Lab 9: JavaFX

Author: Yida Tao

Background

If you are using Java 8, JavaFX is already bundled with your JDK so you do not need to take any further steps. With the advent of Java 11 in 2018, however, JavaFX was removed from the JDK so that it could evolve at its own pace as an independent open-source project guided by Oracle and others in the OpenJFX community.

In this tutorial, we assume that you're using Java 11 or later.

Create a "Hello World" JavaFX Application

Prerequisite

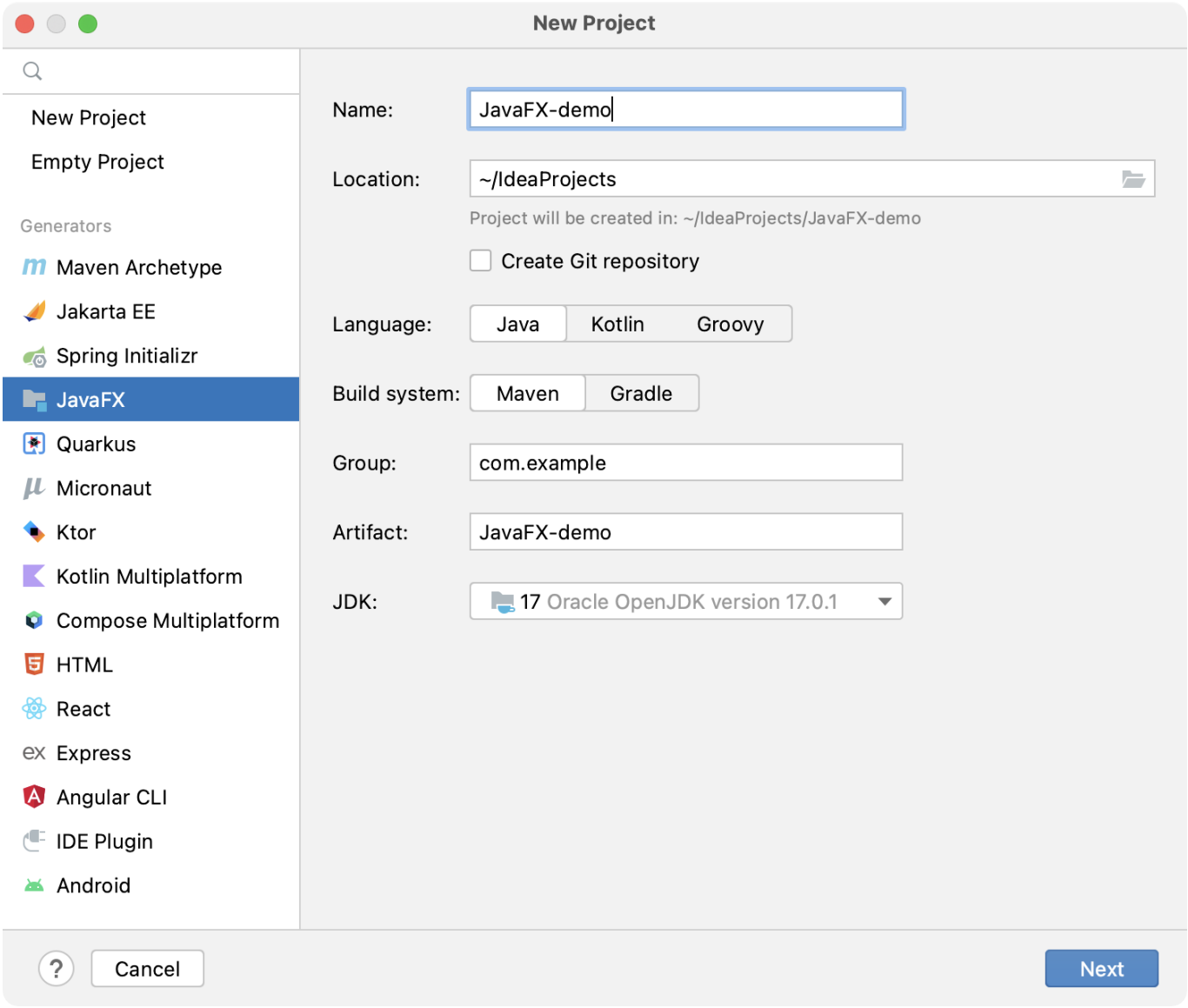
To be able to work with JavaFX in IntelliJ IDEA, the JavaFX bundled plugin must be enabled:

