

DESARROLLO DE INTERFACES

2º DAM

**I.E.S. POLITÉCNICO H.LANZ
JOSÉ MARÍA MOLINA**



TEMA 2-2-1 – JAVA FX - SCENE BUILDER

TEMA 2-1 - JAVA FX - SCENE BUILDER



JavaFX es un conjunto de librerías que permiten crear GUIs de una forma muy rápida y efectiva.

En JavaFX **todo se puede crear y configurar por código** pero por agilizar y simplificar vamos a empezar creando GUIs a partir de un lenguaje declarativo FXML.

Usaremos un IDE que está totalmente integrado en NetBeans: SceneBuilder.

MUY IMPORTANTE! Estos apuntes son **una visión general** para empezar a trabajar, en la siguiente parte ya veremos los atributos por cada componente.

- ✓ 1 SCENE BUILDER
- ✓ 2 EJEMPLOS
- ✓ 3 CLASE CONTROLADOR
- ✓ 4 ANEXO

1 SCENE BUILDER



FUNDAMENTOS DE SCENE BUILDER

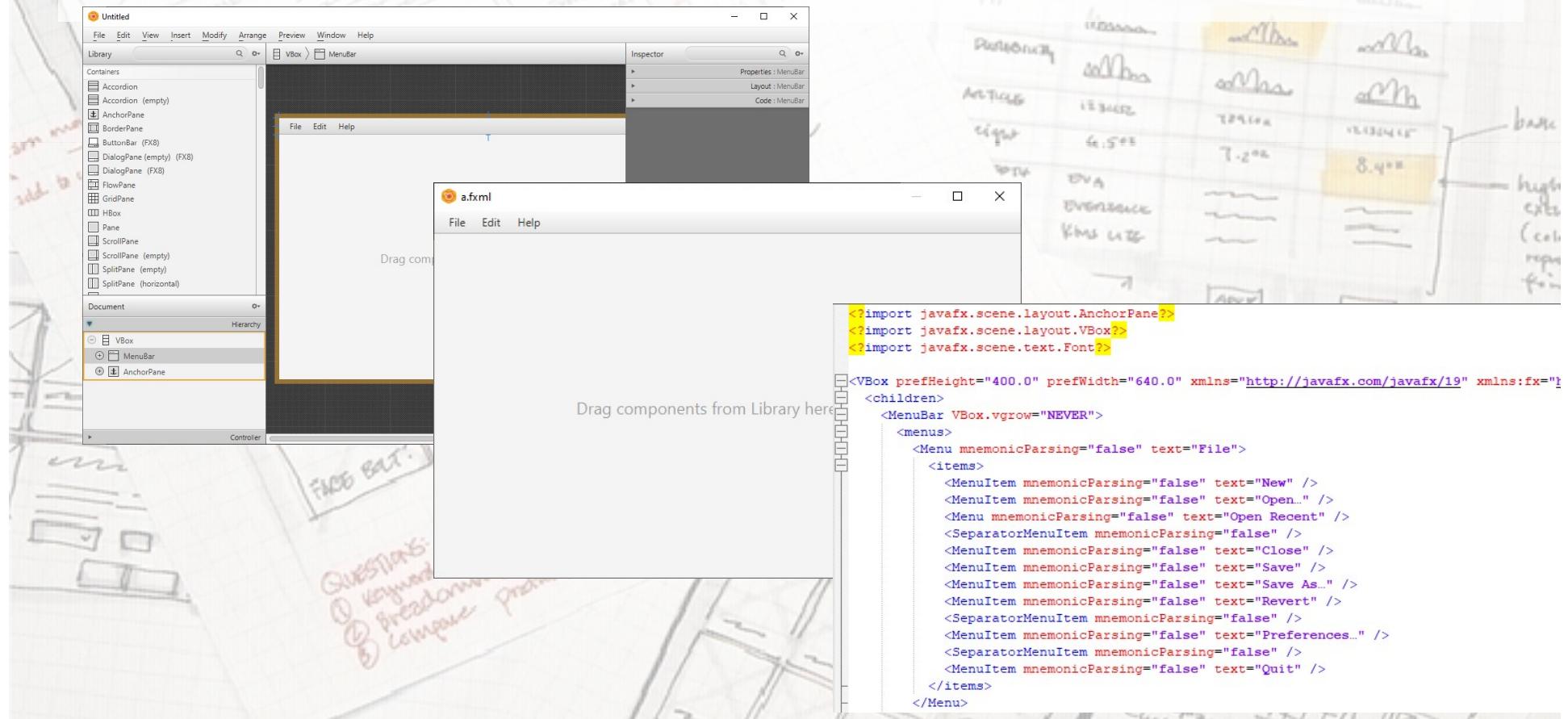
- ✓ Software de diseño de GUI (parte VISTA de MVC)
- ✓ Lenguaje declarativo → Crea código FXML
- ✓ Nodos: componentes en jerarquía
- ✓ Temas:
 - Gluon: tema orientado a móviles/tablets
 - Modena: orientado a Escritorio
- ✓ Permite CSS → separa aún más la VISTA del MVC
- ✓ Componentes básicos: contenedores y controles
- ✓ Componentes avanzados: gráficos, dibujo y 3D

1 SCENE BUILDER



NOS DESCARGAMOS Y ABRIMOS SCENE BUILDER

<https://gluonhq.com/products/scene-builder/> (versión actual 23 de 09/24)

