Workshop 1: A Request Based Approach, the Java Servlet API

DAT076/DIT126 Chalmers/Göteborgs Universitet
Joachim von Hacht

Objectives

The goal for this workshop is to expose the product catalogue of the shop model as a web application using the request based approach.

You need the following tools and skills;

- Environment: NetBeans IDE (including Maven, Tomcat Server) or similar.
- Basic HTML, CSS and JavaScript.
- Bootstrap.
- JEE Web applications and the Servlet API (Servlets, Java Server Pages (JSP), JSP Standard Template Library (JSTL).
- The Front Controller design pattern.
- The Post-Get-Request (PRG) pattern.

Please, have a look at the code samples, everything you need should be there. Also links in slides for more detailed information.

Watch the demo

To see the intended functionality watch the video, see course page.

The Shop model

The core shop model is fully implemented (nothing to do, just use it).

1. Download the model from course page > Workshops. Unzip and open project in NetBeans (it's a Maven Java standalone application).
2. Inspect project. Read README file
3. Switch of testing for ordinary Maven builds: Tools > Options > Java > Maven, check “Skip Tests for any....”

4. Build the project. Will install the model into your local Maven repo (~/.m2 directory) as shop-1.0-SNAPSHOT.jar. Try to find it. When installed in repo it's possible for other applications to add a Maven-dependency on the model (we will use the same model in all workshops).
5. Inspect shop-1.0-SNAPSHOT.jar in NetBeans Files view (tab). Important to check that everything really is included in the jar (more important later).
6. There are a few tests. Run the tests.

Tomcat

Tomcat is our server for now (Servlet container).

1. In NetBeans: Inspect Services-tab > Servers. You should find an Apache Tomcat server. Right click icon > Properties, inspect.
   Note If "Enable HTTP Monitor" is checked it's possible to inspect incoming HTTP requests in HTTP Server Monitor window in NetBeans (pops up at run). Very useful.
2. Start Tomcat, right click > Start. Use a browser to visit http://localhost:8084 (default for Tomcat admin pages).
   Note: There are always one or two administrative applications running in Tomcat, shown as / and /manager. Don't touch!
3. Stop Tomcat.

Web application

1. Download the web application skeleton from course page > Workshops. Unzip and open project in NetBeans (it's a Maven Java Web application). There are probably warnings or errors, ignore for now.
4. There should be a dependency on the shop model. Inspect pom.xml.
5. Inspect the Java version: Mark project, right click > Properties > Sources and Compile. Should indicate Java 8 (1.8).
6. Build the project (Maven possibly will download a lot, … be patient).
7. Now all warnings and errors should be gone, if not contact assistant.
8. Inspect generated .jar file in NetBeans > Files view.
9. Select server. Mark project > Properties > Run > Select Tomcat (possibly already selected).
10. Mark project > Run. Tomcat should start (log windows opens in NetBeans) and the application welcome page (see web.xml) should show up in default browser (adjust browser Tools > Options > General > Web Browser)
11. Try to access different JSP's using the browser address field. Conclusions?
12. Compare browser address field with content in file Web Pages/META-INF/context.xml.
13. Change path in context.xml and run again. As expected? Reset!

NetBeans Notes

- To speed up the deployment of the project, specially when testing small changes to server side code, use “deploy on save”, NetBeans will compile and deploy the application on every save (at severe exceptions possibly have to Build/Run again).
    - In an existing application. Mark application, right click > Select Properties > Run > check Deploy on Save.
    - Select Properties > Build > Compile > Compile On Save: For both application and test…
- **Use clean and build generously**. NetBeans seems to cache very hard ... i.e. unclean builds.
    - Possible delete manually: Files > servlet_shop > target directory to force a clean build.
- Indexing of local Maven repo is very time consuming. Select Tools > Options > Java > Maven > Index Update Frequency: Never
- It's possible to debug a Web application but it's somewhat heavy.
- Use of Logger in applications is recommended (not using System.out). Point into some editor window, right click > Insert code ... > Logger .... Use like below (in general let NetBeans generate as much code as possible):

```
LOG.log(Level.INFO, ".. some message... {0}", somevalue);
```

Implementing missing parts

You need to grasp the FrontController and PRG-Pattern, see lecture slides.
Start implementing missing parts. Files to work in (see //TODO in code)

- ProductServlet (probably best to start here, add some simple GET handling test and develop incrementally)
- products.jsp (next make this one work together with the Servlet)
- delProduct.jsp
- editProducts.jsp

Have fun … (optional)

Add authorization or something else … . For authorization see slides

NetBeans Project structure