- In the `Settings/Preferences` dialog (Ctrl+Alt+S), select `Plugins`.
- Switch to the `Installed` tab and make sure that the JavaFX plugin is enabled. *If the plugin is disabled, select the checkbox next to it*.
- Apply the changes and close the dialog. Restart the IDE if prompted.

Create a new project

When you create a new JavaFX project, IntelliJ IDEA generates a fully configured sample application.

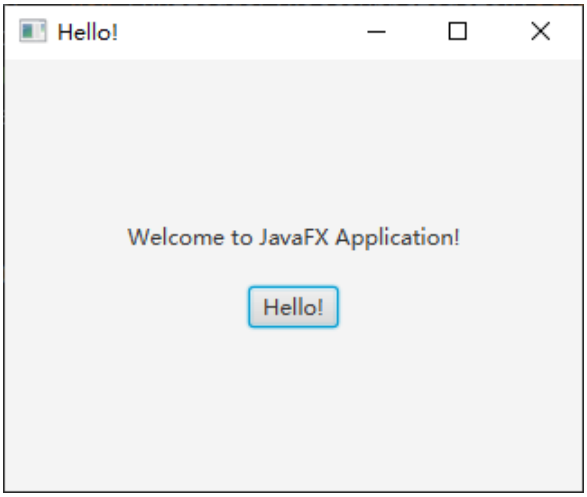
- Launch IntelliJ IDEA. Click `New Project`. Otherwise, from the main menu, select File | New | Project.
- From the Generators list on the left, select `JavaFX`.
- Name the new project, change its location if necessary, and select a language, and a build system.
- In the Group field, specify the name of the package that will be created together with the project.
- From the JDK list, select the JDK that you want to use in your project. If the JDK is installed on your computer, but not defined in the IDE, select Add JDK and specify the path to the JDK home directory. If you don't have the necessary JDK on your computer, select Download JDK.
- Click Next -> Create.



