stack trace :

java.lang.ArrayIndexOutOfBoundsException: 5
        at RuntimeError.get(RuntimeError.java:16)
        at org.w3c.jigsaw.pagecompile.GeneratedFrame.get(GeneratedFrame.java:34)
        at org.w3c.jigsaw.frames.HTTPFrame.perform(HTTPFrame.java:1281)
        at
org.w3c.tools.resources.FramedResource.performFrames(FramedResource.java:581)
        at org.w3c.jigsaw.frames.HTTPFrame.performFrames(HTTPFrame.java:1256)
        at org.w3c.tools.resources.FramedResource.perform(FramedResource.java:599)
        at org.w3c.tools.resources.ResourceFrame.perform(ResourceFrame.java:108)
        at org.w3c.jigsaw.frames.HTTPFrame.perform(HTTPFrame.java:1269)
        at
org.w3c.jigsaw.pagecompile.PageCompileFrame.perform(PageCompileFrame.java:662)
        at org.w3c.jigsaw.http.httpd.perform(httpd.java:1538)
        at org.w3c.jigsaw.http.Client.processRequest(Client.java:384)
        at org.w3c.jigsaw.http.Client.startConnection(Client.java:459)
        at org.w3c.jigsaw.http.socket.SocketClient.run(SocketClient.java:114)
        at org.w3c.util.CachedThread.run(ThreadCache.java:86)
```

Reading the generated java file will help you to find where the error is in the original HTML/Java file. This file is located in the Compiled Pages Directory (on any subdirectory). The generated java file could be in a package, for example if the URL of your HTML/Java file is `/pagecompile/test/Page.jhtml` the package is `pagecompile.test` and the class name is `Page`. So the generated java file is `<Compiled Pages Directory>/pagecompile/test/Page.java` .

Jigsaw Team

$Id: writing.html,v 1.36 1999/09/08 14:58:11 ylafon Exp $

Page Compilation in Jigsaw.

Jigsaw Home / Documentation Overview / Page Compilation Overview

- List of tags
- Variables availables

Note that this is server-side only and this may be not available on all the servers around.

List of tags:

<java> </java> or **<java type=code> </java>**

Define java code for the get method.

<java type=import> </java>

Define one or more import statement.

Ex: **<java type=inport>**import java.net.*;**</java>**

<java type=extends> </java>

Specify the name of the class to extends the GeneratedFrame from.

Ex: **<java type=extends>**MyOwnFrame**</java>**

<java type=implements> </java>

Specify a list of interfaces the GeneratedFrame will implements.

Ex: **<java type=implements>**MyInterface1, MyInterface2 **</java>**

<java type=class> </java>

Define some member variables or some method for the GeneratedFrame.

Ex: **<java type=class>**int counter = 0;**</java>**

<java type=print> </java>

Enclose a Java expression to be sent to the PageCompileOutputStream.

Ex: **<java type=print>**new Date()**</java>**

Variables availables:

request [Request]

The incomming request.

reply [Reply]

The HTTP reply.

out [PageCompileOutputStream]

The reply OutputStream.

Jigsaw Team

$Id: reference.html,v 1.14 1999/09/08 14:58:11 ylafon Exp $

Redirections

Jigsaw Home / Documentation Overview

Here we describe (with examples) what kind of redirection **Jigsaw** is able to perform and how to configure those redirections.

This document has the following sections:

HTTP redirections

When a document has moved, for example `http://host1/Doc.html` is now reachable at `http://host2/Doc.html`, you should use the RelocateFrame.

With JigAdmin, remove the HTTPFrame of Doc1.html (the one at `http://host1/Doc.html`) and add a RelocateFrame, then set the **location** field to `http://host2/Doc.html`. Now, this frame will emit an HTTP redirect reply for every requests. Also, it can be configured to handle the PATH info so that you can remap http://host1/foo/(.*) to http://host2/bar/$1.

Internal redirections

Sometimes, you need to perform internal redirections. For example, if you want requests on all documents with a `foo` extension to be redirected to a cgi script you will need to use the RedirecterFrame. Each document (with foo extension) must be indexed to a FramedResource associated to a RedirecterFrame (read the indexers documentation for more details) and the **target** field of the RedirecterFrame must be set to the cgi script url (eg: /cgi-bin/fooscript). It operates "behind the scene" as the client doesn't get a redirect (unless the target resource generate one). As of avril 29th 1999, it handles PATH_INFO.

Servlet mapping

If you want to map an extension to a specific servlet, for example gnujsp servlet with jsp extension (FAQ), you need to perform an internal redirection that take care of the "path translated". The ServletMapperFrame is what you need. Configure it in the same way that the RedirecterFrame.

Note: this frame doesn't handle pathinfo (it doesn't make sense).

Jigsaw Team

$Id: redirection.html,v 1.4 1999/04/29 09:05:00 bmahe Exp $

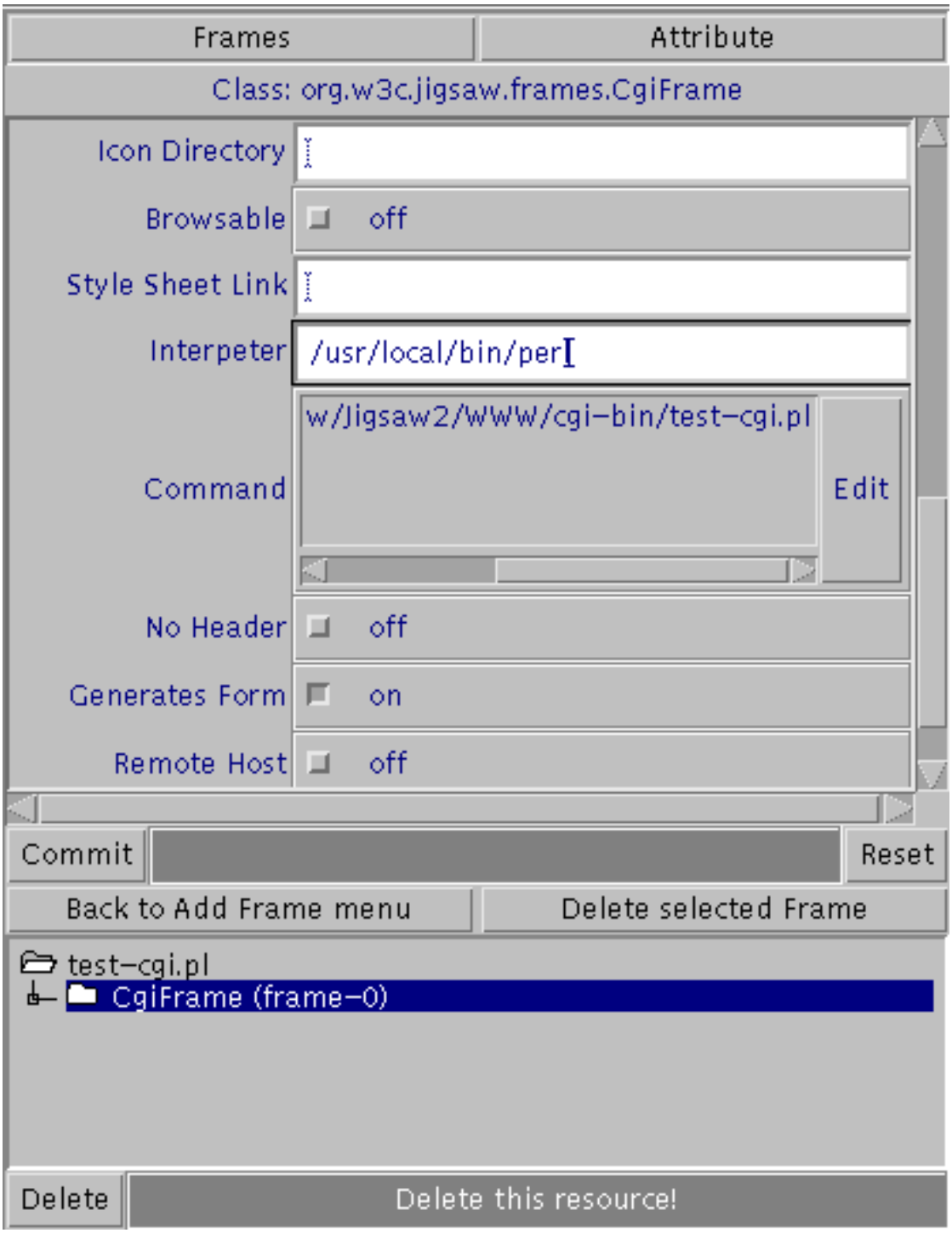
CGI scripts

Jigsaw Home / Documentation Overview

They are two ways to setup **CGI** scripts. The manual way requires that you describe each script to the server. Let's say your script's path relative to the server root is `WWW/cgi-bin/myscript`. You will first have to create an appropriate org.w3c.tools.resources.FileResource with a org.w3c.jigsaw.frames.CgiFrame instance to wrap your script. See this tutorials to know how to create a resource in **Jigsaw**.

Then edit the newly created resource, and setup it's command line (the command line the server will use to run your script). Each line of the text field should represent one argument, the first one being the script full path.

You can also register files of a given extension as scripts, by using a specialized indexer. When required, you can even specify the interpreter to be run to execute the script (for example perl scripts).



In this sample the CGI script is a perl script, as you can see the interpreter location must be specified (in the `Interpreter` field). Read the CgiFrame reference page for more details on CgiFrame configuration.

Note: Some CGI scripts may have problems if the "current directory" is not set, it is something we can't do in Java, so you may have to wrap your script in another one that set the current directory (by using the **cd** command).

`wrapper.bat:`

