P2. SCENE BUILDER

Interfaces Persona Computador

Depto. Sistemas Informáticos y Computación

UPV

Outline

- Concepts of GUI frameworks
- FXML
- SceneBuilder
- NetBeans and the controller class
- Step-by-step example
- Exercise

Concepts of a GUI Framework

Model-View-Controller (MVC)

- Model-View-Controller (MVC) is a popular design pattern that separates the logic, the interface and the data of the application.

  - View: is the visual presentation of the model. It can't modify the model directly and can be notified whenever there is a change in the state of the model

  - Controller: reacts to the user's requests, executing the proper action and updating the proper model. It is also in charge of notifying changes in the model to the view.

  - Model: doesn't have any information about the controller/view. It represents the data (state) and the logic of the application

Concepts of a GUI Framework

The GUI's execution thread

- The GUI runs on a separated thread from the main thread
- The goal is to have always responsive GUIs. They should react as quickly as possible to user actions
- It is necessary to separate the code that runs the UI from the logic of the application
- The user code can run in the GUI thread (for example, in an event handler), but heavy loops or costly actions (network operations or database access) should be executed in a separated thread.

FXML:

- The FXML files have a description of the scene graph that represents the user interface. The syntax of the file is based on the XML format, and it is loaded in runtime to create the instances of the scene nodes described in the file
- HTML and Android work similarly
- The advantages of using FXML are:
  - The designed can work on the interface while the programmer can work on the code without working on the same file
  - The separation between view and controller is enforced, since there is no code in FXML

```
<?xml version="1.0" encoding="UTF-8"?>
…
<StackPane id="root" prefHeight="200" prefWidth="320" xmlns:fx="http://javafx.com/fxml/1"
xmlns="http://javafx.com/javafx/8" >
    <children>
      <Text layoutX="110.0" layoutY="97.0" strokeType="OUTSIDE" strokeWidth="0.0" text="Hi
there!!!"  id="text"/>
    </children>
</StackPane>
```

FXML:

- The controller in a JavaFX application is a Java class that contains the references to the scene graph's nodes and methods for the event handlers.
- The FXML document can define the name of the Java class that will be used as its controller
- The FXML file also associates each event with the name of the method in the controller that will handle them. This considerably reduces the amount of code required to create and register the nodes in the scene
- For associating the variables and methods defined in the controller with the corresponding objects, JavaFX uses injection after the instantiation of the scene graph

JavaFX and MVC :

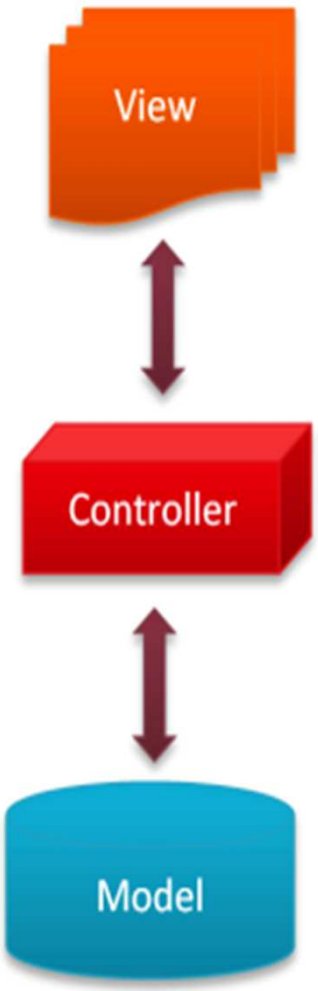
- An FXML file describes the view:

```
<?xml version="1.0" encoding="UTF-8"?>
<?import java.lang.*?>
<AnchorPane id="AnchorPane" prefHeight="200" prefWidth="320"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="hellofxm.FXMLDocumentController">
 <children>
    <Button layoutX="126" layoutY="90" text="Click Me!" onAction="#handleButtonAction"
fx:id="button" />
    <Label layoutX="126" layoutY="120" minHeight="16" minWidth="69" fx:id="label" />
```

