Spring Boot Fundamentals Lab File Setup

This document describes the steps necessary to import the lab and sample files for the SpringBootFundamentals class. The short story is that the code is a bunch of Maven projects. If you are familiar with how to import them into your IDE of choice, you probably don't need to read any further. If not, then the longer story is below with specific instructions for IntelliJ and Eclipse.

Before doing anything you need to clone the github repository at
https://github.com/anildi/SpringBootFundamentals.git

IntelliJ

1. This will give you a folder called **SpringBootFundamentals** which will contain two other folders called **code** and **presentation.**
2. Fire up IntelliJ and ask it to open the **SpringBootFundamentals** folder as a directory.
    1. File → Open → Choose the SpringBootFundamentals directory
3. When it comes up, it should show you the contents of the directory (the two other folders), but IntelliJ still does not know the there is Java code in this directory.
4. Now we need to have IntelliJ import a Maven project.
    1. Right click on the **code/PreSpring/pom.xml** and choose '**Add as Maven Project'.** It should be at the bottom of the menu
    2. Quick sanity check
        1. Open up the **src/test** folder in the **PreSpring** project.
        2. The **java** folder should be displayed in green. If not, then IntelliJ has not properly imported it.
        3. Right click on it and select **Run All Tests**
        4. At this point you may see an error dialog saying the the SDK is not specified for this module.
        5. Probably no need to panic though, because clicking **OK** will take you the Project Structure dialog, where you will have a drop down to select the SDK for the module.
        6. On this same screen, check the version that IntelliJ has chosen for the Project language level. For some strange reason, at least on my machine, it seems to default to Java 12. You need to make sure that the level agrees with the version of the JDK your are using. So, for JDK 11 the Language level should be 11.
        7. You should now theoretically be able to run the tests which should all pass.

8. If you are still seeing errors, check the **Troubleshooting** section below for some possible fixes.

Eclipse

1. Start up Eclipse and ask it to create a new workspace in some convenient location.
2. Import the projects
    1. File → Import… → Maven → Existing Maven Projects
    2. Using the **Browse** button at the top, point at the **code** directory.
    3. Eclipse should then automatically search recursively for Maven projects and import them into the workspace.

Troubleshooting

1. If you keep seeing complaints about missing libraries, and/or the IDE refuses to run anything, one possible issue may be with your maven settings. Maven uses a file called **settings.xml** which usually sits in your local Maven repository directory. You may have your settings.xml file set up to point at internal repositories. One possible workaround to see if this is the issue is to temporarily rename the settings.xml file to, for example, settings.xml.bak, and then try importing the projects again.

    It is, of course, **very important to remember that you did this and change the name back when you need to.**
2. In Eclipse it is sometimes necessary to do a Maven update after importing the projects. If you are seeing mysterious red marks with nothing obviously wrong in the code, try this:
    1. Right click on any project and go to **Maven → Update Project**
    2. This will bring up a dialog from where you can choose which projects you want to update.
    3. If you are seeing this issue on several projects, then try updating all of them.
3. Newer versions of Eclipse default to using JUnit 5 as the test runner. Both version are being used in different projects. If you try to run the **PostSpring** tests from the **src/test/java** directory and nothing happens, you may need to tell Eclipse to use a JUnit 4 test runner. Try this:
    1. Make sure you have tried to run the tests from the **src/test/java** directory at least once. This will make Eclipse create a default run configuration
    2. Right click on the **src/test/java** directory and choose **Run as → Run configurations…**
    3. The dialog that comes up has a drop down that allows you to choose the JUnit runner version. Choose JUnit4 and go click on **Run**

4. If all else fails, sometimes the good old “throw it all away and start from scratch” trick is worth a shot.
5. Those are some of the common issues and fixes that I have encountered so far. But, as we all know, with so many moving parts there can often be specific issues that need to be handled in specific ways. Please get in touch with me if you are stuck, and, of course, we will go over this in class as well.