Iniciación al diseño de interfaces gráficos con Scene Builder, NetBeans y JavaFX

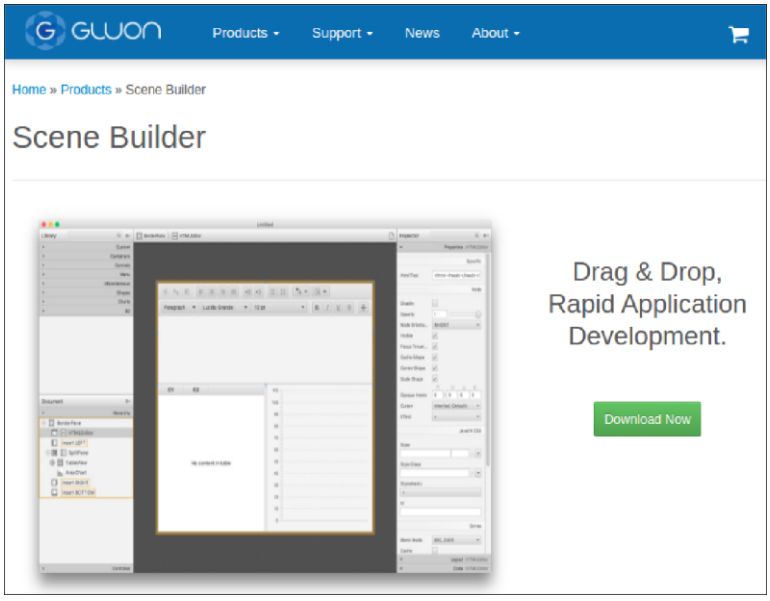
Scene Builder

Para **crear el entorno gráfico** de la aplicación desde Java se pueden utilizar diversas librerías como AWT, Swing, JavaFX, etc, y en este caso se usará **JavaFX** al ser más moderno que los anteriores. Si se desea crear una ventana desde JavaFX se puede hacer completamente usando código Java, o bien utilizar una **herramienta gráfica como *Scene Builder (http://gluonhq.com/products/scene-builder/)*** que facilita el diseño de los interfaces gráficos, ya que ofrece los diferentes elementos que se pueden colocar en la ventana usando una paleta desde la que se pueden arrastrar los elementos deseados en el lugar deseado de la ventana.

Además se pueden ajustar las propiedades de los distintos elementos que formarán parte de la ventana desde esa misma herramienta, de una manera más visual que utilizando directamente el código fuente Java.

Instalación de Scene Builder

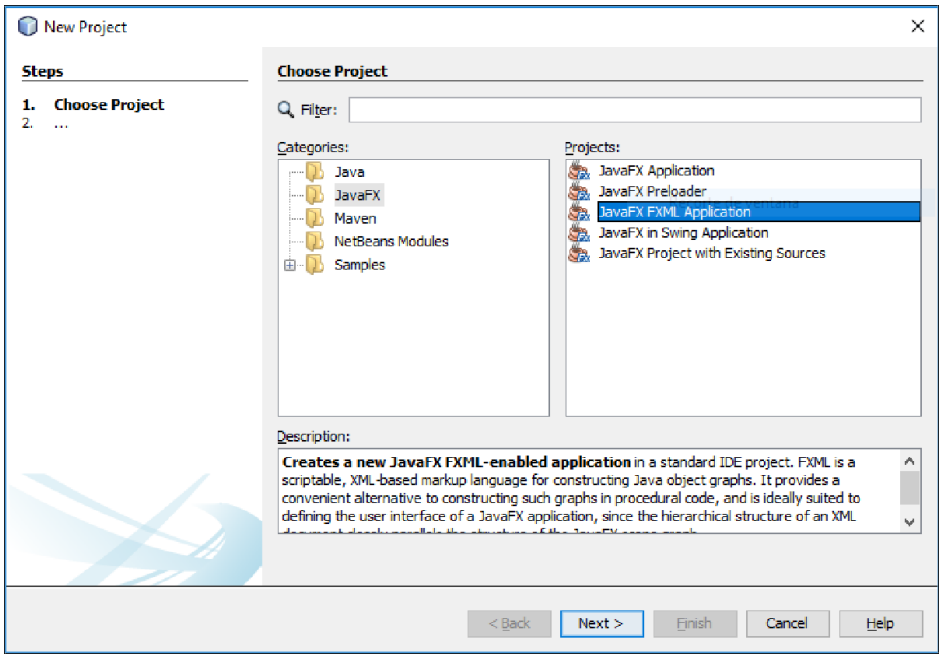
Lógicamente, el primer paso para utilizar *Scene Builder* es instalarlo en nuestro equipo si no lo teníamos ya. Así que accede a la **web de *Scene Builder*** (http://gluonhq.com/products/scene-builder/ (http://gluonhq.com/products/scene-builder/)) y utiliza el botón ***Download Now*** para acceder a la página de descargas, y selecciona a continuación la **descarga correspondiente al sistema operativo** que estés utilizando en tu ordenador.