```
D:
cd c:\foo\bar
mycgi.exe
```

Jigsaw Team

$Id: cgi.html,v 1.10 1999/04/23 15:13:49 bmahe Exp $

Publication in Jigsaw

Jigsaw Home / Documentation Overview

This page describes how to setup an editable space under **Jigsaw**. Today most of the authoring tools allows you to use the PUT method of HTTP to publish a document on a web server.

In the following configuration, the editable URL will be `http://your-server-host/publications`.

The tutorial will go through the following steps:

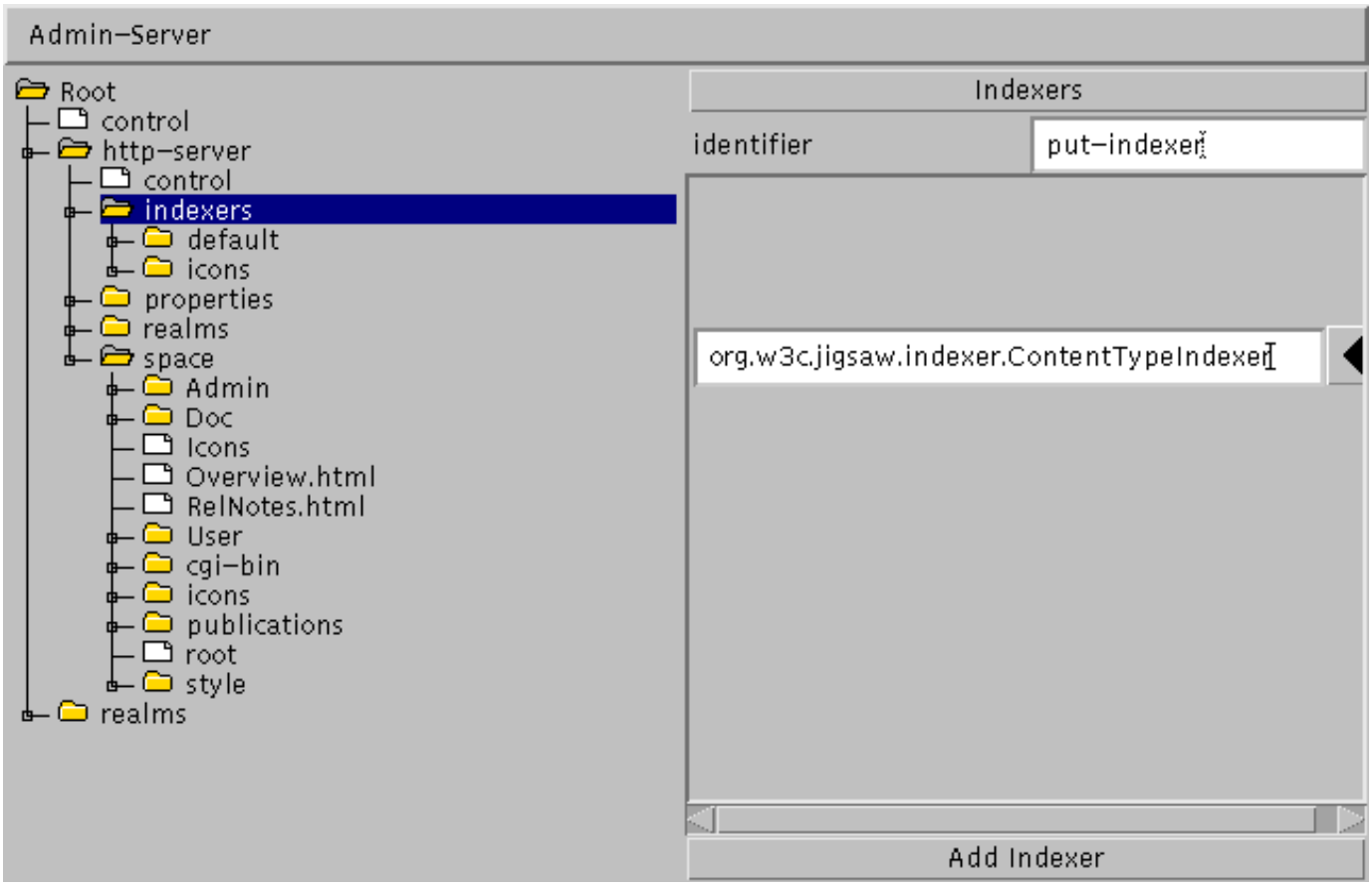
1. Configuring a PUT indexer
2. Configuring the editable space

Configuring a PUT indexer

NOTE: Since Jigsaw2.0beta3, there is a `put-indexer` already configured in the distribution.

First, read the indexer documentation. When you are more familiar with indexers you can go through the following steps:

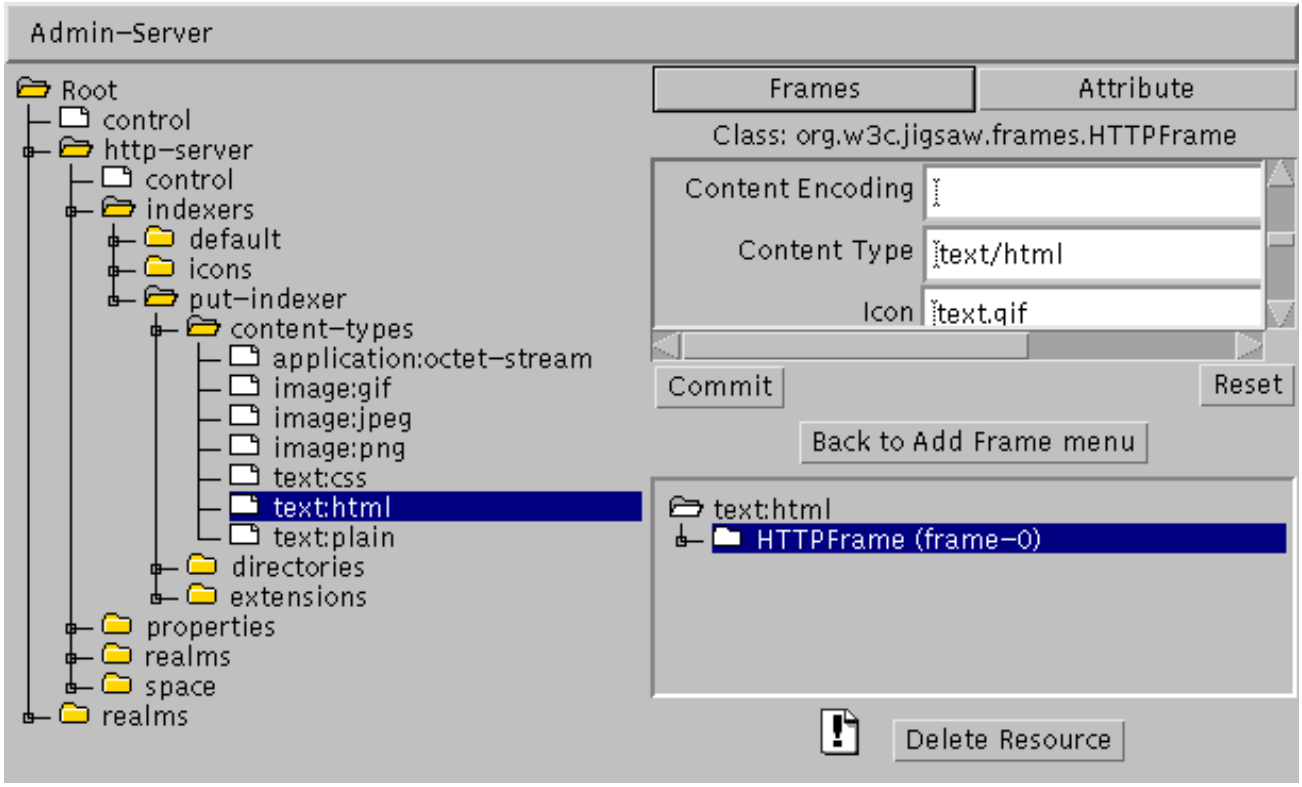
1) Create the indexer, called it `put-indexer` for example. In this case the ContentTypeIndexer is the most appropriate indexer class to choose.



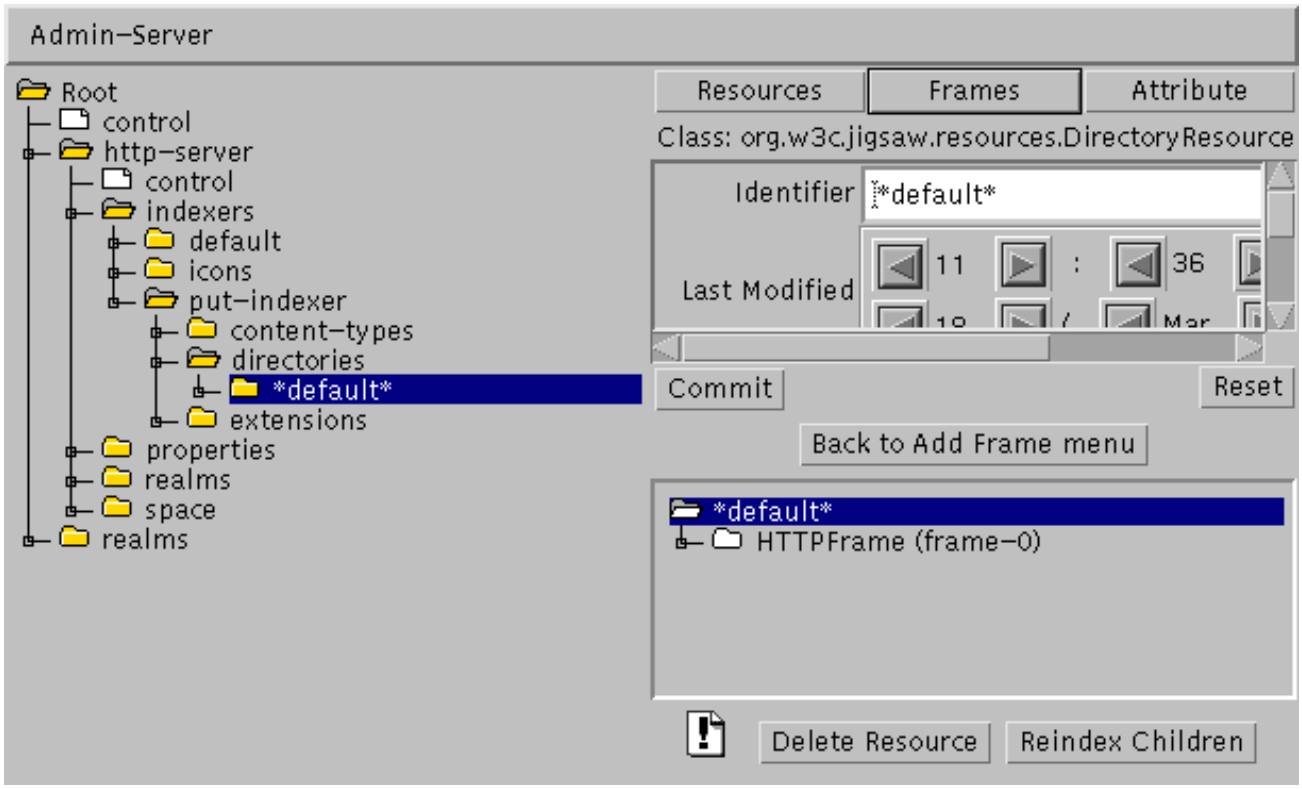
2) The `put-indexer` has three nodes: `content-types`, `directories` and `extensions`.

- `content-types` define some relations between mime types and resources. This is used when the PUT request has a `Content-Type` Header. If the `Content-Type` header is `text/html` then the resource called `text:html` will be the

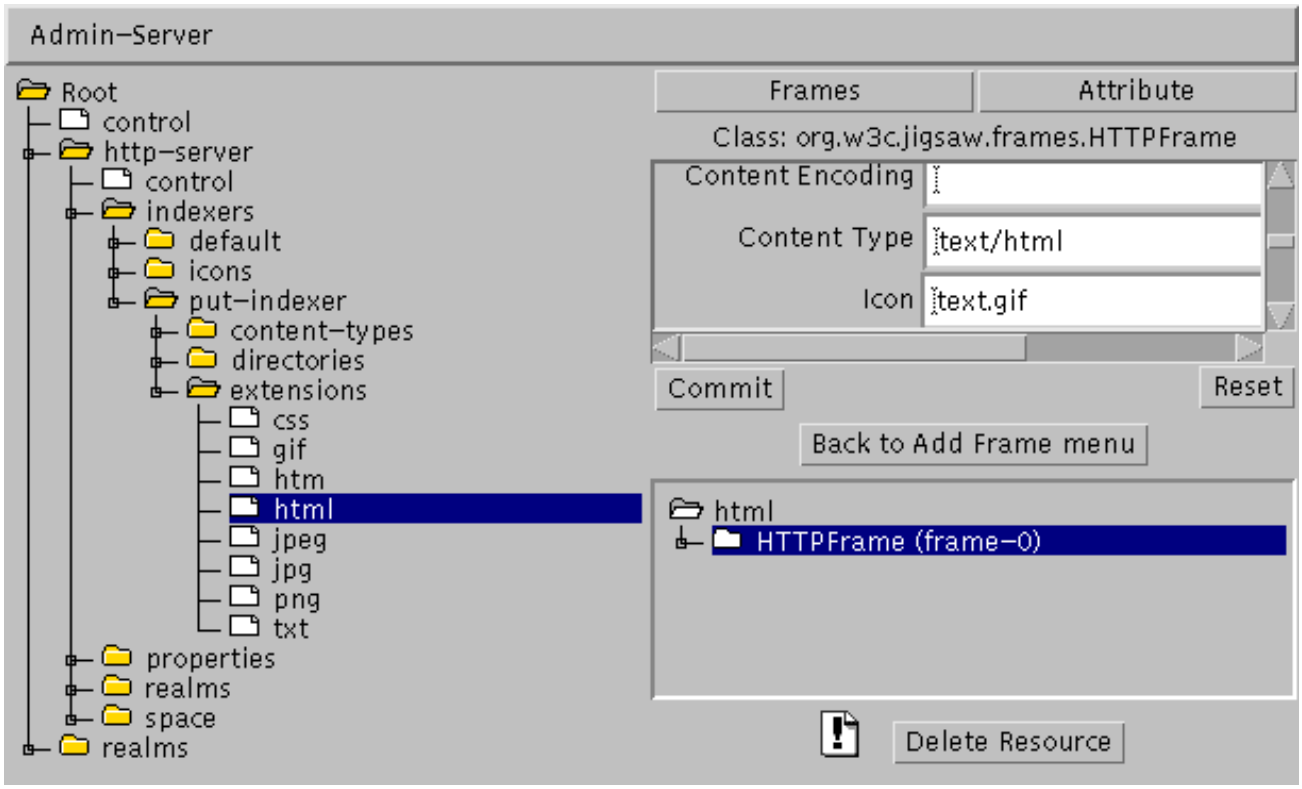
template to use for the new created resource. The template resource `text:html` must be a FileResource associated to an HTTPFrame with its `Putable` flag set to **on** and of course a `Content Type` set to **text/html**.



- `directories` is used to index files matching exactly a name, mainly used to index directories. In this configuration every subdirectories will be indexed as a DirectoryResource associated to an HTTPFrame with its `Putable` flag set to **on**.