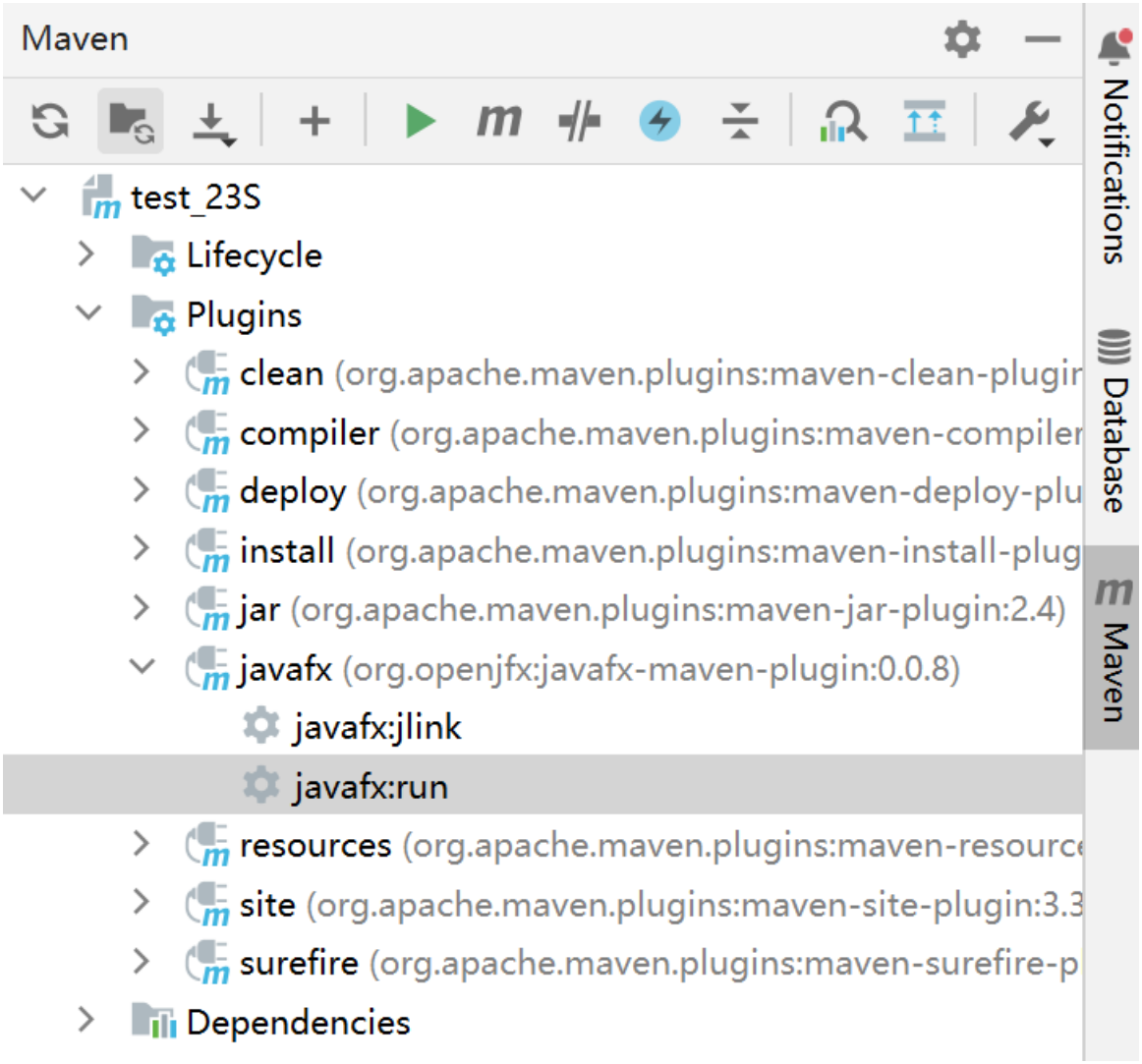
Run the application

Open the `HelloApplication.java` class, click the `Run application` icon in the gutter, and select `Run 'HelloApplication.main()'`. The IDE starts compiling your code.

When the compilation is complete, the application window appears. This means that the project is configured correctly and everything works as it should.



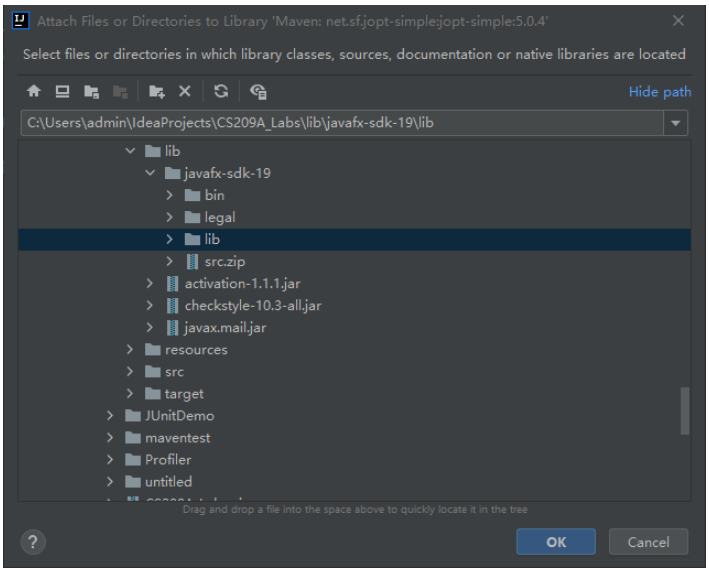
You may also open the "Maven" view, select the current project, then click "Plugins" -> "javafx" -> "javafx:run" to execute the program. Here, we are executing the javafx maven plugin's `run` goal, which is configured in `pom.xml` to execute the main class.



