



## P2. SCENE BUILDER

---

Interfaces Persona Computador

Depto. Sistemas Informáticos y Computación

UPV

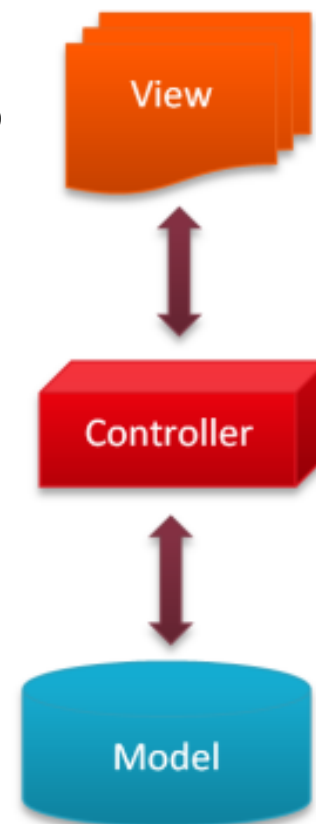
# Índice

- Conceptos de un framework GUI
- Estructura de una aplicación JavaFX
- Los ficheros FXML
- SceneBuilder
- NetBeans y la clase Controller
- Ejemplo guiado
- Ejercicio
- Anexo: pasos en el desarrollo de una ventana en JavaFX

# Conceptos de un Framework GUI

## Modelo-Vista-Controlador (MVC)

- Modelo-Vista-Controlador (MVC) es un patrón de diseño que separa por una parte la lógica de la aplicación, la interfaz y los datos de la aplicación.
  - **Vista:** es la presentación visual del modelo (los datos), no puede cambiar el modelo directamente y puede ser notificada cuando hay un cambio de estado del modelo
  - **Controlador:** reacciona a la petición del usuario, ejecuta la acción adecuada y actualiza el modelo pertinente, o notifica cambios en el modelo a la vista.
  - **Modelo:** no sabe nada del controlador/vista. Representa los datos (estado) y la lógica de la aplicación

