

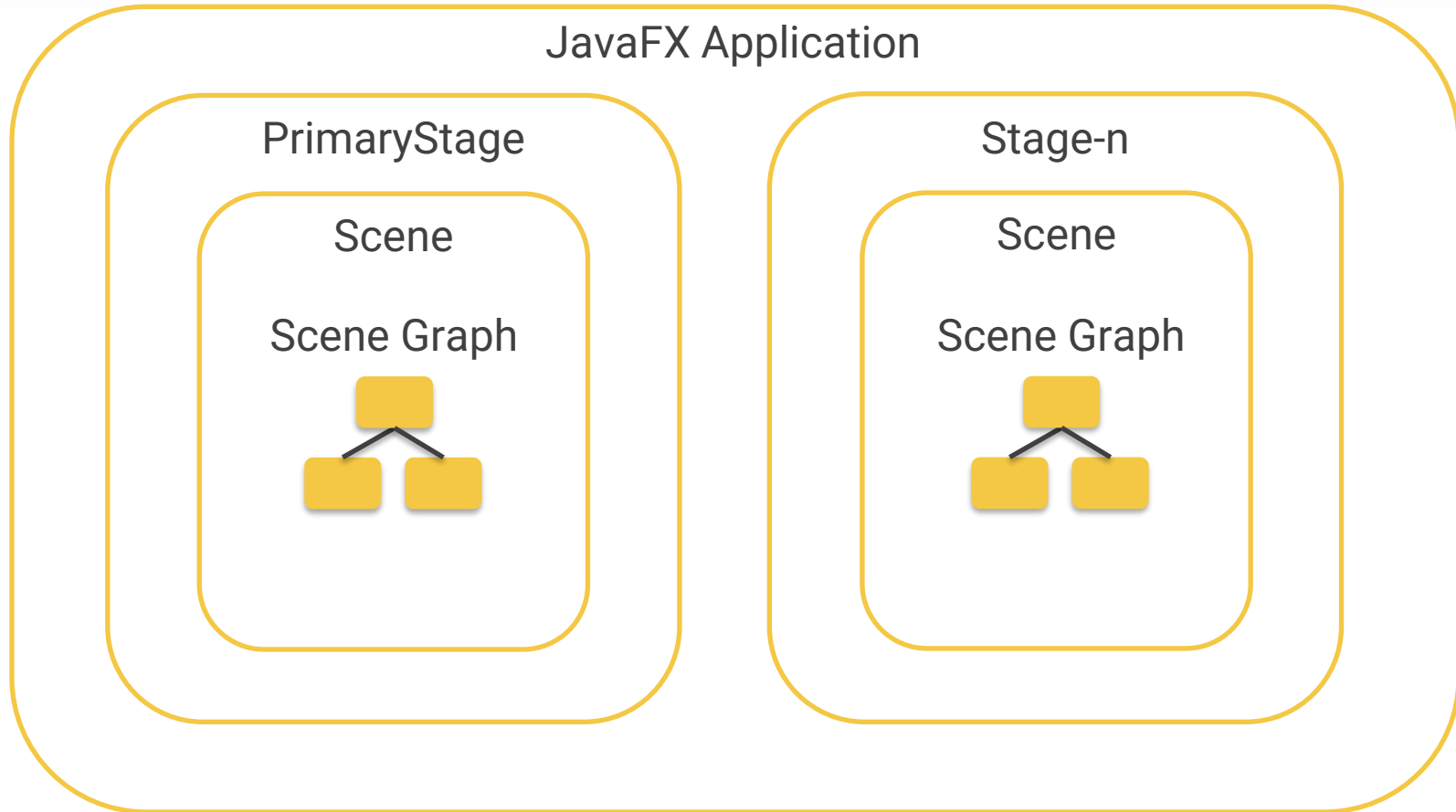
First GUI Project



1. JavaFX Application
2. FXML

JavaFX Application





Primary launch class extending Application

```
import javafx.application.Application;  
public class Main extends Application {  
  
}
```

```
import javafx.application.Application;
```

```
import javafx.stage.Stage;
```

```
public class Main extends Application {
```

```
    @Override
```

```
    public void start(Stage primaryStage) throws Exception{
```

```
        primaryStage.setTitle("Hello World");
```

```
        primaryStage.show();
```

```
    }
```

```
}
```

Implement the abstract method start()

Call the method show to display the Stage

1. JavaFX Application

```
import java  
import java  
public class  
    @Override  
    public  
        p  
        p  
    }  
}
```



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

```
public class Main extends Application {
```

```
    @Override
```

```
    public void start(Stage primaryStage) throws Exception{
```

```
        primaryStage.setTitle("Hello World");
```

```
        Label label = new Label("My first JavaFX Application!");
```

```
        Scene scene = new Scene(label, 400, 200);
```

```
        primaryStage.setScene(scene);
```

```
        primaryStage.show();
```

```
    }
```

```
}
```