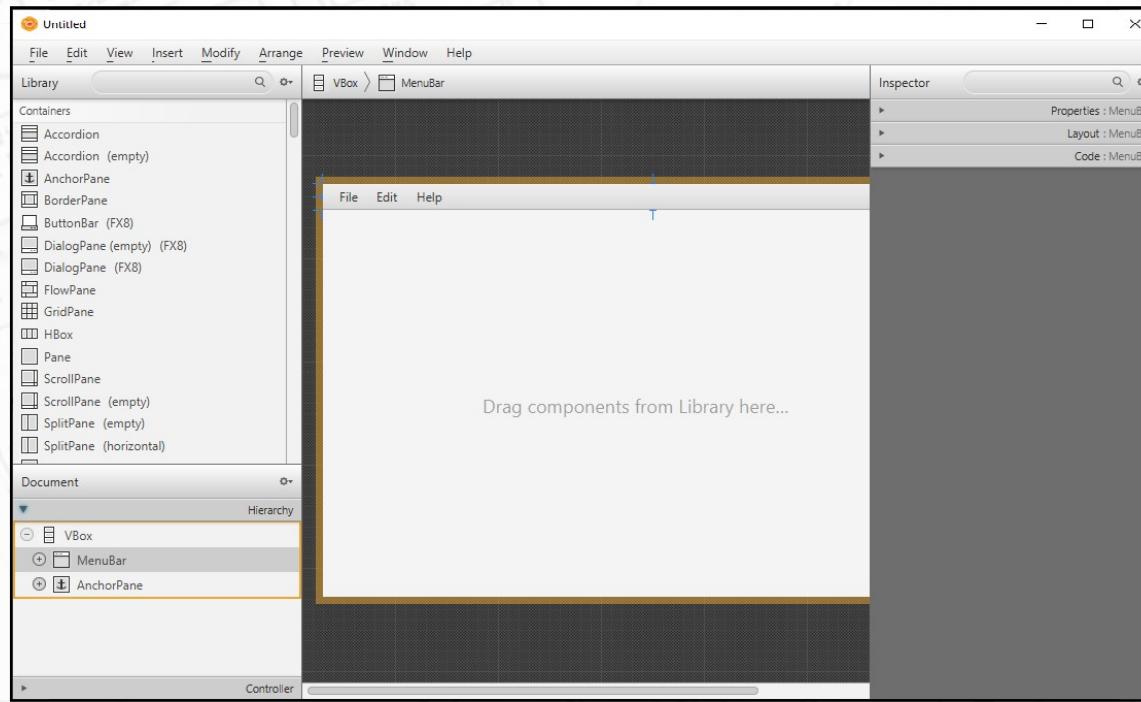


1 SCENE BUILDER



FUNDAMENTOS DE SCENE BUILDER

- ✓ Zonas: Librería de componentes, Jerarquía de nodos, Controlador, Zona de Diseño e Inspector: Propiedades, Layout y Código



1 SCENE BUILDER



FUNDAMENTOS DE SCENE BUILDER

- ✓ Menú de opciones (más útiles). Veremos TODAS:
 - **File/Edit:** opciones generales para crear, cargar/guardar, editar (copiar/pegar/seleccionar)
 - **View:** mostrar/ocultar paneles y crear la clase Controlador (*Show Sample Controller Skeleton*)
 - **Insert:** inserta componentes de la misma forma que desde Librería de forma gráfica.
 - **Modify:** opciones sobre GridPane, efectos visuales, menú contextual y tamaño de scene. Destacar Fit to Parent (ajusta el tamaño del componente al padre en AnchorPane) o Use Computed Sized (ajusta el tamaño del padre al hijo)
 - **Arrange:** mueve componentes entre capas y también añade elementos a un parent (Wrap In) y viceversa (Unwrap).

1 SCENE BUILDER



FUNDAMENTOS DE SCENE BUILDER

- ✓ Menú de opciones (más útiles):
 - **Preview:** muestra cómo quedaría el diseño (Ctrl+P), cambiar temas, hojas de estilo, e idiomas añadidos a la interfaz (internacionalización)
 - **Window:** permite cambiar de proyecto.
 - **Ayuda:** Pulsando de F2 a F5 nos vamos a distintas ayudas sobre JavaFX, la API, construcción de CSS o el lenguaje FXML.

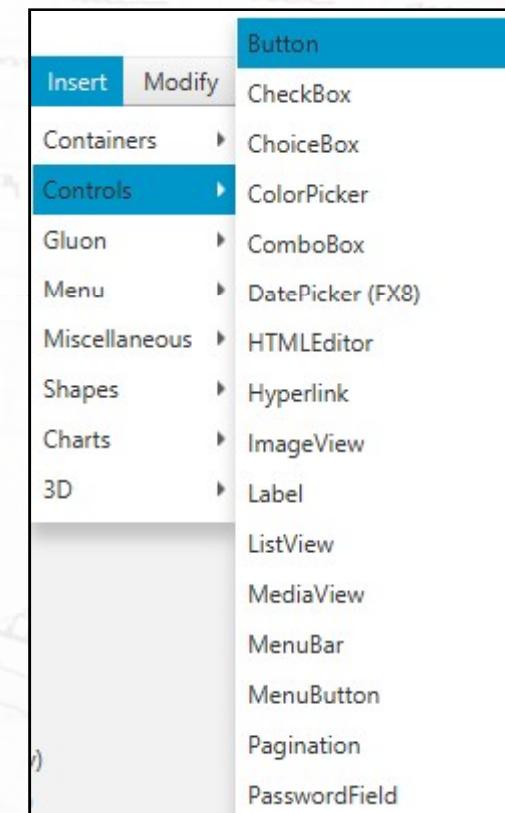
1 SCENE BUILDER



FUNDAMENTOS DE SCENE BUILDER

✓ Clases de componentes (136)

- **Contenedores (26)**: organizan y agrupan otros componentes
- **Controles (36)** : componentes básicos
 - **Gluon (23)**
 - **Menu (7)**: menú superior y contextual
 - **Miscelánea (8)**: otros (tooltip)
 - **Formas (24)**
 - **Diagramas (8)**
 - **3D (4)**



1 SCENE BUILDER



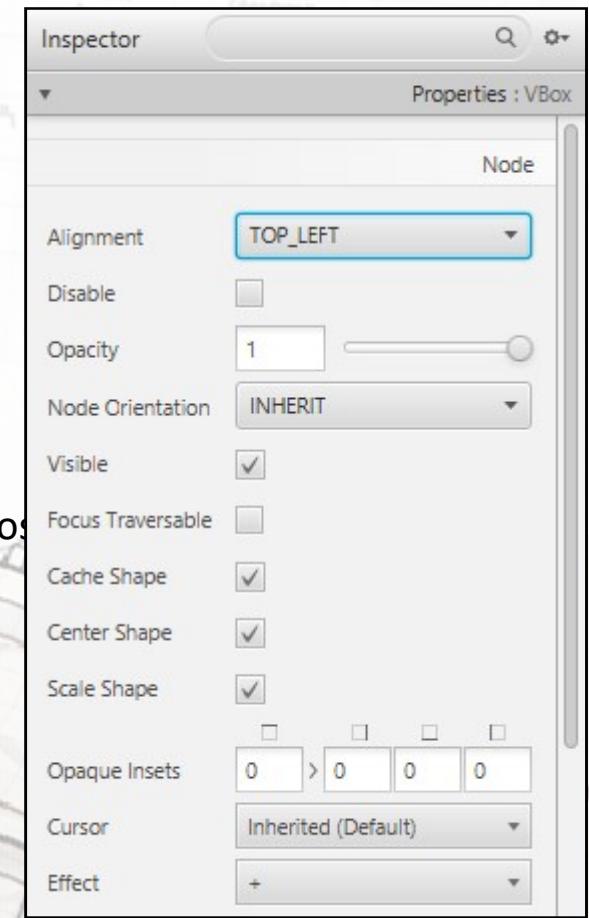
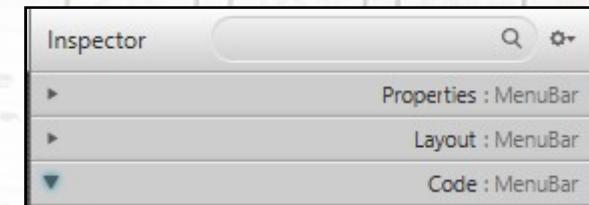
FUNDAMENTOS DE SCENE BUILDER

- ✓ **Inspector:** depende de cada componente y permite configurar y/o buscar las opciones de cada nodo