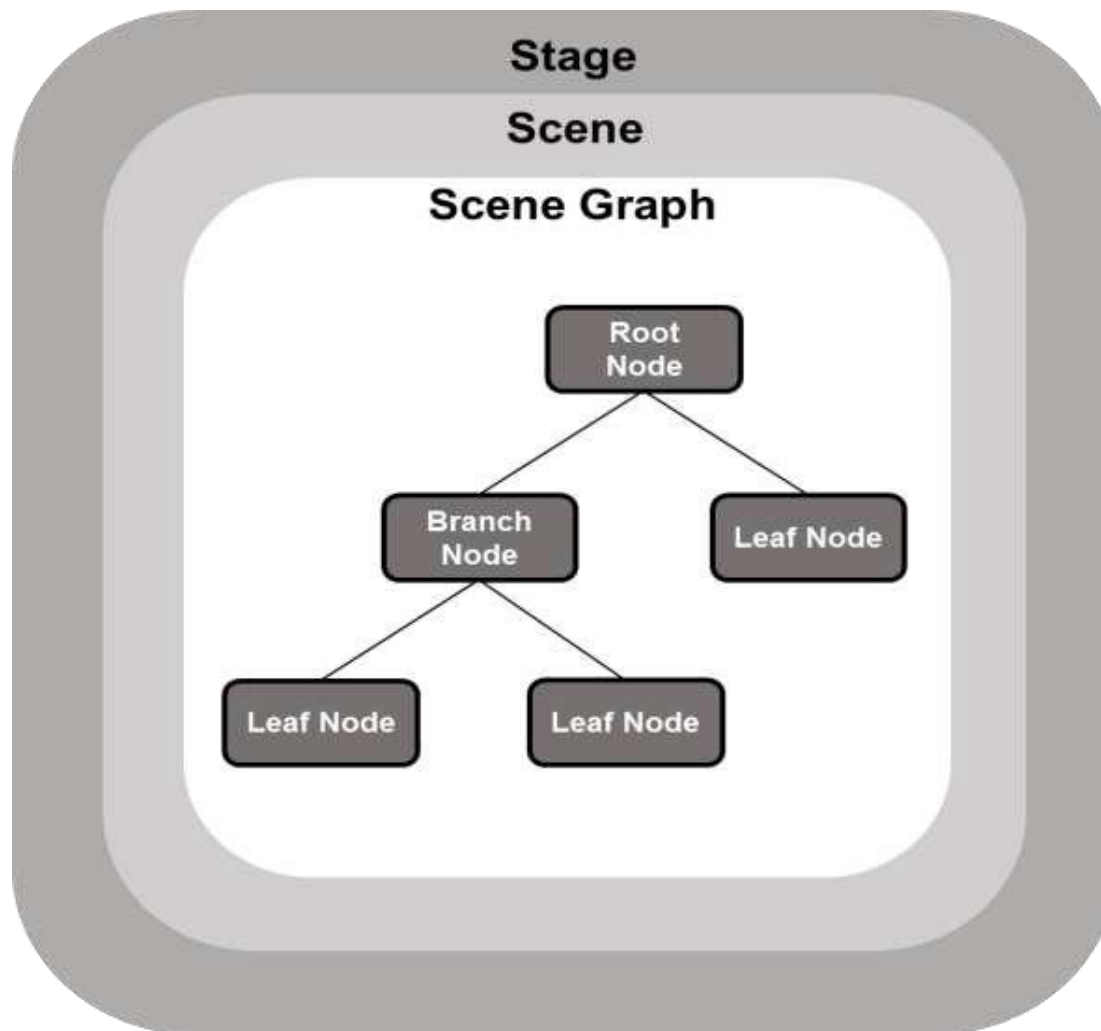


# Conceptos de un Framework GUI

## El hilo de ejecución de un GUI

- El GUI corre sobre un hilo diferente al hilo principal
- Esto se hace así para disponer de una GUI que responda rápidamente a las acciones del usuario
- Hay que separar para ello el código de la Interfaz de usuario del código de la lógica de la aplicación
- El código de usuario puede correr en el hilo de la GUI, pero para grandes bucles o acciones costosas (operaciones de red o bases de datos) es preferible normalmente ejecutar el código en otro hilo

