Your personal archival: Repository Server

José Paulo Leal[1] and Ricardo Queirós[2]
[1] CRACS & DCC-FCUP, University of Porto, Portugal
[2] CRACS & DI-ESEIG/IPP, Porto, Portugal

1 Overview

The architecture of eLearning platforms is moving from centralised, component-based systems to decentralised platforms assembling multiple services [1]. These services can participate in several learning processes that are easily reconfigured to meet changing requirements and demands. The Learning Objects Repositories (LOR) are an essential part of these service oriented platforms since they provide content to several types of services.

The crimsonHex system is a specialised and interoperable repository of programming problems defined as learning objects. The repository is one of several separate components of the EduJudge network and plays a main role on its overall architecture, since it acts as a service provider for the other e-Learning systems such as Learning Management Systems (LMS) and Evaluation Engines (EE).

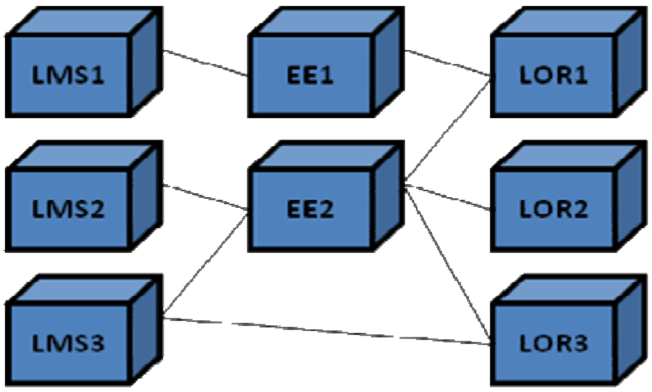


Figure 1: EduJudge Network

The main feature of an interoperable repository is the support for automatic communication with other systems. Hence the architecture of crimsonHex is driven by the need for interoperability. An API is the heart of its architecture, defining functions implemented by a **Core component** and required both by other crimsonHex components (**Web Manager** and **Importer**) and external consumers of crimsonHex's services (**LMS** and **EE**) as depicted in Figure 2.

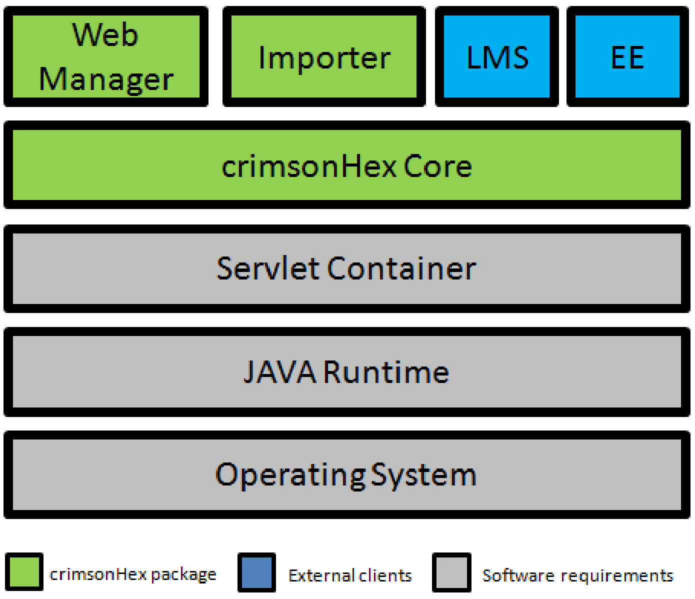


Figure 2: Stack diagram of crimsonHex

The architecture of crimsonHex repository is divided in three main components:

- the **Core** exposes the main features of the repository, both to external services, such as the LMS and the EE, and to internal components - the Web Manager and the Importer;
- the **Web Manager** allows the creation, revision, versioning, uploading/downloading of LOs and related meta-data, enforcing compliance with controlled vocabularies;
- the **Importer** populates the repository with existing legacy repositories. In the remainder we focus on the Core component, more precisely, its functions, communication model and implementation.

The core component of crimsonHex provides a minimal set of functions based on the IMS DRI specification. This specification prescribes a list of basic functions exposed by SOAP web services. We extended this specification both with a RESTfull interface and with new functions, for reporting learning objects' usage data and managing the structure of the repository. The latest type is intended mostly for the user interface layer (Web Manager).