Create a Label control

Create a Scene and add
the Label control to it

The first node added into the scene
is the root of the scene graph

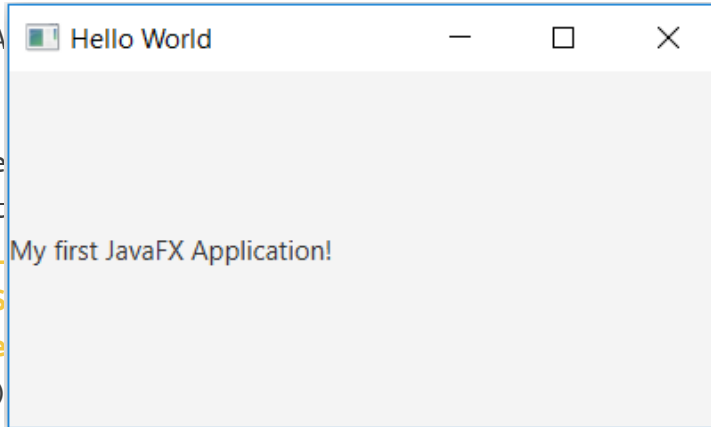
Add the Scene to the stage

1. JavaFX Application

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

```
public class Main extends Application {

    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Hello World");
        Label label = new Label("My first JavaFX Application!");
        Scene scene = new Scene(label);
        primaryStage.setScene(scene);
        primaryStage.show();
    }
}
```



Create a Label control

Create a Scene and add the Label control to it

The first node added into the scene is the root of the scene graph

Add the Scene to the stage

1. JavaFX Application

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

public class Main extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception{
        primaryStage.setTitle("Hello World");
        Label label = new Label("My first JavaFX Application!");
        Scene scene = new Scene(label, 400, 200);
        primaryStage.setScene(scene);
        primaryStage.show();
    }

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

Optionally implement main method

Must call Application.launch



FXML

FXML features

- User interface markup language created by Oracle for JavaFX
- XML format
- Allow separating the layout code from the rest of the JavaFX application
- Can be used both to define the whole app layout or just small parts
- Anything created or implemented in a FXML file can be expressed using JavaFX directly

```
<?import javafx.scene.control.Label?>  
<Label text="Hello, World!"/>
```

FXML features

- User interface markup language created by Oracle for JavaFX
- XML format
- Allow separating the layout code from the rest of the JavaFX application
- Can be used both to define the whole app layout or just small parts
- Anything created or implemented in a FXML file can be expressed using JavaFX directly

```
<?import javafx.scene.control.Label?>  
<Label text="Hello, World!"/>
```

```
BorderPane border = new BorderPane();  
Label toppanetext = new Label("Page Title");  
border.setTop(toppanetext);  
Label centerpanetext = new Label ("Some data here");  
border.setCenter(centerpanetext);
```

JavaFX

```
<BorderPane>  
  <top>  
    <Label text="Page Title"/>  
  </top>  
  <center>  
    <Label text="Some data here"/>  
  </center>  
</BorderPane>
```

FXML

So,... when should use FXML and when should use JavaFX?

FXML performs better than JavaFX and is easier to understand.

- Implement the project GUIs using FXML
- Use JavaFX only to modify the scene in run time

Main.java

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

public class Main extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception{
        primaryStage.setTitle("Hello World");
        Label label = new Label("My first JavaFX Application!");
        Scene scene = new Scene(label, 400, 200);
        primaryStage.setScene(scene);
        primaryStage.show();
    }

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

Main.java

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

public class Main extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception{
        primaryStage.setTitle("Hello World");
        Label label = new Label("My first JavaFX Application!");
        Scene scene = new Scene(label, 400, 200);
        primaryStage.setScene(scene);
        primaryStage.show();
    }

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


Main.java

```
...
@Override
    public void start(Stage primaryStage) throws Exception{
        primaryStage.setTitle("Hello World");
        Label label = new Label("My first JavaFX Application!");
        Scene scene = new Scene(label, 400, 200);
        primaryStage.setScene(scene);
        primaryStage.show();
    }
...
```

myScene.fxml

```
<?import javafx.scene.control.Label?>
<Label text="My first JavaFX Application!"/>
```

Main.java

```
...
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
...
@Override
    public void start(Stage primaryStage) throws Exception{
        primaryStage.setTitle("Hello World");
        Parent root = FXMLLoader.load(getClass().getResource("myScene.fxml"));
        Scene scene = new Scene(root, 400, 200);
        primaryStage.setScene(scene);
        primaryStage.show();
    }
...
```

myScene.fxml

```
<?import javafx.scene.control.Label?>
<Label text="My first JavaFX Application!"/>
```

