JavaFX Adventures part 2 - Application frame

JavaFX Application

Putting together a JavaFX application is pretty simple. The main class must extend the javafx.application.Application. This main method of this class should call the "launch" static method by forwarding the command line parameters, and that's all. Your FX application starts in the "start" method of this class, which receives the reference to the primary stage (javafx.stage.Stage). The stage more or less corresponds to the Swing's JFrame. Next step is initializing a Scene (javafx.scene.Scene), and adding it to the Stage. The scene more or less similar than the root pane of the JFrame, only it is not created automatically, the user should create it and add it to the Stage. Further, the Scene may have no GlassPane added, it might contain only one content pane (called root pane).

The root pane added to the Scene must be a Parent (javafx.scene.Parent), which is the common ancestor of the Control, Group, and Region classes. The first one are the widgets with the capability of user interaction, and the later two resemble to the Panels of the Swing. Although adding one single Control might be useful in some cases (especially if that Control is complex like the TableView), yet in most of the cases, it is a Panel that is used. Initializing the Scene means adding the widgets (GUI controls) to this pane. In JavaFX, the widgets are represented by Node objects (javafx.scene.Node), which is the common ancestor of most of the GUI elements. The available controls are more or less the same than those available in Swing, but in JavaFX the basic 2D shapes are also Nodes, and might be added to the Scene or its root pane.

When one goes this far, one has to recognize the very first difference between the FX and the Swing. Swing uses one general purpose Panel, associated with layout managers responsible for positioning and sizing the panel and its component. There are but a few special panels, like the JScrollPane, and only in the cases when the view must support laying out of the components. JavaFX uses no layout managers - instead it defines several different panels, each has its own internal layout strategy. It's not that different though - when using Swing, in 99.99% of the cases I just create a JPanel instance with a specific layout manager, and won't change that layout manager later. So effectively, I create a specific panel and just use that.

Let's see where we are getting now. We have an empty window, decorated according to the underlying operation system's style.

The empty application

```
public class MyFxApplication extends Application {
        public static void main(final String[] args) {
                launch(args);
        }

        public void start(final Stage primaryStage) throws Exception {
                MyCustomNode root = new MyCustomNode();
                Scene scene = new Scene(root);
                primaryStage.setScene(scene);
                primaryStage.show();
    }
}
```

The MyCustomNode will be the implementation of that visual table I am going to develop; the next post provides more details about it.

Previous pages:

part 1 - Intro

Next pages:

part 3 - Design Considerations (+GridPane)