- It admits a single Controller class with the event handlers

```
public class FXMLDocumentController implements Initializable {
    @FXML
    private Label label;
    @FXML
    private void handleButtonAction(ActionEvent event) {
          label.setText("Hello World!");
    }
    @Override
    public void initialize(URL url, ResourceBundle rb) {
    }
}
```

- Java classes that define the objects of the application



JavaFX Architecture

- Scene Graph (FXML file)

```
<?xml version="1.0" encoding="UTF-8"?>
…
<StackPane id="Raiz" prefHeight="200" prefWidth="320" xmlns:fx="http://javafx.com/fxml/1"
xmlns="http://javafx.com/javafx/8" >
    <children>
      <Text layoutX="110.0" layoutY="97.0" strokeType="OUTSIDE" strokeWidth="0.0" text="Hola a TODOS!!!"
id="texto"/>
    </children>
</StackPane>
```

```
public class HolaFXM extends Application {

    @Override
    public void start(Stage stage) throws Exception {
        Parent raiz = FXMLLoader.load(getClass().getResource("FXMLDocument.fxml"));

        Scene scene = new Scene(raiz);
        stage.setScene(scene);
        stage.show();
    }

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

JavaFX Architecture

- Scene Graph (FXML file)

```
public class HolaFXM extends Application {

    @Override
    public void start(Stage stage) throws Exception {
        Parent raiz = FXMLLoader.load(getClass().getResource("FXMLDocument.fxml"));

        Scene scene = new Scene(raiz);
        stage.setScene(scene);

        stage.show();
    }

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

Create the nodes, build the tree → Create the scene and assign the scene to the window → Draw the window and yield control

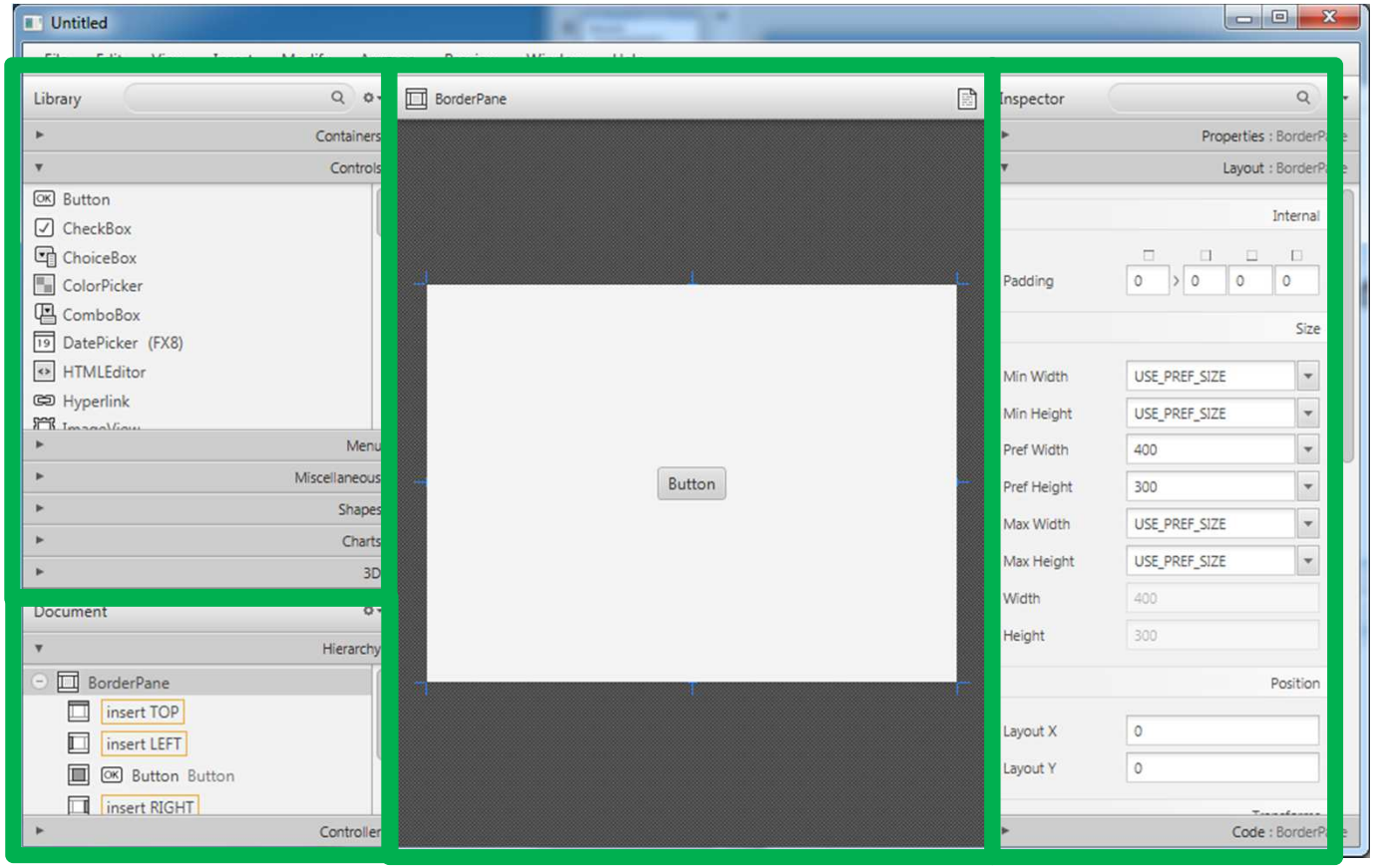
Scene Builder:

- Scene Builder is a standalone editor of FXML files developed by Oracle and now maintained by Gluon (http://gluonhq.com/) that allows us to design our interfaces visually
  - Scene Builder contains all the controls and containers supported by JavaFX
  - Windows are built dragging and dropping controls onto the main work area
  - There is a panel for adjusting the properties of the controls
  - The result is stored in a FXML file
  - It can be integrated into NetBeans or Eclipse

Scene Builder

Main Window

Control Library



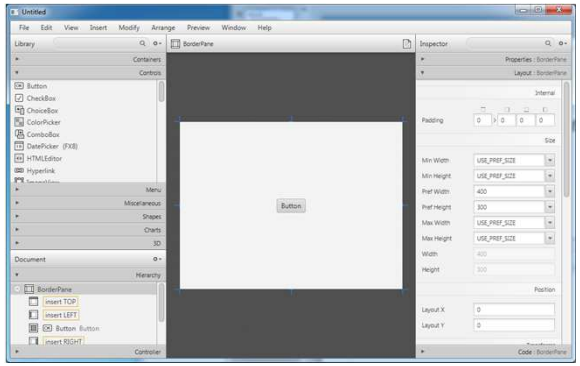
Document Hierarchy

Work area

Inspector

Scene Builder

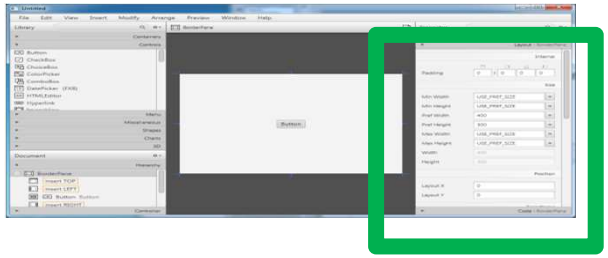
How to use Scene Builder:



- For adding a new control, drag and drop it from the library onto the work area, or onto the Document's hierarchy
- There is a search box at the top of the Library to filter the controls by name
- The Property panel on the right of the window shows the properties of the currently selected control. You can change the control's attributes in that panel (position, size, appearance, etc.)