- `extensions` is used to index files with a specific extension. For example, "html" is a FileResource with an HTTPFrame set to give the "text/html" content type to this file (`Putable` flag set to **on**).



NOTE: Don't forget to set all `Putable` flag to **on** (in associated HTTPFrame).

3) Save your configuration, your new `put-indexer` is ready.

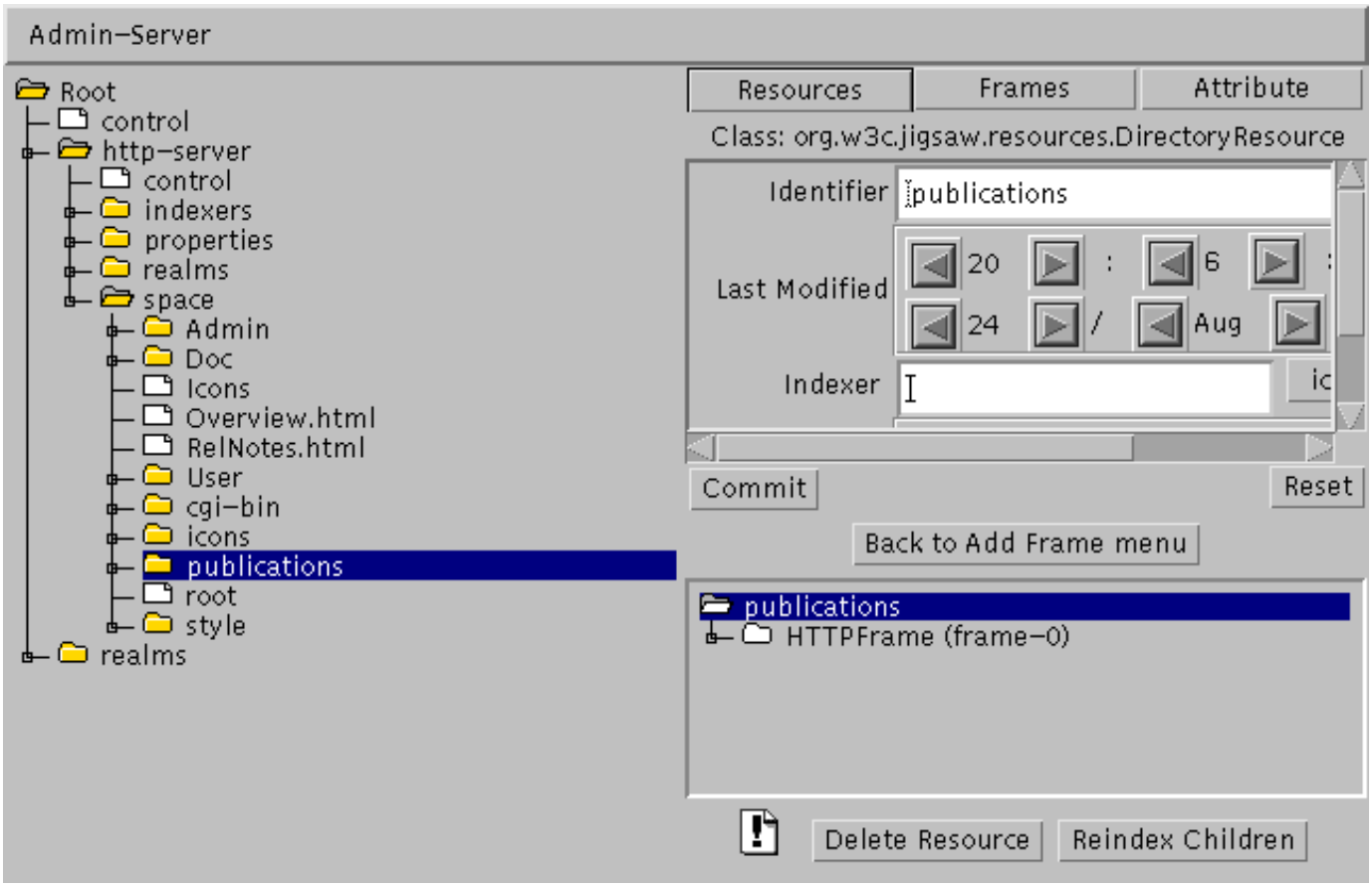
Configuring the editable space

We choose `publications` as our editable directory.

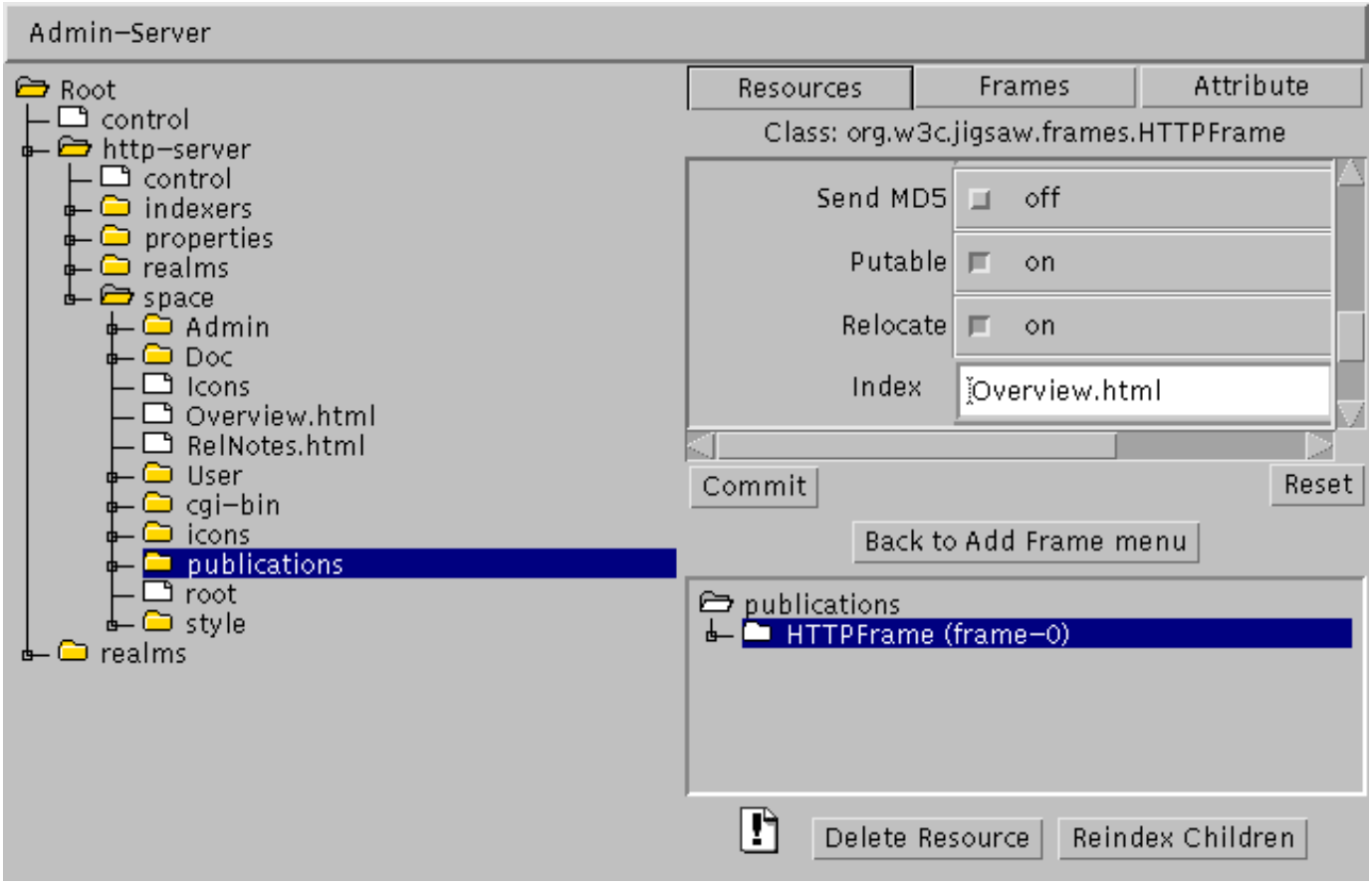
1) Create the directory `<instdir>/Jigsaw/Jigsaw/WWW/publications.`

2) Open a JigAdmin window (see JigAdmin documentation).

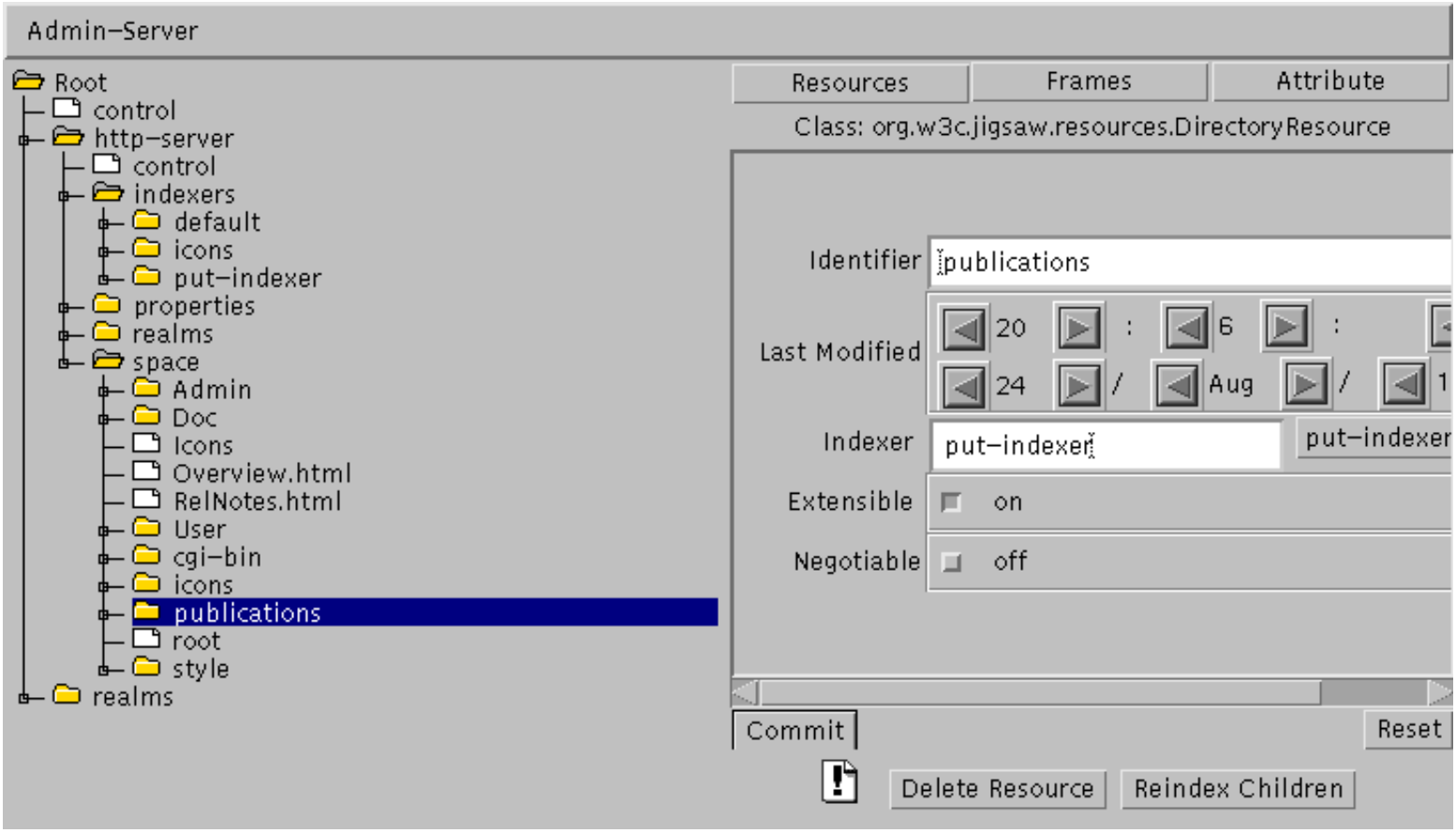
3) Create the `publications` resource. It should be created automatically by the `default` indexer, it should be a DirectoryResource associated to an HTTPFrame (but you can use a PassDirectory with an HTTPFrame).



4) Set the putable flag of the HTTPFrame associated to `publications` to **on**. Don't forget to commit your changes.



5) Set the `put-indexer` as the current indexer. Commit your changes.



6) Save your configuration, now you can PUT some new documents (and some new directories) in `http://your-server-host/publications/` , enjoy!

Jigsaw Team

$Id: publication.html,v 1.20 1999/03/16 13:39:48 bmahe Exp $

JigXML

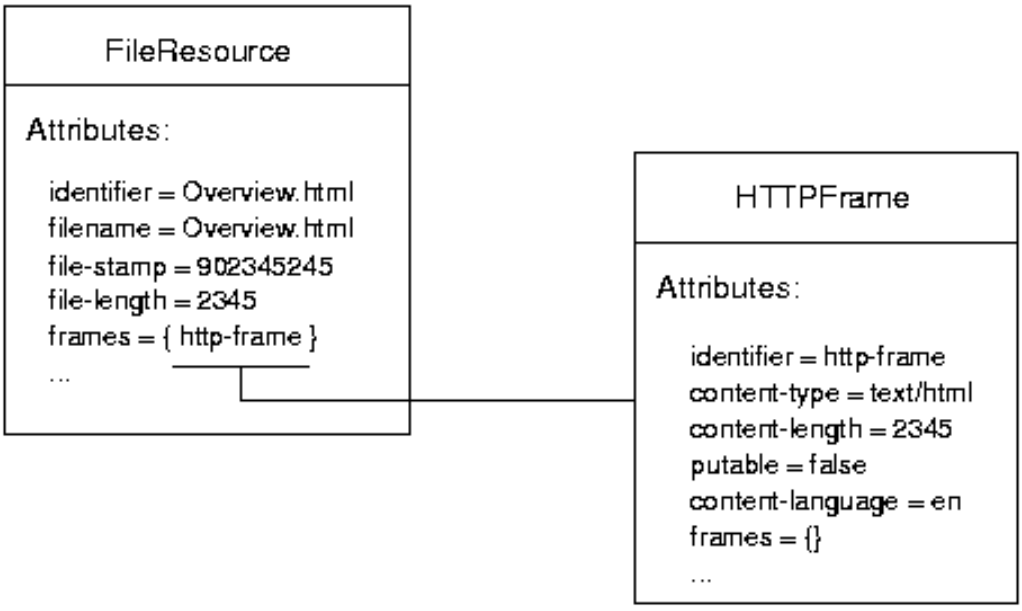
The Jigsaw XML format

Jigsaw Home / Documentation Overview

This document describe the format used by Jigsaw to store the resources metadata. As of 2.1.0, Jigsaw use XML to store these data. A JigXML file describes a set of resources metadata, each container (eg: a directory) save its children's metadata in such a file (via the ResourceStoreManager). It is assumed that you are familiar with the Jigsaw architecture.

Resources and Frames in JigXML

Here is the structure of a resource and its frames:



And here is the corresponding JigXML structure:

```
<jigxml version="1.0" xmlns="http://jigsaw.w3.org/JigXML/JigXML1.0">
  <resource class='org.w3c.tools.resources.FileResource'>
    <attribute name='identifier' flag='6'
```

```
class='org.w3c.tools.resources.StringAttribute'>Overview.html</attribute>
    <resourcearray name='frames' class='org.w3c.tools.resources.FrameArrayAttribute'
length='1'>
      <resource class='org.w3c.jigsaw.frames.HTTPFrame'>
        <attribute name='identifier' flag='6'
class='org.w3c.tools.resources.StringAttribute'>http-frame</attribute>
        <resourcearray name='frames'
class='org.w3c.tools.resources.FrameArrayAttribute' length='0'>
        </resourcearray>
        <attribute name='last-modified' flag='3'
class='org.w3c.tools.resources.DateAttribute'>932126901168</attribute>
        <attribute name='oid' flag='1'
class='org.w3c.tools.resources.IntegerAttribute'>@@NULL@@</attribute>
        <attribute name='quality' flag='2'
class='org.w3c.tools.resources.DoubleAttribute'>1.0</attribute>
        <attribute name='title' flag='2'
class='org.w3c.tools.resources.StringAttribute'>The release notes</attribute>
        <attribute name='content-language' flag='2'
class='org.w3c.jigsaw.frames.LanguageAttribute'>en</attribute>
        <attribute name='content-encoding' flag='2'
class='org.w3c.jigsaw.frames.EncodingAttribute'>@@NULL@@</attribute>
        <attribute name='content-type' flag='2'
class='org.w3c.jigsaw.frames.MimeTypeAttribute'>text/html</attribute>
        <attribute name='content-length' flag='1'
class='org.w3c.tools.resources.IntegerAttribute'>2345</attribute>
        <attribute name='icon' flag='2'
class='org.w3c.tools.resources.StringAttribute'>text.gif</attribute>
        <attribute name='maxage' flag='2'
class='org.w3c.tools.resources.LongAttribute'>@@NULL@@</attribute>
        <attribute name='send-md5' flag='2'
class='org.w3c.tools.resources.BooleanAttribute'>false</attribute>
        <attribute name='allow-delete' flag='2'
class='org.w3c.tools.resources.BooleanAttribute'>false</attribute>
        <attribute name='putable' flag='2'
class='org.w3c.tools.resources.BooleanAttribute'>false</attribute>
        <attribute name='relocate' flag='2'
class='org.w3c.tools.resources.BooleanAttribute'>true</attribute>
        <attribute name='index' flag='2'
class='org.w3c.tools.resources.StringAttribute'>@@NULL@@</attribute>
        <attribute name='icondir' flag='2'
class='org.w3c.tools.resources.StringAttribute'>/icons</attribute>
        <attribute name='browsable' flag='2'
class='org.w3c.tools.resources.BooleanAttribute'>false</attribute>
        <attribute name='style-sheet-link' flag='2'
class='org.w3c.tools.resources.StringAttribute'>/style/directory.css</attribute>
      </resource>
    </resourcearray>
    <attribute name='last-modified' flag='3'
class='org.w3c.tools.resources.DateAttribute'>932126925406</attribute>
    <attribute name='oid' flag='1'
class='org.w3c.tools.resources.IntegerAttribute'>2097344102</attribute>
    <attribute name='filename' flag='2'
class='org.w3c.tools.resources.FilenameAttribute'>Overview.html</attribute>
    <attribute name='file-stamp' flag='1'
```