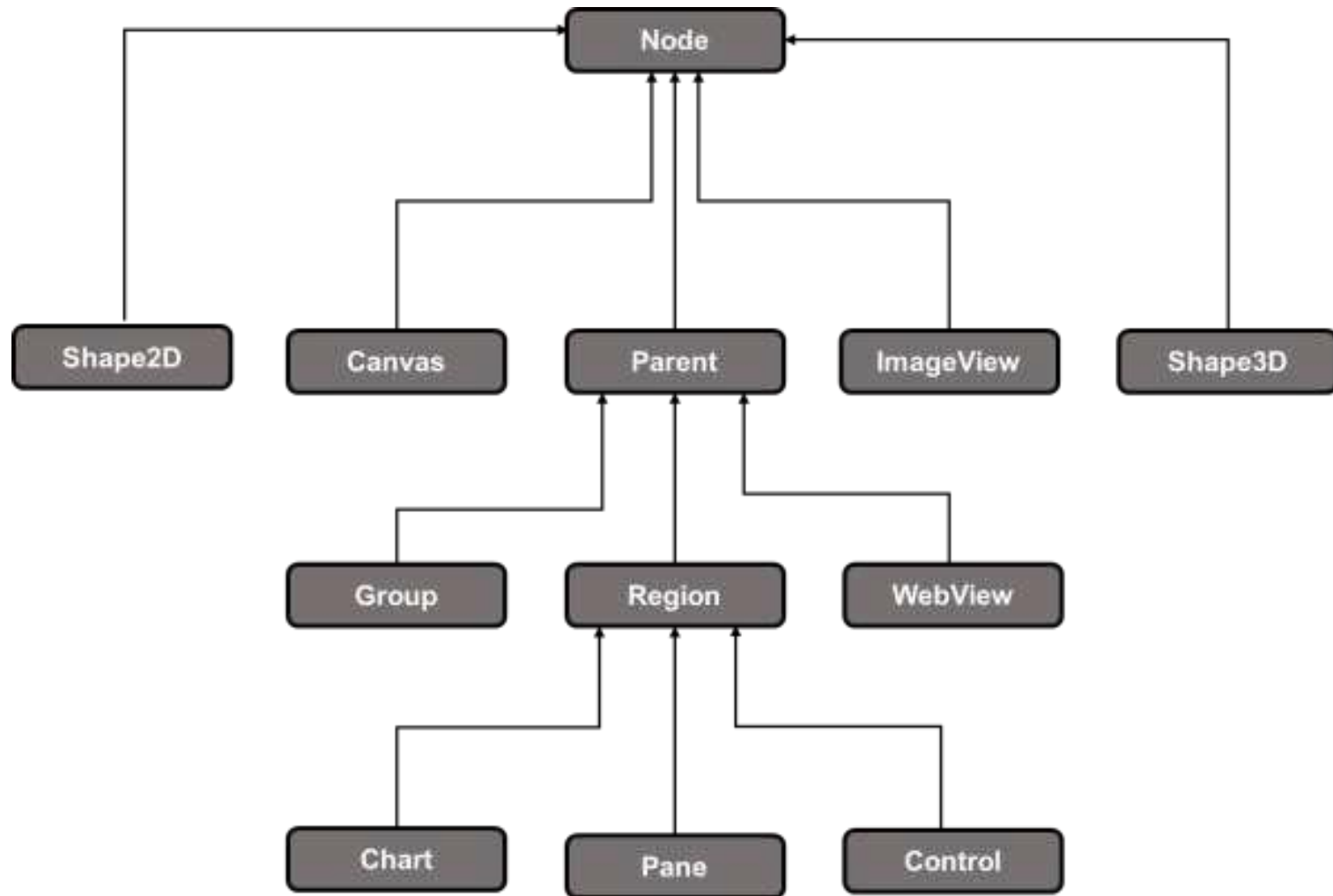
# Estructura de una Aplicación JavaFX



# La clase Application

```
public class JavaFXApplication extends Application {  
    @Override  
    public void start(Stage stage) throws Exception {  
        /*Codigo para el Stage, la Scene y el grafo de escena*/  
        /*-----*/  
  
        /* 1 - preparar el grafo de escena con todos sus nodos*/  
        /* 2 - preparar la escena, requiere de las dimensiones y el nodo raiz */  
        /* 3 - preparar el stage: */  
            /* 3-1 Añadir la escena */  
            /* 3-2 mostrar la escena */  
  
        /*-----*/  
    }  
  
    public static void main(String[] args) {  
        launch(args); // este metodo llama de manera interna al metodo start()  
    }  
}
```

# La clase Node



## Preparando la Scene

```
Scene miEscena = new Scene( nodo_raiz );
```

```
Scene miEscena = new Scene( nodo_raiz, 600,400 );
```

## Preparando el Stage

```
//preparando el titulo de la ventana  
primaryStage.setTitle("Hola");
```

```
//preparando la escena en la ventana,  
primatyStage.SetScene( miEscena);
```

```
//mostrando la ventana  
primaryStage.show();
```

Hay 5 tipos de ventanas:

☐ Decorated ☐ Undecorated ☐ Transparent ☐ Unified ☐ Utility



## Terminando la aplicación

De manera implícita al cerrar la última ventana. Podemos cerrar la ventana por código mediante:

```
primaryStage.Close();
```

```
Platform.Exit( );
```

```
System.Exit( int );
```

# Los ficheros FXML

- contienen la descripción del **grafo de escena** que representa una interfaz de usuario. El fichero tiene formato XML, y se carga en tiempo de ejecución es entonces cuando se **crean todos los objetos del grafo de escena y se integran en el árbol**.

```
Parent nodo_raiz = FXMLLoader.load(getClass().getResource("FXMLDocument.fxml"));
```