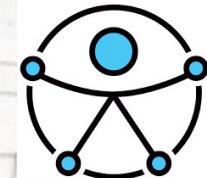
— Properties:

- **Node:** Propiedades generales de NODO: aunque pueden variar, en general se mantienen para todos los nodos: texto, fuente, alineación, etc..
- Y las específicas de cada NODO: Es **IMPOSIBLE** conocer todas (es la verdadera dificultad de esta aplicación, **en la siguiente parte de apuntes veremos los atributos más utilizados**). Por ejemplo, en los botones tiene dos acciones específicas, Default Button y Cancel Button, que asocia botón con Intro y Escape.
- **AYUDA:** Al pinchar sobre la etiqueta de la propiedad nos lleva a la explicación de la documentación oficial. Ej: defaultButton

```
defaultButtonProperty  
  
public final BooleanProperty defaultButtonProperty()  
  
A default Button is the button that receives a keyboard VK_ENTER press, if no other node in the scene consumes it.
```



1 SCENE BUILDER



FUNDAMENTOS DE SCENE BUILDER

- **CSS:** Son los estilos que se pueden aplicar a la interfaz (lo trabajaremos en tema 2-5)
- **Extras:** son propiedades referentes a cómo pinta el componente o cómo lo anima.
- **Accesibilidad:** Si tenemos algún lector de pantallas activo como por ejemplo “Narrador” (Windows) utilizará estos campos para sintetizar el texto en voz.

The image displays three separate windows of the JavaFX Scene Builder application, each showing a different tab of the properties panel:

- JavaFX CSS Tab:** Shows fields for "Style", "Style Class", "Stylesheets" (with a plus sign), and "Id".
- Extras Tab:** Shows settings for "Blend Mode" (SRC_OVER), "Cache" (checkbox), "Cache Hint" (DEFAULT), "Depth Test" (INHERIT), "Insets" (a grid of 5 input fields), "Mouse Transparent" (checkbox), and "Pick On Bounds" (checkbox).
- Accessibility Tab:** Shows fields for "Accessible Text" (text box with "Aceptar"), "Accessible Help" (text box with "Acepta los cambios y cierra"), "Accessible Role" (dropdown menu with "BUTTON" selected), and "Accessible Role Description" (text box with a blue border).

1 SCENE BUILDER



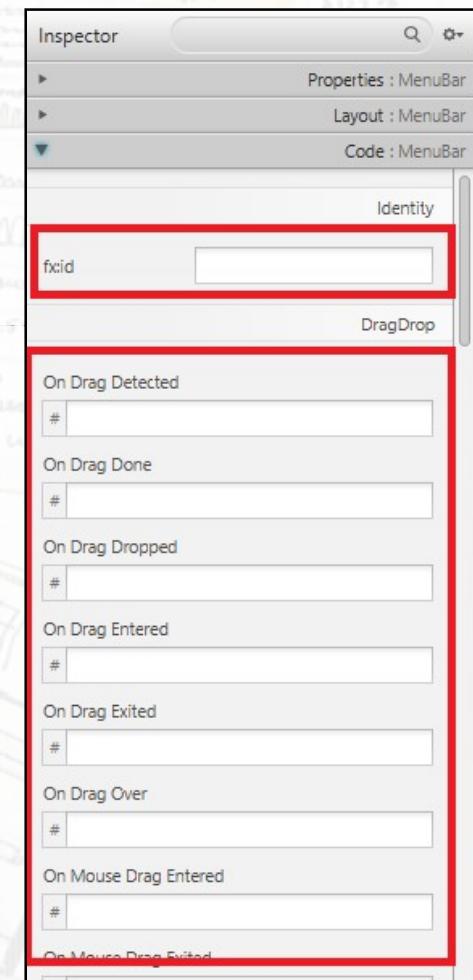
FUNDAMENTOS DE SCENE BUILDER

– Layout:

- Tamaño, bordes, posición, límites, transformaciones, etc..

– Code:

- El identificador principal y TODOS los eventos asociados al componente
- Se crean enlaces mediante el campo fx:id
- Permite asignar muchos tipos de acciones



1 SCENE BUILDER



UTILIDADES Y ACCESOS RÁPIDOS

- ✓ Doble click en componente (librería) para añadirlo al diseño. Hay que fijarse que esté seleccionado (en naranja) un elemento contenedor en la jerarquía (Hierarchy).
- ✓ Ctrl+P (Preview)
- ✓ Modify → Scene Size (Tamaño escena)
- ✓ Hierarchy (botón derecho) → Wrap In (asignar padre). También se puede desasignar (Unwrap) si tiene otro parente por arriba que se pudiera hacer cargo.
- ✓ Hierarchy (botón derecho) → Use Computed Sizes (utiliza tamaños de hijos para establecer el del parente. Es como “quitar aire sobrante” sin tener que ajustar a mano)
- ✓ Preview → Scene Style Sheets (gestionar CSS [tema 2-5])

1 SCENE BUILDER



CONTENEDORES Y CONTROLES

- ✓ Los hemos visto por orden de importancia al crearlos por código, pero aquí se ven por categorías:
- ✓ Aquellos que NO se han visto aún se verán en el proyecto [javafx-ejemplos-basicos-otros](#)

Slider (horizontal)
Slider (vertical)
Spinner (FX8)
SplitMenuItem
TableColumn
TableView
TextArea
TextField
ToggleButton
TreeTableColumn (FX8)
TreeTableView (FX8)
TreeView
WebView

Containers
Accordion
Accordion (empty)
AnchorPane
BorderPane
ButtonBar (FX8)
DialogPane (empty) (FX8)
DialogPane (FX8)
FlowPane
GridPane
HBox
Pane
ScrollPane
ScrollPane (empty)
SplitPane (empty)
SplitPane (horizontal)
SplitPane (vertical)
StackPane
Tab
TabPane
TabPane (empty)
TextFlow (FX8)
TilePane
TitledPane
TitledPane (empty)
ToolBar
VBox

Controls
Button
CheckBox
ChoiceBox
ColorPicker
ComboBox
DatePicker (FX8)
HTMLEditor
Hyperlink
ImageView
Label
ListView
MediaView
MenuBar
MenuButton
Pagination
PasswordField
ProgressBar
ProgressIndicator
RadioButton
ScrollBar (horizontal)
ScrollBar (vertical)
Separator (horizontal)
Separator (vertical)

1 SCENE BUILDER



CONTENEDORES

✓ Destacados:

- **VBox** y **HBox**: permiten crear elementos dispuestos vertical u horizontalmente.
- **Pane** y **AnchorPane**: son paneles de disposición libre, aunque el segundo permite “anclar” elementos al borde del Pane y que los elementos sean responsive (sección Layout → Anchor Pane Constraint).
- **Tabpane**: organización en pestañas
- **TitledPane**: despliega un panel al pinchar en su título
- **Toolbar**: barra de herramientas que normalmente agrupa botones