Scene Builder

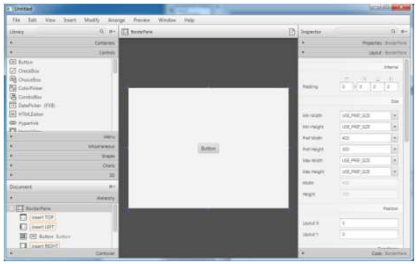
How to use Scene Builder:



- The Inspector panel has three sections: Properties, Layout and Code:
  - The Properties section allows us to change the style of the element selected in the work area. JavaFX uses CSS properties to define the style of the elements (we will study CSS in a later session)
  - The Layouts section allows us to specify how the selected control changes its position and size when the window is resized. Here we can define the allowed range of sizes of the control. This panel configuration depends on the container
  - The Code section specifies the event handlers for the control. The field fx:id determines the name of the variable that holds the reference to this control in the controller class. It is also used to select the proper CSS style
    - This section is very important to properly connect the scene graph with the Java code. We can define both the id and event handler names. Later, NetBeans can generate a template of the controller using the names provided in Scene Builder.
- The Controller class can be selected in the section called Controller of the Document panel (below the Library)

Scene Builder

How to use Scene Builder :

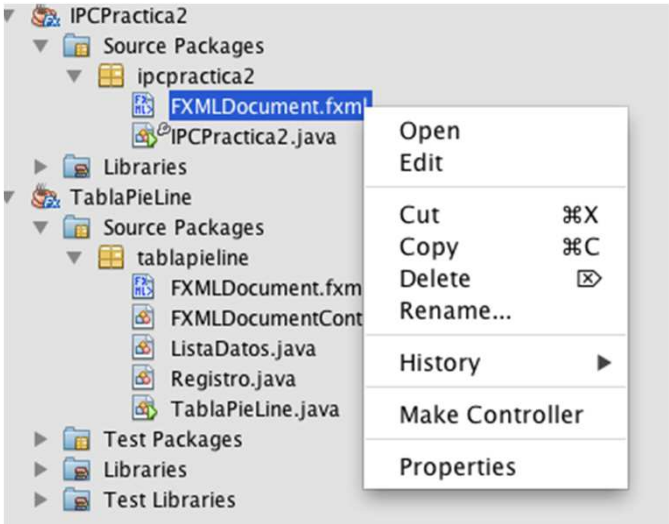


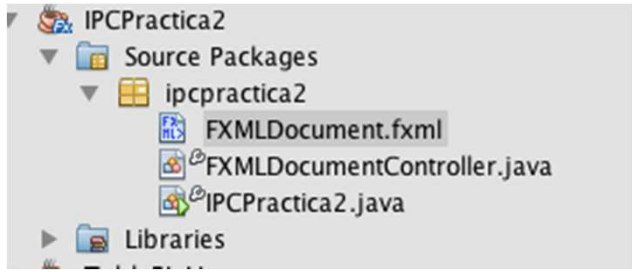
- Some commands in Scene Builder
  - "Wrap in" puts all the selected controls inside a new container of a given type
  - "Unwrap" removes the selected container, without removing its children from the scene graph
  - "Fit to Parent" ask a control to grow to occupy its container space
  - "Use Computed Sizes" resets the control's preferred size to USE_COMPUTED_SIZE
  - "Show/Hide Sample Data" fills the list, table and tree controls with fake data. The data is not stored in the FXML file
  - "Show Preview" shows a separated window with the current state of the GUI
  - "Show Sample Controller Skeleton" shows in a new window a template of the controller, taking into account the current scene graph

NetBeans

How to generate automatically the controller class:

- In NetBeans, right click on the FXML file and select the option Make Controller