Main.java

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.fxml.FXMLLoader;
import javafx.scene.Node;
import javafx.stage.Stage;

public class Main extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception{
        primaryStage.setTitle("Hello World");
        Node root = FXMLLoader.load(getClass().getResource("sample.fxml"));
        Scene scene = new Scene(root, 400, 200);
        primaryStage.setScene(scene);
        primaryStage.show();
    }

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

Main.java

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.fxml.FXMLLoader;
import javafx.scene.Node;
import javafx.stage.Stage;
```

```
public class Main extends Application {
```

```
    @Override
```

```
    public void start(Stage primaryStage) {
```

```
        primaryStage.setTitle("Hello World");
```

```
        Node root = FXMLLoader.load(getClass().getResource("HelloWorld.fxml"));
```

```
        Scene scene = new Scene(root, 300, 200);
```

```
        primaryStage.setScene(scene);
```

```
        primaryStage.show();
```

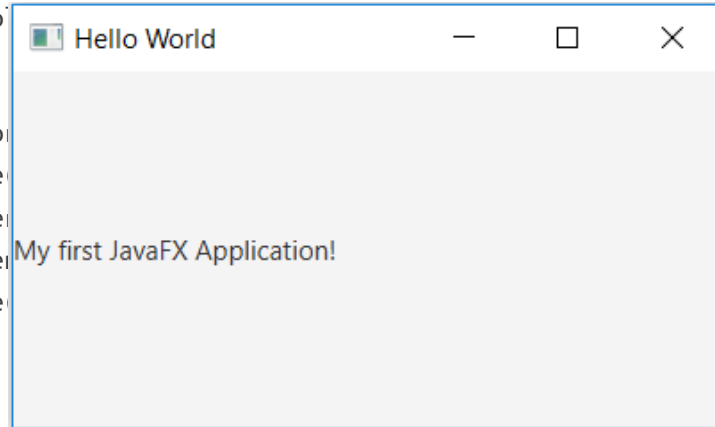
```
    }
```

```
    public static void main(String[] args) {
```

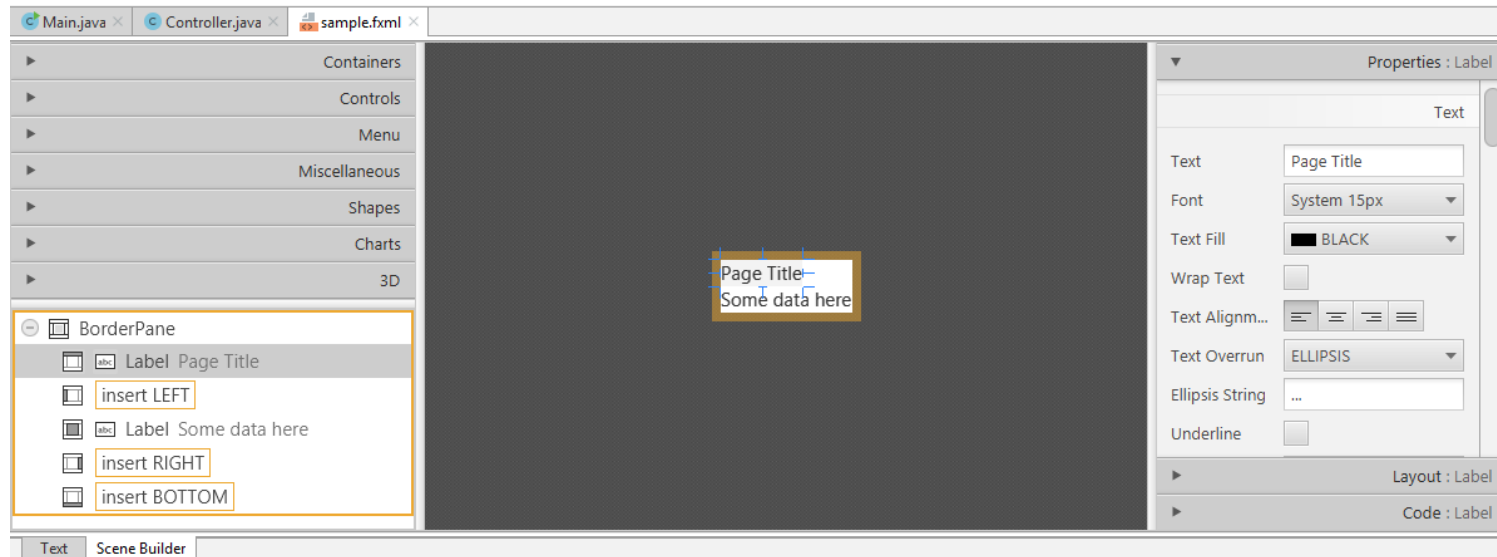
```
        launch(args);
```