The repository is implemented over other software layers depicted as grey rectangles in Figure 2 that must be installed prior to the installation of crimsonHex itself. In the remainder of this chapter we enumerate these system requirements and explain how they can be installed. Finally we present configuration options for securing the repository.

2 System Requirements

The crimsonHex repository is a web system with both a user and an application interfaces. This means that the system itself is deployed to a host server but it also requires client software in order to use. As a rule, servers and clients will run on different machines. However, at least for testing purposes, a single machine can act simultaneously as client and server.

The hardware, operating system and software configurations to host the repository are listed in Table 1.

		Minimum	Recommended	Test with
Hardware	RAM	256MB	512MB	4GB
	Hard Disk[1]	1GB	4GB	2GB
Operating System	Windows	XP	XP	XP and Vista
	Linux	Linux 2.6	Linux 2.6	Mandriva 2009.1
Software	JRE/JDK	1.5	1.6	1.6
	Servlet Container	support to JSP 2.0 and Servlet 2.4	support to JSP 2.1 and Servlet 2.5	Tomcat 6

Table 1: Server requisites

The remainder of this section focuses exclusively on the server installation. On client side a modern web browser is enough to access the user interface. The application interface will depend on the programming platform and thus is out of the scope of this handbook. The client requisites are summarized in Table 2.

		Minimum	Recommended	Tested with
Web browser	IE	6.0	8.0	7.0
	Firefox	1.0	3.5	3.0
	Safari	2.0	4.0	4.0[2]
	Opera	9.0	9.0	9.0
Screen Resolution		1024x768	1280x768	1440x900
Web service clients	REST	HTTP/1.0	HTTP/1.1	HttpURLConnection
	SOAP		SOAP 1.1	Axis 1.0

Table 2: Client requisites

3 JAVA installation

First you must check your server has a Java Runtime Environment (JRE) to execute JAVA applications. The Java web site has detailed instruction on how to download and install Java on Linux, Windows, Solaris and Apple (OS X) operating systems. These instructions are available from the following URL: http://www.java.com/en/download/manual.jsp.

After installing Java on your computer you may be asked to restart it.

[1] Excluding the Operating System
[2] Windows version

4 Servlet Container installation

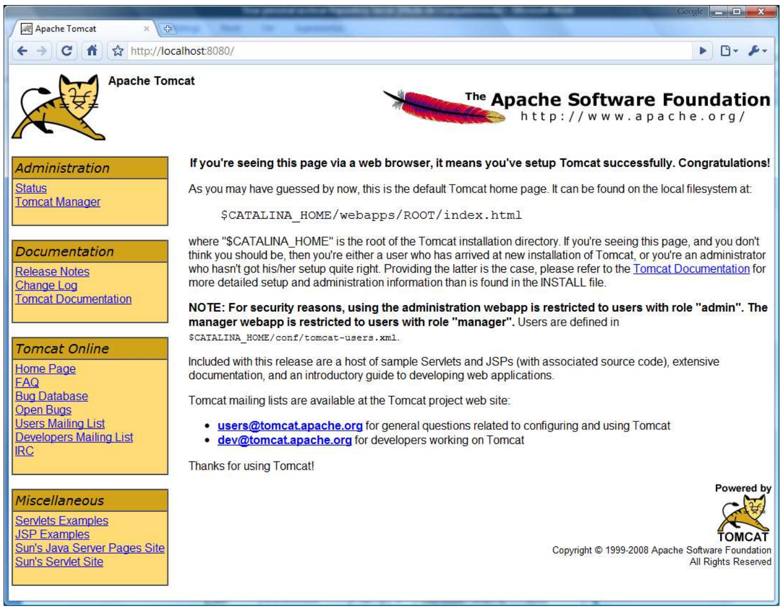
Servlets are Java objects that dynamically process Web requests. A servlet is managed by a servlet container, (also known as Web container) which is essentially a specialized Web server that maps certain URLS to specific servlets and is responsible for managing their lifecycle. The servlet processes each request for its associated URLs and produces a response that is sent back to the client.

