# CST8221 – Java Application Programming
# Hybrid Activity #9
# Visual GUI Builders – The JavaFX Scene Builder

## *Terminology*

Visual GUI builders are specialized tools that let you design the visual appearance of your application and then they generate much (often all) of the GUI code for you. NetBeans IDE offers such a tool for building Swing GUI applications (code name Matisse (a famous French painter)). Eclipse Java IDE offers plug-ins such as Eclipse Visual Editor and WindowBuilder. There is another tool closely associated with NetBeans for building JavaFX GUI. The name of this tool is Scene Builder.

## *The Nature of Things*

In order to use the visual builders you should first learn how to build a user interface manually. Once you understand the basic building blocks of a GUI application, you can use visual builders for Rapid Application Development. At this point of your learning you should have a pretty good understanding of how the Java GUI works. Which means that you a ready to try a visual GUI builder.

JavaFX Scene Builder does not have dependency on any particular IDE. However, since Scene Builder is very closely integrated with NetBeans IDE, you use going to use NetBeans IDE to complete this Hybrid Activity. If Eclipse or IntelliJ IDEA is your preferred IDE, see Using JavaFX Scene Builder with Java IDEs for more information. You can also use Scene Builder as a standalone tool to create your GUI layout and edit the resulting FXML file using a text editor of your choice

This Hybrid Activity presents the step-by-step creation of a simple GUI application using the JavaFX Scene Builder tool. It shows you how to quickly build the user interface (GUI) for a JavaFX application, connect it to the Java source code, and handle the interaction between the data and the user interface. JavaFX Scene Builder generates FXML (FX Markup Language) for defining and arranging JavaFX GUI controls without writing any Java code. The FXML code is separate from the program logic that is defined in the Java source code. Tis separation of the interface (GUI) from the implementation (Java code) make it easier to develop application. You do not need to know FXML or XML to complete this Hybrid Activity.

## Task 1 – Installing NetBeans IDE

If have not already installed NetBeans IDE you have to install it before you can proceed with Task 2. Go to the link below

https://netbeans.org/downloads/

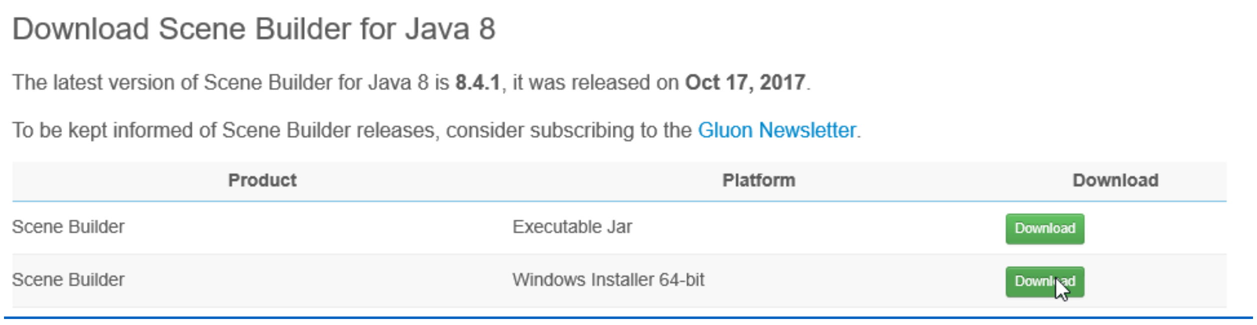
for the latest NetBeans IDE 8 download and installation information. Download the **Java SE** bundle for your operating system platform. The bundle does not include the Java JDK. It will install using your currently installed JDK. Any of the JDK 1.8_XX will work. You will need an Internet connection during the installation. Once NetBeans is installed, launch it and proceed with Task 2.