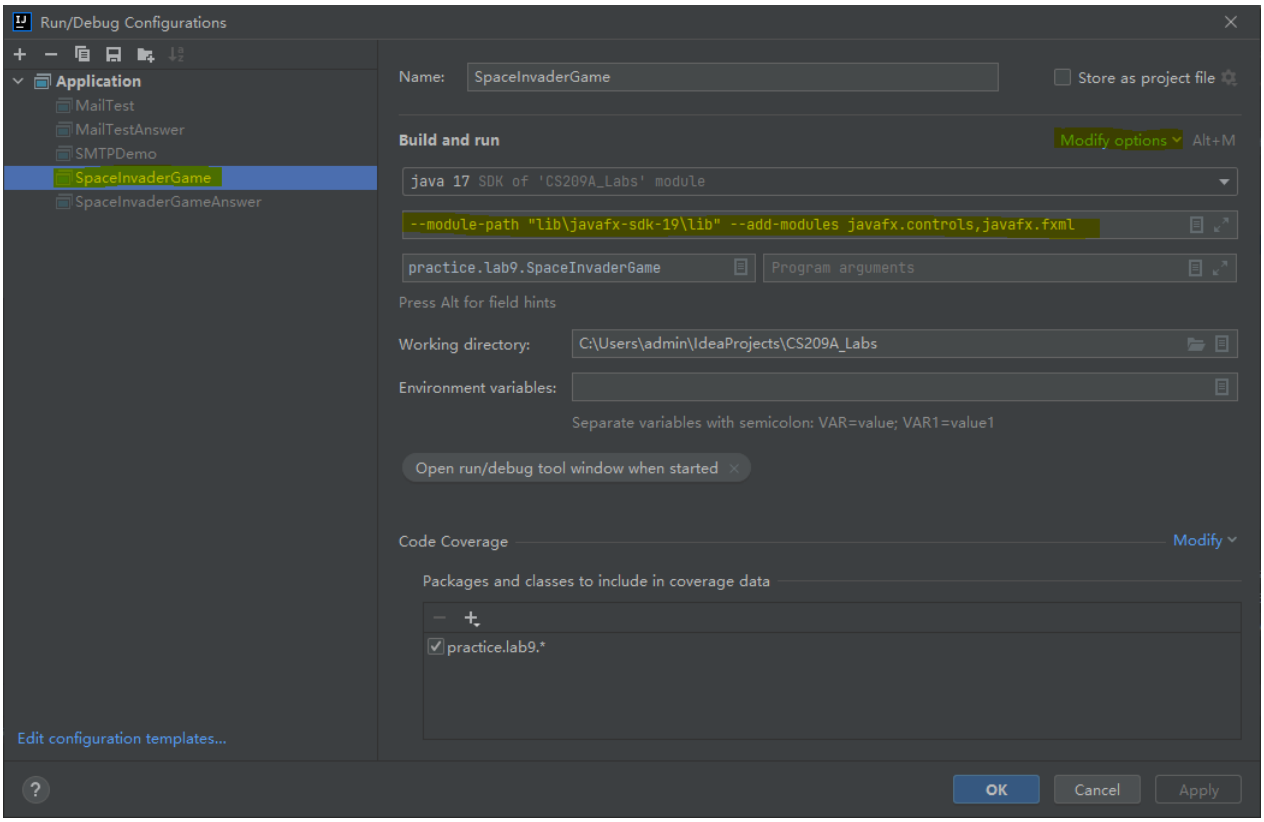
Add JavaFX Functionalities to an Existing Project

Suppose that you have an existing Java project, and you want to use JavaFX in it. You could also achieve this by manually downloading JavaFX and configuring the classpath.

