UNIVERSITA' DI GENOVA – LM IN COMPUTER ENGINEERING
SOFTWARE PLATFORMS - Prof. Massimo Maresca

Homework on Platform Enabling Technologies:

Step 1: Implement a **Thread Pool Based Static Web Server** (tpbsws). The Web Server assigns each request to a new task and submits it to a Thread Pool having size = 2;.
Prepare a folder called **SP** and place the Web Server executable jar file, called **tpbsws.jar**, and a subfolder called "**staticcontentrepository**" containing the HTML files to be downloaded in such a folder. Create an HTML file called "file1.html" to test the server and place it in the "staticcontentrepository" subfolder.

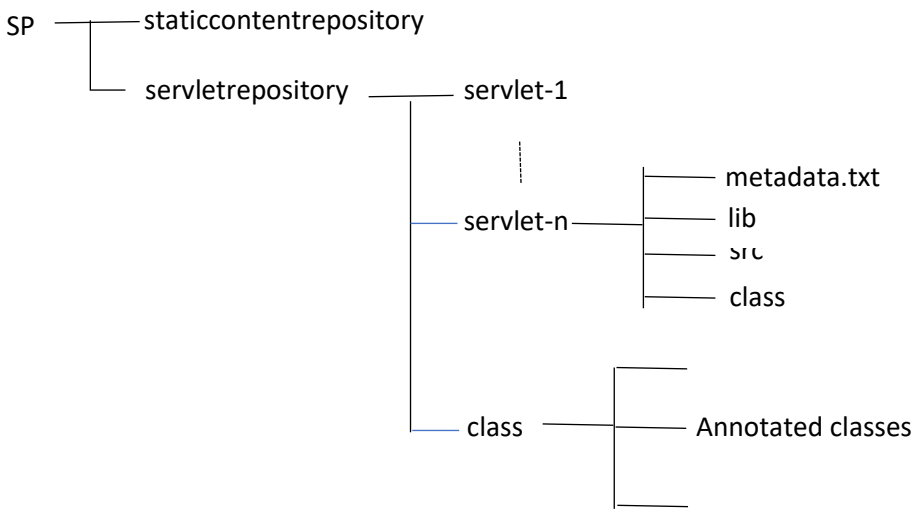
Step 2: Extend the Thread Pool Based Static Web Server to a **Thread Pool Based Servlet Container** (tpbsc), able to process both static content requests and servlet activation requests. These latter requests are triggered by the "servlet" keyword in the URL request string (e.g., http://localhost:7654/servlet/<servlet name>).
Servlets must be associated to subdirectories in an external servlet repository located in an external servlet repository, i.e., a folder called "**servletrepository**", to be placed in the SP folder. The name of the servlet folder inside the external servlet repository is used by clients as a servlet id in the URL request string. The servlet folder must include a file called "**metadata.txt**", which contains the name of the servlet class in the form: **ServletClassName=<class name>**, a subfolder called "**class**", which contains the executable class, a subfolder called "**lib**", which includes the required libraries if needed, and a folder called "**src**", which includes the servlet source files.
The Thread Pool Based Servlet Container must include a **Management Console**, running in a separate thread, in addition to the main thread listening on port 7654. Through such a console the container administrator populates and manages an internal servlet repository (e.g., a Hashtable). The Management Console must support the following commands: "**load <servlet name>**", "**remove <servlet name>**", "**list**", where <servlet name> refers to the name of the servlet folder. The "load" command checks that the servlet requested 1. is in the external servlet repository and 2. is not in the internal servlet repository. If both checks give positive results, then it loads the servlet class in the internal servlet repository, otherwise it issues a notification. The "remove" command checks that the servlet to be removed is in the internal servlet repository. If the check gives a positive result, then it removes the servlet, otherwise it issues a notification. Finally, the "list" command lists the name of the servlets in the internal servlet repository.
Create an executable jar file called **tpbsc.jar** and place it in the SP folder. Create a servlet called **myservlet** and place it in the external repository (SP/servletrepository). The servlet source file, written in Java, must print the English Alphabet (A..Z) one letter at a time at a pace of one letter every 400 msecs, so that the whole print takes about 10 seconds.

Step 3: Upgrade the Management Console in such a way to make it able to manage the loading of servlets in which the association between the servlet class name and the URL request string is expressed in the form of an annotation. Upgrade the existing servlet code in such a way that it includes the following annotation: @MyAnnotation (name = "URLServletName", value =

"myservlet2"). The annotation associates the URL request string to the executable class name in the class folder.
Create an executable jar file called **tpbsca.jar** and place it in the SP folder. Place the annotated servlet source code in a subfolder named after the class name in the external servlet repository and compile it. Place the servlet .class file in the **class** folder in the "**servletrepository**" folder.
Include a new command named "**load-with-annotations <servlet-name>**" which loads the .class file included in the "**class**" folder in the "servletrepository" folder.

In the end students are supposed to deliver just an executable jar file, which includes the software components that they were able to complete. The executable jar must be included in a folder called **SP** which also includes a folder called "**staticcontentrepository**", hosting the html files, and a folder called "**servletrepository"**. The servlet repository includes the servlet folders as well as a folder named **class**, hosting the annotated classes.

Here is the folder scheme.



Program writing, debugging and testing can be done using either a CLI or any IDE, e.g., Eclipse.