If you do not have a container installed on your host system, you need to install one that implements the Java Servlet 2.5 and the JavaServer Pages 2.1 (JSP) specifications from Sun Microsystems. There are several containers available, such as: Apache Tomcat, Sun GlassFish, Red Hat JBoss, BEA WebLogic Server, Caucho's Resin Server, IBM WebSphere Server or Apple WebObjects. From the previous list, the one that has been tested was the Apache Tomcat, but according to the JAVA specifications supported by the other containers the repository should work in any of them.

Apache Tomcat is an open source servlet container developed by the Apache Software Foundation (ASF). Tomcat version 6.0 implements the Java Servlet 2.5 and the JavaServer Pages 2.1 (JSP) specifications and provides an HTTP web server environment for Java code to run. To install Tomcat you should follow the following instructions:

1. Locate the latest production version of Tomcat from `http://tomcat.apache.org/download-60.cgi`.
2. Choose the format type of the core binary distribution: zip (Windows) or tar.gz (Unix)
3. Save the file (named something like `apache-tomcat-6.0.20.zip`) to your computer
4. Unzip the file into a directory of your choice such as `c:\ (windows)` or `/usr/local` (Unix). The program is contained in a subdirectory named something like apache-tomcat-6.0.20
5. Open a command shell and change your directory to the Tomcat directory such as `c:\apache-tomcat-6.0.20` (Windows) or /usr/local/apache-tomcat-6.0.20 (Unix)
6. Change to the bin subdirectory
7. Type startup.bat (Windows) or startup.sh (Unix) followed by ENTER

Point your web browser to `http://localhost:8080`. You should get the default Tomcat home page, similar to Figure 3:



If you're seeing this page via a web browser, it means you've setup Tomcat successfully. Congratulations!

As you may have guessed by now, this is the default Tomcat home page. It can be found on the local filesystem at:

```
$CATALINA_HOME/webapps/ROOT/index.html
```

where "$CATALINA_HOME" is the root of the Tomcat installation directory. If you're seeing this page, and you don't think you should be, then you're either a user who has arrived at new installation of Tomcat, or you're an administrator who hasn't got his/her setup quite right. Providing the latter is the case, please refer to the Tomcat Documentation for more detailed setup and administration information than is found in the INSTALL file.

NOTE: For security reasons, using the administration webapp is restricted to users with role "admin". The manager webapp is restricted to users with role "manager". Users are defined in `$CATALINA_HOME/conf/tomcat-users.xml`.

Included with this release are a host of sample Servlets and JSPs (with associated source code), extensive documentation, and an introductory guide to developing web applications.

Tomcat mailing lists are available at the Tomcat project web site:

- **users@tomcat.apache.org** for general questions related to configuring and using Tomcat
- **dev@tomcat.apache.org** for developers working on Tomcat

Thanks for using Tomcat!