- Los beneficios son:
  - el diseñador puede trabajar con la interfaz mientras que el programador puede trabajar con el código sin la necesidad de trabajar sobre el mismo fichero
  - separación entre vista y controlador

```
<?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>
```

# La clase Controlador (solo con FXML)

- Cuando utilizamos ficheros FXML es necesario crear una clase de Java donde añadir el código necesario para los nodos que se crearan en la carga de fichero fxml.
- Esta clase se llama el Controlador ( Controller class)
- Cada fichero FXML solamente puede tener asociada una clase Controller
- Para enlazar los métodos y variables definidos dentro de esta clase con los objetos que forman el grafo de escena se utiliza inyección. Este proceso requiere que el programador codifique los métodos y variables de una manera determinada. **Este proceso se detalla después con el sceneBuilder, y se puede automatizar con el netBeans**

# JavaFX y el patrón MVC

- JavaFX permite definir el grafo de escena de manera independiente a su codificación en Java, es decir la **vista**.
  - El diseño del árbol o de una rama del grafo de escena se puede guardar en un fichero FXML (formato XML). La vista se puede separar en tantos ficheros FXML como sea necesario
  - En tiempo de ejecución la clase FXMLLoader permite la creación completa del árbol a partir del fichero de la vista.
- El código java para cada fichero FXML se agrupa en una clase de Java, **Controller class**
  - Cada fichero FXML solamente se puede asociar a una clase de Java
- Las clases de Java que forman el **modelo** se suelen crear aparte en la inicialización de la aplicación



# JavaFX y el patrón MVC

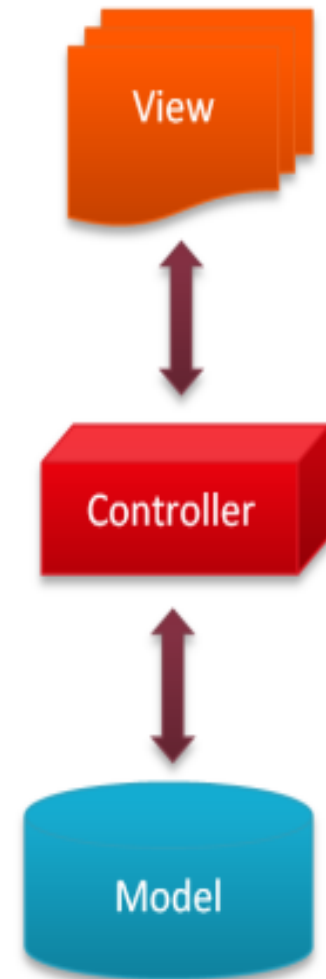
- Un fichero FXML es una descripción de la vista

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

- Admite una única clase Controlador con métodos para manejar los eventos

```
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) {
    }
}
```

- Clases de Java que definen los objetos de la aplicación.