## Task 2 – Installing Scene Builder

Go to the link below

http://gluonhq.com/products/scene-builder/#download

and download Scene Builder for Java 8 (not 9).
Scene Builder   Windows Installer 64-bit    SceneBuilder-8.4.1.exe
See the screen capture below.



Download Scene Builder for Java 8

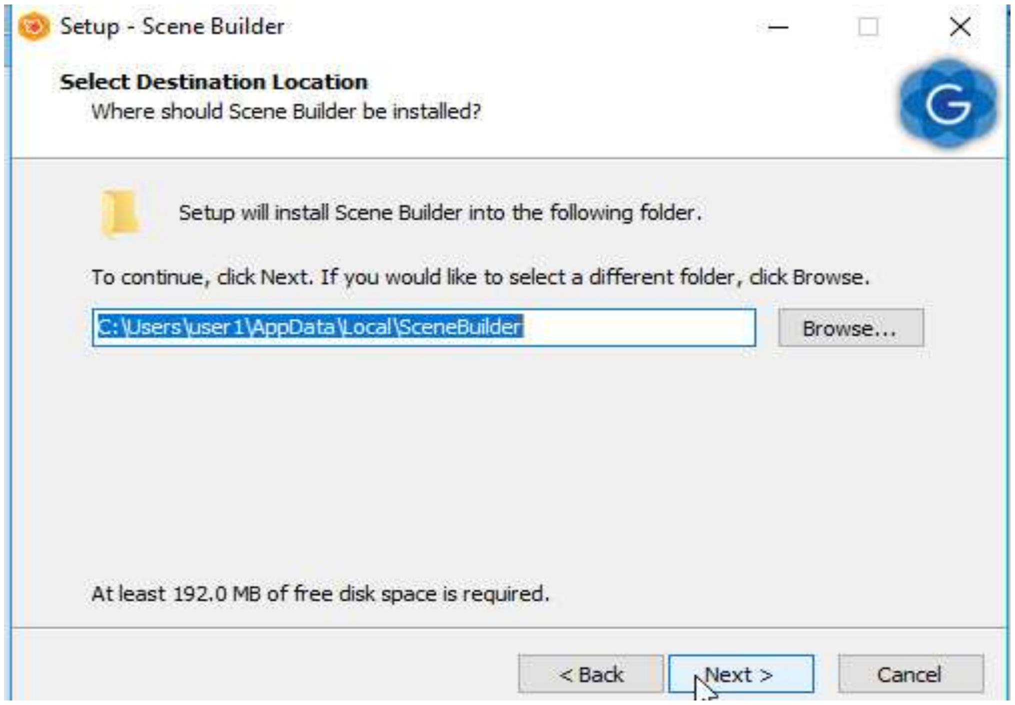
The latest version of Scene Builder for Java 8 is **8.4.1**, it was released on **Oct 17, 2017**.

To be kept informed of Scene Builder releases, consider subscribing to the Gluon Newsletter.

| Product | Platform | Download |
|---|---|---|
| Scene Builder | Executable Jar | Download |
| Scene Builder | Windows Installer 64-bit | Download |

You can also download the executable jar which is a stand-alone application not integrated with any IDE.

Run the installer. When the Select Destination Location dialog appears **remember** the destination location and click Next.



Setup - Scene Builder

Select Destination Location
Where should Scene Builder be installed?