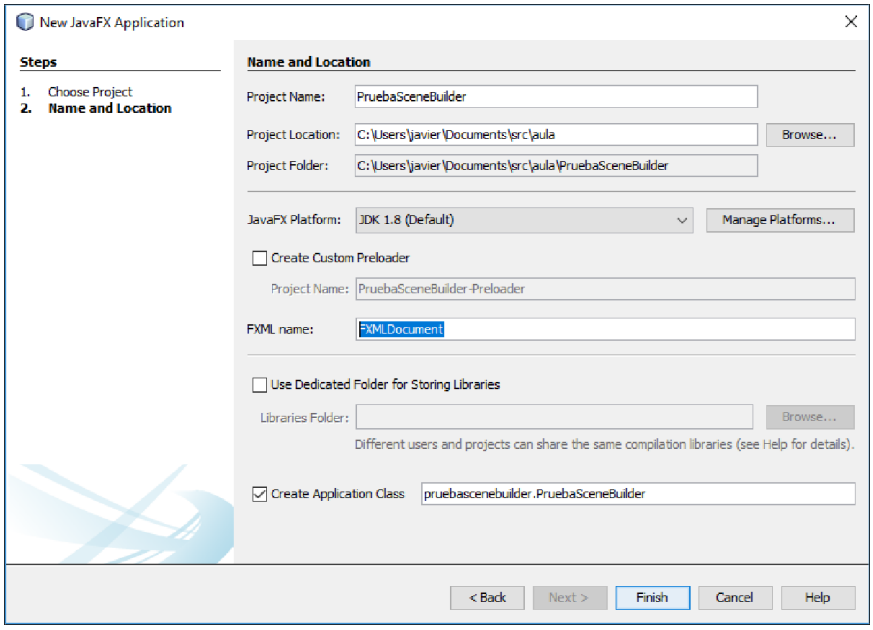
Una vez completada la descarga, realiza la **instalación** de este producto de la manera habitual.

Archivo FXML y clase controladora

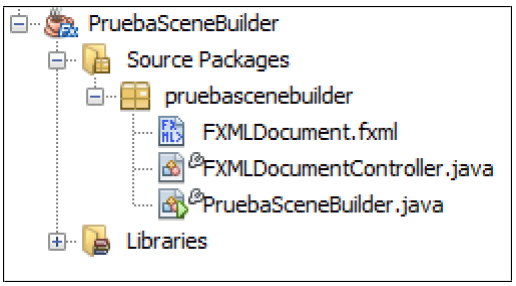
Para empezar a conocer el modo de funcionamiento de esta herramienta, puedes crear un proyecto (*File* > *New Project*) de prueba en NetBeans desde la categoría *JavaFX*, seleccionando la opción ***JavaFX FXML Application***.



El asistente de creación de este tipo de proyectos pregunta en la siguiente pantalla aspectos comunes a otros proyectos Java (como el nombre del proyecto, la carpeta donde se ubicará, el nombre de la clase principal de la aplicación, etc), y en este caso también permite indicar el **nombre del archivo que contendrá el diseño del interfaz gráfico** que podrás editar con *Scene Builder*. El nombre asignado por defecto es *FXMLDocument*, aunque lo puedes cambiar por otro si lo deseas.



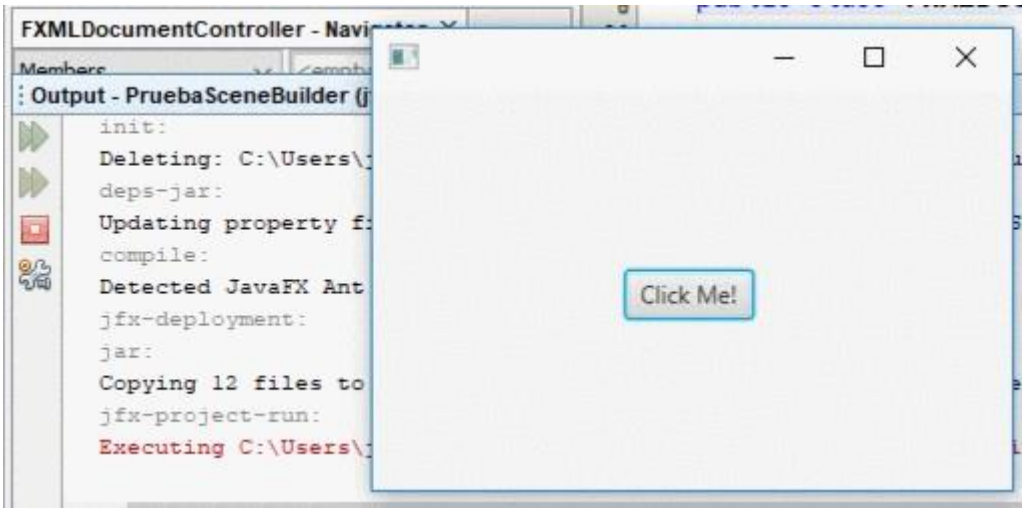
Una vez finalizado el asistente puedes ver que se han creado los siguientes archivos en el paquete de fuentes:



