**LAB 06: Maven intro**

# Ex 1: Maven Project Dependencies

For this exercise, you will prepare the Maven project for your course main project (you can work in groups, same group as in the course main project).

1. Create a maven project with the name of your course main project (see course main project description).
2. In src/main/java folder, create a Class with a main method, make sure you name your project using a namespace that reflects your team/company, for example: psu.se411.myprojectname

Example:

A screenshot of a computer

Description automatically generated

1. Import JavaFX into your maven project by adding a dependency and a plugin:

|  |  |
| --- | --- |
| Declaration of a dependency to JavaFX in pom.xml | 1. Add a dependencies section inside the project element   <project>  …........  **<dependencies>**  **</dependencies>**  </project>   1. Add the JavaFx dependency inside the dependencies element:   **<dependency>**  <groupId>org.openjfx</groupId>  <artifactId>javafx-controls</artifactId>  <version>23</version>  **</dependency>** |
| Declaration of a JavaFX plugin in pom.xml | 1. Add a build section inside the project section.   <project>  …........  <dependencies>  </dependencies>  **<build>**  **</build>**  </project>   1. Add a plugins section inside the build section:   **<build>**  <plugins>    </plugins>  **</build>**   1. Add the JavaFX plugin that has the “run” goal. Make sure you change the package name and the main class name accordingly.   **<plugin>**  <groupId>org.openjfx</groupId>  <artifactId>javafx-maven-plugin</artifactId>  <version>0.0.8</version>  **<configuration>**  <mainClass>packageName**.MainClass**</mainClass>  **</configuration>**  **</plugin>** |

3. Test your project by creating a very basic JavaFX application in your main class: Make your Main class inherit from the JavaFx Application class:

public class MainClass extends **Application ..**

Then implement the basic methods of a JavaFX Application: For Example:

public static void main(String[] args) {

  launch();

}

@Override

public void start(Stage primaryStage) {

  try {

    primaryStage.setTitle("My Project");

    primaryStage.show();

  } catch (Exception e) {

    e.printStackTrace();

  }

}

4. Finally, execute the clean and javafx:run goals using Maven. The window should show up.

# EX 2: Maven Site Plugin

Continue working of the same project.

You will add basic information to your project and generate a website as documentation:

1. In pom.xml add the following elements:

|  |  |
| --- | --- |
| 1. Add information about your project | <**name**>MyProject Name</**name**>  <**description**>This project is……  </**description**>  <**url**>https://yourProjectUrl.dom</**url**> |
| 1. Add information about the license of the product | <**licenses**>  <**license**>  <name>Apache License 2.0</name>  <url>https://www.apache.org/licenses/LICENSE-2.0</url>  </**license**>  </**licenses**> |
| 1. Add information about your company (here it is a team) | <**organization**>  <name>SE411-Tech-Company</name>  <url>https://mycompanyUrl.dom</url>  </**organization**> |
| 1. Add information about the developers (your team members) | <**developers**>  <**developer**>  <name>Salah Ahmed</name>  <email>salah.ahmed@domain.com</email>  </**developer**>  </**developers**> |

1. Generate the documentation website by running the basic site plugin: clean site
2. Inside the target/site folder, you should find an index.html page. Open it by right-clicking -> open with -> system editor.
3. (optional) open a generative AI tool (copilot, chatgpt, gemini, claude…) and find how you can improve your generated site by including other useful information.

**END**