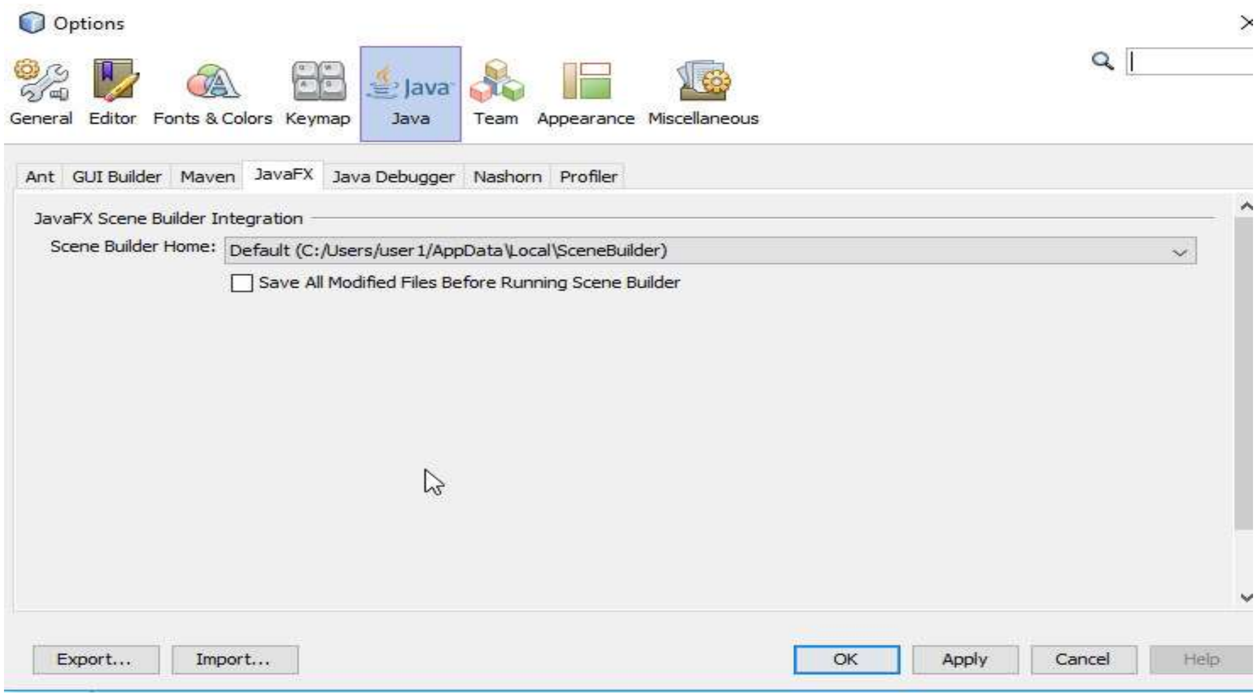
Setup will install Scene Builder into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

C:\Users\user1\AppData\Local\SceneBuilder

At least 192.0 MB of free disk space is required.

After the installation Scene Builder will launch. Close the application.
If the NetBeans IDE has not been launched, launch it NetBeans. From the main menu click Tools and select Options. Select Java, then select JavaFX. See the screen capture below.