- **FXML** (*FXMLDocument.fxml*)
  - Archivo XML que contiene el diseño con la estructura de los elementos que forman la ventana.
- **FXML Controller** (*FXMLDocumentController.java*)
  - Clase Java que se encargará de establecer las acciones que deben realizar los elementos que forman la ventana.
- **JavaFX Main Class** (*PruebaSceneBuider.java*)
  - Clase Java que se encargará de mostrar el diseño realizado en el archivo FXML, y que permite la ejecución de la aplicación.

> Puede ser que en algún caso tuvieras en **proyecto creado anteriormente, que no fuera de tipo *FXML Application***, pero al que desees agregar un diseño de interfaz con FXML. Esto puedes hacerlo añadiendo un archivo nuevo al paquete de fuentes de tipo ***Empty FXML***. Si este tipo de archivos no puedes encontrarlo inicialmente en la opción *New* del menú contextual del proyecto, podrás verlo en la sección *New* > *Other* > *JavaFX*. En este caso deberás tener en cuenta que la carga del contenido del archivo FXML deberás hacerla escribiendo el código necesario, que será similar al que verás a continuación, el cual se genera automáticamente con el asistente de creación de proyectos de tipo FXML.

Si ejecutas la aplicación verás que aparece una ventana con un único botón, que al pulsarlo muestra un mensaje bajo el botón y en la salida estándar.



Carga del archivo FXML

Si observas el **código de la clase principal** encontrarás que se hace **referencia al archivo FXML** que se ha creado.

```
public void start(Stage stage) throws Exception {
    Parent root = FXMLLoader.load(getClass().getResource("FXMLDocument.fxml"));
    Scene scene = new Scene(root);
    stage.setScene(scene);
    stage.show();
}
```

De esa manera se está consiguiendo que todo el **contenido del archivo FXML se cargue** dentro de un contenedor genérico de tipo *Pane* de JavaFX (que se almacena en la variable *root*). Dicho contenedor *Pane* será el contenedor principal de los elementos gráficos de la aplicación, y por ello en la línea siguiente se le asigna a la escena (*Scene*) de la ventana de la aplicación.

Archivo FXML

Como ya se ha comentado antes, el archivo FXML contiene (en formato XML) la **estructura de los elementos gráficos** que formarán parte del interfaz de usuario. Viendo la parte principal de su contenido:

```
<AnchorPane id="AnchorPane" prefHeight="200" prefWidth="320" xmlns:fx="http://javafx.com/fxml/1" fx:c
ontroller="pruebascenebuilder.FXMLDocumentController">
    <children>
        <Button layoutX="126" layoutY="90" text="Click Me!" onAction="#handleButtonAction" fx:id="but
ton" />
        <Label layoutX="126" layoutY="120" minHeight="16" minWidth="69" fx:id="label" />
    </children>
</AnchorPane>
```

De manera general observa que declara un contenedor principal de tipo ***AnchorPane***(permite anclar los elementos que contenga a otros elementos), y dentro de él como hijos (***children***) aparecen un botón (***Button***) y una etiqueta de texto (***Label***).

> En el artículo Introduction to FXML (http://docs.oracle.com/javase/8/javafx/api/javafx/fxml/doc-files/introduction_to_fxml.html) puedes encontrar información sobre los distintos elementos XML que se pueden utilizar, aunque **lo normal será que hagas el diseño desde la aplicación *Scene Builder*** de manera más gráfica y sencilla.

Dentro de cada etiqueta aparecen diversas **propiedades y los valores** que tienen asociadas. Por ejemplo, se indica el texto (*Click Me!*) que debe aparecer en el botón:

```
text="Click Me!"
```

Para tipo de elemento de FXML tendrá sus propiedades que serán diferentes a las de otros elementos, pero **desde *Scene Builder* podrás conocer de manera visual las posibles propiedades** que podrás aplicar a cada elemento.