- Download JavaFX SDK from the official site. For example, download the Windows/x86/SDK and unzip the downloaded zip for Windows users.
- In the existing project, click `File` -> `Project Structures` -> `Project Settings` -> `Libraries`, click `+`, find your downloaded javafx-sdk folder and select the `lib` subfolder.



- Click `Run` -> `Edit Configurations`, select your JavaFX class on the left, click `Modify options` -> `Add VM options`, and add `--module-path "your-path-to-sdk\javafx-sdk-19\lib" --add-modules javafx.controls,javafx.fxml`

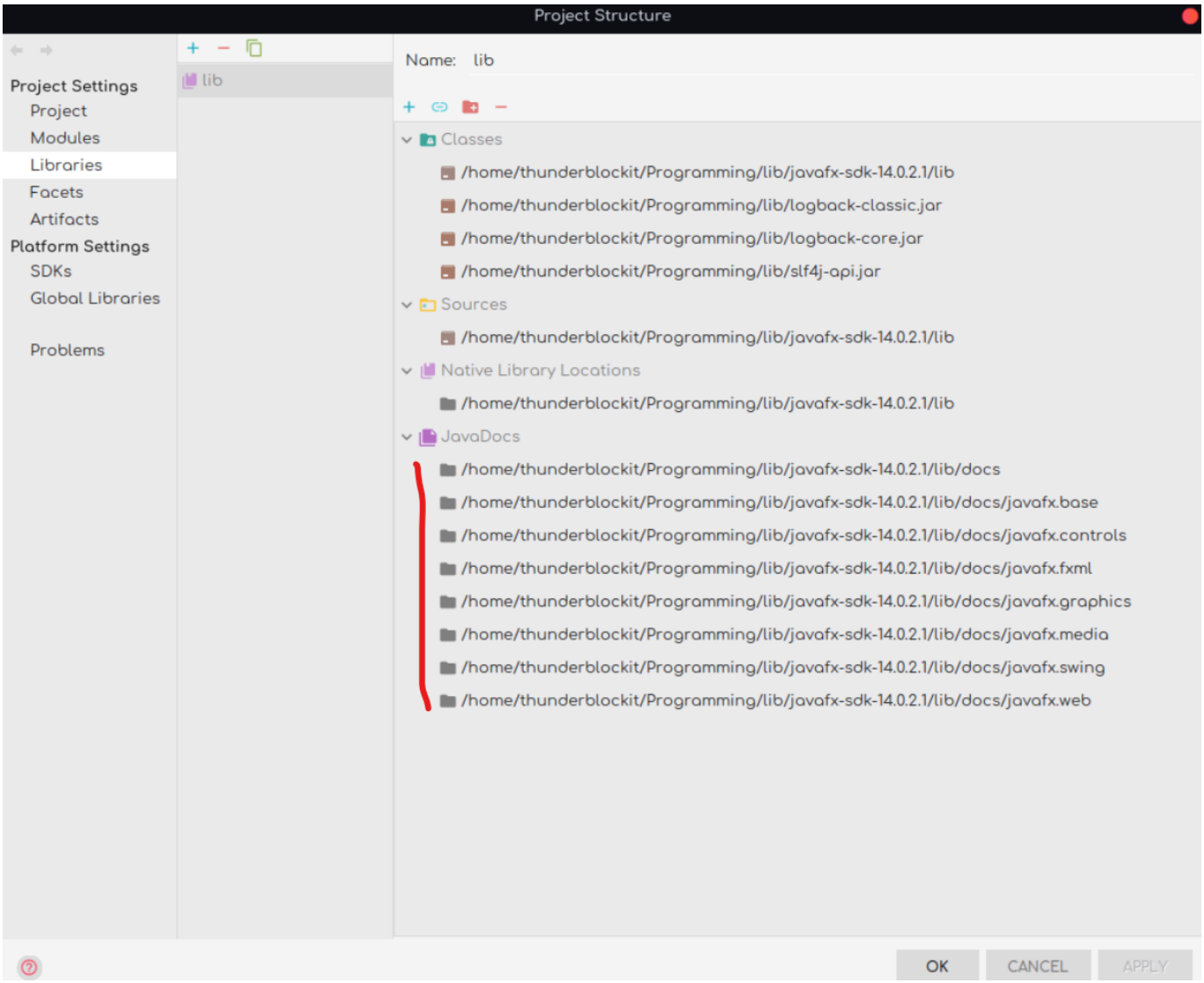