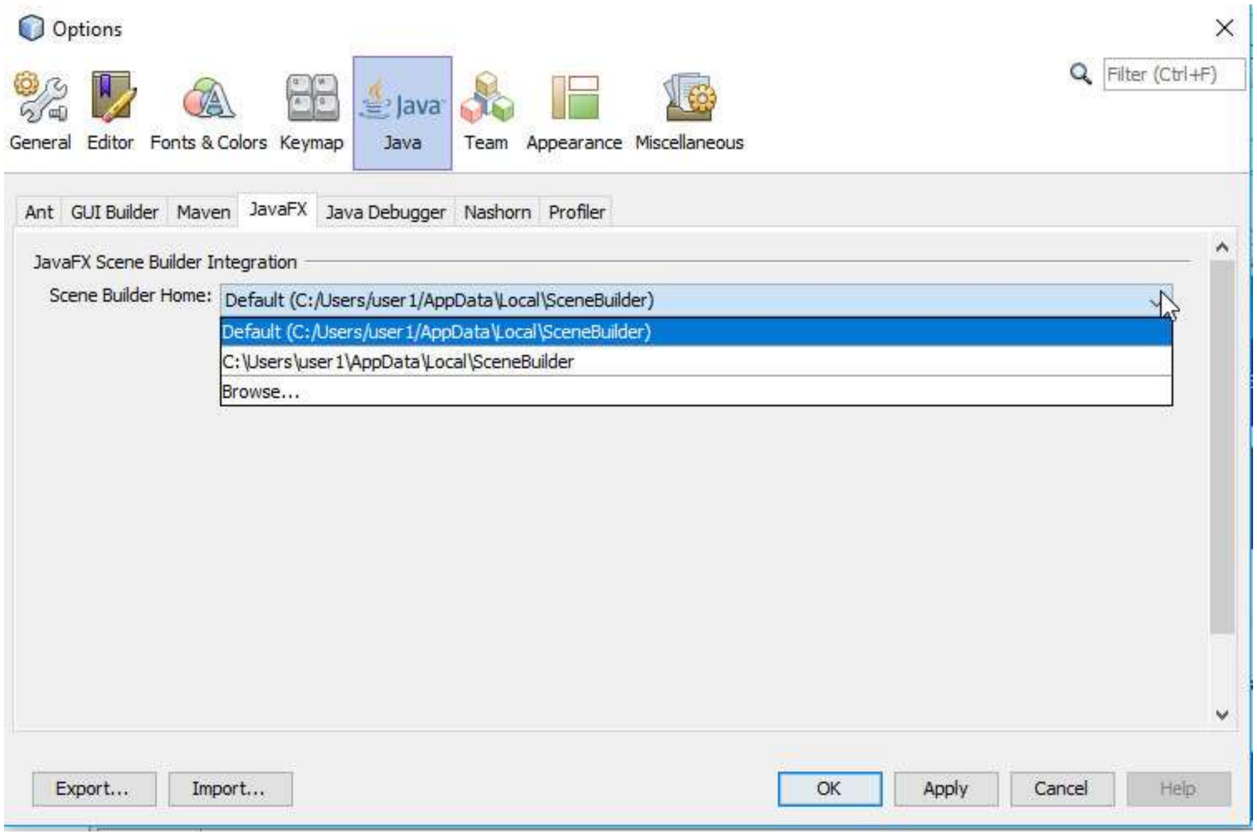


JavaFX Scene Builder Integration

Scene Builder Home: Default (C:/Users/user1/AppData\Local\SceneBuilder)

☐ Save All Modified Files Before Running Scene Builder

Check if the Scene Builder Home location is the same as the one you have previously installed Scene Builder. If it is the same click OK. If it is not the same, click on the down arrow and select *Browse….* Find the location, Click Apply then OK. You may also decide to check the *Save All Modified Files*… checkbox.



JavaFX Scene Builder Integration

Scene Builder Home: Default (C:/Users/user1/AppData\Local\SceneBuilder)

Default (C:/Users/user1/AppData\Local\SceneBuilder)
C:\Users\user1\AppData\Local\SceneBuilder
Browse...

Now you are ready to proceed with Task 3.