Otro aspecto **importante** a conocer del archivo FXML es la **referencia a la clase Java** que va a hacer las funciones de clase **controladora**. En este ejemplo puedes ver que es *FXMLDocumentController* como aparece en la **propiedad *fx:controller*** dentro de la etiqueta *AnchorPane*.

```
fx:controller="pruebascenebuilder.FXMLDocumentController"
```

Clase controladora

Cada archivo FXML debe tener asociada una clase Java a la que se suele denominar "controladora". En esta clase se deben declarar las instrucciones Java que se deseen ejecutar

cuando se inicialice el interfaz gráfico contenido en al archivo FXML, o también, por ejemplo, declarar el método que debe ejecutarse cuando el usuario haga clic en un botón, o el código necesario para cambiar un campo de texto, rellenar una lista de datos, etc.

El asistente de NetBeans que ha creado el proyecto genera una clase controladora con un código como el siguiente:

```
public class FXMLDocumentController implements Initializable {

    @FXML
    private Label label;

    @FXML
    private void handleButtonAction(ActionEvent event) {
        System.out.println("You clicked me!");
        label.setText("Hello World!");
    }

    @Override
    public void initialize(URL url, ResourceBundle rb) {
        // TODO
    }

}
```

Observa que contiene declarado el **método *initialize***, aunque su contenido está vacío. Las instrucciones Java que indiques dentro de este método se ejecutarán cada vez que se cargue el interfaz gráfico durante la ejecución de la aplicación.

Encima de él se encuentra el **método *handleButtonAction***que posee la **anotación @FXML**. Esa anotación indica que el elemento que aparece a continuación de él está asociado a algún elemento del archivo FXML. En este caso, el método handleButtonAction aparecía asociado al botón, como puedes recordar en esta línea resumida:

```
<Button onAction="#handleButtonAction" />
```

De esta manera se consigue que **cuando el usuario haga clic en el botón** (*onAction*), se ejecuten las sentencias contenidas en el método *handleButtonAction*. En el código que estamos analizando, este método muestra en la salida estándar el mensaje "*You clicked me!*" y cambia el texto de la etiqueta que está contenida en interfaz gráfico con el mensaje "*Hello World!*".

Sobre la declaración de este método está la declaración de la variable *label* que también tiene la anotación @FXML. En este caso esta variable está asociada al elemento *Label* del archivo FXML, ya que se le ha asignado el **identificador (id) *label***, como puedes recordar con esta línea resumida del archivo FXML:

```
<Label fx:id="label" />
```

> El **identificador** que se asigne a un elemento FXML se debe corresponder con el nombre de la **variable** que se declare en la clase controladora.

Observa que en la clase controladora se le ha asignado ese mismo nombre a la variable:

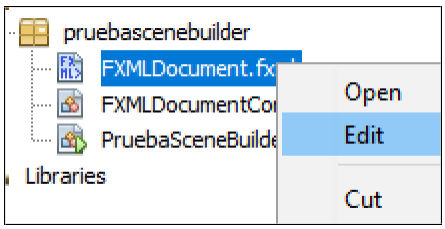
```
    @FXML
    private Label label;
```

De esta manera, cualquier acción que se realice sobre el objeto al que hace referencia la variable tendrá efecto sobre el elemento FXML que tenga ese identificador. En el código que estamos analizando, dentro del método asociado a la acción del botón, se realiza el cambio del texto que contiene esa etiqueta con la sentencia:

```
label.setText("Hello World!");
```

