

# JAVAFX

# JavaFX

JavaFX es una tecnología Java para el diseño de aplicaciones con interfaces gráficas interactivas multiplataforma. Las aplicaciones JavaFx pueden ser ejecutadas con el mismo resultado en multitud de dispositivos: pc, móviles, tv, consolas, etc. JavaFX amplía la potencia de Java permitiendo a los desarrolladores utilizar cualquier biblioteca de Java en aplicaciones JavaFX. Los desarrolladores pueden ampliar sus capacidades en Java y utilizar la tecnología de presentación que JavaFX proporciona para crear experiencias visuales que resulten atractivas.

JavaFX es una tecnología Java para el diseño de aplicaciones con interfaces gráficas interactivas multiplataforma. Las aplicaciones JavaFx pueden ser ejecutadas con el mismo resultado en multitud de dispositivos: pc, móviles, tv, consolas, etc. JavaFX amplía la potencia de Java permitiendo a los desarrolladores utilizar cualquier biblioteca de Java en aplicaciones JavaFX. Los desarrolladores pueden ampliar sus capacidades en Java y utilizar la tecnología de presentación que JavaFX proporciona para crear experiencias visuales que resulten atractivas.

# JavaFX

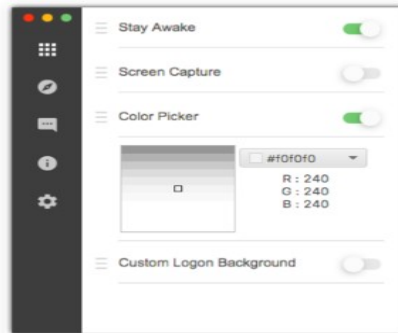
- ✓ **Java APIs:** las APIs están escritas en código nativo Java compatibles con otros lenguajes soportados por la máquina virtual.
- ✓ **FXML and Scene Builder:** FXML es un lenguaje de marcado que describe las interfaces de usuario. Se pueden escribir directamente o usar la herramienta JavaFX Scene Builder para crearlos con una interfaz gráfica.
- ✓ **WebView:** permite embeber páginas HTML en las aplicaciones JavaFX. Ofrece soporte para JavaScript.
- ✓ **Built-in UI controls and CSS:** proporciona cantidad de controles para construir aplicaciones completas. El estilo de los controles puede ser modificado con CSS.
- ✓ **Canvas API:** para dibujar directamente en la pantalla.
- ✓ **Multitouch Support:** soporte para gestos táctiles múltiples en función de las posibilidades de la plataforma subyacente.
- ✓ **Hardware-accelerated graphics pipeline:** haciendo uso de la GPU se consiguen animaciones gráficas fluidas en las tarjetas gráficas soportadas, si la gráfica no está soportada se hace uso de la pila de software Java2D.
- ✓ **High-performance media engine:** soporta la reproducción de contenido multimedia con baja latencia basándose en GStreamer.
- ✓ **Self-contained application deployment model:** las aplicaciones contenidas tienen todos los recursos y una copia privada de los entornos de ejecución de Java y JavaFX. Son distribuidos como paquetes instalables y proporcionan la misma experiencia de instalación e inicio que las aplicaciones nativas del sistema operativo.

# JavaFX

JavaFX es un conjunto de paquetes de gráficos y medios que permite a los desarrolladores diseñar, crear, probar, depurar e implementar aplicaciones de cliente enriquecido que ejecutan y se visualizan perfectamente en diversas plataformas.

Con JavaFX se pueden crear muchos tipos de aplicaciones. Por lo general, son aplicaciones que cumplen con los requisitos de ejecución en múltiples plataformas y muestran información en una interfaz de usuario moderna de alto rendimiento que incluye audio, vídeo, gráficos y animación. Esta tecnología aumenta la productividad y mejora la mantenibilidad de nuestras interfaces de usuario, manteniendo de forma separada los ficheros que implementan la lógica de negocio de los que contienen el diseño de las interfaces. En este sentido, JavaFX supera con creces las posibilidades con respecto Swing, donde crear una simple animación o dar un aspecto semiprofesional es una tarea tediosa y no demasiado simple.

En el siguiente enlace tienes acceso a la web del proyecto [JavaFX](#).



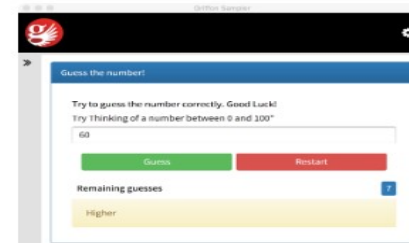
## Actlist

JavaFx Utility Platform to easy and simply execute your own act list.



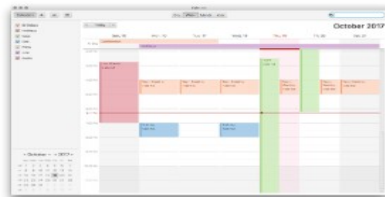
## AsciiDocFX

An AsciiDoc editor to build PDF, Epub, Mobi and HTML books, documents and slides



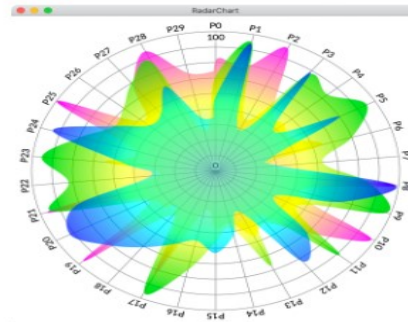
## BootstrapFX

Twitter's Bootstrap CSS for JavaFX



## CalendarFX

A Java framework for creating sophisticated calendar views



## Charts

A library for scientific charts in JavaFX.



## ControlsFX

De-facto JavaFX controls library

# *Instalación de JavaFX*

El primer paso será descargar el paquete apropiado según nuestro sistema operativo.

Existen dos paquetes diferentes: versión no modular y versión modular del proyecto. En nuestro caso utilizaremos la versión no modular. Además podemos descargar la versión 11 LTS o la última versión estable que es la 19.

[Descarga de JavaFX SDK](#)

# Instalación de JavaFX

Por lo tanto, si utilizamos Windows, puedes descargar el paquete que se observa en la imagen:

GLUON

Products ▾

Developers

Pricing

Services

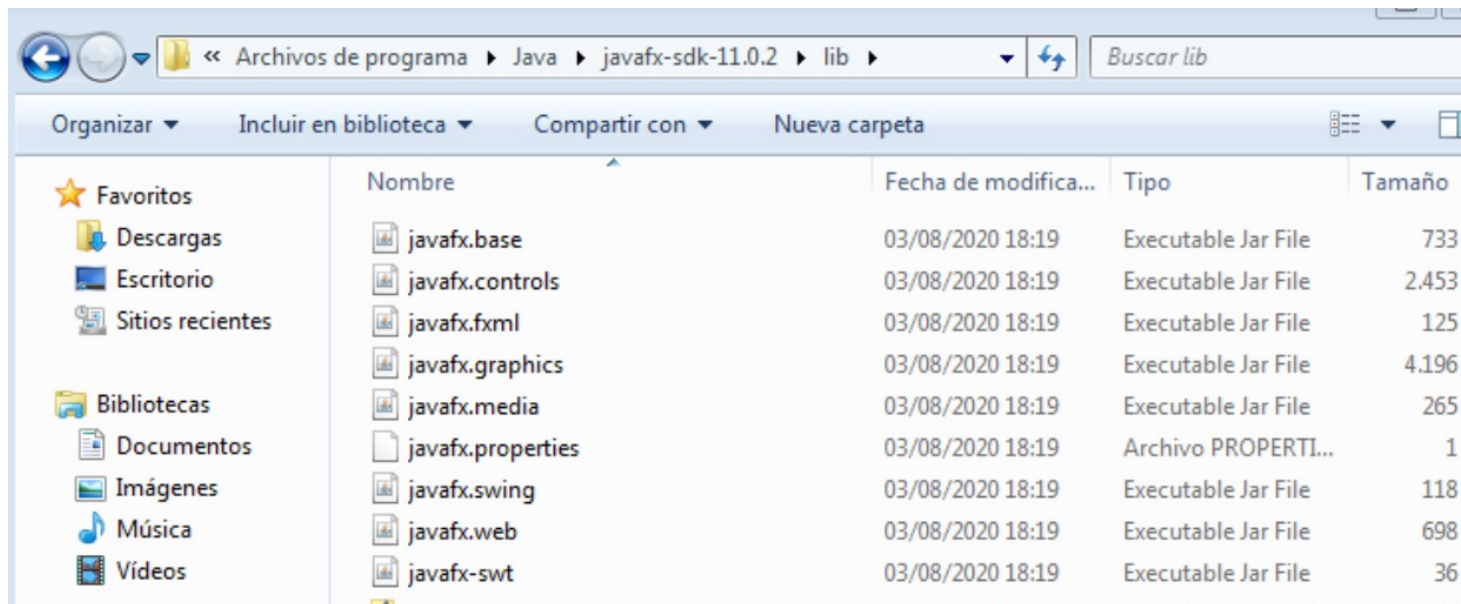
Insights ▾

Contact ▾

Linux	19	aarch64	SDK	<a href="#">Download</a> <a href="#">[SHA256]</a>
Linux	19	aarch64	jmods	<a href="#">Download</a> <a href="#">[SHA256]</a>
Linux	19	aarch64	Monocle SDK	<a href="#">Download</a> <a href="#">[SHA256]</a>
Linux	19	arm32	SDK	<a href="#">Download</a> <a href="#">[SHA256]</a>
Linux	19	x64	SDK	<a href="#">Download</a> <a href="#">[SHA256]</a>
Linux	19	x64	jmods	<a href="#">Download</a> <a href="#">[SHA256]</a>
macOS	19	aarch64	SDK	<a href="#">Download</a> <a href="#">[SHA256]</a>
macOS	19	aarch64	jmods	<a href="#">Download</a> <a href="#">[SHA256]</a>
macOS	19	aarch64	Monocle SDK	<a href="#">Download</a> <a href="#">[SHA256]</a>
macOS	19	x64	SDK	<a href="#">Download</a> <a href="#">[SHA256]</a>
macOS	19	x64	jmods	<a href="#">Download</a> <a href="#">[SHA256]</a>
macOS	19	x64	Monocle SDK	<a href="#">Download</a> <a href="#">[SHA256]</a>
Windows	19	x64	SDK	<a href="#">Download</a> <a href="#">[SHA256]</a>
Windows	19	x64	jmods	<a href="#">Download</a> <a href="#">[SHA256]</a>
Windows	19	x64	Monocle SDK	<a href="#">Download</a> <a href="#">[SHA256]</a>

# Instalación de JavaFX

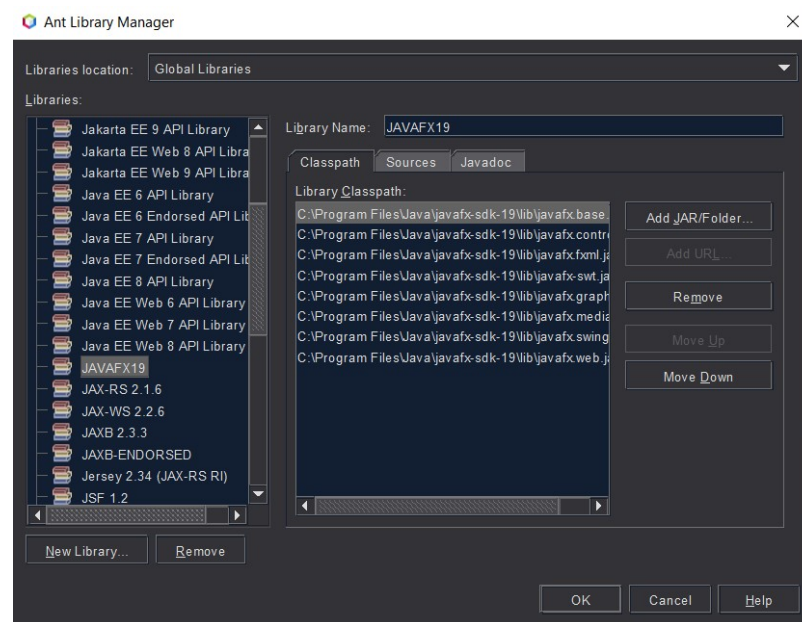
No será necesario ejecutar ningún archivo, tan solo descomprimir el paquete zip descargado en la ruta que consideres oportuna. Un lugar apropiado puede ser la carpeta de instalación de Java.





# Instalación de JavaFX. Netbeans

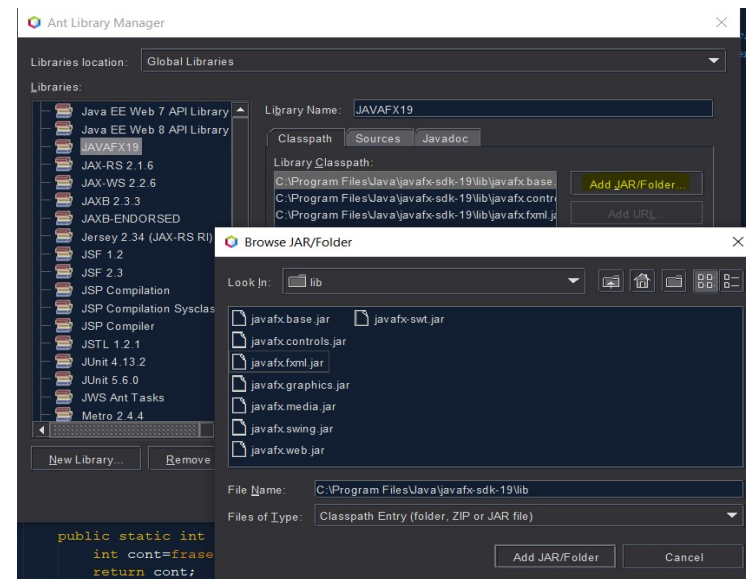
Accedemos al menú de Netbeans Tools - Library. Vamos a crear una nueva librería que contenga los ficheros del JavaFX SDK. Pulsamos en el botón New Library: tan solo tendremos que asignarle un nombre, por ejemplo, JavaFX19



