TP Servlets

Cours d’Applications Réparties – RICM4- Polytech

Contacts: Fabienne Boyer (Fabienne.Boyer@imag.fr), Philippe Morat (mailto:Philippe.Morat@imag.fr)

Source: TP servlets de Sara Bouchenak, LIG.

1- Installing the Servlet server

In order to install Apache Tomcat Servlet server, please follow these steps:

1. Download the Tomcat core module from http://tomcat.apache.org/ (version 7.0 if your Java version is minimum 1.6, check this through the command *java –version*)
2. Uncompress the downloaded file
3. Define the *CATALINA_HOME* environment variable in such a way that it points to your installation directory of Apache Tomcat, as follows:
    - In Unix / tcsh: *setenv CATALINA_HOME ~/AppliRep/apache-tomcat-6.0.14*
    - In Unix / bash: *export CATALINA_HOME=~/ AppliRep /apache-tomcat-6.0.14*
    - In Windows: *set CATALINA_HOME=~/AppliRep /apache-tomcat-6.0.14*

(check the shell you are using through the command *echo $SHELL)*

4. Make sure that the *JAVA_HOME* environment variable is defined (check it with: *echo $JAVA_HOME*); otherwise, define it in such a way that it points to the installation directory of Java.

2- Starting the Servlet server

The Apache Tomcat Servlet server is started on a particular port specified in the *$CATALINA_HOME/conf/server.xml* configuration file. The default port for accepting HTTP connections is *8080* (see the corresponding "*Connector port*" value in the *server.xml* configuration file).

In order to start the Apache Tomcat Servlet server, follow these steps:

1. Make sure that the port number specified in the Servlet server configuration file is not used.
2. Run *$CATALINA_HOME/bin/startup.sh* on a Unix machine (or *$CATALINA_HOME/bin/startup.bat* for Windows).

(make the scripts executable if necessary through the command *chmod +x bin/*.sh*)

3- Accessing a Servlet-based web application

In order to access a web application hosted by the Servlet server, please do the following:

- Open a web browser and type the following URL *http://<host>:<port>/* where *<host>* and *<port>* are respectively the host name and the port on which the Servlet server is running (e.g., http://localhost:8080/).
- Follow the *Examples* (Developper Quick Start) that provides several Servlet examples. Run the HelloWorld example and understand the Java source.

4- Stopping the Servlet server

In order to stop the Apache Tomcat Servlet server, please do the following:

- Run *$CATALINA_HOME/bin/shutdown.sh* on a Unix machine (or *$CATALINA_HOME/bin/shutdown.bat* for Windows).

5. HelloWorld Servlet

1. Examine the directory content of *$CATALINA_HOME/webapps/*
    - *$CATALINA_HOME/webapps/* contains one sub-directory per web application (e.g. *$CATALINA_HOME/webapps/ROOT/* is the directory of the default web application, i.e. the application accessible via *http://<host>:<port>/*).
    - A web application deployed on the Servlet server is organized as follows :
        - ***.html, etc.** - The HTML pages, along with other files that must be visible to the client browser (such as images or text files) for your application.
        - **/WEB-INF/web.xml** - The *Web Application Deployment Descriptor* for your application. This is an XML file describing the servlets and other components that make up your application, along with any initialization parameters.
        - **/WEB-INF/classes/** - This directory contains any Java class files (and associated resources) required for your application, including both servlet and non-servlet classes, that are not combined into JAR files. If your classes are organized into Java packages, you must reflect this in the directory hierarchy under `/WEB-INF/classes/`. For example, a Java class named `com.mycompany.mypackage.MyServlet` would need to be stored in a file named `/WEB-INF/classes/com/mycompany/mypackage/MyServlet.class`.

        - **WEB-INF/lib/** - This directory contains JAR files that contain Java class files (and associated resources) required for your application, such as third party class libraries or JDBC drivers.
    - Examine, in particular, the content of the following files or directories: *$CATALINA_HOME/webapps/ROOT/index.html*, *$CATALINA_HOME/webapps/ROOT/WEB-INF/web.xml*, *$CATALINA_HOME/webapps/ROOT/classes/*
2. Verify that any servlet can be accessed through an url of the form http://host:port/<application-name>/<url-servlet>, where url-servlet is specified in the url-pattern field of the web.xml file.

   (example : http://localhost:8080/examples/servlets/servlet/HelloWorldExample)

6. Defining a new servlet

1. In the WEB-INF/classes directory, copy the HelloWorldExample class in a file named HelloToExample class
2. Modify the HelloToExample class such that the servlet receives a GET request with two parameters : a name and a surname (example : http://localhost:8080/examples/servlets/servlet/HelloToExample?firstname=paul&lastname=durand). The servlet produces an HTML response including a sentence like this: hello Paul Durand
3. Compile your newly defined servlet class (do not forget to reference servlet-api.jar from *$CATALINA_HOME/bin in your classpath)*
4. Modify accordingly the web.xml file
5. Run your newly defined servlet

Remark : any time you modify your servlet class, you need to stop and restart the servlet server to take into account the new class file. When developping and debugging servlets, this may be cumbersome. To avoid this, Tomcat allows to set a flag named reloadable to true, ensuring that the server will periodically check for servlet classes modifications and reload the necessary classes. To do this, insert the following directive in your web.xml file associated to the example application :

```
<Context reloadable="true"/>
```

7. Come back

1. Modify the application in order for a web client to be able, from the web page resulting from a "Hello World" request, to return back to the *http://<host>:<port>* main page (i.e. the *index.html* page).

8. Session management

Sessions:

1. Execute the SessionExample servlet from the Examples
2. Understand the Java source of the SessionExample servlet
3. Shut down the Servlet server (while keeping the web browser running), and reload the SessionServlet several times in the same web browser. What are the produced results? How do you explain it?
4. Modify your HelloToExample servlet to integrate in the HTML response the number of time the servlet has been invoked from the same client
5. Run the newly defined servlet

Cookies:

1. Execute the CookieExample servlet from the Examples
2. Understand the Java source of the CookieExample servlet
3. Shut down the Servlet server (while keeping the web browser running), and reload the CookieServlet several times in the same web browser. What are the produced results? How do you explain it?
4. Modify your HelloToExample servlet to integrate in the HTML response the number of time the servlet has been invoked from the same client, but use cookies instead of sessions.
5. Run the newly defined servlet