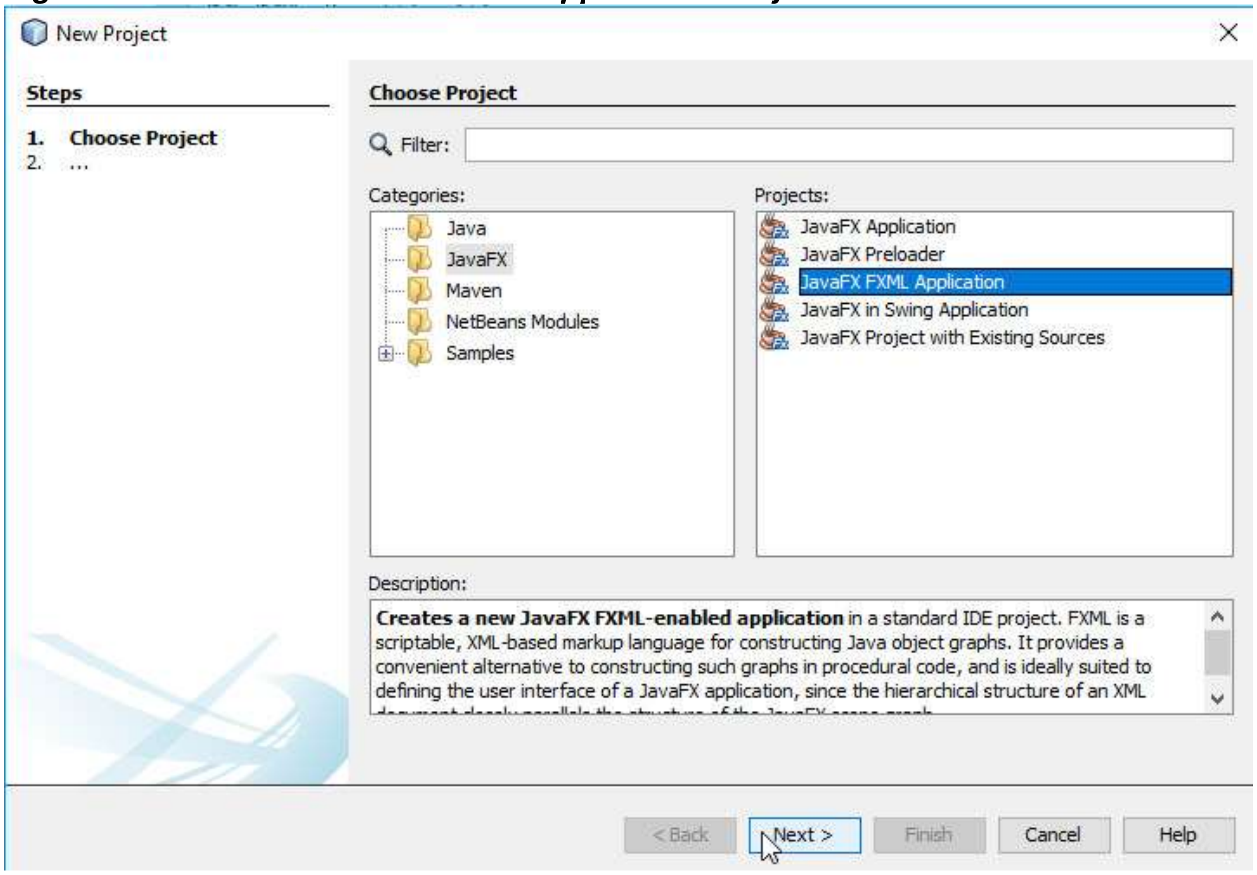
## Task 3 – Creating JavaFX Application with NetBeans IDE and Scene Builder

In Task 3 you are going to create a simple JavaFX Metric Converter application. Following the steps you will create a new JavaFX FXML project, start Scene Builder from within the IDE, and run Scene Builder sample applications. The integration of JavaFX Scene Builder with NetBeans IDE provides optimal development workflow.

### Step 1 - Creating a New JavaFX FXML Project

In NetBeans IDE, you use the **New** wizard to create a new JavaFX FXML Application, which is a JavaFX project that is based on an FXML layout. After the project is created, you can edit the FXML file using Scene Builder.
From the NetBeans IDE Main menu, select **File**, and then choose **New Project**. In the **New Project** dialog box, choose the **JavaFX** category and J**avaFX FXML Application** project, as shown in *Figure 1*. Click **Next**.