# Instalación de JavaFX. Netbeans

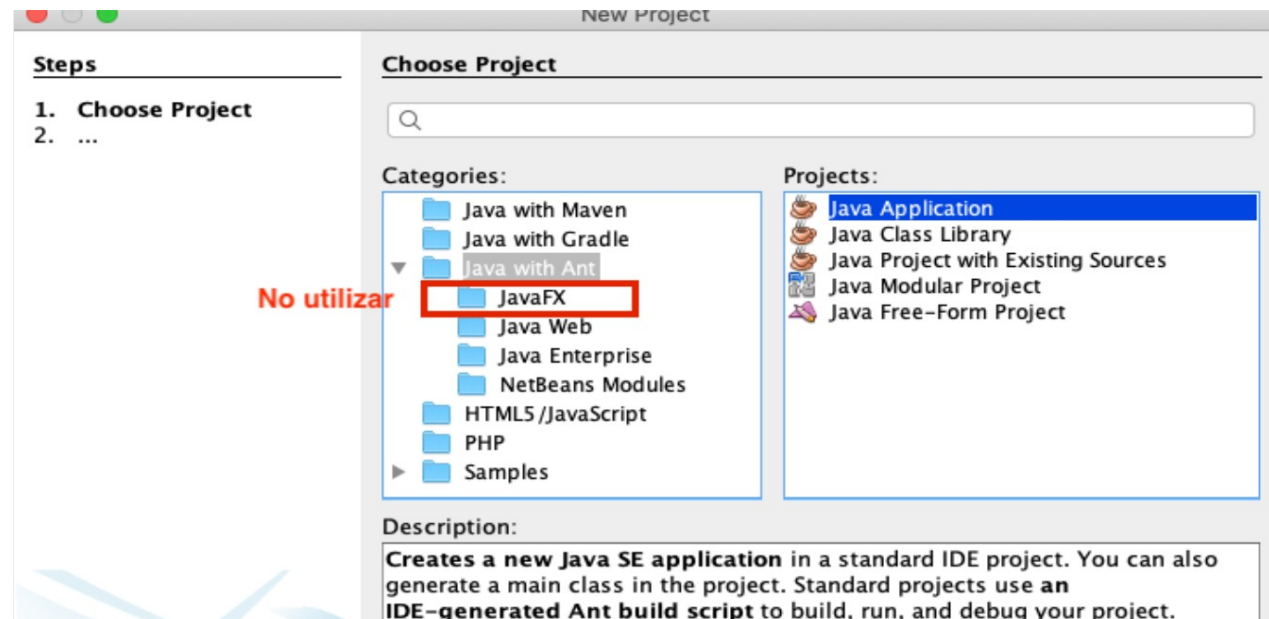
Por último, vamos a añadir a la librería creada los ficheros .jar contenidos en la carpeta lib de la carpeta de instalación del jdk.

IMPORTANTE: No incluir el fichero src.zip que también está contenido en esa carpeta, puede generar errores en la ejecución.



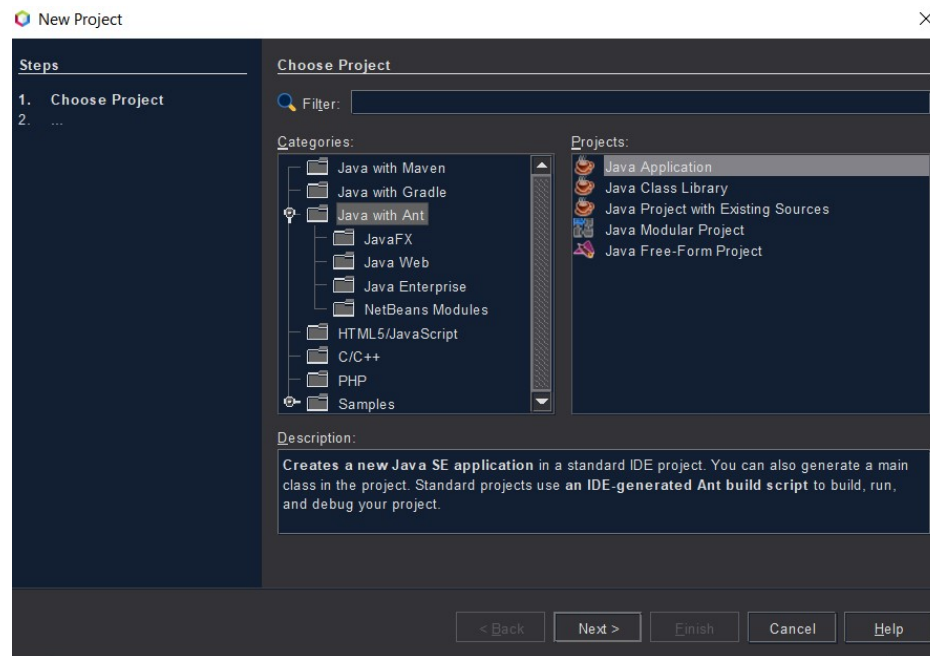
# Primera aplicación JavaFX

Una vez instalado el JavaFX SDK en Netbeans, vamos a crear un nuevo proyecto que haga uso del mismo: HolamundoFX. **No intentes crear un proyecto FX directamente.**

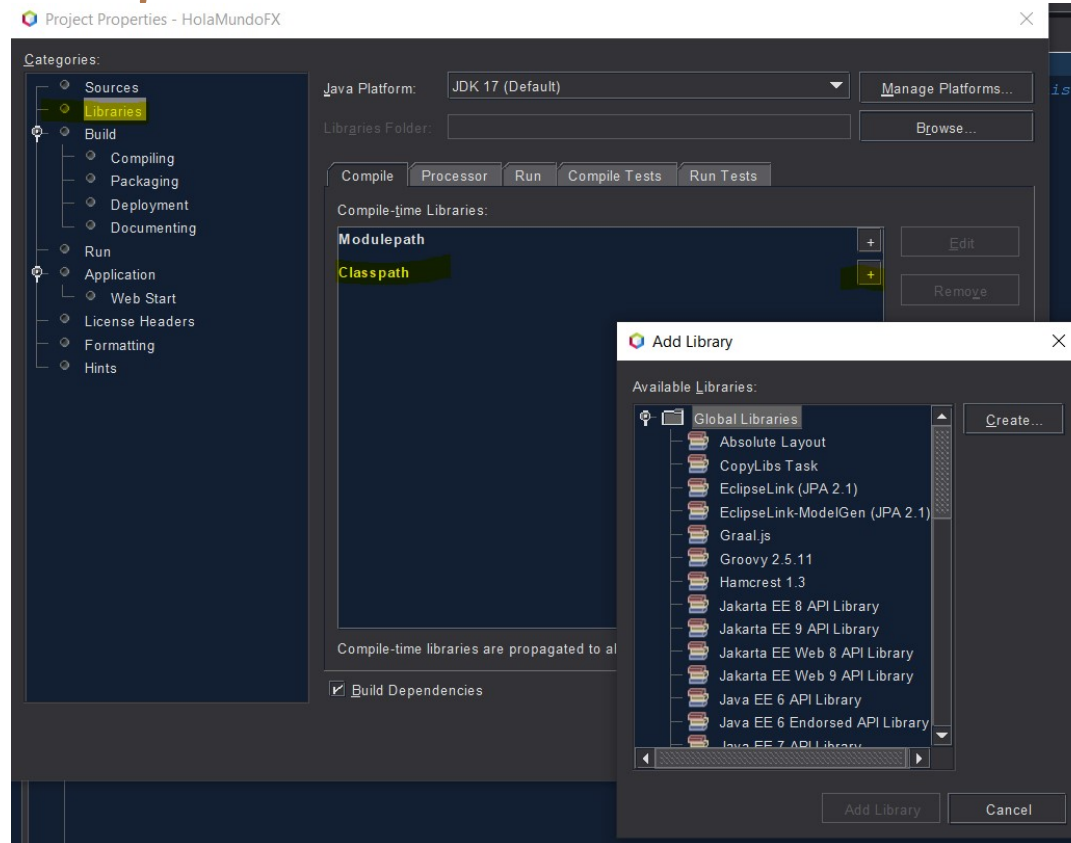


# Primera aplicación JavaFX

Creamos un proyecto como siempre:

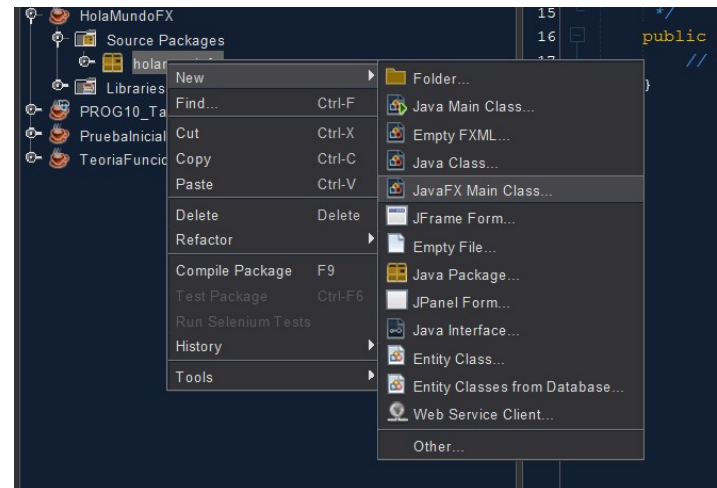


# Primera aplicación JavaFX



# Primera aplicación JavaFX

Nuestra aplicación ya tiene soporte para JavaFX. Vamos a crear una clase principal en el paquete por defecto que implementa la ventana principal de nuestra aplicación. Tan solo tendrás que darle un nombre. La ventana contendrá un botón que al pulsar mostrará "Hello World" en la consola. Observa la opción a elegir en la siguiente imagen:



# *Primera aplicación JavaFX*

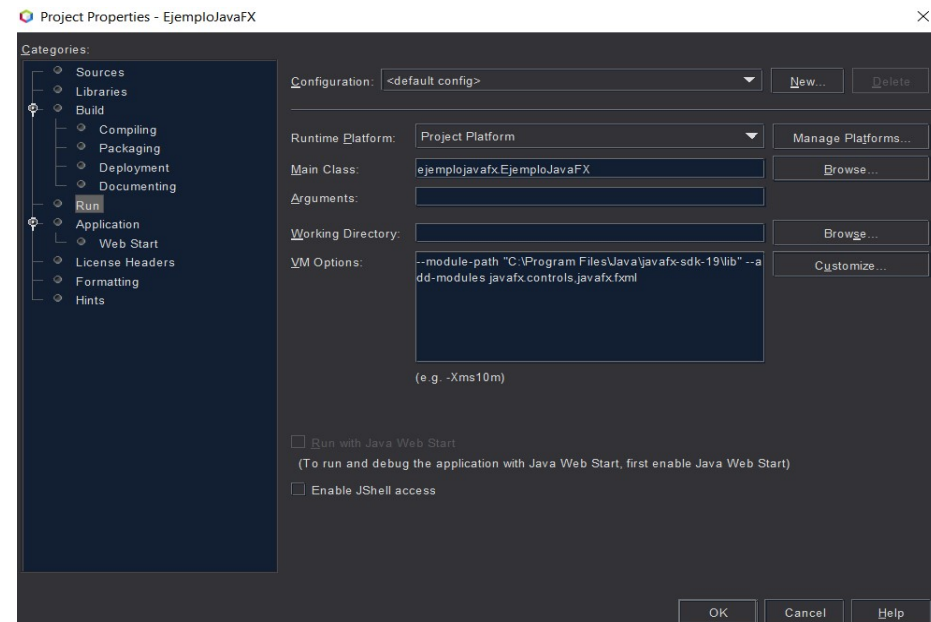
Si ejecutamos la aplicación como un aplicación tradicional obtendremos errores.

Debemos lanzar la aplicación parametrizando la jvm, indicándole las librerías de JavaFX que debe utilizar. ¿Porqué?. Si te fijas, la clase principal hereda de Application, por lo tanto necesita el soporte de JavaFx para lanzarla la ejecución.

# Primera aplicación JavaFX

Vamos al proyecto y a la opción Properties y añadimos en VM Options el path dónde tenemos el sdk de JavaFX:

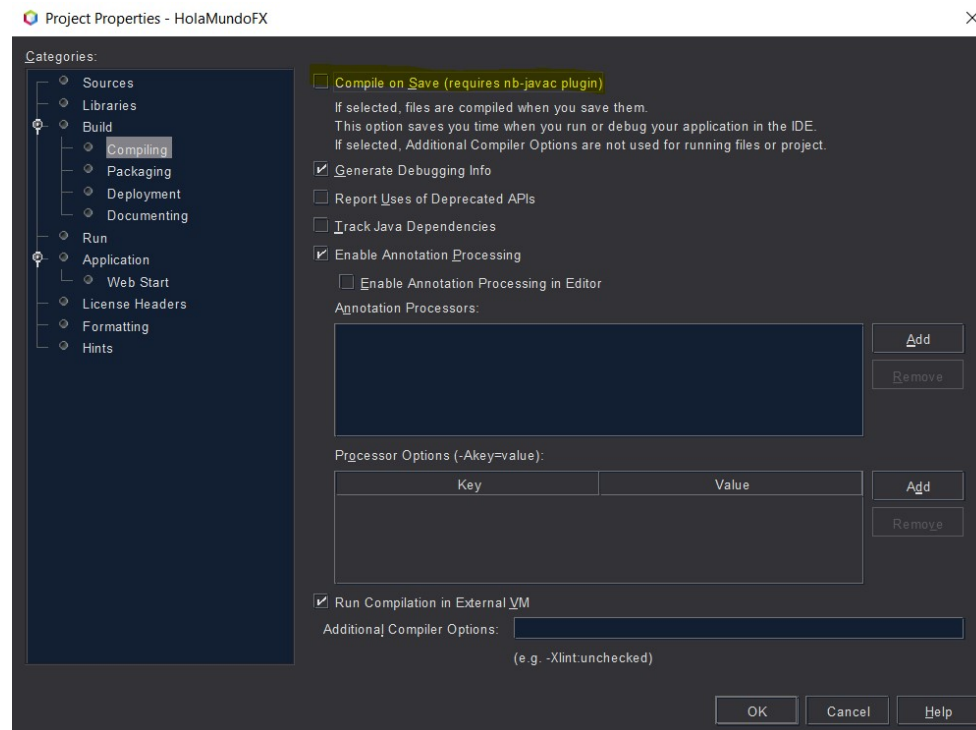
```
--module-path "\path\to\javafx-sdk-19\lib" --add-modules javafx.controls,javafx.fxml
```