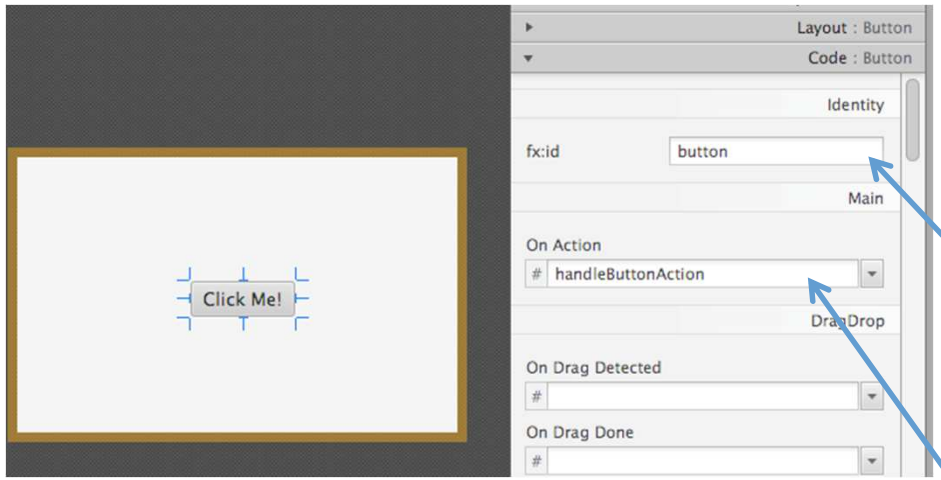


- If there is already a controller, it is updated with the latest changes in the FXML file. NetBeans does not remove existing code.

NetBeans

How to generate automatically the controller class:

- Besides creating the controller class with the event handlers and the references to the controls, NetBeans updates the FXML file adding a reference to the controller class




```
package ipcpractica2;

import java.net.URL;
import java.util.ResourceBundle;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;
import javafx.scene.control.Label;

/**
 * FXML Controller class
 *
 * @author jsoler
 */
public class FXMLDocumentController implements Initializable {

    @FXML
    private Button button;
    @FXML
    private Label label;

    /**
     * Initializes the controller class.
     */
    @Override
    public void initialize(URL url, ResourceBundle rb) {
        // TODO
    }

    @FXML
    private void handleButtonAction(ActionEvent event) {
    }

}
```

Loading an FXML file

- FXML files are loaded using the `load()` method of the class `FXMLLoader`. There are two options: using a static method or creating a instance of the loader:

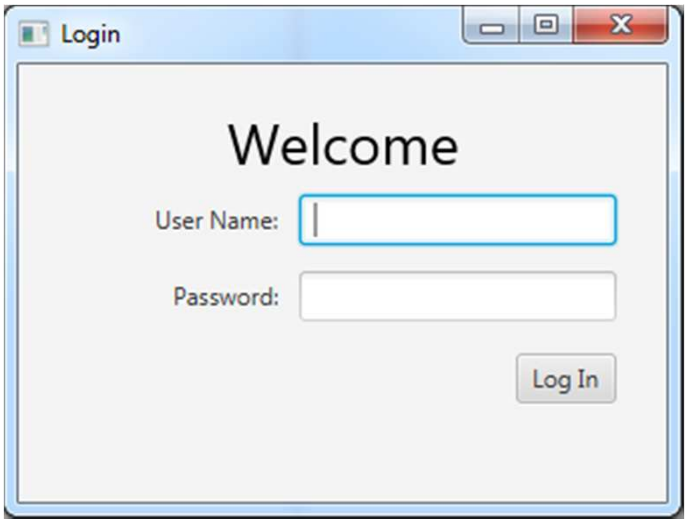
```
(option 1) Parent root = FXMLLoader.load(getClass().getResource("FXMLDocument.fxml"));

(option 2) FXMLLoader loader= new FXMLLoader();
           loader.setLocation(MainApp.class.getResource("FXMLDocument.fxml"));
           Parent root = loader.load();
```

- The method load performs the following tasks:
  1. Objects defined in the FXML file are instantiated and the scene graph is built
  2. The controller class is instantiated. The scene nodes are injected into the references of the controller, and the event handlers are registered
  3. If present, the method Initialize in the controller instance is executed. In this method we can add additional initialization code, if required

Step by step example

- Login example



Exercise

- Create a JavaFX FXML project with the following view:

References

- Find more information in:
  - https://docs.oracle.com/javase/8/javase-clienttechnologies.htm
  - https://docs.oracle.com/javase/8/javafx/get-started-tutorial/get_start_apps.htm
  - http://docs.oracle.com/javafx/2/api/javafx/fxml/doc-files/introduction_to_fxml.html
  - http://docs.oracle.com/javafx/scenebuilder/1/overview/jsbpub-overview.htm
  - http://code.makery.ch/library/javafx-8-tutorial/es/
- Online documentation:
  - Java: http://docs.oracle.com/javase/8/docs/api/
  - JavaFX: http://docs.oracle.com/javase/8/javafx/api/
- Carl Dea *et al*. JavaFX 8.
  Introduction by Example. Apress 2014.