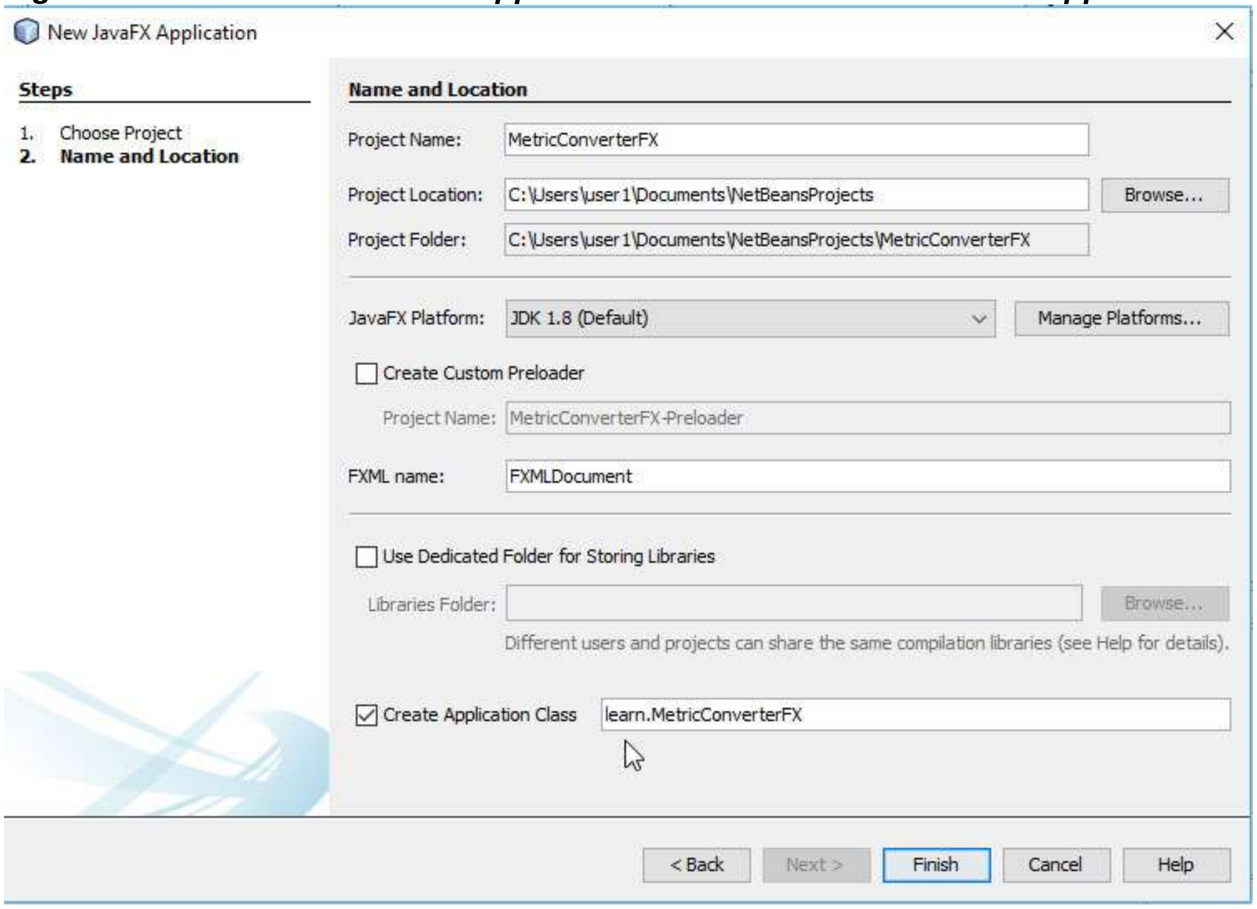
***Figure 1 Create New JavaFX FXML Application Projec****t*



**Creates a new JavaFX FXML-enabled application** in a standard IDE project. FXML is a scriptable, XML-based markup language for constructing Java object graphs. It provides a convenient alternative to constructing such graphs in procedural code, and is ideally suited to defining the user interface of a JavaFX application, since the hierarchical structure of an XML

In the **New JavaFX Application** dialog box, enter the values you would like to use for the project name, project location, and FXML file name. Click **Finish** to complete the project creation. *Figure 2* gives the values you have to enter or accept as default. The name of the project must be *MetricConverterFX* and the name of the **Application Class** must be *learn.MetricConverterFX*. If you want, can change the project default location and folder.