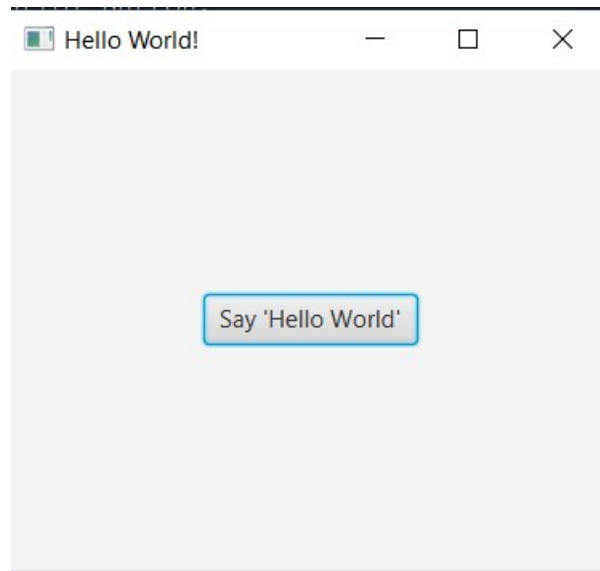
# Primera aplicación JavaFX

Deberemos también deseleccionar la opción indicada en la imagen en las propiedades del proyecto.



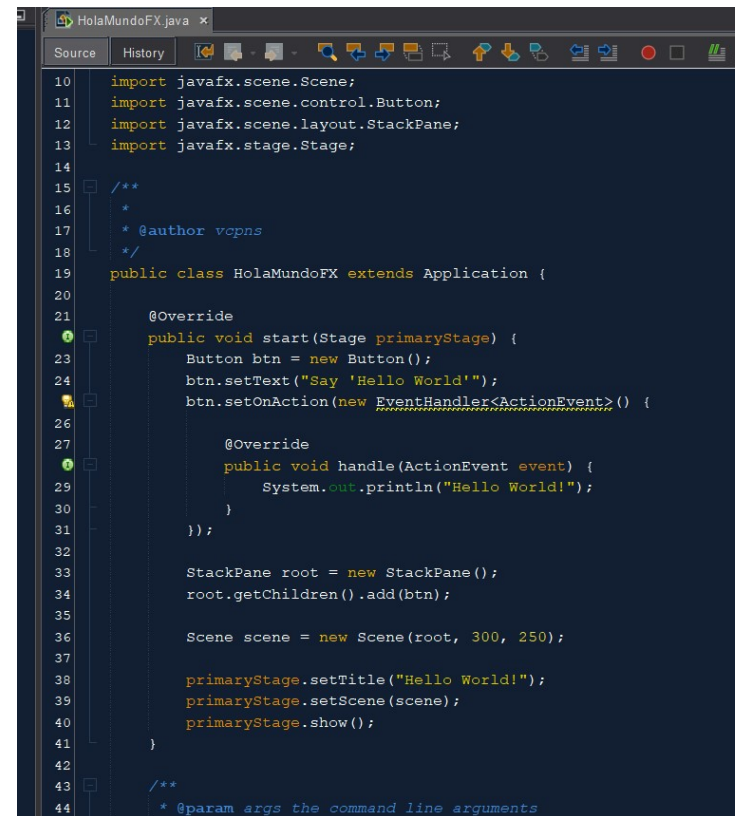
# *Primera aplicación JavaFX*

Ejecutamos el proyecto y ya nos sale el mensaje:



# Estructura de una aplicación JavaFX

Analicemos el fichero HolaMundoFX.java



```
10 import javafx.scene.Scene;
11 import javafx.scene.control.Button;
12 import javafx.scene.layout.StackPane;
13 import javafx.stage.Stage;
14
15 /**
16  *
17  * @author vcpns
18  */
19 public class HolaMundoFX extends Application {
20
21     @Override
22     public void start(Stage primaryStage) {
23         Button btn = new Button();
24         btn.setText("Say 'Hello World'");
25         btn.setOnAction(new EventHandler<ActionEvent>() {
26
27             @Override
28             public void handle(ActionEvent event) {
29                 System.out.println("Hello World!");
30             }
31         });
32
33         StackPane root = new StackPane();
34         root.getChildren().add(btn);
35
36         Scene scene = new Scene(root, 300, 250);
37
38         primaryStage.setTitle("Hello World!");
39         primaryStage.setScene(scene);
40         primaryStage.show();
41     }
42
43     /**
44      * @param args the command line arguments
```

## *Estructura de una aplicación JavaFX*

1. La clase creada hereda de `javafx.application.Application`. El método `start()` es el punto de entrada a cualquier aplicación JavaFX.
2. La aplicación JavaFX define un contenedor para la aplicación que contiene un stage (recibido por parámetro en el método `start`) y un scene.
  - La clase `Stage` representa en JavaFX un contenedor de máximo nivel.
  - La clase `Scene` representa un contenedor donde incluimos todos los demás elementos de la ventana (botones, etiquetas, gráficos, etc).
3. En las líneas del 38 a 40 se crea una escena con un determinado tamaño. Se añade un título al `Stage`, se le añade la escena creada y se hace visible.
4. En JavaFX, el contenido de una escena es representado como una jerarquía de nodos gráficos. En el ejemplo, el nodo raíz es un objeto de tipo `StackPane`, el cual es un nodo contenedor redimensionable. Es decir, que modificará su tamaño automáticamente si es redimensionado el `Stage`.
5. El nodo raíz contiene un nodo hijo, un botón con texto, con un manejador de eventos que imprime un mensaje cuando el botón es pulsado.
6. El método `main` no es necesario para lanzar aplicaciones JavaFX. Sin embargo, es aconsejable utilizarlo e invocar desde él el método `launch()`. Eso permite por ejemplo, que aplicaciones Swing (necesitan el método `main`), pueden embeber código Java FX.

# *Instalación del Scene Builder*

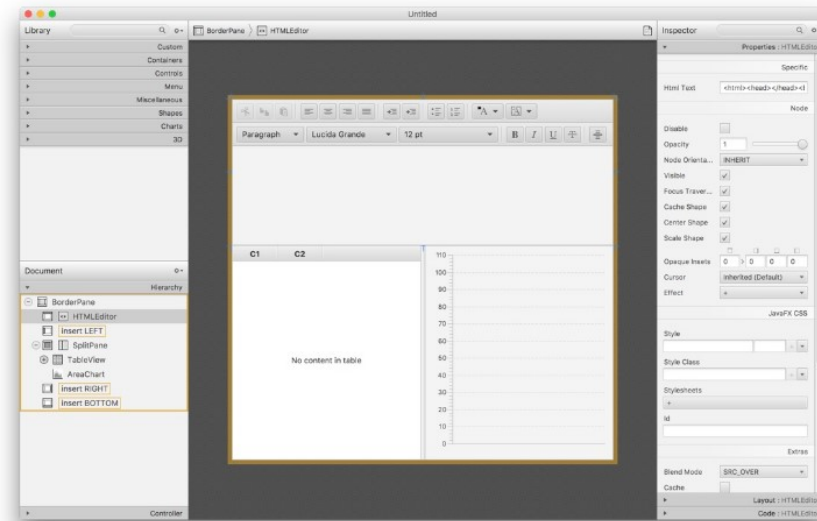
Scene Builder es una herramienta que trabaja con JavaFX y permite el diseño de interfaces gráficas de usuario de forma gráfica utilizando un potente entorno.

La instalación de Scene Builder y su integración en Netbeans es un proceso sencillo.

