Servlet examples

Here is a description of how to use the two programs named MyHttpServer0 and MyHttpServer1. Both programs require that directory C:\Temp\WebContent includes the following files:

* static.html, used by both programs as a static resource. It can be loaded by issuing a request to <http://localhost:8080/static.html>
* HelloWorld.class, used by MyHttpServer0 as a servlet. It can activated by issuing a request to <http://localhost:8080/servlet/HelloWorld>
* Myservlet directory, used by MyHttpServer1 as a servlet. It can activated by issuing a request to <http://localhost:8080/servlet/myservlet>

The folder named WebContent can be taken as a reference. Copy it to C:\Temp\WebContent.

The folder named HelloWorld includes the source code of the HelloWorld servlet as well as two scripts. Script makeHelloWorld compiles the HelloWorld.java program and produces the HelloWorld.class file. Script

MyHttpServer0, the first of the two programs, aims at showing the distinction between static web access and dynamic web access. When receiving a request, the server checks whether the request is associated to static content or to dynamic content and activates the appropriate processor. In case of a request associated to dynamic content, the program loads the required class from the servlet repository (C:\Temp\WebContent\servletrepository).

For example, if <http://localhost:8080/servlet/HelloWorld> is indicated as a target URL, the program loads the HelloWorld.class class and activates it.

MyHttpServer1 is an evolution toward a platform, i.e., a servlet container. In particular, it distinguishes **servlet management** from **servlet activation**. A Management Console, based on a CLI, allows loading/unloading servlets, where each servlet is associated to a different folder in C:/Temp/WebContent/servletrepository. Each servlet folder, for example the myservlet folder, includes two files, the first associated to the servlet class (HelloWorld.class), and the second associated to the servlet metadata (named metadata.txt). The metadata.txt file contains the class name.

You can test MyHttpServer0 as follows. Run the project on Eclipse. The program does not require commands. Just run it. Then run a browser and access the following URLS:

<http://localhost:8080/static.html> to retrieve the static content

<http://localhost:8080/servlet/HelloWorld> to activate the servlet

You can test MyHttpServer1 as follows. Run the project on Eclipse. A CLI should appear in the management console. The CLI accepts three commands, namely load <servlet> , unload <servlet> and quit. If you do not type any command you are able to access the static content in the same way as in the previous case, but you cannot access any servlet.

Now load a servlet through the following command

load myservlet

Then you should be able to access the servlet on a browser through the following URL:

<http://localhost:8080/servlet/myservlet>

Remember that when you activate servlet myservlet, you are activating class HelloWorld.

Finally, you can unload the servlet and see that you cannot access it any more.