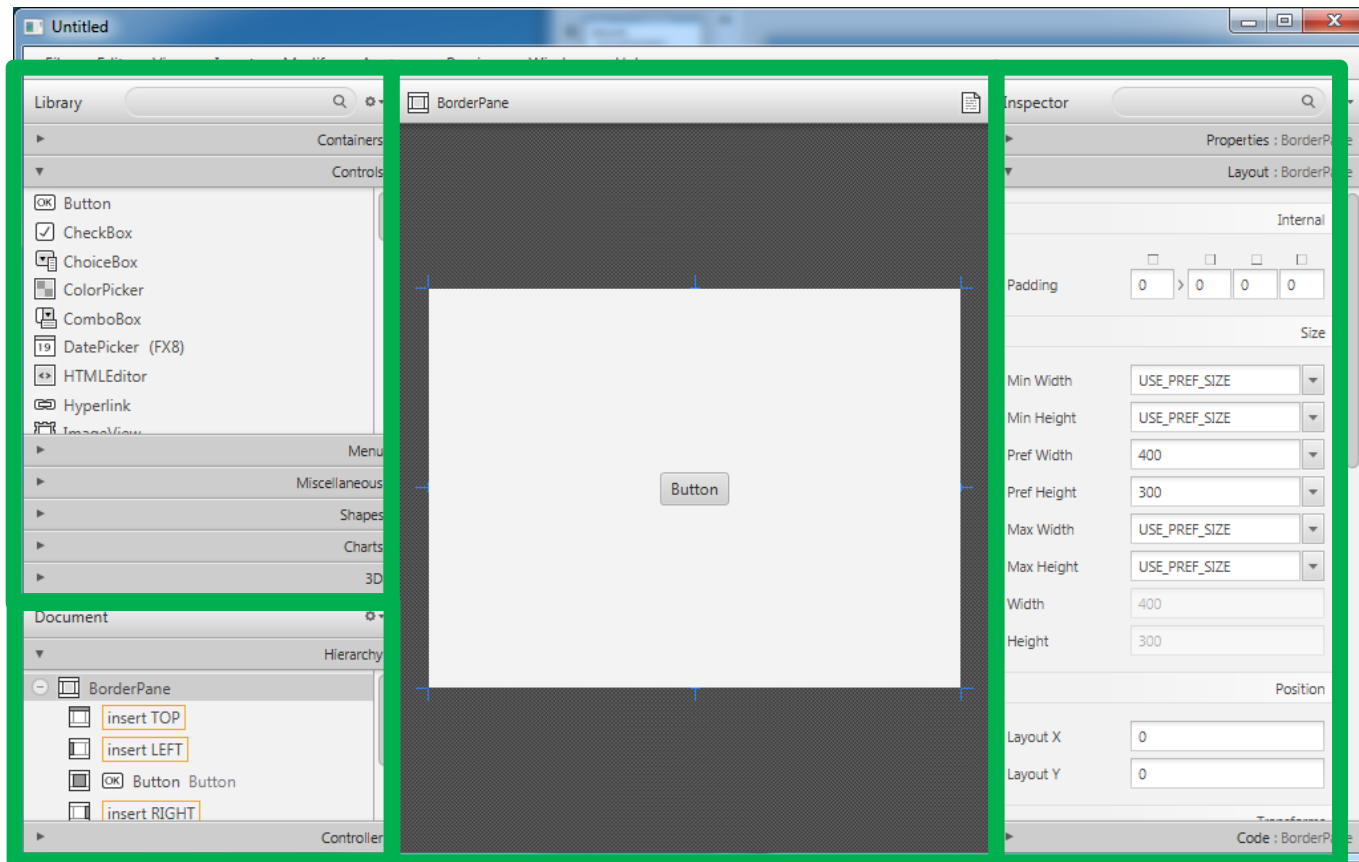
# Scene Builder:

- **Scene Builder** es un editor externo de ficheros FXML desarrollado por Oracle y ahora continuado por Gluon (<http://gluonhq.com/>) que nos permite diseñar nuestras interfaces de forma visual
  - Scene Builder contiene una paleta con todos los nodos definidos en JavaFX (podemos añadir controles personalizados)
  - Las ventanas se configuran arrastrando y soltando los nodos sobre el área de trabajo
  - Se pueden ajustar las propiedades de los nodos en un panel separado
  - El resultado se almacena en un fichero XML (con extensión FXML)
  - Se puede integrar con Netbeans o Eclipse

