CS480 Software Engineering
Assignment 5 - Maven Exercise

Due Date
Sunday, February 8, 2015

Score
5

Questions and Directions

Note: This is an individual assignment for each team member. If you are not using Java Spring, you should do something equivalent with other build tools or libraries.

1. Add a 3rd party Java libraries to your team project using Maven

You can add any library dependency for your project. Ideally, you want to find and include libraries that are useful for your team project. However, if it is too early to decide or you do not know what good libraries are available, you can just use some unrelated libraries such as JSoup as we demoed in the class for the purpose of practicing Maven. Here are some popular Java libraries (feel free to Google other interesting ones):

- JSoup (A popular Java library to parse HTML pages) http://jsoup.org/
- Commons IO (a library of utilities to assist with developing IO functionalities) http://commons.apache.org/proper/commons-io/
- Commons Math (a library of lightweight, self-contained mathematics and statistics components) http://commons.apache.org/proper/commons-math/
- Google Guava (contains several of Google's core libraries that we rely on in our Java-based projects: collections, caching, primitives support, concurrency libraries, common annotations, string processing, I/O, and so forth) https://code.google.com/p/guava-libraries/

Of course, we would like to use Maven to automatically configure and add the libraries to our project dependencies. You need to search the Maven package dependency information for the library, and modify the `pom.xml` file in your web project. The official maven package repository is located at: http://search.maven.org/

You need to commit your change on the `pom.xml` to GitHub. Team members should NOT use the same library.

2. Add a new method that uses your library

Similar to Assignment 3, you need to add a new method in the `WebController.java` (add a new HTTP URL), which does something interesting with the library you added from Maven (e.g., print out all the hyperlinks in a certain web page if you use JSoup).

You can google tons of code examples for your library. For instance, the example we used during the class is at: http://www.mkyong.com/java/jsoup-html-parser-hello-world-examples/

You don't need to spend a lot of time on learning the new library, since the main goal here is to ensure that you can use the Maven to import libraries and integrate them with the existing web service. A simple hello world example would be enough.

Again, you need to commit the changes on the new method to GitHub.

3. Practice Maven build lifecycles

Make sure you have installed Maven command line tool (http://maven.apache.org/). You can run command "mvn" in your iphoto-web folder to verify it. Make sure you have "PATH" and "JAVA_HOME" environment variables configured correctly.

If it runs correctly, you can practice and try the following different Maven build goals:

```
mvn compile
mvn package
```

With `mvn package`, you should be able to see the generated jar file in `target` folder. Then, try to run your project in the shell through the following command (make sure you have JDK installed and configured):

```
java -jar demo-web-project-1.0.jar
```

(note: your project file name may be different)

You should be able to see the running web application through command above. Go to your web browser to verify it with the new URL you just added.

You do not need to submit anything for Question 3. However, make sure you can run and build your project in the command-line with Maven. The future assignments will be depending on this step.