```
class='org.w3c.tools.resources.DateAttribute'>922721334000</attribute>
    <attribute name='file-length' flag='1'
class='org.w3c.tools.resources.IntegerAttribute'>2345</attribute>
    <attribute name='backup' flag='2'
class='org.w3c.tools.resources.BooleanAttribute'>false</attribute>
  </resource>
</jigxml>
```

This sample describe the metadata of only one resource: "`Overview.html`". A resource is a set of attributes, each attribute is described by an `<attribute>` tag. The frames of a resource are described in a `<resourcearray>` tag, in this example the resource has only one frame (an HTTPFrame), and this frame is called "`http-frame`". A frame is a resource, so it's also a set of attributes.

How to find the JigXML description of a resource?

The usual resources are stored in a repository (a file) located in the `<INSTDIR>/Jigsaw/Jigsaw/config/stores` directory. A repository is used to store the children's configurations of one (and only one) container (eg: DirectoryResource). For example, the children's configurations of the root resource are stored in the file `<INSTDIR>/Jigsaw/Jigsaw/config/stores/root.xml`.

The associations between containers and repositories are stored in the file `stores/<server-name>-index.xml` (eg: `http-server-index.xml`). Each container has a key, the index file stores the association between this key and the repository containing the children's configurations.

Let me show you an example, in the description of the DirectoryResource (eg: images) you can read the following container key:

```
<resource class='org.w3c.jigsaw.resources.DirectoryResource'>
  <attribute name='identifier' flag='6'
class='org.w3c.tools.resources.StringAttribute'>images</attribute>
  <resourcearray name='frames' class='org.w3c.tools.resources.FrameArrayAttribute'
length='1'>
    <resource class='org.w3c.jigsaw.frames.HTTPFrame'>
      <attribute name='identifier' flag='6'
class='org.w3c.tools.resources.StringAttribute'>frame-0</attribute>
      <resourcearray name='frames'
class='org.w3c.tools.resources.FrameArrayAttribute' length='0'>
      </resourcearray>
      <attribute name='last-modified' flag='3'
class='org.w3c.tools.resources.DateAttribute'>932127175430</attribute>
      <attribute name='oid' flag='1'
class='org.w3c.tools.resources.IntegerAttribute'>@@NULL@@</attribute>
      <attribute name='quality' flag='2'
class='org.w3c.tools.resources.DoubleAttribute'>1.0</attribute>
      <attribute name='title' flag='2'
class='org.w3c.tools.resources.StringAttribute'>@@NULL@@</attribute>
      <attribute name='content-language' flag='2'
class='org.w3c.jigsaw.frames.LanguageAttribute'>@@NULL@@</attribute>
      <attribute name='content-encoding' flag='2'
class='org.w3c.jigsaw.frames.EncodingAttribute'>@@NULL@@</attribute>
      <attribute name='content-type' flag='2'
class='org.w3c.jigsaw.frames.MimeTypeAttribute'>text/html</attribute>
      <attribute name='content-length' flag='1'
```

```
class='org.w3c.tools.resources.IntegerAttribute'>@@NULL@@</attribute>
      <attribute name='icon' flag='2'
class='org.w3c.tools.resources.StringAttribute'>dir.gif</attribute>
      <attribute name='maxage' flag='2'
class='org.w3c.tools.resources.LongAttribute'>@@NULL@@</attribute>
      <attribute name='send-md5' flag='2'
class='org.w3c.tools.resources.BooleanAttribute'>false</attribute>
      <attribute name='allow-delete' flag='2'
class='org.w3c.tools.resources.BooleanAttribute'>false</attribute>
      <attribute name='putable' flag='2'
class='org.w3c.tools.resources.BooleanAttribute'>false</attribute>
      <attribute name='relocate' flag='2'
class='org.w3c.tools.resources.BooleanAttribute'>true</attribute>
      <attribute name='index' flag='2'
class='org.w3c.tools.resources.StringAttribute'>Overview.html</attribute>
      <attribute name='icondir' flag='2'
class='org.w3c.tools.resources.StringAttribute'>@@NULL@@</attribute>
      <attribute name='browsable' flag='2'
class='org.w3c.tools.resources.BooleanAttribute'>false</attribute>
      <attribute name='style-sheet-link' flag='2'
class='org.w3c.tools.resources.StringAttribute'>/style/directory.css</attribute>
    </resource>
  </resourcearray>
  <attribute name='last-modified' flag='3'
class='org.w3c.tools.resources.DateAttribute'>932127175430</attribute>
  <attribute name='oid' flag='1'
class='org.w3c.tools.resources.IntegerAttribute'>@@NULL@@</attribute>
  <attribute name='key' flag='1'
class='org.w3c.tools.resources.IntegerAttribute'>83855963</attribute>
  <attribute name='dirstamp' flag='1'
class='org.w3c.tools.resources.DateAttribute'>932031181000</attribute>
  <attribute name='indexer' flag='2'
class='org.w3c.tools.resources.StringAttribute'>default</attribute>
  <attribute name='extensible' flag='2'
class='org.w3c.tools.resources.BooleanAttribute'>true</attribute>
  <attribute name='negotiable' flag='2'
class='org.w3c.tools.resources.BooleanAttribute'>false</attribute>
</resource>
```

and you will find the corresponding repository in the file `http-server-index.xml`:

```
<resource class='org.w3c.tools.resources.store.NewStoreEntry'>
  <attribute name='repository' flag='4'
class='org.w3c.tools.resources.StringAttribute'>6/st-6</attribute>
  <attribute name='key' flag='4'
class='org.w3c.tools.resources.IntegerAttribute'>83855963</attribute>
</resource>
```

So the children's configurations of the DirectoryResource `images` are stored in the repository `stores/6/st-6`.

Specific resources like indexers and realms are stored in two other directories, `<INSTDIR>/Jigsaw/Jigsaw/config/indexers` and `<INSTDIR>/Jigsaw/Jigsaw/auth`.

Note: You can find the dtd (document type definition) of the JigXML format at http://jigsaw.w3.org/JigXML/JigXML.dtd.

The administration protocol

The new administration protocol use the XML serialiation.

Here is a sample request (add a DirectoryResource called "new" under docs_space, docs_space is an alias for the root resource).

```
REGISTER-RESOURCE /http-server/docs_space/ HTTP/1.1
Date: Tue, 10 Aug 1999 15:17:53 GMT
Content-Length: 212
Content-Type: application/xml;type=jigsaw-config
Accept: */*
Host: ender.inria.fr:8009
User-Agent: Jigsaw/2.1.0

<?xml version='1.0' encoding='UTF-8'?>
<jigxml version="1.0" xmlns="http://jigsaw.w3.org/JigXML/JigXML1.0">
 <description class='org.w3c.tools.resources.DirectoryResource' name='new'>
 </description>
</jigxml>
```

Another one (set the "putable" flag to true on the HTTPFrame of style):

```
SET-VALUES /http-server/docs_space/style?frame-0 HTTP/1.1
Date: Tue, 10 Aug 1999 15:15:06 GMT
Content-Length: 994
Content-Type: application/xml;type=jigsaw-config
Accept: */*
Host: ender.inria.fr:8009