Vista de diseño del archivo FXML

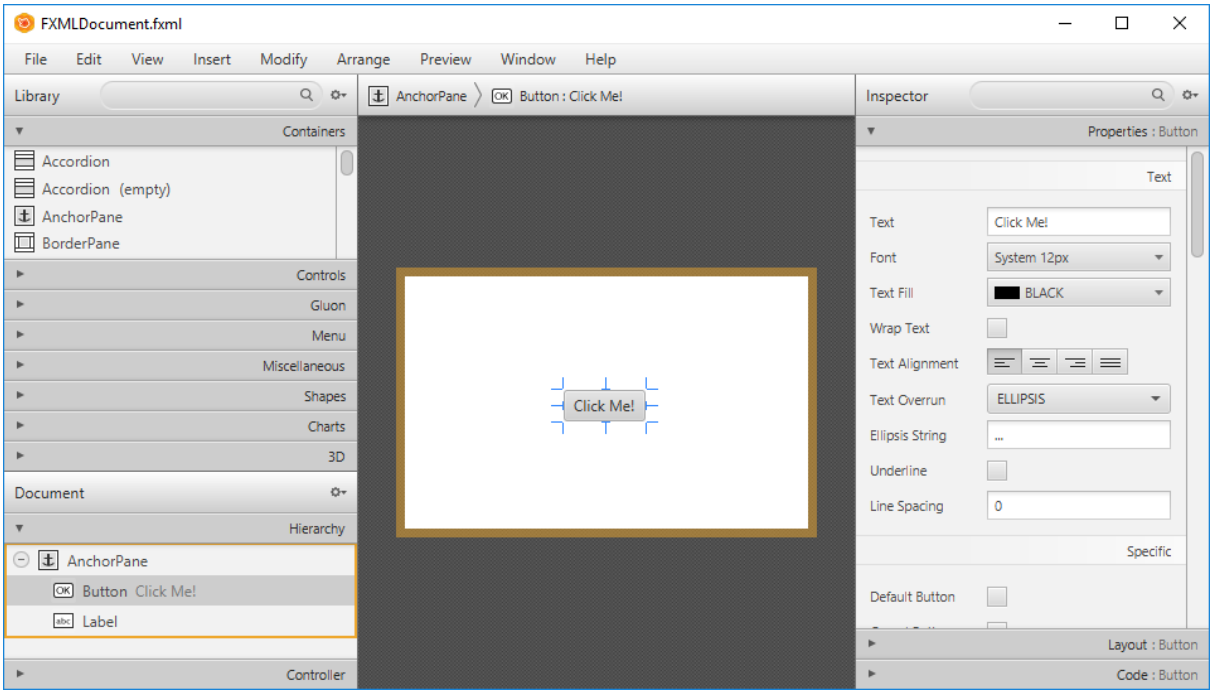
El contenido del archivo FXML has podido verlo y editarlo a través del código XML que contiene, y que ha mostrado NetBeans en una de las pestañas abiertas al crear el proyecto. Si en algún momento necesitaras acceder de nuevo a la vista del código y tuvieras esta pestaña cerrada, deberás abrir el archivo usando la opción *Edit* del menú contextual del archivo.



Pero lo más cómodo generalmente es editar el interfaz gráfico en modo diseño, utilizando para ello la aplicación que has instalado, Scene Builder. La manera más rápida de **abrir el archivo FXML en Scene Builder** es haciendo **doble clic sobre él**, o seleccionando la opción ***Open*** del menú anterior.

> En caso de que no se abra Scene Builder directamente desde NetBeans al abrir un archivo FXML, **abre el menú *Tools* > *Options*** de **NetBeans** y accede a la **sección *Java*** y desde ahí a la **pestaña *JavaFX***. En la **lista desplegable *Scene Builder Home*** selecciona la **opción *Browse*** y localiza la carpeta donde se encuentre instalado *Scene Builder*.

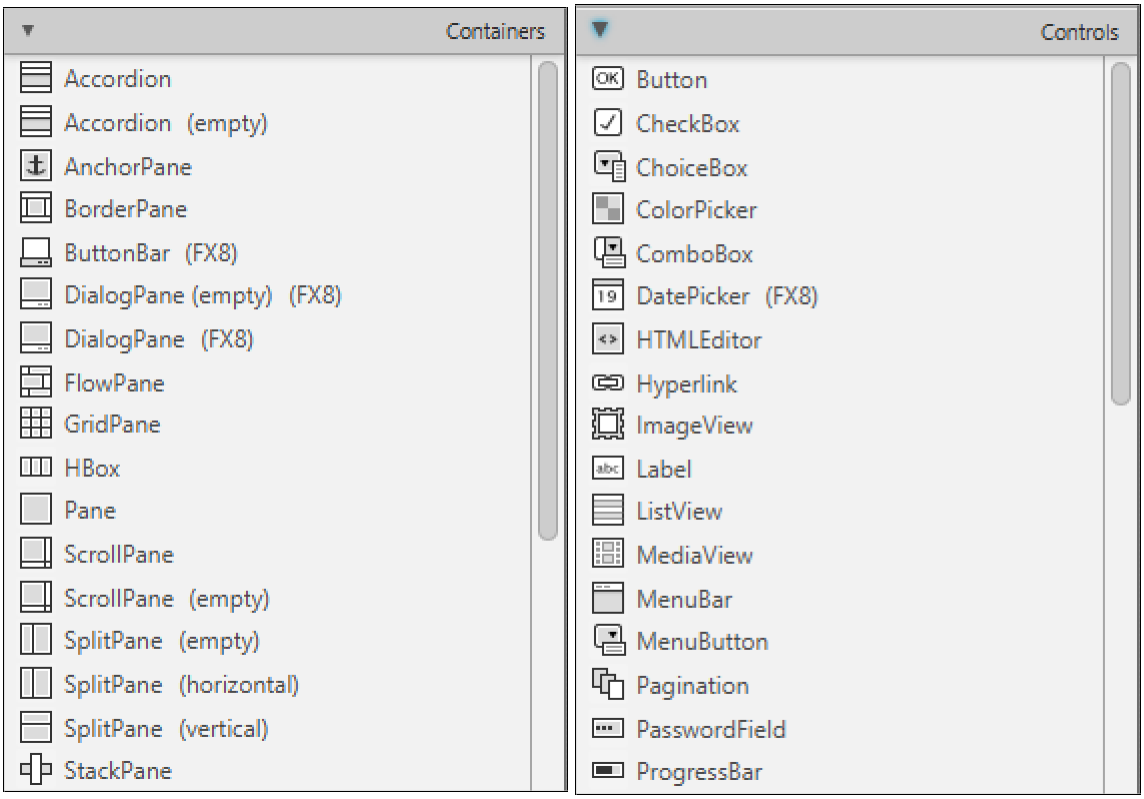
Se debe mostrar la misma ventana de la aplicación, pero en modo diseño:



Sección *Liibrary*

En la parte derecha de la ventana de Scene Builder puedes encontrar esta sección que engloba varios apartados como: *Containers, Controls, Gluis, Menu*, etc. Ahí puedes encontrar los elementos gráficos que puedes añadir al interfaz gráfico que estés diseñando.

Dentro de todos esos apartados los más importantes pueden ser el de ***Containers***, donde se encuentran los distintos tipos de contenedores, como *HBox, VBox, BorderPane, GridPane, ScrollPane*, etc, que permiten organizar los elementos que se quieran añadir al diseño, y la sección ***Controls***que contiene los elementos más comunes de la mayoría de los interfaces gráficos como: botones, etiquetas de texto, campos de texto, casillas de verificación, listas, etc.



Para **añadir un elemento al diseño, tan sólo debes arrastrarlo** hacia el lugar deseado, teniendo en cuenta que los elementos como los controles siempre deben colocarse dentro de un elemento de tipo contenedor. Por defecto se crea un contenedor general de tipo *AnchorPane*, que son los que permiten colocar elementos de manera bastante libre, anclando los elementos respecto a la posición de otros.

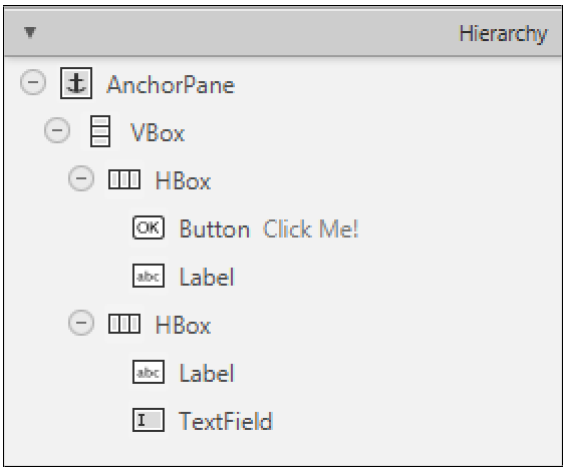
Sección *Document*

Esta sección sólo incluye los apartados *Hierarchy* (jerarquía) y *Controller*.

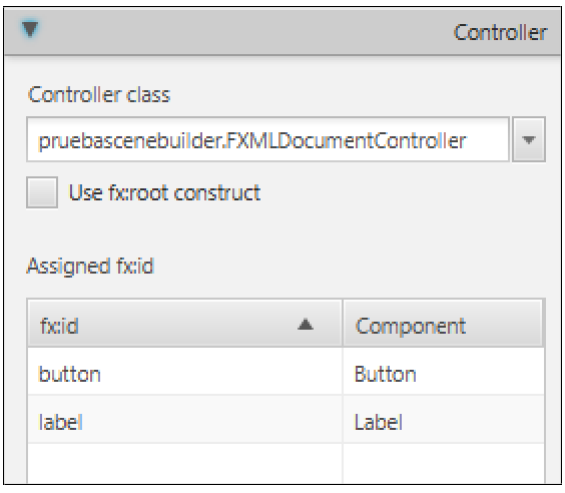
En el apartado ***Hierarchy***puedes ver de manera detallada **qué elementos están contenidos dentro de otros**, y puede resultar de bastante utilidad cuando desees **reorganizar** los elementos dentro de los distintos contenedores que tengas en el diseño.

Puede ser más **cómodo arrastrar los elementos a esta sección** en lugar de arrastrarlos directamente a la vista de diseño de la zona central.

Observa este **ejemplo** donde dentro del contenedor principal (*AnchorPane*) se ha colocado un contenedor vertical (*VBox*) dentro del cual hay 2 contenedores horizontales (*HBox*) con algunos controles.