Containers	
Accordion	
Accordion (empty)	
AnchorPane	
BorderPane	
ButtonBar (FX8)	
DialogPane (empty) (FX8)	
DialogPane (FX8)	
FlowPane	
GridPane	
HBox	
Pane	
ScrollPane	
ScrollPane (empty)	
SplitPane (empty)	
SplitPane (horizontal)	
SplitPane (vertical)	
StackPane	
Tab	
TabPane	
TabPane (empty)	
TextFlow (FX8)	
TilePane	
TitledPane	
TitledPane (empty)	
ToolBar	
VBox	

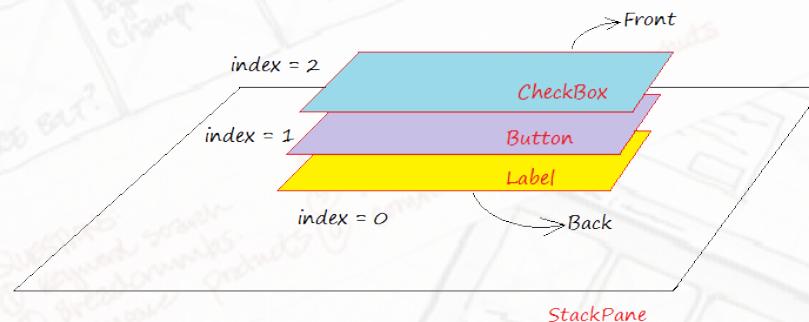
1 SCENE BUILDER



CONTENEDORES

– **Stackpane**: permite establecer elementos por “capas” (es el equivalente al JLayeredPane de Swing).

- Solo el que está en Front puede tener el foco, el resto serán visibles a no ser que se oculten con .setVisible()
- nodo.toBack(): envía atrás del todo
- nodo.toFront(): envía al primer plano



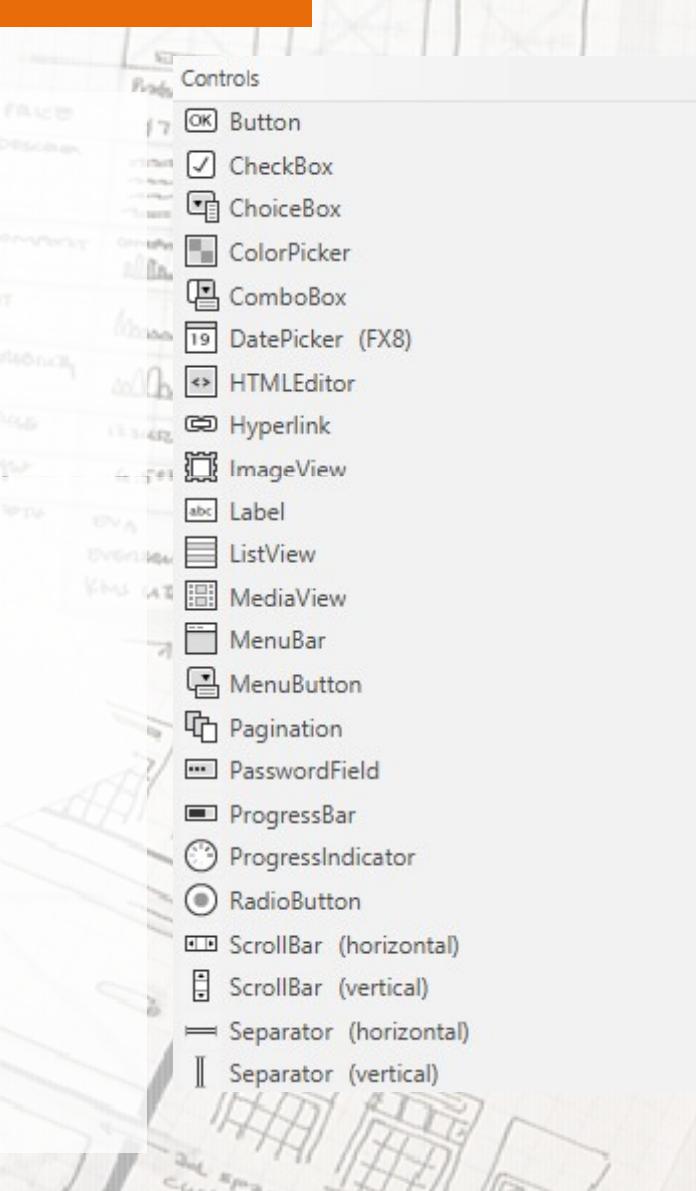
Containers	
Accordion	
Accordion (empty)	
AnchorPane	
BorderPane	
ButtonBar (FX8)	
DialogPane (empty) (FX8)	
DialogPane (FX8)	
FlowPane	
GridPane	
HBox	
Pane	
ScrollPane	
ScrollPane (empty)	
SplitPane (empty)	
SplitPane (horizontal)	
SplitPane (vertical)	
StackPane	
Tab	
TabPane	
TabPane (empty)	
TextFlow (FX8)	
TilePane	
TitledPane	
TitledPane (empty)	
ToolBar	
VBox	

1 SCENE BUILDER



CONTROLES BÁSICOS I

- ✓ Destacados:
 - **Button:** botón de acción
 - **Checkbox:** botón de activo/inactivo
 - **Combobox/Choicebox:** despliegue para selección
 - **DatePicker:** selección de fecha
 - **ImageView:** imágenes
 - **Label:** etiqueta
 - **PasswordField:** campo de password
 - **RadioButton:** botón de selección dependiente
 - **Menubar:** barra de menú superior



1 SCENE BUILDER



CONTROLES BÁSICOS II

✓ Destacados:

- **ListView**: útil cuando queremos mostrar una serie de datos en una lista. En principio NO se puede llenar desde aquí, solo configurar.
- **TableView**: muy utilizado para mostrar datos en columnas. En principio NO se puede llenar desde aquí, solo configurar.
- **TextArea**: área de texto para insertar text
- **TextField**: campo de texto para insertar datos