Descarga el fichero de instalación de Scene Builder teniendo en cuenta la versión del sistema operativo que estás utilizando.

[Enlace Descarga de Scene Builder](#)

## Scene Builder

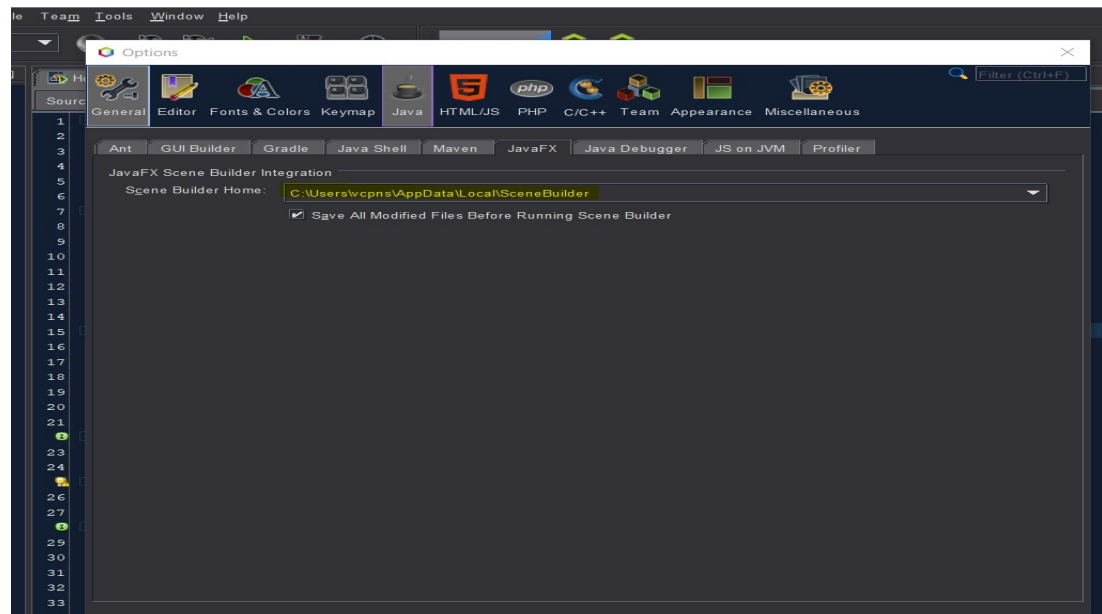


Drag & Drop,  
Rapid Application  
Development.

[Download Now](#)

# Integración de Scene Builder en Netbeans

Integrar Scene Builder en Netbeans es fácil. Una vez instalado, tan solo tendremos que decirle a Netbeans en qué ruta se encuentra instalado. Para ello, debemos acceder a las preferencias de Netbeans y configurar el parámetro que se observa en la imagen:



# Primera aplicación JavaFx con Scene Builder

El primer paso será crear un proyecto con soporte para JavaFX (nómbrale MiniCalculadoraFX)

- Crearemos una aplicación java tradicional.
- Añadimos la librería de JavaFX.
- Parametrizamos la jvm para poder lanzar a ejecución la aplicación.

El siguiente paso será crear, tal y como hicimos en el proyecto anterior, una clase principal JavaFX. Una vez creada, eliminamos el código del método start.

```
public class EjemploJavaFX extends Application {  
    public void start(Stage primaryStage) throws IOException {  
    }  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        launch(args);  
    }  
}
```

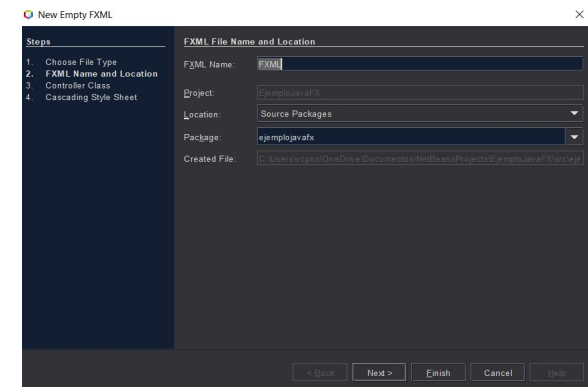
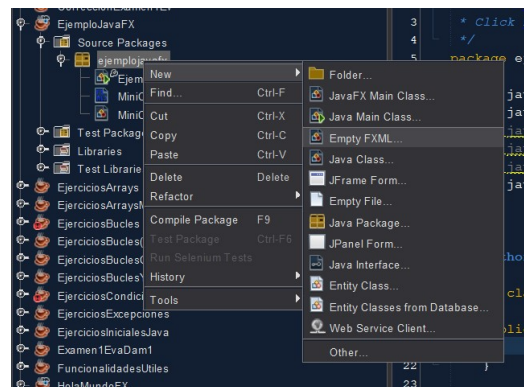


# Primera aplicación JavaFx con Scene Builder

El siguiente paso consistirá en crear dos ficheros que darán soporte a la interfaz de usuario de nuestra aplicación:

1. MiniCalcVentana.fxml: Fichero declarativo que contiene la definición de la interfaz de usuario: controles, posición, características, etc. Este fichero no se editará a mano sino a través del Scene Builder.
2. MiniCalcVentanaControlador.java: Fichero java que contendrá el código asociado a la ventana, es decir, toda la lógica de negocio. Por ejemplo, los manejadores de eventos de la ventana MiniCalcVentana estarán implementados en este controlador.

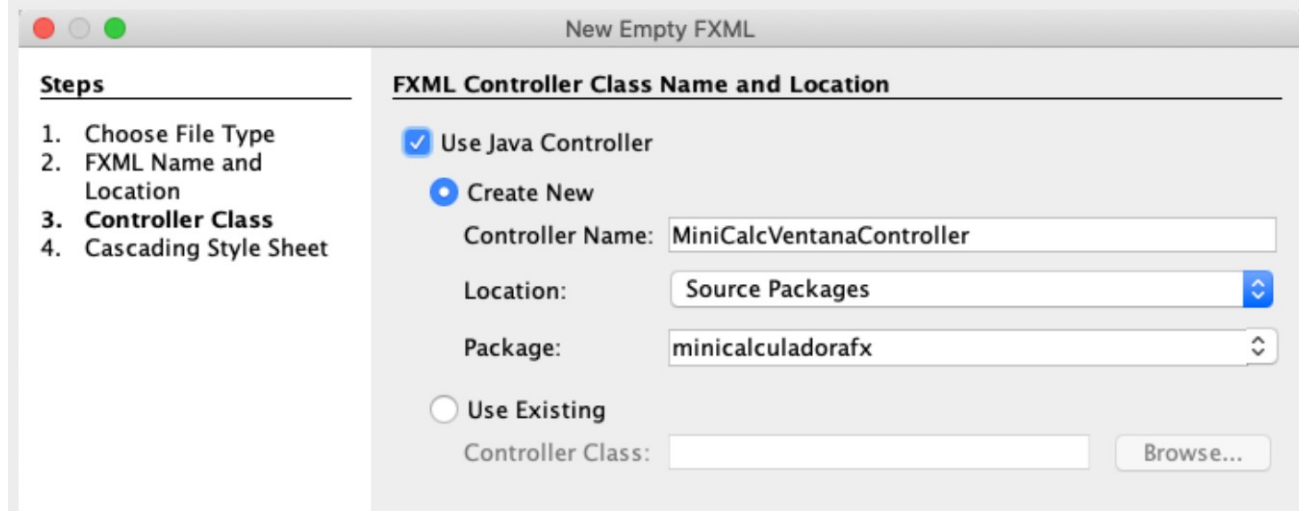
Para ello, en el paquete por defecto, **seleccionamos en Netbeans New - Empty FXML**. Aparecerá la siguiente ventana:



# Primera aplicación JavaFx con Scene Builder

Indicamos el nombre para nuestro fichero FXML: MiniCalcVentana. Pulsamos Next.