# Scene Builder

## Organización de la pantalla principal

Librería  
de  
controles



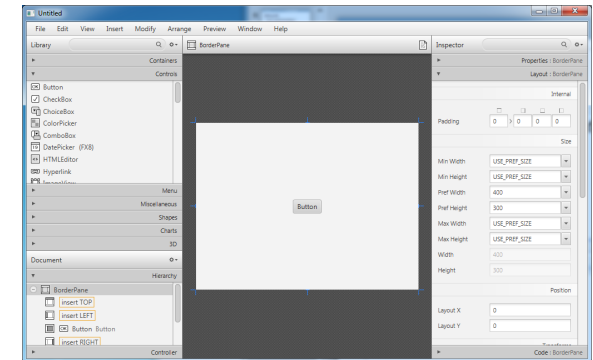
Árbol de  
escena

Zona de trabajo

Inspector

# Scene Builder

## Cómo utilizar Scenebuilder:

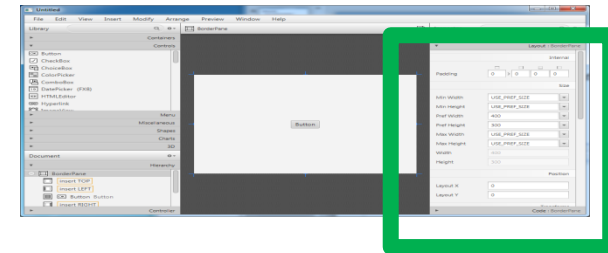


- Para añadir un elemento lo arrastraremos desde la librería de controles hasta la zona de trabajo o hasta la jerarquía de controles en el documento.
- Es posible filtrar los controles por nombre
- Con el control seleccionado podremos modificar cualquiera de sus propiedades. Las propiedades son los atributos disponibles de cada control (posición, tamaño, apariencia, etc.)



# Scene Builder

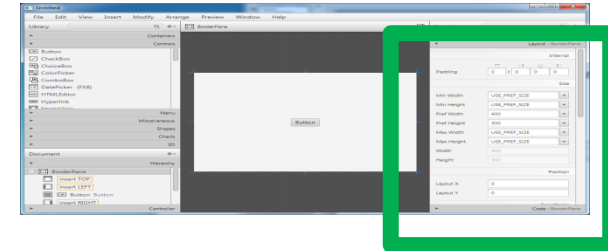
Cómo utilizar Scenebuilder:



- En el panel *Inspector* tenemos las secciones Properties, Layout y Code:
  - La sección **Properties** nos permite definir el estilo del elemento seleccionado en el área de trabajo. En JavaFX se utiliza plantillas CSS para definir el estilo de los elementos (lo veremos mas adelante)
  - La sección **Layouts** nos permite especificar el comportamiento en tiempo de ejecución del contenedor cuando cambiamos el tamaño de la ventana. También permite definir el tamaño del control. La información que aparece en esta sección dependerá del contenedor

# Scene Builder

## Cómo utilizar Scenebuilder:

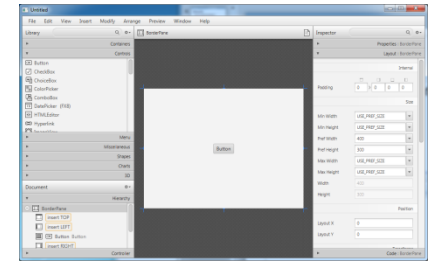


- La sección **Code** es **muy IMPORTANTE** pues en ella se especifica el nombre que tomarán los nodos en tiempo de ejecución y los métodos que estarán codificados en la clase controladora y queremos que se ejecuten ante la interacción del usuario sobre el control.
- El campo `fx:id` determina el nombre que debe tener el objeto de java dentro de la clase controlador. También sirve como identificador para la hoja de estilos CSS.
- Si definimos previamente en SceneBuilder los id de los nodos y los métodos que se van a ejecutar ante eventos, la generación automática de código desde NetBeans creará la clase controladora con los identificadores de los objetos gráficos y sus manejadores de eventos, y nos evitará errores tipográficos sobre los nombre de métodos y objetos.
- Para asignar una clase Java como el Controlador sobre el fichero fxml, debemos de indicar su nombre en el apartado Controller.