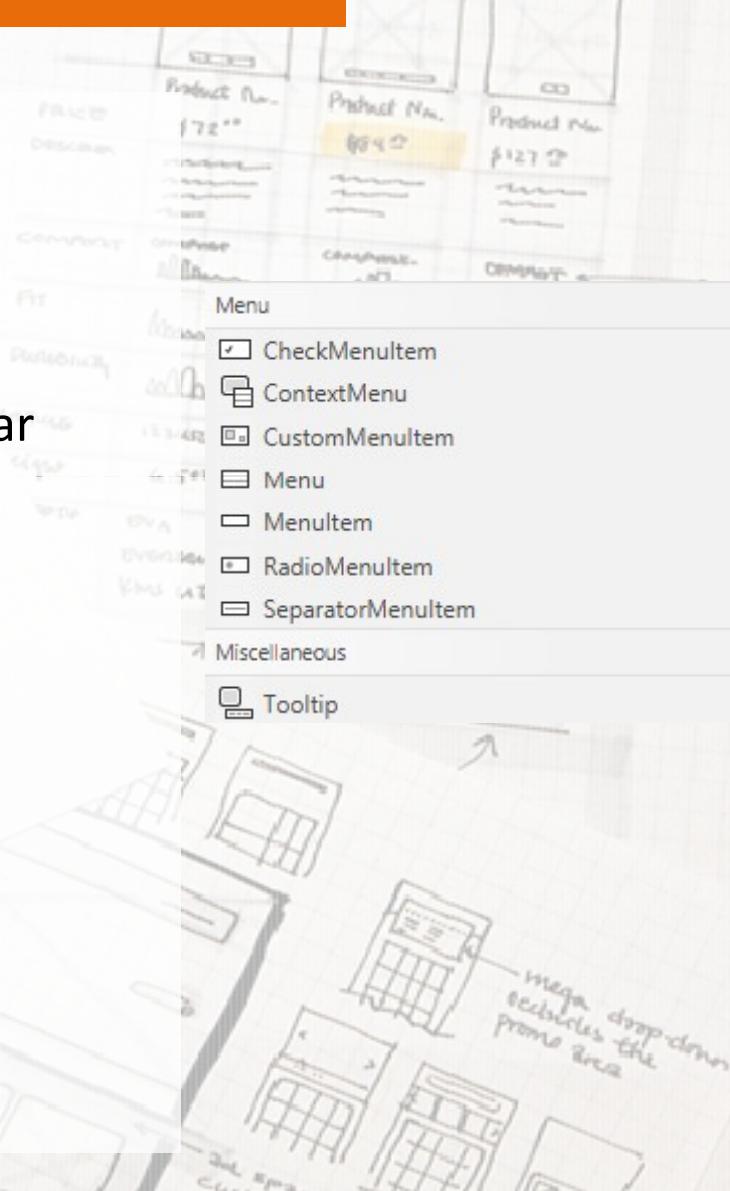
Slider (horizontal)
Slider (vertical)
Spinner (FX8)
SplitMenuItem
TableColumn
TableView
TextArea
TextField
ToggleButton
TreeTableColumn (FX8)
TreeTableView (FX8)
TreeView
WebView

1 SCENE BUILDER



MENÚS Y MISCELÁNEA

- ✓ Destacados:
 - **MenuItem**: componente de menú tanto Menubar como de ContextMenu
 - **Check/Radio/Separator-MenuItem**: checkbox, radiobutton y separador pero dentro de un menú.
 - **Tooltip**: ayuda (usabilidad)



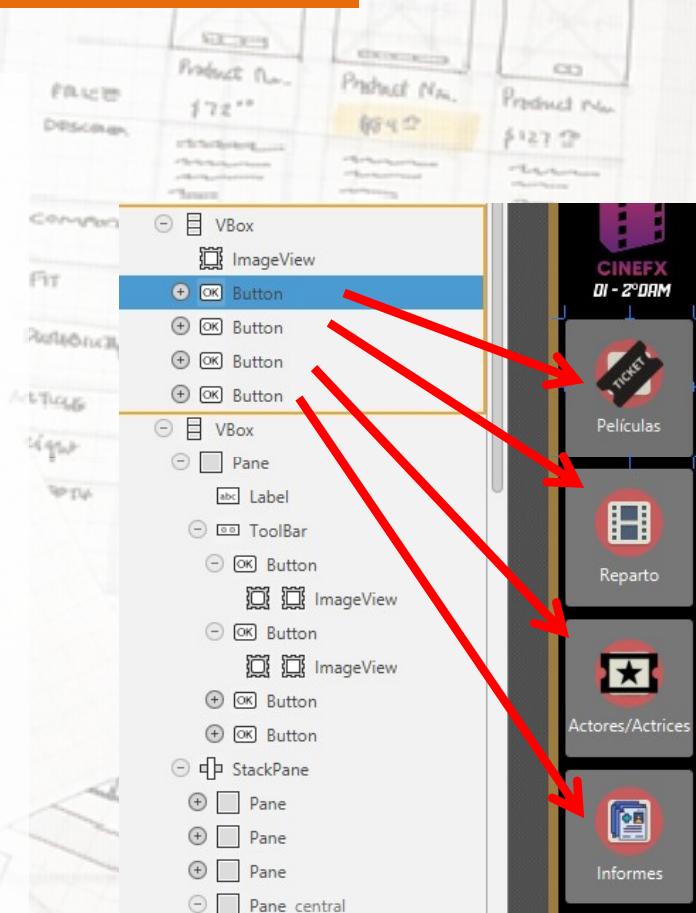
1 SCENE BUILDER



MENÚS Y MISCELÁNEA

✓ TABORDER:

- Orden de tabulación con tecla de TAB, o con Shift+TAB para orden inverso
- También se puede navegar con las flechas
- Lo establece el orden dentro de la jerarquía (más alto va antes). Ejemplo:

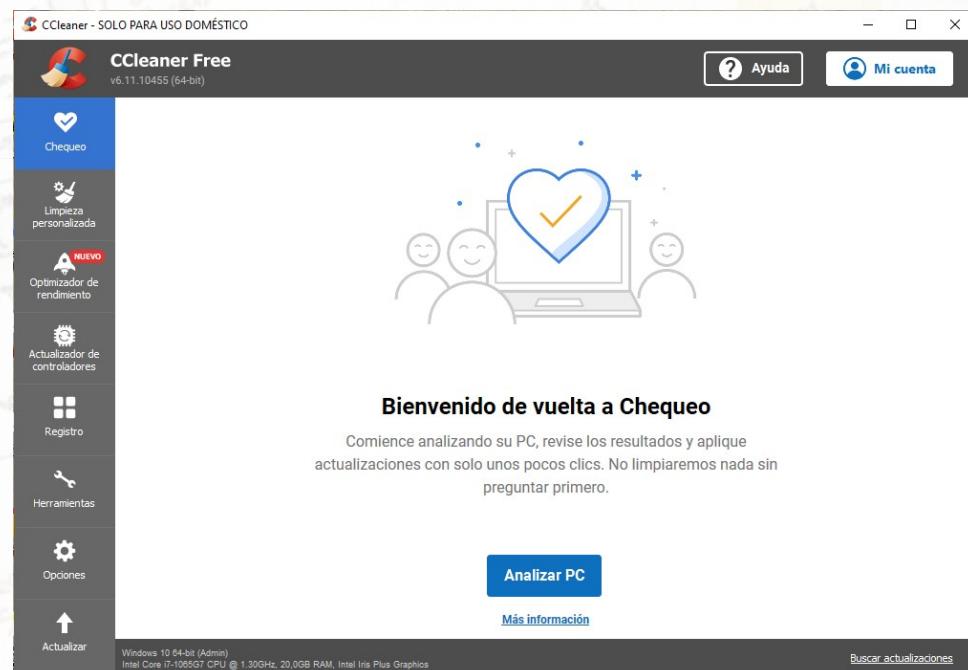


2 EJEMPLOS



EJEMPLO: Intentar “recrear” una GUI similar a la de CCleaner siguiendo estos pasos. **ALTERNATIVA A:**

1. **Identificar esqueleto:** 1 HBOX (2 col) conteniendo 2 VBOX:
 1. VBOX Izq: 9 filas o componentes
 2. VBOX Der: Contiene 3 Paneles (porque hace falta separar componentes)
 3. Tamaño y posición de botones y textos
2. **Identificar iconos, colores, tipo de letra y aplicar CSS.**



Prácticas: recrear el resto de pantallas: limpieza, Optimizador, Registro, Herramientas, etc..

