CSCI 2120:
Software Design & Development II

UNIT4: UI management

GUI framework

Your First JavaFX Application

Overview

1. Introduction
2. The JavaFX Application Class
3. Implementing start()
4. Adding a main() Method
5. Adding a Scene

Introduction

In this lecture I will show you how to create your first JavaFX application. This lecturel thus serves both to introduce you to the core JavaFX concepts, as well as to give you a some JavaFX code you can use as template for your own experiments.

The JavaFX Application class

A JavaFX application needs a primary launch class. This class has to extend the `javafx.application.Application` class which is a standard class in Java since Java 8.

Here is an example subclass of `Application`:

```java
import javafx.application.Application;

public class MyFxApp extends Application {

}
```

Implementing start()

All subclasses of the JavaFX `Application` class must implement the abstract `start()` method of the `Application` class (or be an abstract subclass of `Application` itself).

The `start()` method is called when the JavaFX application is started. Here is the example with the `start()` method implemented:

```java
import javafx.application.Application;
import javafx.stage.Stage;

public class MyFxApp extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception {
        primaryStage.setTitle("My First JavaFX App");

        primaryStage.show();
    }

}
```

Code Explanation:

The `start()` method takes a single parameter of the type `Stage` . The stage is where all the visual parts of the JavaFX application are displayed. The `Stage` object is created for you by the JavaFX runtime.

This example sets a title on the stage object and then calls `show()` on it. That makes the JavaFX application visible in a window with the title in the top bar of the window.

If you do not call `show()` on the stage object, then nothing is visible. No window is opened. In case your JavaFX application does not become visible when launched, ensure that the Stage show() method is called from inside `start()`

Adding a main() Method

You can actually launch a JavaFX application without a main() method. But, if you want to pass command line parameters to the application you need to add a main() method. In general I prefer to add a main() method because it makes it more explicit which code launches the application.

Here is the example from above with a main() method added:

```java
import javafx.application.Application;
import javafx.stage.Stage;

public class MyFxApp extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception {
        primaryStage.setTitle("My First JavaFX App");
        primaryStage.show();
    }

    public static void main(String[] args) {
        Application.launch(args);
    }
}
```

Code Explanation:

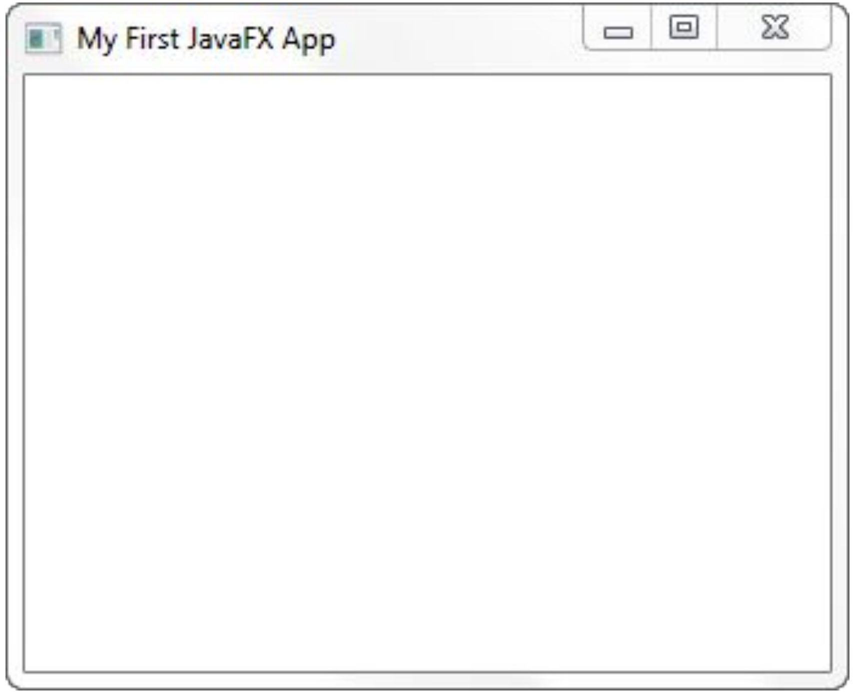
As you can see, the main() method calls the static `launch()` method with the command line parameters. The `launch()` method is a static method located in the `Application` class. This method launches the JavaFX runtime and your JavaFX application.