Copyright © 1999-2008 Apache Software Foundation
All Rights Reserved

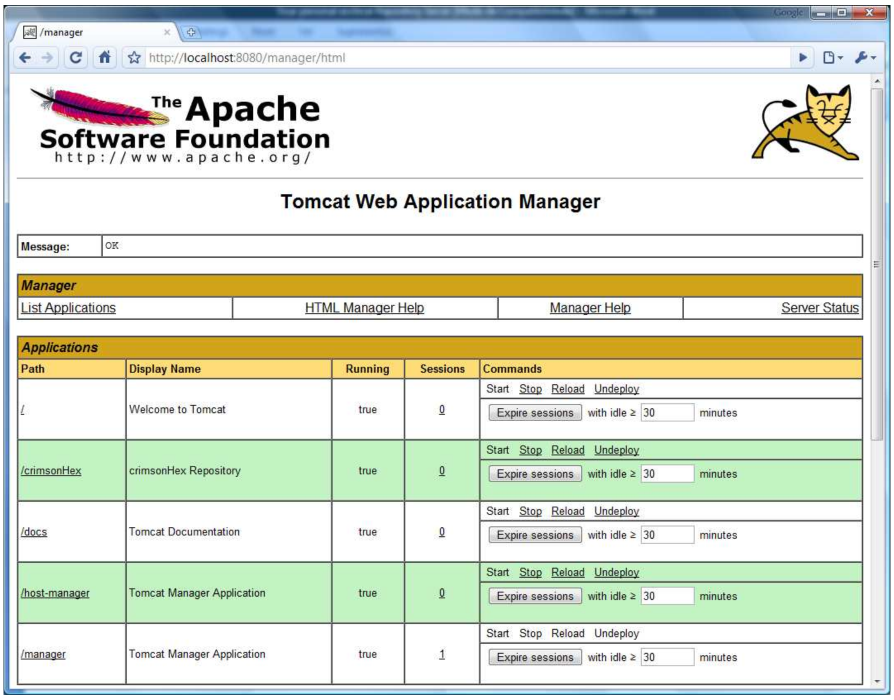
Figure 3: Default Tomcat home page

At this moment Tomcat is successfully installed and the HTML source of the page presented in Figure 3 can be found on the local file system at: `$CATALINA_HOME/webapps/ROOT/index.html`, where "$CATALINA_HOME" is the root of the Tomcat installation directory. To stop the Tomcat server, type `shutdown.bat` (Windows) or `shutdown.sh` (Unix) followed by ENTER.

The default Tomcat home page presents four sections named Administration, Documentation, Tomcat Online and Miscellaneous. For security reasons, using the administration webapp is restricted to users with role "admin". The manager webapp is restricted to users with role "manager". Users are defined in `$CATALINA_HOME/conf/tomcat-users.xml`. The default content of this file includes a user with the previous two rules:

```
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
  <role rolename="manager"/>
  <role rolename="admin"/>
  <user username="admin" password="" roles="admin, manager"/>
</tomcat-users>
```

The Administration section provides a Status link to inform the Administrator about relevant Server information (e.g. software versions, memory status, and thread statistics). The Tomcat Manager link allows you to list all the applications in the servlet container and allows the Administrator to start/stop, reload and undeploy an existent application.


/manager

http://localhost:8080/manager/html

Tomcat Web Application Manager

Message:	OK

Manager

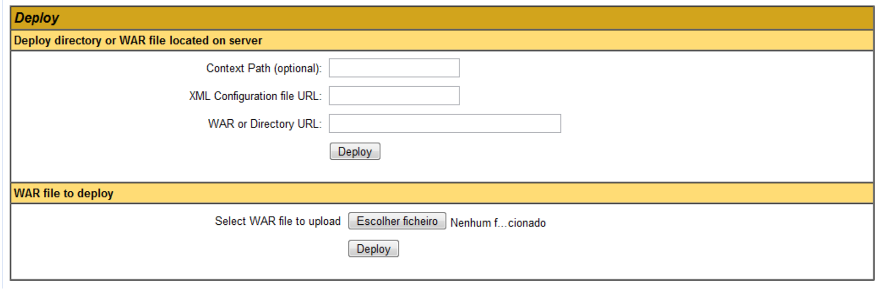
List Applications	HTML Manager Help	Manager Help	Server Status

Applications

Path	Display Name	Running	Sessions	Commands
/	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/crimsonHex	crimsonHex Repository	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	Tomcat Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

Figure 4: Managing applications in Tomcat

Using the Manager (Figure 5), you can also deploy your application to the servlet container as a Web application archive (WAR) file. You can either deploy a WAR file already on the server or upload a WAR file from the computer running your web client. Alternatively, you can just drop the WAR file on the `$CATALINA_HOME/webapps` directory of your tomcat server.


Deploy

Deploy directory or WAR file located on server

Context Path (optional):	
XML Configuration file URL:	
WAR or Directory URL:	
	Deploy

WAR file to deploy

Select WAR file to upload	Escolher ficheiro Nenhum f...cionado
	Deploy

Figure 5: Deploying WAR files in Tomcat

A WAR file contains a complete Java web application,, including server-side classes, static Web content (HTML, image, and sound files). A WAR has a specific directory structure. The top-level directory is where JSP pages, client-side classes and static Web content are stored. The top-level directory contains a special subdirectory called WEB-INF, which contains the following files and directories:

- web.xml: Web application deployment descriptor
- classes: directory with server-side classes (servlets and other classes)
- lib: directory that contains JAR archives of libraries