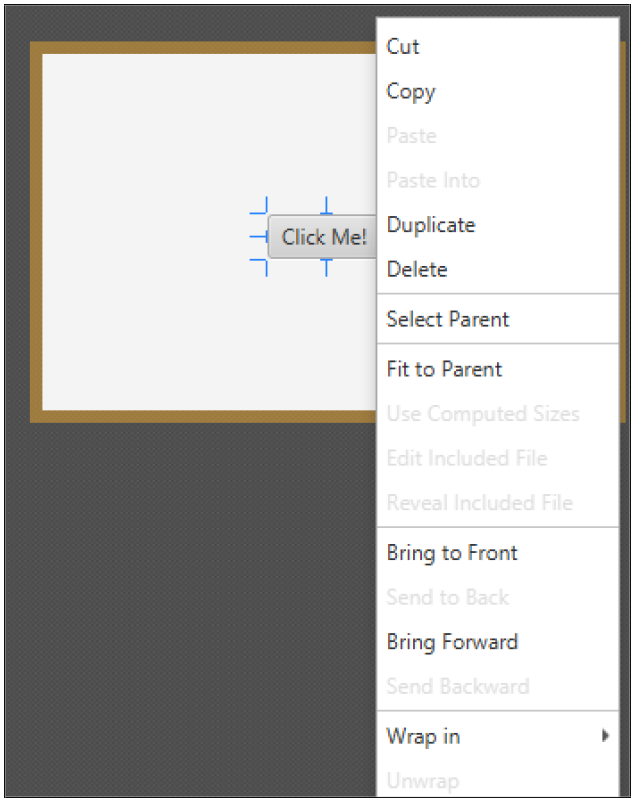
En el apartado ***Controller***puedes ver y modificar el **nombre de la clase** que va a hacer las funciones de **controladora** de este archivo FXML. También muestra los **identificadores** que se han asignado a los elementos del diseño que lo tengan. Ten en cuenta que si cambias el nombre de la clase controladora desde NetBeans o la mueves a otro paquete de fuentes, deberás cambiar los nuevos datos en este apartado.



Zona de diseño

En la parte central se encuentra la zona de diseño donde puedes ir colocando los distintos elementos de la librería o ver cómo irían quedando si los añades al árbol de jerarquía que se ha comentado hace un momento.

Si seleccionas un determinado elemento puedes modificar su tamaño tirando con el ratón desde los indicadores azules de las esquinas o laterales, y abriendo el menú contextual podrás encontrar varias acciones que podrás realizar sobre ese elemento.