# Scene Builder

## Cómo utilizar SceneBuilder:

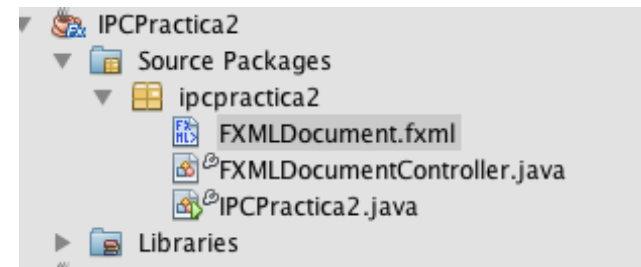
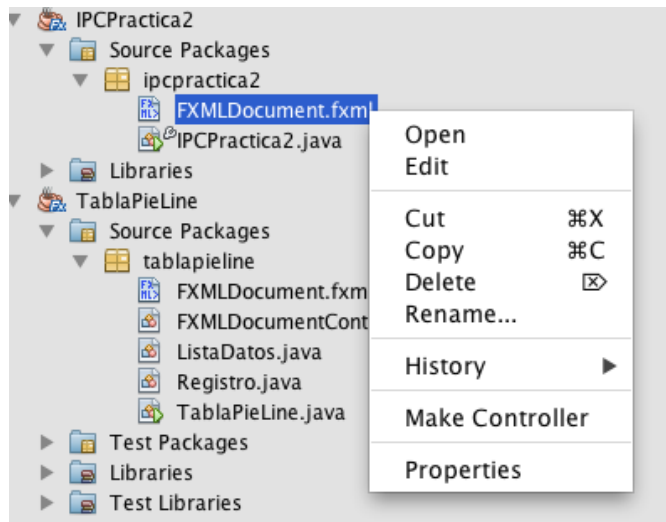


- Con el comando “**Wrap in**” situamos los controles seleccionados dentro de uno de los contenedores disponibles
- El comando “**Unwrap**” elimina el contenedor seleccionado pero deja sus controles inalterados
- Con el comando “**Fit to Parent**” cambiamos el tamaño del control seleccionado hasta que ocupe el área de su contenedor
- El comando “**Use Computed Sizes**” resetea los valores de las propiedades del contenedor a USE\_COMPUTED\_SIZE
- El comando “**Show/Hide Simple Data**” muestra datos ficticios en aquellos controles del tipo lista, tabla o árbol. Los datos no se guardan en el fichero FXML
- El comando “**Show Preview**” muestra en una ventana el resultado final del fichero FXML que se está editando
- El comando “**Show Sample Controller Skeleton**” abre una ventana y muestra una plantilla de código para crear una clase controlador a partir del fichero FXML

# NetBeans, clase Controlador

Cómo generar de manera automática la clase controlador:

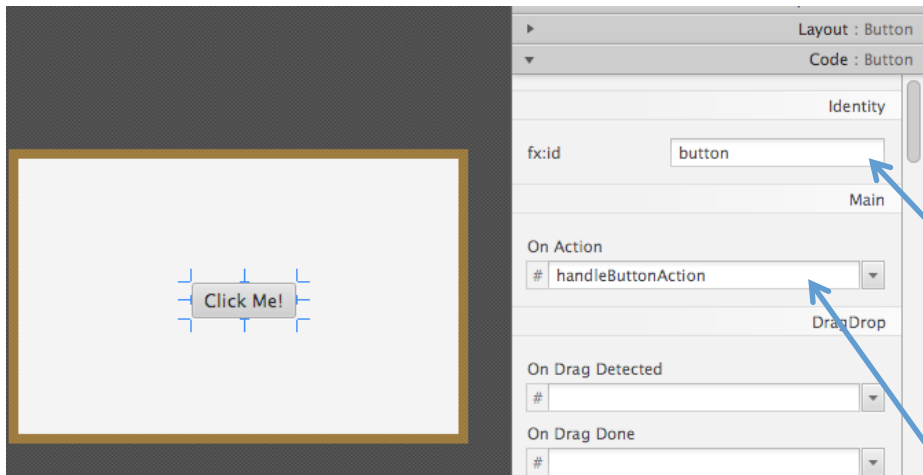
- Desde el explorador de NetBeans seleccionaremos el fichero FXML y con el botón derecho accederemos al menú **Make Controller**



# Netbeans, clase Controlador

Cómo generar de manera automática la clase controlador:

- Además de generar la Clase Controlador con los manejadores y las referencias a los controles, modifica el fichero FXML y le añade la referencia a la clase controlador.



```
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) {
    }

}
```

## Creación de los objetos del grafo y la clase Controladora

```
Parent nodo_raiz = FXMLLoader.load(getClass().getResource("FXMLDocument.fxml"));
```

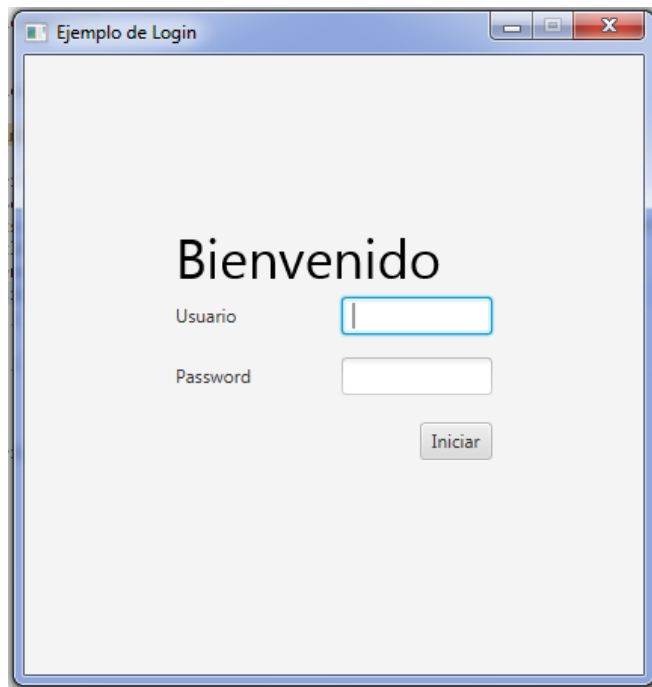
- Al llamar al método load de la clase FXMLLoader se van a realizar las siguientes acciones:
  - Creación de todos los nodos descritos en el fichero fxml como objetos de Java.
  - Construcción del grafo de escena con estos nodos.
  - Creación de un objeto de la clase controlador
    - Cuando se realiza un new sobre la clase controlador se invoca al constructor. Se lleva a cabo la INYECCION, es decir se les asigna a los objetos definidos en la clase bajo la etiqueta **@FXML** la referencia del objeto creado con anterioridad
    - Si la clase controlador implementa la interfaz **Initializable** se invoca al método  

```
@Override  
public void initialize(URL location, ResourceBundle resources){  
    }  
}
```

para que realice las oportunas inicializaciones

# Ejercicio guiado

- Ejemplo de login



The image shows a screenshot of a login window titled "Ejemplo de Login". The window has a light gray background and a blue border. At the top, there is a title bar with the text "Ejemplo de Login" and standard window control buttons (minimize, maximize, close). The main content area displays the word "Bienvenido" in a large, bold, black font. Below this, there are two input fields: one labeled "Usuario" and another labeled "Password". The "Usuario" field is a simple text box with a blue border, and the "Password" field is a text box with a white background and a blue border. Below the "Password" field, there is a button labeled "Iniciar" with a gray background and a blue border.

# Ejercicio guiado

- Crear un proyecto JavaFX FXML con la siguiente vista





# Bibliografía

- Puedes encontrar más información en:
  - <https://docs.oracle.com/javase/8/javase-clienttechnologies.htm>
  - [https://docs.oracle.com/javase/8/javafx/get-started-tutorial/get\\_start\\_apps.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/2/api/javafx/fxml/doc-files/introduction_to_fxml.html)
  - <http://docs.oracle.com/javafx/scenbuilder/1/overview/jsbpub-overview.htm>
  - <http://code.makery.ch/library/javafx-8-tutorial/es/>
- Documentación online:
  - Java: <http://docs.oracle.com/javase/8/docs/api/>
  - JavaFX: <http://docs.oracle.com/javase/8/javafx/api/>
- Carl Dea y otros. JavaFX 8. Introduction by Example. Apress 2014.