```
    }
```

```
}
```



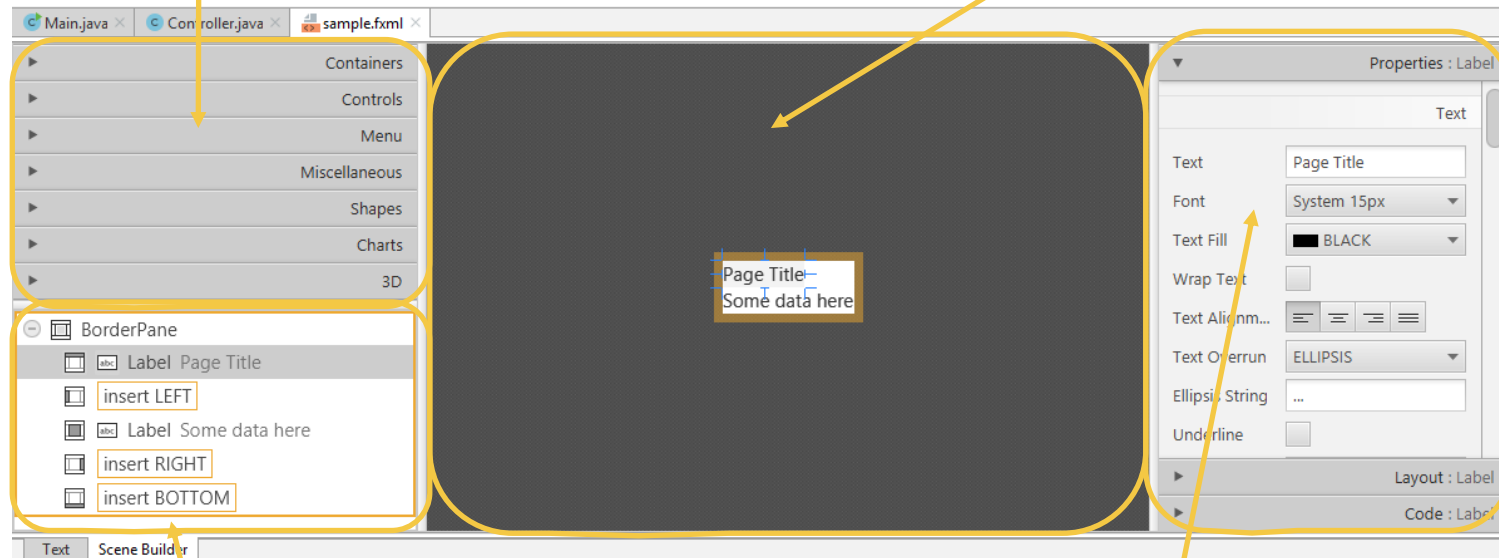
Scene Builder Tool



Available containers and controls

Scene Builder Tool

Scene prototype



Scene hierarchy

Control properties, layout and event handling

2. FXML

The screenshot displays the JavaFX IDE interface, divided into two main sections: the top half for visual design (Scene Builder) and the bottom half for code editing (Code Editor).

Top Section (Scene Builder):

- Left Panel (Component Palette):** Lists various UI components. Under the 'Containers' category, 'BorderPane' is selected. Its children are listed: 'Label Page Title' (top), 'insert LEFT' (left), 'Label Some data here' (center), 'insert RIGHT' (right), and 'insert BOTTOM' (bottom).
- Center Canvas:** A visual representation of the BorderPane layout. It shows a dark gray background with a white rectangular area in the center. This area contains two text labels: 'Page Title' at the top and 'Some data here' below it. The labels are highlighted with a blue selection border.
- Right Panel (Properties):** Titled 'Properties : Label', it shows the configuration for the selected 'Label' component. Properties include:
 - Text:** 'Page Title'
 - Font:** 'System 15px'
 - Text Fill:** 'BLACK'
 - Wrap Text:** (unchecked)
 - Text Alignm...:** (Left, Center, Right, Justify buttons)
 - Text Overrun:** 'ELLIPSIS'
 - Ellipsis String:** '...'
 - Underline:** (unchecked)

Bottom Section (Code Editor):

- Code Editor:** Displays the XML code for the scene graph. The code is as follows:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <?import javafx.scene.control.Label?>
4 <?import javafx.scene.layout.BorderPane?>
5
6
7 <BorderPane>
8   <top>
9     <Label text="Page Title"/>
10  </top>
11  <center>
12    <Label text="Some data here"/>
13  </center>
14 </BorderPane>
15
```
- Bottom Bar:** Shows 'Text' and 'Scene Builder' tabs, with 'Scene Builder' currently active.

“Existen tres posibles respuestas a un diseño:
Si, No y WOW!.
WOW es a lo que hay que aspirar.”

Milton Glaser, diseñador gráfico