2 EJEMPLOS

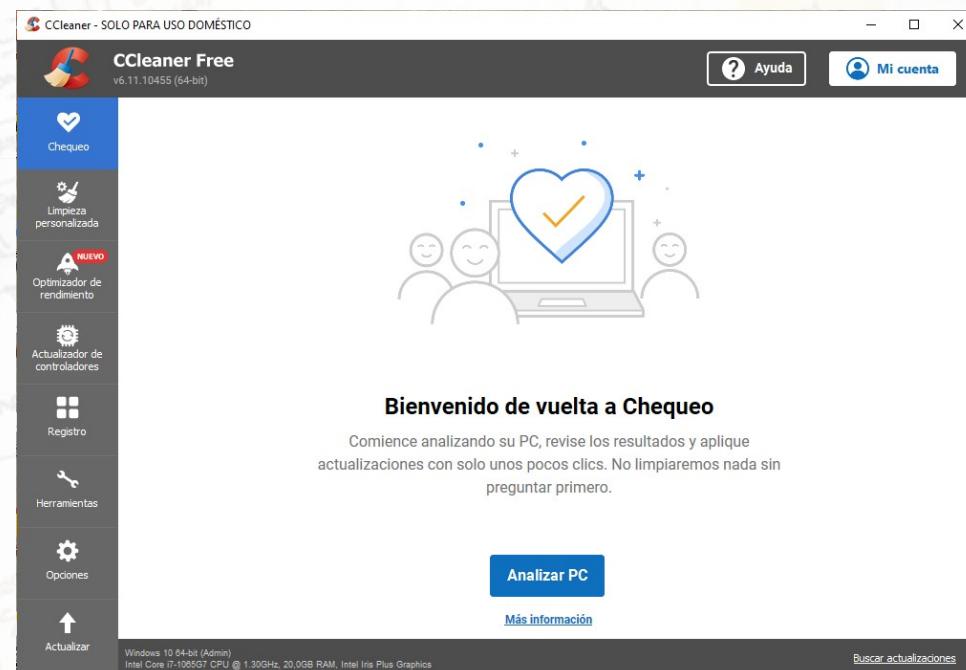


EJEMPLO: Intentar “recrear” una GUI similar a la de CCleaner siguiendo estos pasos. **ALTERNATIVA B:**

1. Identificar esqueleto: 1 BORDERPANE:

1. A Izquierda: VBOX 8 filas o componentes
2. Al centro: un StackPane
3. Arriba: un Pane con imágenes y 2 botones
4. Abajo: un Pane con labels

2. Identificar iconos, colores, tipo de letra y aplicar CSS.



Prácticas: recrear el resto de pantallas: limpieza, Optimizador, Registro, Herramientas, etc..

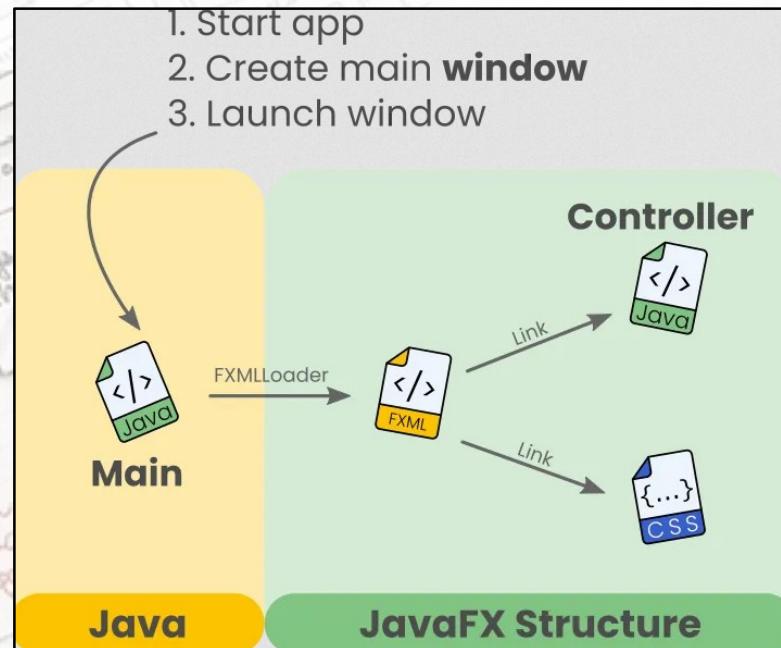
3 CLASE CONTROLADOR



FUNDAMENTOS DE SCENE BUILDER

- ✓ Hasta ahora se han creado los nodos con código, pero eso no usa el MVC
- ✓ Controlador: clase JAVA que hace de intermediario entre el FXML y la APP (aunque ya hemos visto que no es obligatorio, simplifica mucho la APP) .
- ✓ Funcionamiento:

Nos vamos apoyando en el proyecto [javafx-fxml](#)

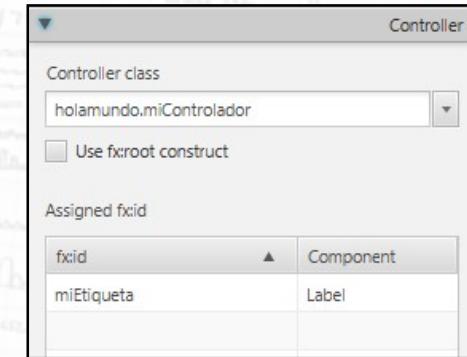


3 CLASE CONTROLADOR



FUNDAMENTOS DE SCENE BUILDER

- ✓ **Controlador:** se define en la sección Controller indicando el nombre del package donde irá (en el caso de no colocarlo en la misma carpeta que el FXML).



Error muy común! No poner la ruta correcta al Controlador (revisar si está en otro package o en el propio)

- ✓ A) Se crean enlaces mediante el campo *fx:id*
- ✓ B) Le ponemos un nombre en Controller class (en la sección Controller)
- ✓ C) Vemos las etiquetas que se han creado de forma automática desde View → Show Sample Controller Skeleton (es un ejemplo orientativo, no hay que tomarlo como código válido ni definitivo!)