En el siguiente paso, indicamos a Netbeans que genere automáticamente el controlador asociado a la vista.



## Primera aplicación JavaFx con Scene Builder

Dejamos el nombre por defecto y pulsamos Next. En el siguiente paso podemos comprobar como incluso podemos asociar un fichero de estilos (CSS) a nuestra vista. Esto no es posible hacerlo en Swing pero si en JavaFX. En nuestro caso no lo hacemos. Pulsamos en finalizar y ya tenemos los dos ficheros creados en nuestro paquete.

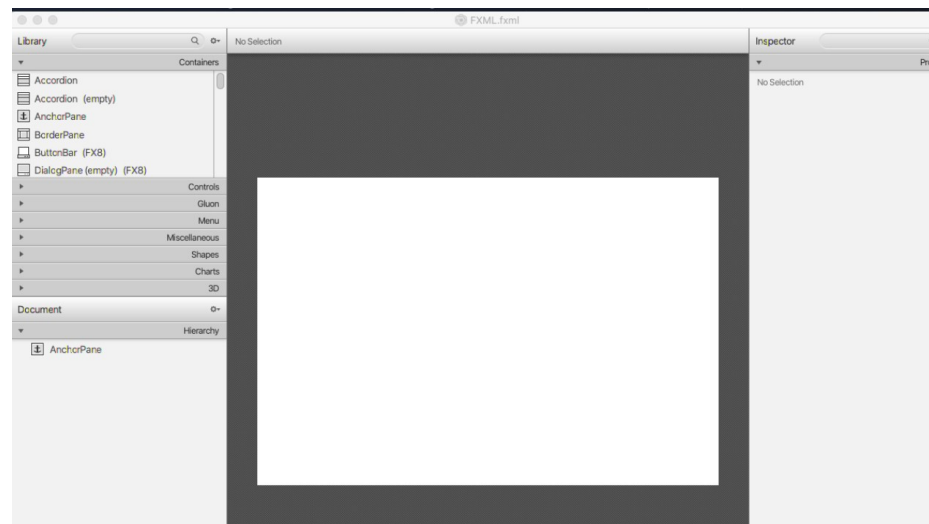
Antes de diseñar nuestra interfaz, vamos a incluir el código java necesario en la clase principal de nuestra aplicación para que cuando lancemos a ejecutar, cargue la vista creada MiniCalcVentana. Observa el código del método start:

```
public void start(Stage primaryStage) throws IOException {  
  
    Parent root= FXMLLoader.load(getClass().getResource("MiniCalculadoraInt  
    Scene scene= new Scene(root);  
    primaryStage.setTitle("Mini Calculadora JavaFX");  
    primaryStage.setScene(scene);  
    primaryStage.show();  
}
```

# Primera aplicación JavaFx con Scene Builder

Parte de este código ya nos es familiar, pues se crea la escena y se asocia al Stage. En la primera línea del código, a través del FXMLLoader, indicamos que cargue nuestro fichero fxml. En nuestra "primera aplicación JavaFX" creamos la interfaz a través de código Java y en este caso lo haremos a través de nuestro fichero fxml, que editamos con Scene Builder.

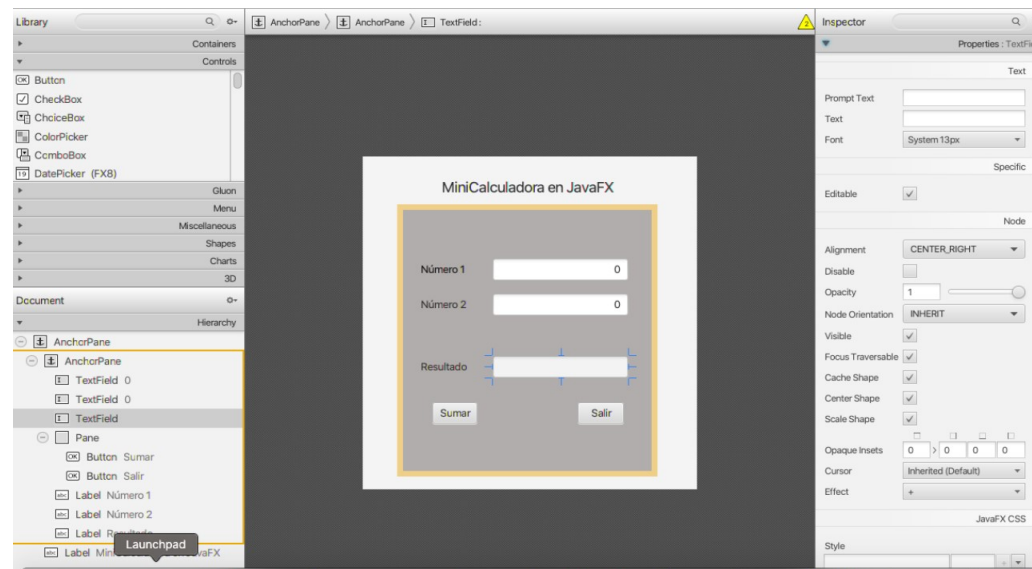
Llega el momento de diseñar la interfaz. Para ello, pulsamos sobre el fichero fxml con el botón derecho y pulsamos Open: si la instalación del Scene Builder se realizó correctamente, se abrirá y cargará el fichero MiniCalcVentana.fxml, que estará vacío.



# Primera aplicación JavaFx con Scene Builder

La interfaz de Scene Builder es parecida al editor de Swing de Netbeans.

- El panel superior izquierdo contiene todo tipo de controles y layouts.
- Justo debajo se encuentra el navegador, donde se irán colocando de forma jerárquica todos los elementos que añadamos a nuestra interfaz.
- En la parte central tendremos el editor: podemos añadir elementos arrastrando con el ratón.
- El panel de la parte derecha contiene las propiedades de un elemento seleccionado.



