Sample Spark Web-App

Overview

Follow along with these instructions using the sample webapp project provided to you. This guide will walk you through setting up your workspace, compiling and running a Java application from the command line using Maven, and importing the project for use with IntelliJ IDEA. Finally, you will be walked through the process of creating a GitHub repository and adding your project to it.

Prerequisites

Prior to beginning, ensure you have installed the following tools. Download links and guides are available on the Class Resources webpage.

- Java JDK 1.8 (download here)
- IntelliJ IDEA (download here)
- Maven (download here)

Instructions

Follow along with these instructions. If you run into any issues, see the Troubleshooting section for common issues you may run into, and how to resolve them.

Building and running the application from the command line using Maven

1. Download the sample webapp zip file provided to you on the course schedule. Extract all of the files in the zip into an easy to locate directory.
2. Using Command Prompt (Windows) or Terminal (Mac), navigate to the top level of the folder you extracted in Step 1. From there, use the following command to compile and execute the Java application: `mvn compile exec:java`
    a. "mvn" – run the Maven executable.
    b. "compile" – tells Maven to compile the application.
    c. "exec:java" – tells Maven to run the Java executable that is output from the compile operation.

    NOTE: the first time you run Maven on a new project it is likely that Maven will download dozens of plugins and other libraries used by the project. This is could take a few minutes and is perfectly normal.
3. Your project should now be running. Open a web browser and navigate to http://localhost:4567/ or http://127.0.0.1:4567/ to see it in action.

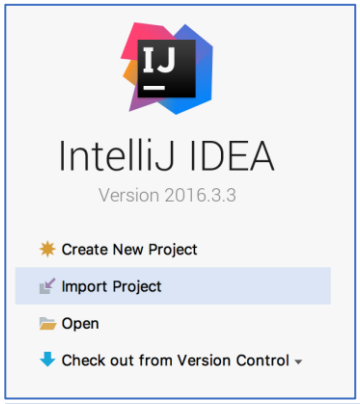
```
➜ guessing-game mvn compile exec:java
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building guessing-game 0.0.1-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ guessing-game ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 3 resources
[INFO]
[INFO] --- maven-compiler-plugin:3.6.0:compile (default-compile) @ guessing-game ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 7 source files to /Users/Shayde/SWEN_261/guessing-game/target/classes
[INFO]
[INFO] >>> exec-maven-plugin:1.2.1:java (default-cli) > validate @ guessing-game >>>
[INFO]
[INFO] <<< exec-maven-plugin:1.2.1:java (default-cli) < validate @ guessing-game <<<
[INFO]
[INFO] --- exec-maven-plugin:1.2.1:java (default-cli) @ guessing-game ---
[Thread-1] INFO spark.webserver.SparkServer - == Spark has ignited ...
[Thread-1] INFO spark.webserver.SparkServer - >> Listening on 0.0.0.0:4567
[Thread-1] INFO org.eclipse.jetty.server.Server - jetty-9.0.2.v20130417
[Thread-1] INFO org.eclipse.jetty.server.ServerConnector - Started ServerConnector@649ef709{HTTP/1.1}{0.0.0.0:4567}
```

Expected output from compiling and executing the project using Maven

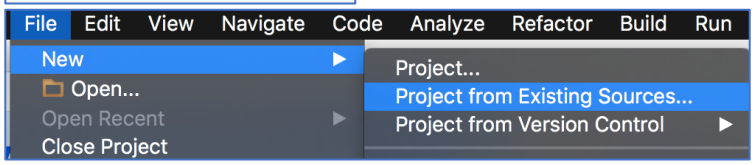
Importing the Maven project into IntelliJ IDEA

1. Launch the IntelliJ IDEA program. Select the option to **Import a project** using one of the methods in the images below.

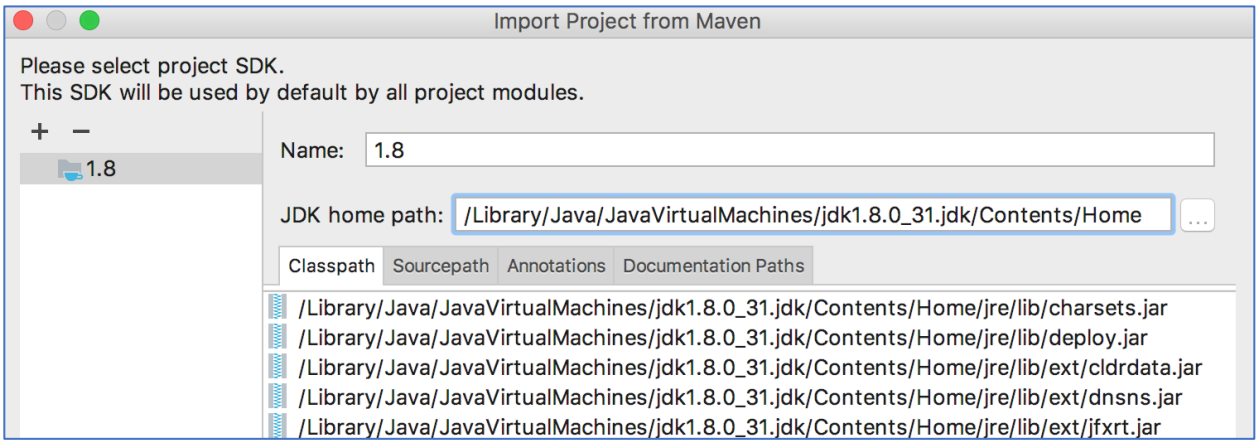
a.



b.