3 CLASE CONTROLADOR



FUNDAMENTOS DE SCENE BUILDER

```
Sample Skeleton for 'ejemplo.fxml' Controller Class

package holamundo;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Label;

public class miControlador {

    @FXML
    private Label miEtiqueta;

    @FXML
    void botonPulsado(ActionEvent event) {
    }
}
```



```
public class miControlador implements Initializable {

    @FXML
    private Label miEtiqueta;

    @FXML
    private void botonPulsado() {
        miEtiqueta.setText(string: "Hola Mundo");
        System.out.println(: "Hola Mundo!!!");
    }

    @Override
    public void initialize(URL url, ResourceBundle rb) {
        miEtiqueta.setText(string: "");
    }
}
```

- ✓ D) Copiamos todas las etiquetas @FXML, además, eliminamos los parámetros de las cabeceras de las funciones.
- ✓ y E) Con este código creamos una clase **public class Controlador implements Initializable** (implementa la interface **Initializable**). Nos obligará a implementar todos los métodos abstractos, esto es, a crear el método *initialize* (inicialización de componentes). Eliminamos la excepción que lanza y personalizamos el contenido.

3 CLASE CONTROLADOR



JERARQUÍA DE ÁRBOL Y MVC

- **Clase Main:** es el punto de inicio del programa, extiende de **Application** y su constructor llama al método launch (importante: solo se puede llamar a un launch por aplicación)
- A su vez, el launch llama al método start **donde :**
 - Se carga un fichero FXML con la GUI (no es obligatorio como ya hemos hecho con los ejercicios ya vistos en tema 2-1) y se crea un nodo principal del árbol
 - Se configura el Stage inicial
 - Se crea una escena (Scene), se le asigna al Stage y se muestra
 - **Abrimos el proyecto holamundoFXML (nos servirá de Plantilla!)**

```
public class Main extends Application {  
  
    public static void main(String[] args) {  
        launch(args);  
    }  
  
    @Override  
    public void start(Stage primeraEscena) throws Exception {  
        Parent root = FXMLLoader.load(getClass().getResource(name: "ejemplo.fxml"));  
  
        Scene scene = new Scene(parent: root);  
        primeraEscena.setScene(scene);  
        primeraEscena.setTitle(string: "Primer Ejemplo JavaFX - Hola Mundo");  
        primeraEscena.show();  
    }  
}
```

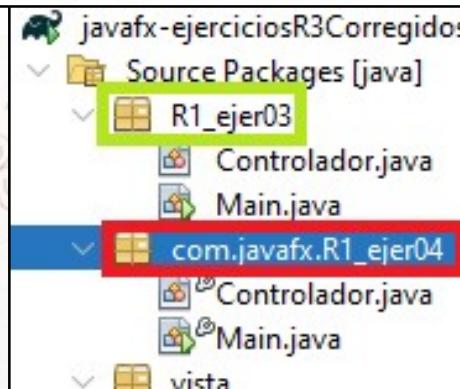
3 CLASE CONTROLADOR



JERARQUÍA DE ÁRBOL Y MVC

- **Carga de FXML:** Se puede cargar el FXML desde otro package pero hay que tener cuidado
 - Si el package tiene un carácter extraño como por ejemplo un nombre con . no cargará el fichero
 - En el ejemplo de abajo, la clase Main de R1_ejer03 sí cargará el fxml que está en vista, pero no lo hará la clase Main de R1_ejer04.
 - Ejemplo correcto (R1_ejer03):

```
@Override  
public void start(Stage stage) throws Exception {  
    Parent root = FXMLLoader.load(getClass().getResource(name: ".../vista/Empleados.fxml"));
```



3 CLASE CONTROLADOR



JERARQUÍA DE ÁRBOL Y MVC

- **Carga de Recursos:** es importante tener en *build.gradle* una línea que indique que podemos cargar recursos a la hora de hacer el **despliegue**
- En este caso, indicamos que queremos que pueda cargar los *.fxml que cuelguen de la carpeta /src/main/java:
- También se pueden añadir otros como css, html...
- Ni decir tiene que no es obligatorio que los fxml estén junto al código, se podrían haber llevado a Resources (esto es un tema organizativo)

```
sourceSets {  
    main {  
        resources {  
            srcDirs = ["src/main/java"]  
            includes = ["**/*.fxml"]  
        }  
    }  
}
```

3 CLASE CONTROLADOR



JERARQUÍA DE ÁRBOL Y MVC

- **Clase Controller:** gestiona los eventos de todos los elementos de la parte gráfica definidos en el archivo fxml enlazado
- Tiene un método *initialize* obligatorio (se carga al inicio e inicializa todo lo que nos haga falta)
- Contiene una etiqueta @FXML por cada elemento o acción que quiere enlazar desde la interfaz gráfica.

```
public class miControlador implements Initializable {  
  
    @FXML  
    private Label miEtiqueta;  
  
    @FXML  
    private void botonPulsado() {  
        miEtiqueta.setText(string: "Hola Mundo");  
        System.out.println(x: "Hola Mundo!!");  
    }  
  
    @Override  
    public void initialize(URL url, ResourceBundle rb) {  
        miEtiqueta.setText(string: "");  
    }  
}
```

3 CLASE CONTROLADOR



JERARQUÍA DE ÁRBOL Y MVC

- **Fichero FXML:** define la parte gráfica, aunque no los tengamos que escribir, hay que saber interpretarlos
- Hay que indicarle cual es la clase controlador
- Por cada elemento “enlazado” tendrá que haber una etiqueta `fx:id` o en caso de evento un `on<evento>` (ej: `onAction`, `onKeyTyped`, etc..)

