Homework for Maven

Assignment:

For this week's assignment I would like you to take the basic ReST service project from earlier in the semester and convert it to a Maven based project. This means adding an appropriate pom.xml file, structuring the project in accordance with that pom file and replacing the physical libraries added to the project with dependency references in your pom.

To ensure that you are all working with the "same" template for maven, I will be providing an Eclipse based solution that you MUST USE as a starting point for this assignment. Since this solution will be for the previous assignment, I will be providing it to you, individually, on Saturday after the deadline for the ReST assigment when I receive your submission. Note that you will not receive this template until after you complete and submit the previous assignment, so it would benefit you to turn it in on time.

I will provide some assistance for the steps necessary, which are relatively straightforward with a tool like Eclipse. If you don't have an Eclipse version of the assignment from last week I would suggest you create an empty dynamic web project to start and then move the relevant components (source code, web content, web.xml) over manually.

You can also complete this assignment without Eclipse, using the command line to generate a maven project using the maven-archetype-webapp template. You would then need to move your project components into that project, updating the directory since it does not have a java directory to start, and then manually updating the pom.xml file. However the pom file that is generated is far less complete for this assignment then the one generated through the conversion of a Dynamic Web Project in Eclipse to a Maven project and would require the addition and configuration of plugins in addition to dependencies. Something that is not trivial.

Here are the overall steps for converting your Eclipse Dynamic Web project to a Maven project:

1. Prepare your environment
    1. Make sure that you have maven installed (as per the lecture notes).
    2. You MUST add the BhcUtils.jar to your local repository so it can be added to the project as a maven dependency using the following command:

        ```
        mvn install:install-file -Dfile=path-to-your-BhcUtils.jar -DgroupId=edu.jhu.rfs -DartifactId=BhcUtils -Dversion=1.0 -Dpackaging=jar
        ```

        Make sure that you used the EXACT values for groupId, artifactId and version so they match the dependency information in your pom file and will ultimately be represented the same way in my repository when I test the build.
2. Open your project and right click on the overall project item in the project explorer pane. Follow the contextual menu that appears to the Configure item near the bottom and in that sub-menu select "Convert to Maven Project".
3. After converting, remove all of the libraries that you added to support jersey and jax-rs from your project.
4. Remove all references to Apache Tomcat in your project. First from the build path (right click on project, contextual menu to build path and configure build path) and then as a runtime for the project. (right click on project, contextual menu to Project Facets, select the Runtimes tab on right and deselect Apache Tomcat). You should also remove the BhcUtils.jar file from the project as it is provided to maven through the local repository.
5. Now you need to add dependencies for the crucial jar files that you removed (note that not all of those removed were critical). I would like you to add three dependencies to your pom file. jersey-container-servlet-core version 2.43, jersey-hk2 version 2.43 and the BhcUtils.jar file that was provided as part of your project (use the groupId, artifactId and version that you used in the repository install in the earlier step). You can look up the appropriate maven details (Group Id, Artifact Id) using google for the other libraries.
6. You can update your project now to make sure there are no issues. Right click on the project, contextual menu to Maven, then Update Project. Or you can just use the Alt+F5 key combination.
7. If all checks out you can then build your war file by right clicking on your project, contextual menu to Run As, Maven Install. This will compile and build your war file. You will see maven output in the Console window in Eclipse. If all is good the last line will say Build Successful.
8. Your resulting war file will be in the target directory that was created under the project. You can test it locally, by physically copying or moving it to your local Tomcat webapps folder, starting Tomcat and walking through your links. Remember the name of the war file will precede the other path elements to your pages and service in the URL.

Test Cases:

I will not be running several test cases since I provided the initial template. I am not looking in detail to see if your running application executes all of the positive and negative cases indicated in previous assignments. I will likely just run one postive case through your converted code to see that it runs (or not).

Submission:

You should deploy your tested war file to the class server upon completion using the naming pattern identified in the syllabus. Please don't forget to change the host in any links before doing so and for your own good, test it on the class server after deploying it there.

What you submit will be used to assess your grade on the assignment. For your Canvas submission I DO NOT want you to just submit your entire project folder. I am expecting a zip file with the minimal maven project comprised of ONLY the necessary components (source code (java and client side), configuration (web.xml) and the pom.xml file. There should be NO inluded jar files as they are pulled by Maven during the build and there should be NO other configuration files from the IDE. Make sure it's structured properly so I can just deploy your project for testing using maven (install) from the command line inside your project folder. You will lose credit if you provide unecessary folders, jar files, and anything not necessary for my testing. I also expect you to post the URL of your deployed project to Canvas as well.

Grading:

Grading for this assignment will be slightly different that for all previous assignments. There will be three "starting points" in the grading. If I can build (using maven)and run your Canvas submission, you will be starting with a score of 100% before other deductions. If I can build (using maven) your submission, but it doesn't run locally for me, you will be starting with a 90% before other deductions. If I can neither build (using maven) nor run your submission locally, you will be starting with an 80% prior to other deductions.

Other deductions will be assessed for additional items including (but not limited to):

1. Inneficiencies in your POM file (aka unneeded dependencies and/or plugins or other items that add no value to the maven project.
2. Providing additional items in your project that are not necessary for the maven build
3. Your class server deployment does not run
4. Failing to provide a URL in the text area of your Canvas submission