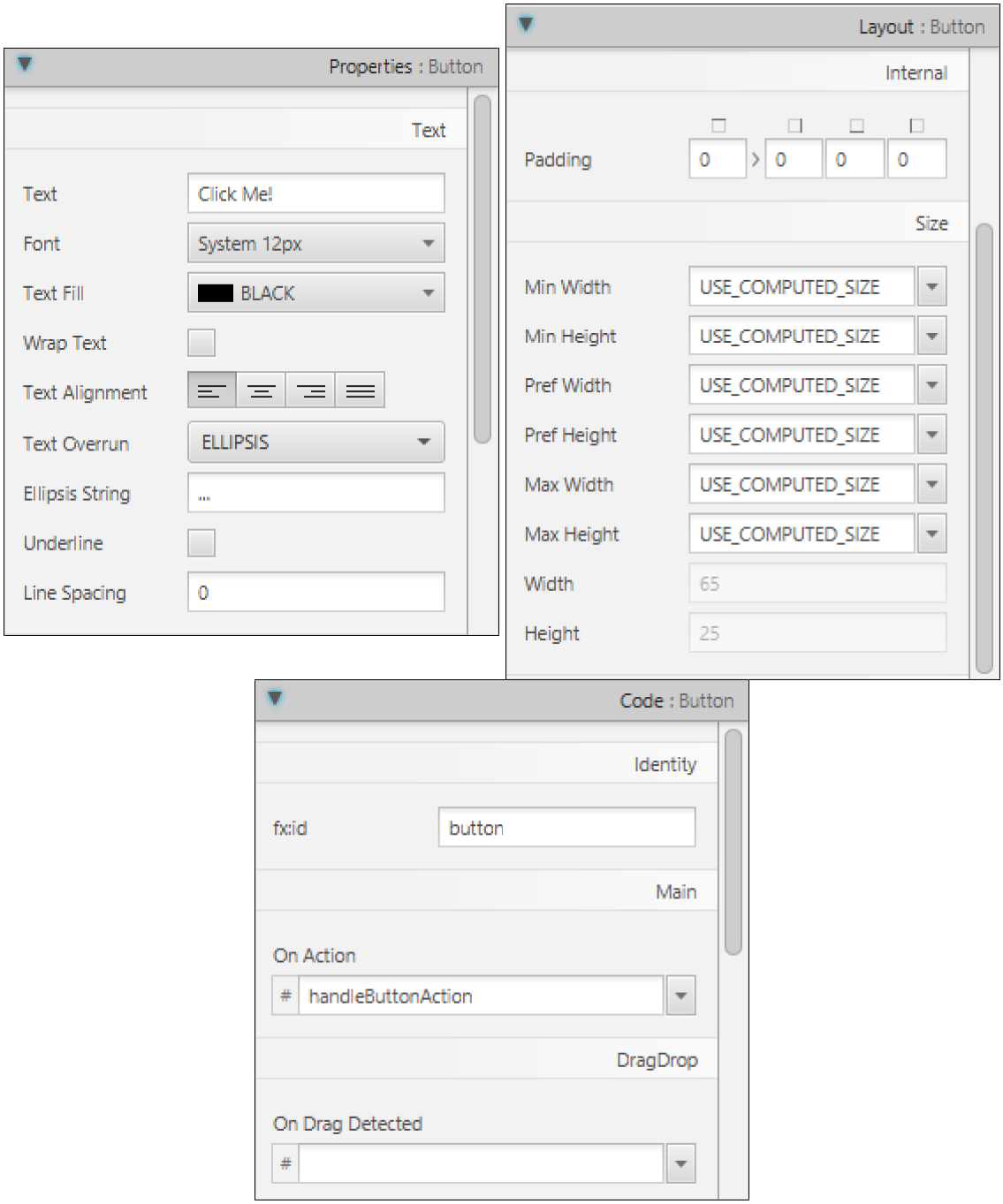
Sección Inspector

Contiene los apartados *Properties, Layout, y Code*.

En el apartado ***Properties***podrás consultar y modificar los **valores asignados a las propiedades** que ofrece el elemento que se encuentre seleccionado. Según el tipo de elemento que sea, se mostrarán distintas propiedades. Aquí podrás cambiar, por ejemplo, el texto de un elemento, su color, tipo de fuente, alineaciones, o asignarle un estilo CSS de JavaFX.

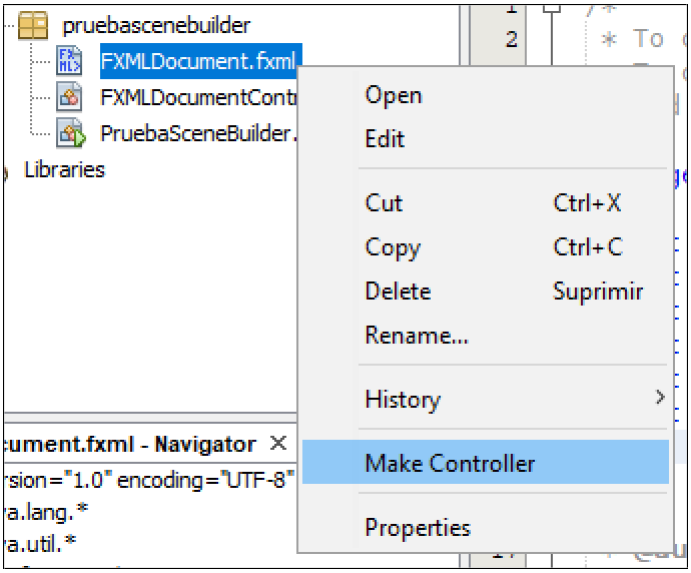
El apartado ***Layout***permite consultar y modificar las aspectos relacionados con la **distribución del elemento en pantalla**. Es decir, ahí podrás asignar márgenes, tamaños, escalas, etc.

Por último, el apartado ***Code***ofrece las **opciones relacionadas con el código Java** asociado al elemento seleccionado. Es **importante** conocer que ahí es donde se le puede **asignar un identificador** (*fx:id*). También aquí se indicará el nombre del **método que se desea ejecutar** cuando se produzca un determinado evento sobre el elemento, por ejemplo, cuando se haga clic en un botón (evento *On Action*).



Make Controller

Debes recordar que **tras cualquier cambio que realices en el apartado *Code***, y tras **guardar los cambios** del archivo en Scene Buider, debes actualizar la información en la clase controladora de NetBeans. Esto debes hacerlo con la **opción *Make Controller*** del menú contextual del archivo FXML.



Es decir, si se indica un nombre de un método para una acción, por ejemplo, cuando el usuario mueva el ratón dentro del botón, se debe seleccionar *Make Controller* para que se declare ese método en el controlador:

Recuerda pulsar la tecla Intro después de escribir el nombre del identificador o de un método para que se tenga en cuenta dicho nombre al guardar el documento FXML desde Scene Builder.

```
@FXML
private Label label;

@FXML
private void handleButtonAction(ActionEvent event) {
    System.out.println("You clicked me!");
    label.setText("Hello World!");
}

@Override
public void initialize(URL url, ResourceBundle rb) {
    // TODO
}

@FXML
private void ratonDentroBoton(MouseDragEvent event) {
}
```