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.layout.VBox?>

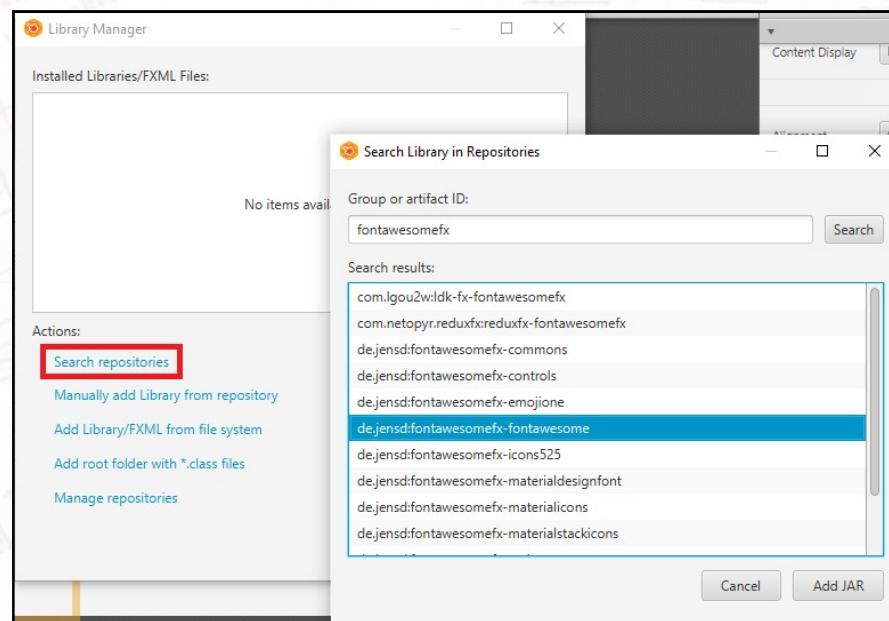
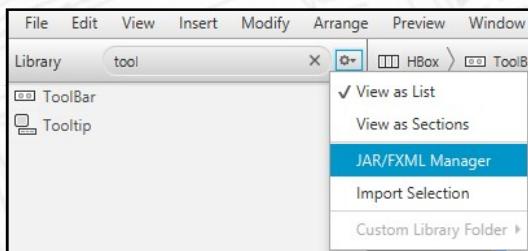
<VBox alignment="CENTER" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity"
       minWidth="-Infinity" prefHeight="115.0" prefWidth="365.0"
       xmlns="http://javafx.com/javafx/19" xmlns:fx="http://javafx.com/fxml/1" fx:controller="holamundo.miControlador">
    <children>
        <Button mnemonicParsing="false" onAction="#botonPulsado" text="Púlsame!" />
        <Label fx:id="miEtiqueta" text="Label" />
    </children>
</VBox>
```

4 ANEXO



AÑADIR LIBRERÍAS EXTERNAS

- ✓ Doble click en



- ✓ En realidad se trata de bajar un jar con un componente personalizado..
- ✓ El **tema 3 de Creación de Componentes** consiste en esto: crear componentes gráficos, añadirlos y reutilizarlos. Permite definir propiedades nuevas en el inspector de propiedades.

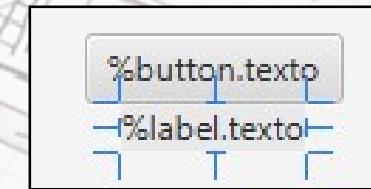
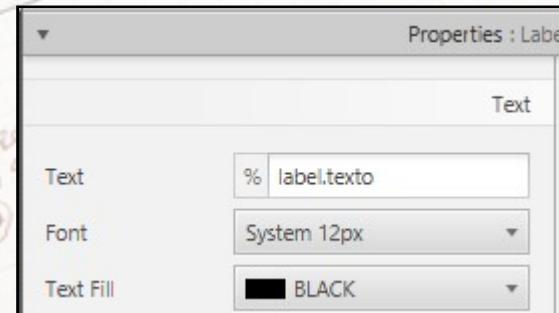
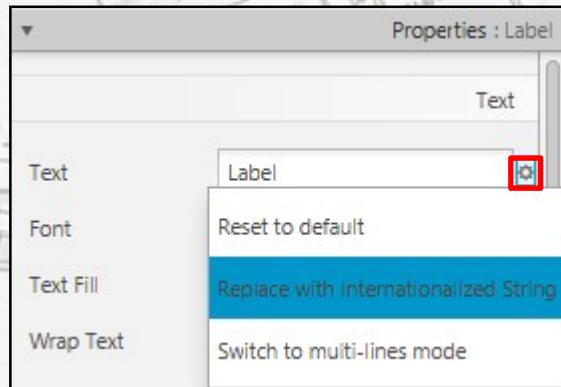
4 ANEXO



INTERNACIONALIZACIÓN

Nos vamos apoyando en el proyecto [javafx-fxml](#)

- Vamos a suponer que queremos tener un mecanismo de traducción semiautomático para los textos de la interfaz.
- De forma normal habría que crear un fxml para cada idioma, pero existe un método sencillo de traducción.
- PASO 1: En el atributo Text de cualquier Nodo que lo admita, pinchamos en la rueda pequeña de la derecha
- PASO 2: Le damos el formato de internacionalización formado por un campo de dos palabras separadas por un punto, por ejemplo:
label.texto

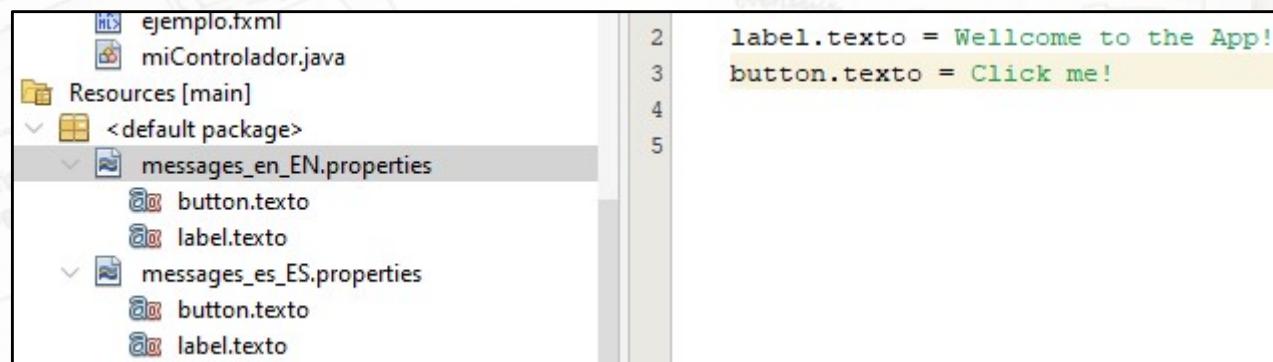


4 ANEXO



INTERNACIONALIZACIÓN

- PASO 3: Nos vamos a la carpeta resources del proyecto y creamos un fichero de propiedades. Rellenamos tantos campos de texto como traducciones quiero tener:



- En este ejemplo se crean 2 ficheros para inglés y castellano



4 ANEXO

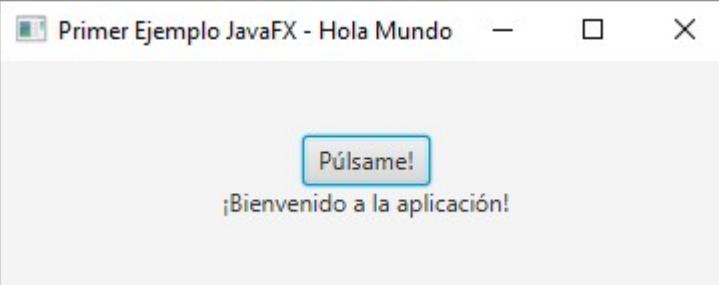


INTERNACIONALIZACIÓN

- PASO 4: Por último, podemos elegir el lenguaje usando un bundle u otro. Lo suyo sería crear otro fichero de propiedades que me permitiera elegirlo. Dicho fichero debería estar fuera del proyecto (veremos un ejemplo de esto cuando nos metamos con el acceso a las Bases de Datos)

```
//Se establecen los lenguajes y regiones
Locale castellanoLocale = new Locale.Builder().setLanguage("es").setRegion("ES").build();
Locale englishLocale = new Locale.Builder().setLanguage("en").setRegion("EN").build();
ResourceBundle bundle = ResourceBundle.getBundle("messages", castellanoLocale);
//ResourceBundle bundle = ResourceBundle.getBundle("messages", englishLocale);

FXMLLoader loader = new FXMLLoader(getClass().getResource("ejemplo.fxml"));
loader.setResources(bundle);
```



4 ANEXO



INTERNACIONALIZACIÓN

- También se pueden cargar los ficheros de recursos de idioma desde Preview → Internationalization → Set Resource.
- Se busca el fichero de properties y muestra el resultado.