The `launch()` method will detect from which class it is called, so you don't have to tell it explicitly what class to launch.

That is all it takes to create a JavaFX application!

Adding a main() Method

Here is a screenshot of the window being opened as a result of running the above JavaFX application:



Adding a Scene

The previous JavaFX examples only open a window, but nothing is displayed inside this window. To display something inside the JavaFX application window you must add a `Scene` to the `Stage` object. This is done inside the `start()` method.

All components to be displayed inside a JavaFX application must be located inside a scene. The names for "stage" and "scene" are inspired by a theater. A stage can display multiple scenes, just like in a theater play. Similarly, a computer game could have a menu scene, a game scene, a game over scene, a high score scene etc.

Adding a Scene

Here is an example of how to add a `Scene` object to the `Stage` along with a simple `Label`:

```java
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.stage.Stage;

public class MyFxApp extends Application {
   @Override
   public void start(Stage primaryStage) throws Exception {
       primaryStage.setTitle("My First JavaFX App");

       Label label = new Label("Hello World, JavaFX !");
       Scene scene = new Scene(label, 400, 200);
       primaryStage.setScene(scene);

       primaryStage.show();
   }

   public static void main(String[] args) {
       Application.launch(args);
   }
}
```

Code Explanation:

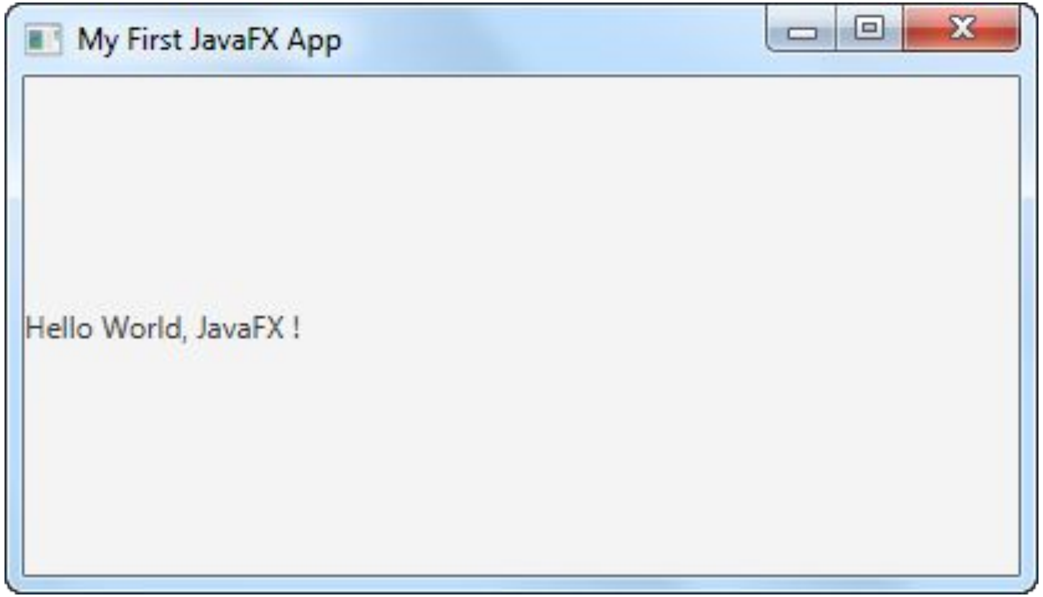
Three lines have been added to this example. First a `Label` object is created. Then a `Scene` object is created, passing the `Label` as parameter along with two parameters representing the width and height of the scene.

The first parameter of the `Scene` constructor is the root element of the *scene graph*. The scene graph is a graph like object structure containing all the visual components to be displayed in the JavaFX application - for instance GUI components.

The width and height parameters sets the width and height of the JavaFX window when it opened, but the window can be resized by the user.

Adding a Scene

Here is how the opened window looks with the `Scene` and `Label` added:



Hello World, JavaFX !

END