## Primera aplicación JavaFx con Scene Builder

Una vez que tenemos creada nuestra interfaz, es hora de darle funcionalidad. Recuerda que todo el código Java asociado se incluirá en el controlador MiniCalcVentanaController.java. Observa el código que vamos a añadir a nuestro controlador.

```
public class MiniCalculadoraInterfazController implements Initializable {

    @FXML
    private TextField tfNumero1;

    @FXML
    private TextField tfNumero2;

    @FXML
    private TextField tfResultado;

    @FXML
    private void buttonSumarHandler (ActionEvent event){
        //Variables para almacenar los operandos a sumar.
        int num1, num2, resul;
```

## *Primera aplicación JavaFx con Scene Builder*

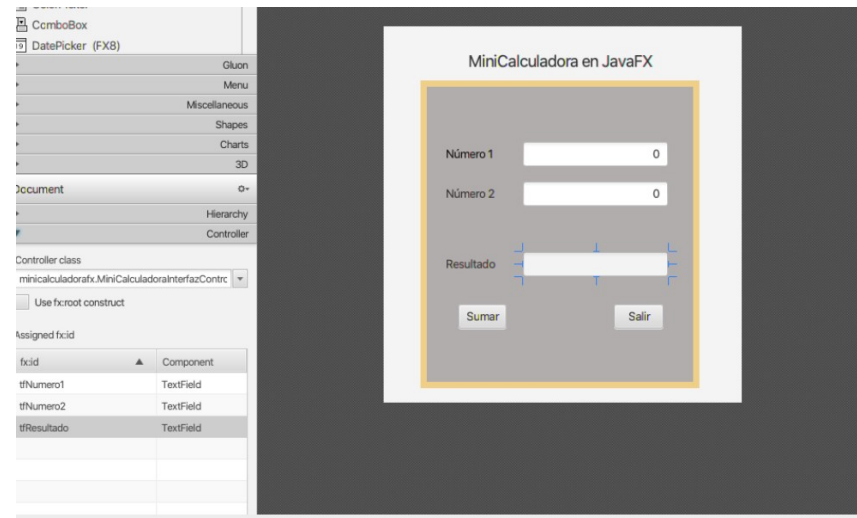
1. Declaramos una referencia a cada uno de los elementos de la interfaz a los que tenemos que acceder desde el controlador. Para ello utilizamos la etiqueta `@FXML` (incluir el import si hay errores de compilación). Evidentemente, hemos de asignar un nombre a cada referencia. En nuestro caso, necesitamos acceder a los tres campos de texto: a dos de ellos para recuperar los valores de los números a sumar (`tfNumero1` y `tfNumero2`) y el tercero para mostrar el resultado (`tfResultado`). Observa que esas referencias son del tipo de objeto que utilizamos al incluir el elemento en la interfaz, en nuestro caso, `TextField`. Líneas 3-10.
2. A continuación, declaramos un método que va a actuar como manejador de eventos del botón Sumar. Se encargará de realizar la suma y mostrar el resultado en el campo de texto apropiado. El nombre del método lo decidimos nosotros.
3. Creamos un tercer método que se encargará de manejar eventos producidos por el botón Salir.

# Primera aplicación JavaFx con Scene Builder

¿Cómo asociamos el controlador al fichero fxml? Esta asociación consiste en:

1. Asociar el fichero MiniCalcVentanaController.java al fichero MiniCalcVentana.fxml.
2. Asociar cada campo de texto en el controlador a su homólogo en el fichero fxml.
3. Asociar el manejador creado a cada uno de los botones en el fichero fxml.

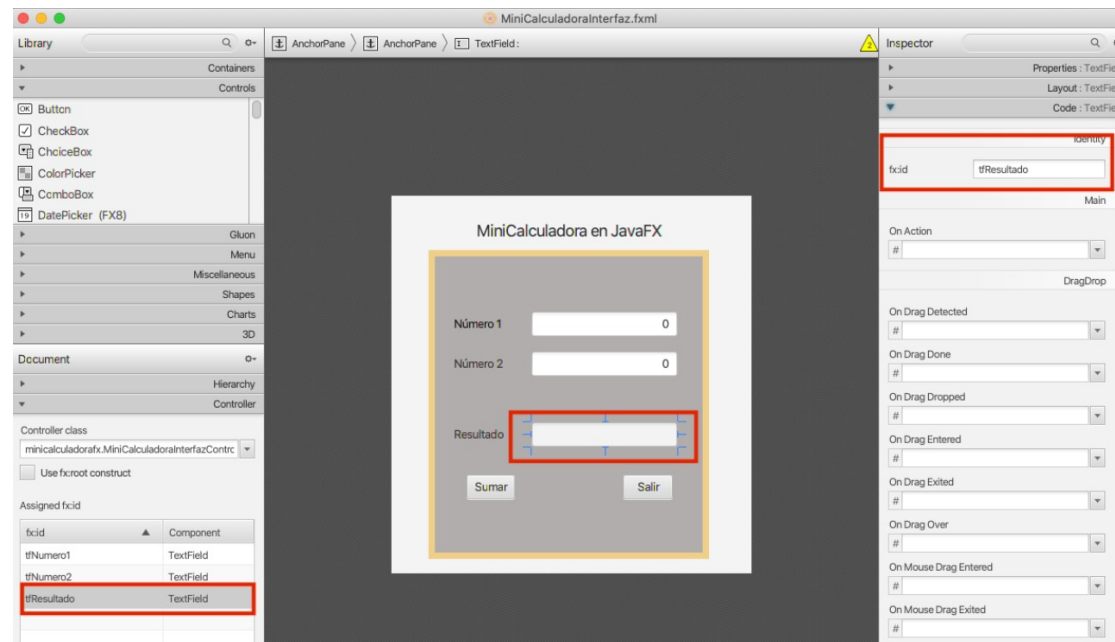
La asociación de ficheros ya está hecha, pues al generar el fichero fxml indicamos a Netbeans que generara su fichero controlador. De todas formas, se podría modificar esa asociación en Scene Builder.





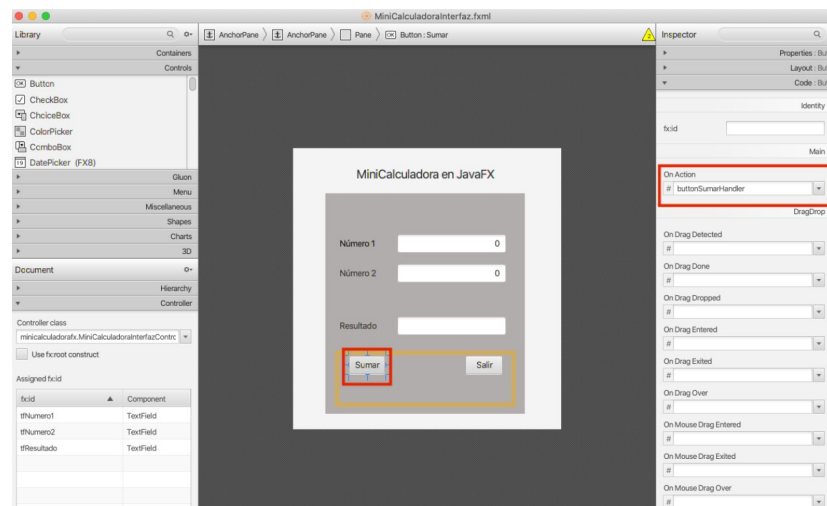
# Primera aplicación JavaFx con Scene Builder

Para asociar los campos de texto utilizamos el panel Code situado en la parte derecha. Observa como en la imagen, para el elemento seleccionado (campo de texto con el resultado), asociamos el id tfResultado que previamente hemos definido en el controlador: esta asociación permite que desde el controlador se pueda recuperar el valor de este campo de texto. Habría que lo mismo para los otros dos campos de texto.



# Primera aplicación JavaFx con Scene Builder

Solo nos queda asociar los manejadores de eventos a los botones. Utilizamos el mismo panel. Obsérvalo en la imagen:



Al evento OnAction del botón, que se lanzará cuando el botón sea pulsado, le asociamos el manejador buttonSumarHandler, que previamente hemos definido en el controlador.

Puedes asociar el manejador apropiado al botón Salir.