Now, you should be able to run your JavaFX code in this project.

Adding JavaDoc Support

Documentation is super helpful for us to learn a library. If you want to bring up JavaFX's documentation in IDEA, download the JavaFX's Javadoc here (be sure to use the same version as JavaFX SDK), unzip it, then add it as "JavaDocs" in `Project Structure` -> `Project Settings` -> `Libraries`.

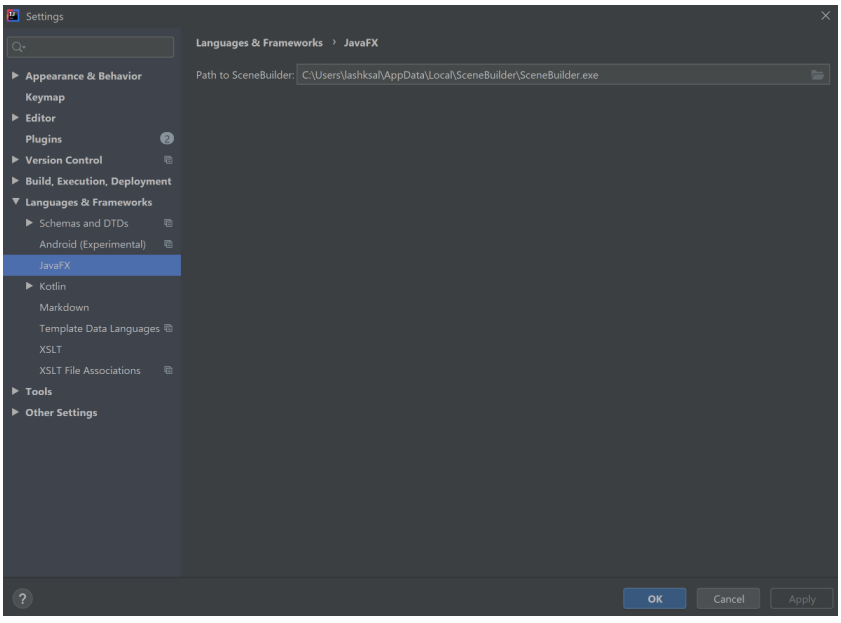
Once applying this configuration, whenever you hover your mouse to a certain JavaFX entity in IDEA, you should be able to see its JavaDoc automatically pops up.