The **web.xml** describes how to deploy a web application in a servlet container. If the web application uses servlets, then the servlet container uses this file to map an URL request to a specific servlet. Here is a short example:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app …>
  <display-name>MyCompany App</display-name>
  <description>Details of MyCompany App</description>
  <servlet>
       <servlet-name>MyServlet</servlet-name>
       <servlet-class>MyCompany.MyServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>MyServlet</servlet-name>
    <url-pattern>/logic/*</url-pattern>
  </servlet-mapping>
  <welcome-file-list>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
  </web-app>
```

web-app Elements	Description
`display-name`	Application short name (displayed in Tomcat Manager).
`description`	Optional description tag (not used by Tomcat).
`servlet/servlet-name`	Symbolic servlet name.
`servlet/servlet-class`	Fully qualified class name for the servlet class
`servlet-mapping/servlet-name`	Symbolic servlet name linking this URL mapping to a servlet definition
`servlet-mapping/url-pattern`	URL patterns which will trigger the servlet
`welcome-file-list/welcome-file`	Define, in order of preference, the default file name for the web application and its sub-directories.

Table 3: Main elements in the web.xml file

The information in Table 3 can be complemented reading the full schema of the Web application deployment descriptor file [2].

Included in the distribution of Tomcat are sample Servlets and JSPs (with source code), extensive documentation and an introductory guide to developing Java web applications.

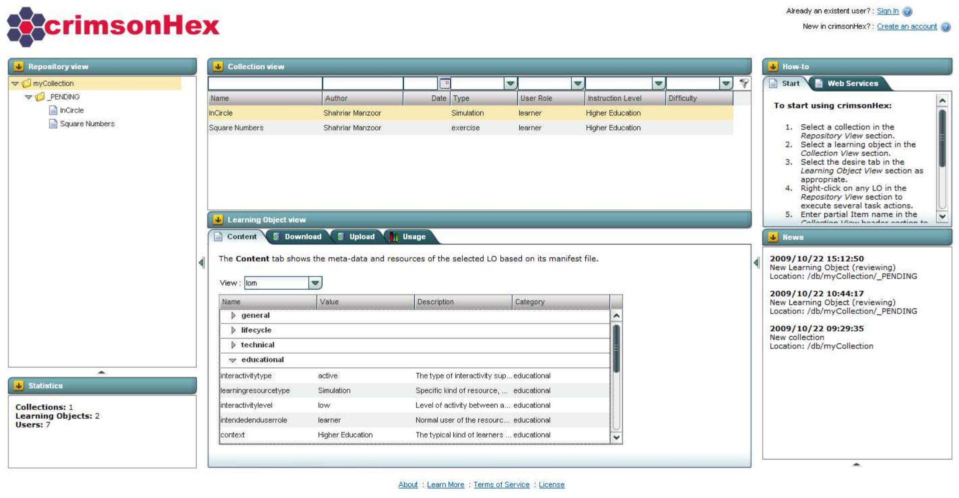
5 crimsonHex installation

The crimsonHex repository is distributed under a GNU-GPL v2 License. For more details we advise you to read the GNU General Public License on the GNU web site [3].

The repository is distributed as a single package, including the Core and the Web Manager. To install it follow these instructions:

1. Open a web browser and locate the latest version of crimsonHex from the web address http://mooshak.dcc.fc.up.pt/~edujudge/crimsonHex/distrib/
2. Download the latest version of the repository as a single WAR file;
3. Deploy the WAR file to the servlet container (e.g. Tomcat: use the Tomcat Manger to upload the WAR file to the server or just drop the WAR file in webapps directory of Tomcat installation);
4. Open your favourite web browser and type http://localhost:8080/crimsonHex.

You should get the home page of the web interface of the crimsonHex repository called **Web Manager** as illustrated in Figure 6.



Name	Author	Date	Type	User Role	Instruction Level	Difficulty
InCircle	Shahriar Manzoor		Simulation	learner	Higher Education	
Square Numbers	Shahriar Manzoor		exercise	learner	Higher Education	

The **Content** tab shows the meta-data and resources of the selected LO based on its manifest file.

View : lom

Name	Value	Description	Category
general			
lifecycle			
technical			
educational			
interactivitytype	active	The type of interactivity sup...	educational
learningresourcetype	Simulation	Specific kind of resource, ...	educational
interactivitylevel	low	Level of activity between a...	educational
intendedenduserrole	learner	Normal user of the resourc...	educational
context	Higher Education	The typical kind of learners ...	educational

To start using crimsonHex:

1. Select a collection in the *Repository View* section.
2. Select a learning object in the *Collection View* section.
3. Select the desire tab in the *Learning Object View* section as appropriate.
4. Right-click on any LO in the *Repository View* section to execute several task actions.
5. Enter partial Item name in the

2009/10/22 15:12:50
New Learning Object (reviewing)
Location: /db/myCollection/_PENDING

2009/10/22 10:44:17
New Learning Object (reviewing)
Location: /db/myCollection/_PENDING

2009/10/22 09:29:35
New collection
Location: /db/myCollection

Collections: 1
Learning Objects: 2
Users: 7

Figure 6: Web Manager home page

The Web Manager allows users to browse the repository and access learning objects using a web interface. This component of the crimsonHex repository is detailed in the section "Feeding the beast: Managing your collections of problems". The Web Manager contains also a tutorial on accessing the repository using web services.

6 Web service security