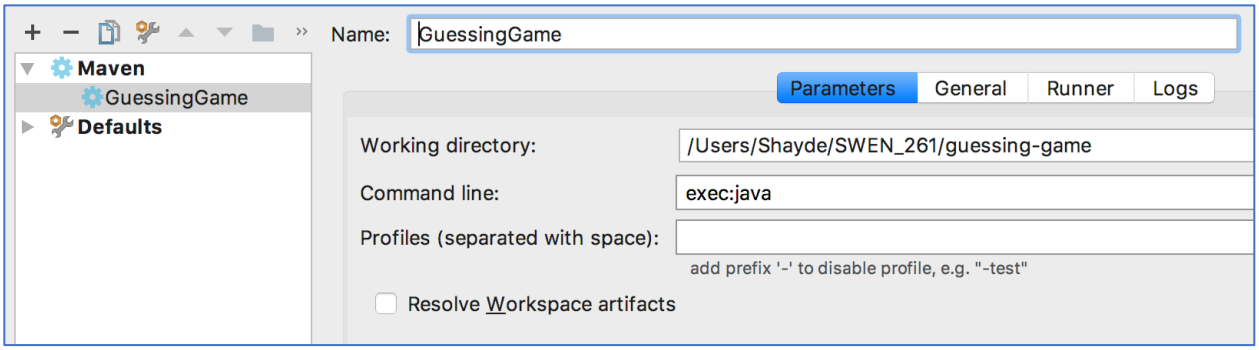
2. In the dialog window that appears, navigate to the *guessing-game* folder, select the `pom.xml` file and click 'Open'. The "Import Project from Maven" window should appear.
3. Leave the default settings as-is and click 'Next'.
4. Select the *com.example:guessing-game…* Maven project if it is not selected by default and click 'Next'.
5. Select the project SDK. For this class, we will be using the **Java 1.8 JDK**. If it isn't shown on the left, click the '+' button in the top left and select 'Java' to import it manually. After highlighting it on the left, click 'Next' to continue. *Note: the JDK home path should match your machine's JAVA_HOME path.*



6. Leave the project name and locations as-is and click 'Finish'. A `guessing-game.iml` project file will be created. The 'iml' file is a project file specific to IntelliJ. Select this file when you want to open the project in the IntelliJ IDE.

Compiling and Running the project using IntelliJ

1. Open the *guessing-game* project in IntelliJ using the `guessing-game.iml` file.
2. To **compile** your project, select 'Build module *guessing-game*' from the 'Build' menu.
3. To **run** your project, you're going to need to create a new Maven run configuration. To do this, click 'Edit Configurations' from the 'Run' menu.
4. Click the '+' button in the top left corner of the "Run/Debug Configurations" screen and select Maven.
5. In the editor screen that appears, fill in the information as follows:
    a. Name: **GuessingGame**
    b. Working Directory: *The default path should be the root of the project directory*
    c. Command line: **exec:java**
    d. Profiles: *Leave this line blank*



6. Click 'Apply', then 'OK'.
7. Run the project by opening the 'Run' menu from the toolbar and clicking 'Run'. Specify the run configuration we created above, **GuessingGame**. A console window should appear and the output should be like what you saw when running the project manually.
8. Your project should now be running. Open a web browser and navigate to http://localhost:4567/ or http://127.0.0.1:4567/ to see it in action.

Version Control using GitHub

In this section, we will walk through creating a private repository on GitHub and adding your Java project to that repository. To create private repositories, you will need to sign up for the GitHub Student Developer Pack. Note that it may take a few days to activate.

1. Begin by logging into your GitHub account. If you have not created one, you will need to do so before continuing.
2. From the main screen, click the '+' button in the **top right corner**, and select the option to create a new repository.
3. On the "create a new repository page", give the repository a name and select your account as the owner if it is not by default.
4. Change the repository type to **Private** by selecting its radio button.
5. Do **not** initialize the repository with a README – one has already been provided for you.
6. Do **not** add a `.gitignore` file – one has already been provided for you.
7. When you are ready, click the "Create repository" button to proceed. You should be redirected to your (empty) repository's main page.
8. Using Command Prompt (Windows) or Terminal (Mac), navigate to the root of your project's directory. From there, initialize Git version control by running:
   ```
   git init
   ```
9. Next, add the remote repository you created on GitHub using your project's URL by running:
   ```
   git remote add origin [HTTPS://GITHUB.COM/NAME/REPOSITORY]
   ```
10. To test this, try committing your README.md file and pushing it to the GitHub repository by using the following commands:
    ```
    git add README.md
    git commit -m README.md "created readme file"
    git push origin master
    ```
11. Enter your GitHub credentials, if prompted to.
12. After the push process completes, refresh your GitHub repository page and you should see your README.md file. Push the entirety of your local repository to GitHub by running these commands:
    ```
    git add --all
    git commit -am "initial commit"
    git push origin master
    ```

Tip: Don't want to type your GitHub credentials every time? Cache your GitHub password in Git by following this guide for Mac and Windows.

Troubleshooting

- **JAVA_HOME / Path** variable issues - If you get an error mentioning Java versions, or an inability to locate a Java file path, ensure that you have **Java 1.8** installed and that it is correctly specified in your JAVA_HOME/PATH variable. Oracle has a decent guide on this for Windows and Mac/Unix.
- A Maven error mentioning "no goals have been specified" or similar may mean you are not executing from the correct location. You'll want to run Maven commands from the root of the project directory containing the `pom.xml` file which specifies how the Maven project should be built.
- "Port number already in use" or similar errors indicate that the port being requested is in use by another process. This most likely means you already have an instance of the program running on your machine. You'll have to locate / terminate that running process before your new one will run.
- Having issues with Git / Github? Give GitHub's Interactive tutorial a shot. Additional tutorials are available on the Class Resources page.