Using Scene Builder for UI Design

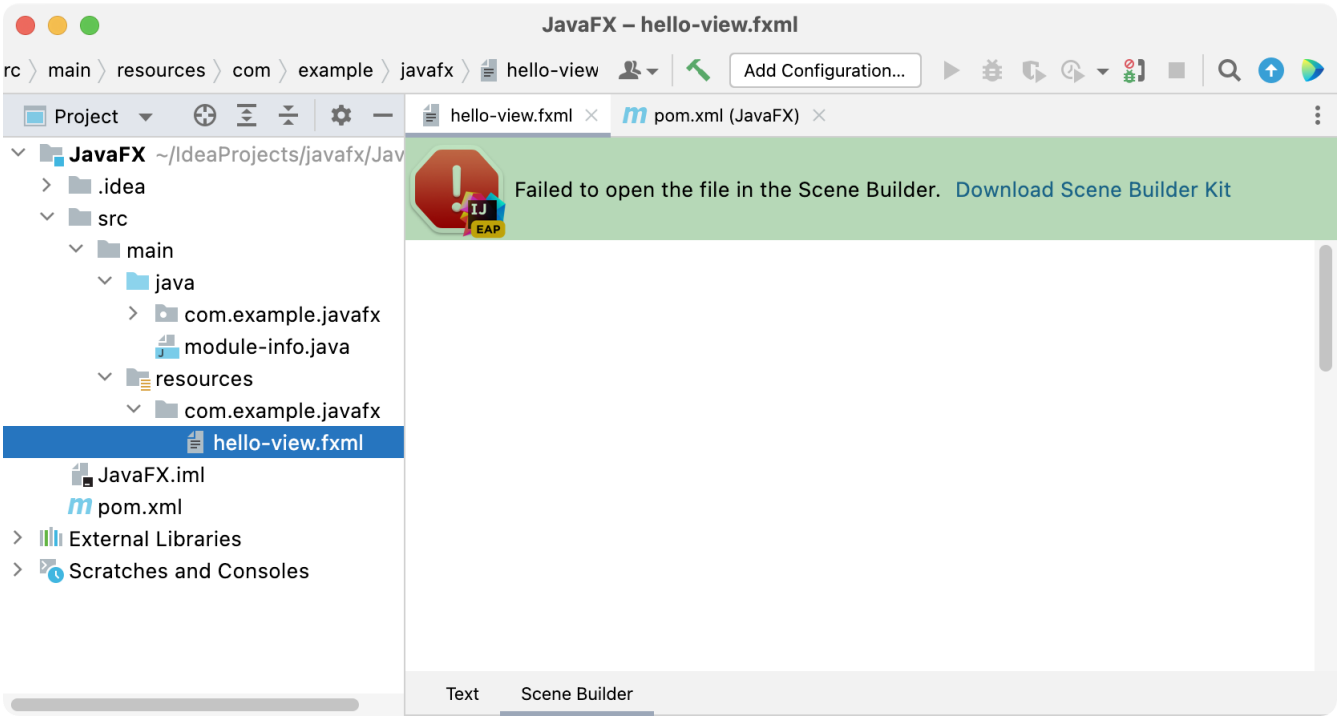
First, download Scene Builder here and install it.

Next, click "File" -> "Setting" -> "Language & Frameworks" -> "JavaFX", and specify the installation path for Scene Builder.