<?xml version='1.0' encoding='UTF-8'?>
<jigxml version="1.0" xmlns="http://jigsaw.w3.org/JigXML/JigXML1.0">
 <resource class='org.w3c.jigsaw.frames.HTTPFrame'>
  <inherit class='org.w3c.tools.resources.ProtocolFrame'>
   <inherit class='org.w3c.tools.resources.ResourceFrame'>
    <inherit class='org.w3c.tools.resources.FramedResource'>
     <inherit class='org.w3c.tools.resources.Resource'>
      <inherit class='org.w3c.tools.resources.AttributeHolder'>
       <inherit class='java.lang.Object'>
       </inherit>
      </inherit>
     </inherit>
    </inherit>
   </inherit>
  </inherit>
  <implements class='org.w3c.tools.resources.event.AttributeChangedListener'/>
  <implements class='org.w3c.tools.resources.event.FrameEventListener'/>
  <implements class='java.lang.Cloneable'/>
  <attribute name='putable' flag='2'
class='org.w3c.tools.resources.BooleanAttribute'>true</attribute>
 </resource>
</jigxml>
```

XSL example

Just a little example of what can be done with JigXML and XSL. In this example, the JigXML file is transformed to an HTML file using the following XSL rules.

Jigsaw Team

$Id: JigXML.html,v 1.18 1999/07/29 16:05:19 bmahe Exp $

Jigsaw
SSI extension tutorial

Jigsaw Home / Documentation Overview

The SSIFrame (org.w3c.jigsaw.ssi.SSIFrame) is a Jigsaw frame that provides a flexible way of generating part of the content of a document from individual pieces. This may sound too general, and that's because there is little constraint on the way the constituent pieces are generated. For example, one use of the SSIFrame is the traditional one: the content of any resource can be embedded within any document exported by the SSIFrame, by using the `include` command from the default command registry. Some other of the default commands allow you to include the size of the document, the time of day, the hit count, and other general data.

One of the goals of this tutorial is to show that the SSIFrame is useful beyond its traditional use, as a powerful way of creating documents with a dynamically generated content. It is assumed that you are familiar with the administration of Jigsaw in general.

Commands and registries

The SSIFrame will scan through the text of the file looking for a special kind of HTML comment. If it finds something of the form **`<!--#command par_1=val_1 par_2=val_2 ... par_n=val_n -->`**, it will interpret it as a command. **`par_1 ... par_n`** are the names of the parameters, and **`val_1 ... val_n`** are their values. The values can optionaly be enclosed in single or double quotes; otherwise they are delimited by ASCII white space. For example, the string **`<!--#include virtual="doc.html"-->`** denotes a call to a command called "include", with one parameter called "virtual" that has a value of "doc.html".

Upon finding a command, the SSIFrame will look it up in an object called the *command registry*. The command registry returns the *command* that is registered by that name. Then, it will call the command's `execute` method with the specified parameters, and with other contextual data.

Command registries are objects of class org.w3c.jigsaw.ssi.commands.CommandRegistry. Since this is an abstract class, a concrete implementation of one must be available for SSIFrame to work. One such implementation is supplied with the distribution: it is org.w3c.jigsaw.ssi.commands.DefaultCommandRegistry, which includes the bread-and-butter SSI commands. Commands are implementations of the org.w3c.jigsaw.ssi.commands.Command interface or org.w3c.jigsaw.ssi.commands.ControlCommand. The SSIFrame declares a `registryClass` attribute, which is set to the particular command registry to use in parsing a given document.

Therefore, the way to extend the SSIFrame is to create (either from scratch or by subclassing an existing one) a command registry that knows about the new commands that are being added. A good way to become familiar with these classes is too look at the code for DefaultCommandRegistry and its superclass, BasicCommandRegistry, and at the code for the default commands (in rough order of complexity): SampleCommand, CountCommand, ConfigCommand, FSizeCommand, FLastModCommand, EchoCommand, IncludeCommand, jdbcCommand, CounterCommand, ServletCommand.

SSIFrame allows you to create control commands like loop and test. These commands implements the org.w3c.jigsaw.ssi.commands.ControlCommand interface. Here is the code of the default control commands :

IfCommand, ElseCommand, EndifCommand, LoopCommand, ExitloopCommand, EndloopCommand. The org.w3c.jigsaw.ssi.commands.Command interface has been modified, a new method was added (getValue). This method is used by some control commands (if) to get some value relative to the command.

Let's have a look of what can be done with control commands :

This shtml page display the content of the users database.

```
<html>
  <head>
    <title>Database SSI</title>
  </head>

  <body>
      <h1>Database SSI</h1>
      <p>This Server Side Include extension allows you to query a database,
         to make some loop and some tests.
        (which I am doing right now)

<!--#jdbc select="SELECT * FROM users" name="result"
driver="COM.imaginary.sql.msql.MsqlDriver"
url="jdbc:msql://www43.inria.fr:4333/users" -->

        <p>The query has run, here is all the results:<p>
        <table border=2>
        <tr><td><b>Name</td><td><b>Login</td>
        <td><b>Email</td><td><b>Age</td></tr>
<!--#loop name="loop1" -->
      <!--#jdbc name="result" next="true" -->

      <!--#if name="if1" command="jdbc" var="result" equals="empty" -->
                <!--#exitloop name="loop1" -->
      <!--#endif name="if1" -->

      <!-- the three lines above can be changed in : -->

      <!--#exitloop name="loop1" command="jdbc" var="result" equals="empty" -->

        <tr><td>
        <!--#jdbc name="result" column="1" -->
        </td><td>
        <!--#jdbc name="result" column="2" -->
        </td><td>
        <!--#jdbc name="result" column="3" -->
        </td><td>
        <!--#jdbc name="result" column="4" -->
        </td></tr>
<!--#endloop name="loop1" -->
        </table>
      <hr>
  </body>
</html>
```

IfCommand : the source code

This command implements the *classic* if statement. This command can only be used with EndifCommand and (optionnaly) with ElseCommand.

```
package org.w3c.jigsaw.ssi.commands;

import java.util.*;

import org.w3c.jigsaw.http.* ;
import org.w3c.www.http.* ;
import org.w3c.tools.resources.* ;
import org.w3c.util.* ;
import org.w3c.jigsaw.ssi.*;

/**
 * Implementation of the SSI if command.
 * @author Benoit Mahe :bmahe@sophia.inria.fr
 */

public class IfCommand implements ControlCommand {
  private final static String  NAME  = "if";
  private final static boolean debug = true;

  // The parameters accepted by the if command
  private static final String keys[] = {
    "name",
    "command",
    "var",
    "equals"
  };

  // Used to store the position of each if command
  protected static Hashtable ifstore = null;

  static {
     ifstore = new Hashtable(23);
  }

  /**
   * Returns the (String) value of the given variable.
   * @return a String instance.
   */
  public String getValue(Dictionary variables, String var) {
    return null;
  }

  protected static int getPosition(String name)
    throws ControlCommandException
  {
    Integer pos = (Integer)ifstore.get(name);
```

```
    if (pos == null)
      throw new ControlCommandException(NAME,"Position unknown.");
     else return pos.intValue();
  }


  /**
   * register the command position in the structure
   * witch store the SSIFrame.
   */
  public void setPosition(SSIFrame ssiframe,
                          Request request,
                          CommandRegistry registry,
                          ArrayDictionary parameters,
                          Dictionary variables,
                          int position)
  {
    Object values[] = parameters.getMany(keys);
    String name     = (String) values[0];
    if (name != null)
      ifstore.put(ssiframe.getResource().getURLPath()+":"+name, new
Integer(position));
  }

  /**
   * Executes this command. Might modify variables.
   * Must not modify the parameters.
   * It may handle conditional requests, except that if
   * it replies with a status of HTTP.NOT_MODIFIED, it must
   * still reply with a content (the same content that it would have
   * returned for an inconditional request).  This is because
   * further SSI commands down the line may decide thay they have
   * been modified, and then a content must be emitted by SSIFrame.
   * @param request the original HTTP request
   * @param parameters The parameters for this command
   * @param variables The global variables for the parse
   * @return a Reply with the output from the command */

  public Reply execute(SSIFrame ssiframe,
                       Request request,
                       ArrayDictionary parameters,
                       Dictionary variables)
  {
    // Empty reply
    return ssiframe.createCommandReply(request, HTTP.OK);
  }

  protected boolean check(CommandRegistry registry,
                          Request request,
                          ArrayDictionary parameters,
                          Dictionary variables)
  {
```

```
    Object values[] = parameters.getMany(keys);
    String name     = (String) values[0];
    String command  = (String) values[1];
    String var      = (String) values[2];
    String equals   = (String) values[3];

    if ((command == null) || (var == null) || (equals == null))
      return false;
    Command cmd = registry.lookupCommand(command);
    String value = cmd.getValue(variables,var);
    // here is the test
    return value.equals(equals);
  }

  /**
   * Give the next position in the structure witch
   * store the SSIFrame.
   */
  public int jumpTo(SSIFrame frame,
                    Request request,
                    CommandRegistry registry,
                    ArrayDictionary parameters,
                    Dictionary variables)
    throws ControlCommandException
  {
    Object values[] = parameters.getMany(keys);
    String name     = (String) values[0];
    if (name != null) {
      if (check(registry,parameters,variables))
        return getPosition(frame.getURLPath()+":"+name)+1;
      try {
        return (ElseCommand.getPosition(frame.getURLPath()+":"+name)+1);
      } catch (ControlCommandException ex) {
        return (EndifCommand.getPosition(frame.getURLPath()+":"+name)+1);
      }
    }
    throw new ControlCommandException(NAME,"name not initialized.");
  }

  /**
   * Returns the name of this command. (Case sensitivity is up to
   * the lookupCommand method in the command registry.)
   * @return the name of the command
   * @see org.w3c.jigsaw.ssi.commands.CommandRegistry#lookupCommand
   */
  public String getName() {
    return NAME;
  }