Security is a major concern in a repository of learning objects. The architecture of crimsonHex defines two access points to the repository: using the Web Manager or interacting directly with the Core. User authentication and authorization in the Web Manager is based on access credentials and in detailed in the chapter entitled

“Feeding the beast: Managing your collections of problems”. Web services security (e.g. LMS, EE) is described in the remainder of this section.

Following the design principles of simplicity and efficiency we decided to avoid the management of users and access control in the Core. This decision does not preclude the security of this component since we can control these features in the servlet container (e.g. Tomcat). By default, you do not need to authenticate to access Tomcat resources. To authenticate users accessing Tomcat resources we can use the following mechanisms:

- **HTTP Basic Authentication**: allow a web browser to provide credentials (username and password) when making a request. If SSL is not used, then the credentials are sent as plaintext and could be intercepted easily;
- **HTTP Digest Authentication**: is also performed by the browser upon request by the web server. However, in this case the password is digested with the secure MD5 algorithm before it is sent by the browser;
- **HTTPS Client Authentication**: secure the channel using HTTPS (HTTP over SSL - Secure Sockets Layer). The authentication is achieved through the verification of client certificates provided by SSL. To implement this approach it’s necessary to configure the servlet container (e.g. Tomcat) to support HTTPS requests with authorized certificates.

Authenticated clients can be granted access to resources by configuring the web.xml referred on section 4. However, you can authorize access to resource based on client identification rather than on user authentication. This approach does not require user management and provided a reasonable security, although without ensuring the confidentiality of the exchanged data.

Client identification based authorization in Tomcat uses **valve components**. A Valve is a component inserted in the request processing pipeline of Tomcat. The following valves can be used for client identification:

- **Remote Address Filter** - allows you to compare the IP address of the client that submitted this request against one or more regular expressions, and either allow the request to continue or refuse to process the request from this client;
- **Remote Host Filter** - allows you to compare the hostname of the client that submitted this request against one or more regular expressions, and either allow the request to continue or refuse to process the request from this client.

To exemplify the use of valves for this purpose we present the configuration steps to restrict the access to a crimsonHex deployed application in Tomcat using the Remote Address Filter:

1. Open the `server.xml` file located on the local file system at `$CATALINA_HOME/conf`, where "$CATALINA_HOME" is the root of the Tomcat installation directory.
2. Inside the existent `host` element put the following code:

```
<Context path="/crimsonHex">
     <Valve
            className="org.apache.catalina.valves.RemoteAddrValve"
            allow="192.168.40.190"/>
</Context>
```

3. Start Tomcat

The previous example sets the access of the crimsonHex repository to the computer with the IP defined in the `allow` attribute of the `Valve` element. Note that the `Valve` element is surrounded by a `Context` element representing a web application, which is run within a particular virtual host (defined in the `host` element). The valve has a `className` attribute that must be set to the specific filter. The request client address will be checked against a configured list of "accept" and/or "deny" filters, which are defined using the Regular Expression syntax supported by the Jakarta Regexp regular expression library. Requests that come from locations that are not accepted will be rejected with an HTTP "Forbidden" error.

References

[1] Dagger, D., O'Connor, A., Lawless, S., Walsh, E., Wade, V.: Service Oriented eLearning Platforms: From Monolithic Systems to Flexible Services. In: IEEE Internet Computing Special Issue on Distance Learning, (2007)

[2] Sun Microsystems: XML Schema for the Servlet 2.5 Web ARchive (WAR) File

[3] GNU General Public License http://www.gnu.org/copyleft/gpl.html.