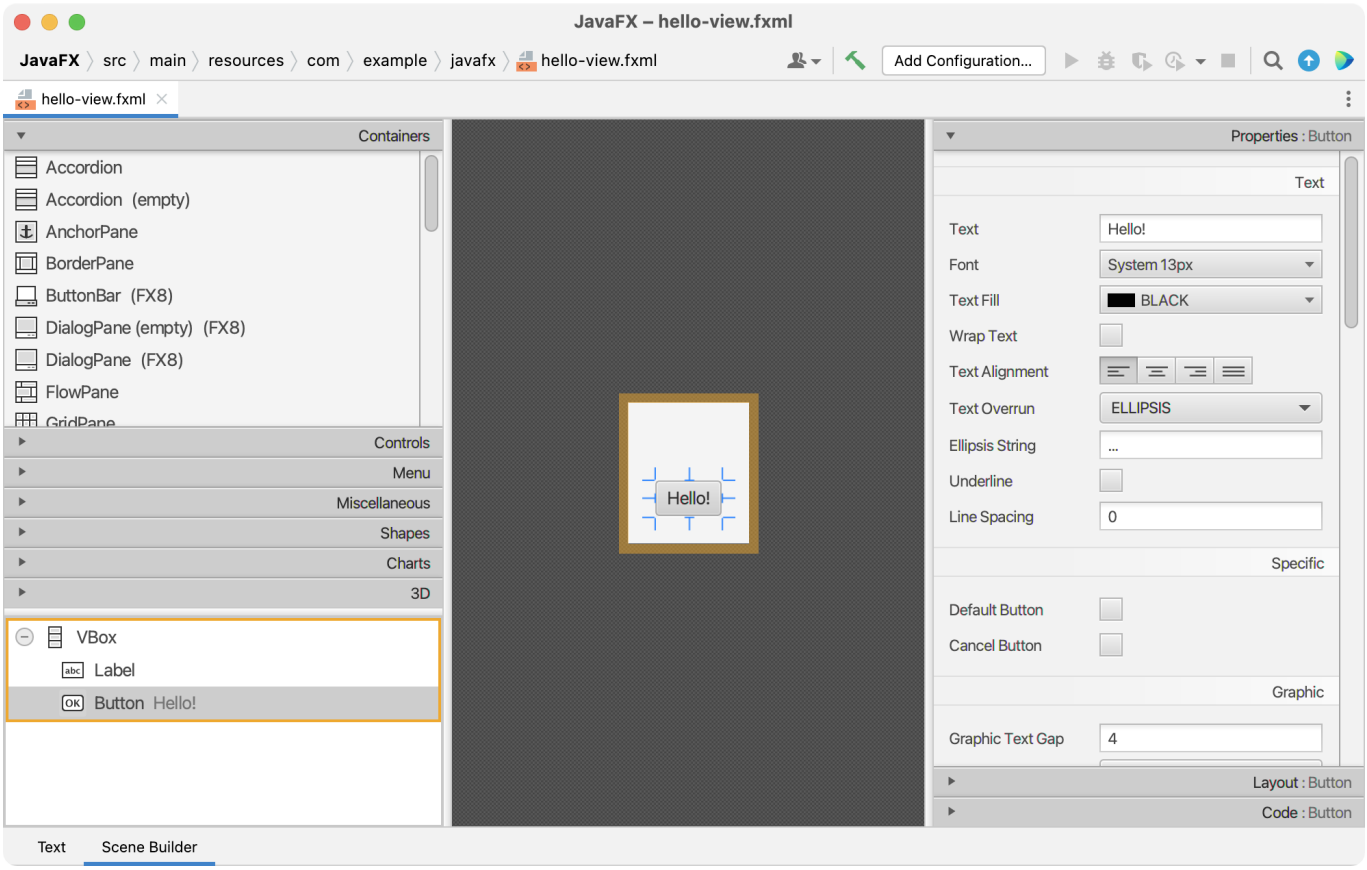


Now, when you open an `.fxml` file in the editor, there are two tabs underneath the editing area: the Text tab is for developing the markup, and the Scene Builder tab is for editing the file in Scene Builder.

If you cannot view `.fxml` in the Scene Builder tab, you'll see a notification like below. In that case, click `Download Scene Builder Kit` in the notification to download and install the tool.



Next, you could open the `.fxml` in the Scene Builder tab and design the UI visually.



If this still doesn't work, you could right-click .fxml and select "Open in Scene Builder", which will open the file in Scene Builder rather than inside IDEA.

JavaFX Concurrency

Run `ConcurrencyExample.java`, observe the results. The reason why the window hangs is because the JavaFX scene graph, which represents the graphical user interface of a JavaFX application, is not thread-safe and can only be accessed and modified from the UI thread also known as the `JavaFX Application thread`. As we're implementing a long-running task on the JavaFX Application thread, the UI becomes unresponsive until the task is done.

For this reason, we typically do long-running tasks on one or more **background threads** and inform the JavaFX Application thread to update the UI. `Platform.runLater` is one way to achieve this. In `ConcurrencyExample.java`, comment the bad-practice code and uncomment the good-practice code, then observe the execution results.

In addition, you could also use the APIs provided in `javafx.concurrent` package to enable the communication between background threads and the JavaFX Application thread. See here for a further introduction on this package.

Reference:

- https://www.jetbrains.com/help/idea/javafx.html
- https://www.jetbrains.com/help/idea/opening-fxml-files-in-javafx-scene-builder.html#open-in-scene-builder
- https://jenkov.com/tutorials/javafx/concurrency.html