}
```

With this in mind, let's implement a useful extension of SSIFrame.

A server statistics page with SSIFrame

There is an existing Jigsaw resource (org.w3c.jigsaw.status.StatisticsFrame) that is used to display the internal statistics of the server. In what follows, we will mimic the functionality of this frame with an SSI command. There is an object that supplies all these statistics for us; its class is org.w3c.jigsaw.http.httpdStatistics and it can be obtained from the server. Our SSI command will query this object and emit the values. We'd like to be able to use it like this: <!--#stat data=<type> -->, where <type> specifies the particular statistic that is going to be inserted, and is one of:

- `serverLoad`
- `freeThreads`
- `idleThreads`
- `totalThreads`
- `hitCount`
- `meanReqTime`
- `maxReqTime`
- `maxReqURL`
- `minReqTime`
- `minReqUrl`
- `emittedBytes`

Each of them will correspond to one of the methods in httpdStatistics.

Writing the `stat` command

This command can be written in a very straightforward manner. All we have to do is:

1. Obtain the httpdStatistics instance from the server.
2. Call in it the appropriate method, according to the data parameter.
3. Return a reply with this value as content.

This translates to the following java class, which will be called StatCommand:

```
package org.w3c.jigsaw.tutorials ;

import java.util.* ;

import org.w3c.jigsaw.http.* ;
import org.w3c.www.http.HTTP ;
import org.w3c.jigsaw.ssi.* ;
import org.w3c.util.* ;
import org.w3c.jigsaw.ssi.commands.* ;

public class StatCommand implements Command {
    private static final String NAME = "stat" ;

    public final String getName()
    {
        return NAME ;
    }

    // Unuseful here
```

```
    public String getValue(Dictionary variables, String variable) {
        return null;
    }

    public Reply execute(SSIFrame frame,
                         Request request,
                         ArrayDictionary parameters,
                         Dictionary variables)
    {
        // Obtain the statistics from the server
        httpdStatistics stats = frame.getServer().getStatistics() ;

        // Get the parameter specifying the kind of statistic to emit.
        String data = (String) parameters.get("data") ;

        // If the parameter is not supplied, do nothing
        if(data == null)
            return null ;

        // Otherwise, compare it against the possible different keywords
        // (Since there are no "pointers to methods", this is the simplest way it
        // can be written)
        long result = -1 ;
        String urlResult = null ;
        if(data.equalsIgnoreCase("serverload")) {
            result = stats.getServerLoad() ;
        } else if(data.equalsIgnoreCase("freethreads")) {
            result = stats.getFreeThreadCount() ;
        } else if(data.equalsIgnoreCase("idlethreads")) {
            result = stats.getIdleThreadCount() ;
        } else if(data.equalsIgnoreCase("totalthreads")) {
            result = stats.getTotalThreadCount() ;
        } else if(data.equalsIgnoreCase("hitcount")) {
            result = stats.getHitCount() ;
        } else if(data.equalsIgnoreCase("meanreqtime")) {
            result = stats.getMeanRequestTime() ;
        } else if(data.equalsIgnoreCase("maxreqtime")) {
            result = stats.getMaxRequestTime() ;
        } else if(data.equalsIgnoreCase("maxrequrl")) {
            urlResult = stats.getMaxRequestURL().toExternalForm() ;
        } else if(data.equalsIgnoreCase("minreqtime")) {
            result = stats.getMinRequestTime() ;
        } else if(data.equalsIgnoreCase("minrequrl")) {
            urlResult = stats.getMinRequestURL().toExternalForm() ;
        } else if(data.equalsIgnoreCase("emittedbytes")) {
            result = stats.getEmittedBytes() ;
        } else return null ;

        // Make a reply with the datum and return it
        Reply reply = frame.createCommandReply(request, HTTP.OK) ;
        reply.setContent( urlResult == null
                         ? Long.toString(result)
```

```
                            : urlResult ) ;
        return reply ;
    }
}
```

Listing 1: The command class

The Command interface defines three methods. The `getName` method simply returns a String with the name of the command. The getValue method returns a value relative to the given parameter (Used by control commands).The `execute` method is the one that does the work. This method can be thought of as the `get` method in a frame: it takes, among other things, a Request object, and it produces a Reply object. The SSIFrame will insert the contents of the replies of each of the commands (*partial replies*) into the main, global, content, and it will also merge the relevant headers of the partial replies into the headers of the global reply. Besides taking a request, the `execute` method takes these arguments as well:

`org.w3c.jigsaw.ssi.SSIFrame frame`

This is the SSIFrame that is executing the command.

`org.w3c.util.ArrayDictionary parameters`

The parameters that the command is called with. An ArrayDictionary is a subclass of java.util.Dictionary. The parameters are stored as strings with the parameter names as keys.

`java.util.Dictionary variables`

The current set of variables. A command may change its behavior according to the values of these variables, and it can also modify the variables. The meaning of the variables is almost completely command- and command registry-dependent. The DefaultCommandRegistry uses the variables to keep state across different command calls in the same document, such as the current date and time formats.

The `execute` method can also return **`null`**, which is interpreted as the absence of output. There are some subtle differences between the `execute` method and a regular frame `get` method. In particular, care must be taken if dealing with conditional requests. This example is simple enough that this is not a concern.

Now that the command itself is finished, we need to make it part of a command registry, so that it can be actually used in documents.

Writing a command registry

Since we'd like to be able to use the "standard" SSI commands in adition to our brand-new `stat` command, it's not a bad idea to make our new registry a subclass DefaultCommandRegistry. The way to do this is very straightforward:

```
package org.w3c.jigsaw.tutorials ;

import org.w3c.jigsaw.ssi.* ;
import org.w3c.jigsaw.ssi.commands.* ;

public class MyCommandRegistry extends DefaultCommandRegistry {
    public MyCommandRegistry()
    {
        registerCommand(new StatCommand()) ;
    }
}
```

Listing 2: The command registry class

The constructor simply calls the `registerCommand` method (defined in BasicCommandRegistry), with a new instance of the command that we're adding.

We're now ready to use this command in a future document.

Using the new registry

One way of using the newly-created command registry is to change the `registryClass` attribute defined for the `.shtml` extension to "`org.w3c.jigsaw.tutorials.MyCommandRegistry`". After doing that, Jigsaw will use the new registry when indexing new files with the `.shtml` extension (or reindexing old files). Then we can create a file that makes use of the new command, and place it in a Jigsaw-accesible directory. For example, we could do this:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>
  <head>
    <meta http-equiv="Refresh" content="5">
    <title>Server Statistics</title>
  </head>
  <body>
      <ul>
        <li>hits: <!--#stat data=hitCount  -->
        <li>bytes: <!--#stat data=emittedBytes  -->
      </ul>
      <p>Request processing times:
      <table border>
        <tr>
          <th align="center"> min
          <th align="center"> avg
          <th align="center"> max
        </tr>
        <tr>
          <th align="center"> <!--#stat data=minReqTime  -->
          <th align="center"> <!--#stat data=meanReqTime  -->
          <th align="center"> <!--#stat data=maxReqTime  -->
        </tr>
      </table>
      <p>Thread counts:
      <table border>
        <tr>
          <th align="center"> free
          <th align="center"> idle
          <th align="center"> total
        </tr>
        <tr>
          <th align="center"> <!--#stat data=freeThreads  -->
          <th align="center"> <!--#stat data=idleThreads  -->
          <th align="center"> <!--#stat data=totalThreads  -->
      </table>
      <p>Current load: <!--#stat data=serverLoad  -->
  </body>
</html>
```

Listing 3: A possible use of the new command

The above document will produce exactly the same output that the Statistics frame would emit.

What have we gained?

At this point we can compare two different approaches to generating HTML dynamically. The first one involves writing a new, specialized, frame. The approach illustrated in this tutorial consists of writing an SSI command and serving the document with the SSIFrame. Doing it this way has these advantages:

- The structure of the served document is not hard-coded. Instead, the Java code only expresses the minimal needed functionality (in this case, that of emitting server statistics), while the markup of the document can be modified without having to recompile.
- Different functionalities can be mixed in a more orthogonal way. That is, it is inefficient at best to create a resource that combines the function of two existing resources. By distilling the functionality into SSI commands, it is straightforward to make a registry that has all the needed commands.

One disadvantage of the SSI approach is the extra overhead incurred at serve-time of constructing the content from the pieces supplied by the commands. The SSIFrame tries to avoid this overhead as much as possible. The most important optimization in this respect is the fact that the parsing of the document (i.e., scanning the text for commands, and reading the parameters) is done only when the file is modified. Even then, each command needs to check its parameters, which *does* add to serve-time overhead.

Jigsaw Team

$Id: SSI.html,v 1.8 1999/03/16 13:38:12 bmahe Exp $