***Figure 2 Set the Name and the Application Class of the New JavaFX Application***


New JavaFX Application

Steps
1. Choose Project
2. **Name and Location**

**Name and Location**

Project Name: MetricConverterFX

Project Location: C:\Users\user1\Documents\NetBeansProjects

Project Folder: C:\Users\user1\Documents\NetBeansProjects\MetricConverterFX

JavaFX Platform: JDK 1.8 (Default)

☐ Create Custom Preloader

Project Name: MetricConverterFX-Preloader

FXML name: FXMLDocument

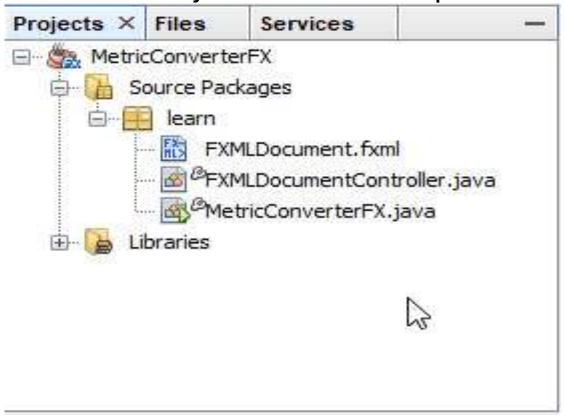
☐ Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

☑ Create Application Class learn.MetricConverterFX

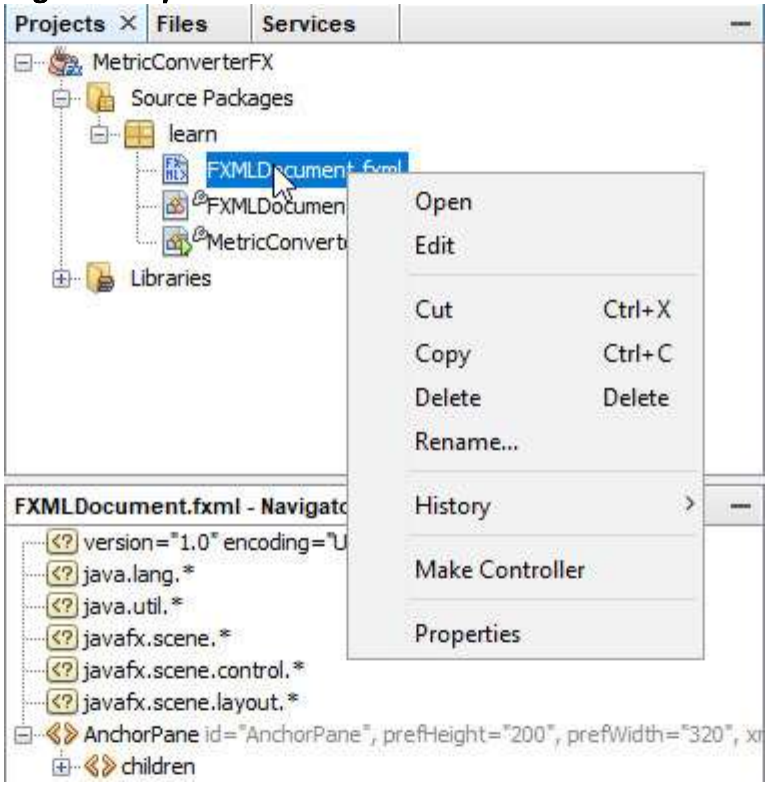
The IDE's Projects window is updated with the new JavaFX application you just created.


```
MetricConverterFX
    Source Packages
        learn
            FXMLDocument.fxml
            FXMLDocumentController.java
            MetricConverterFX.java
    Libraries
```

## Editing an FXML File Using Scene Builder

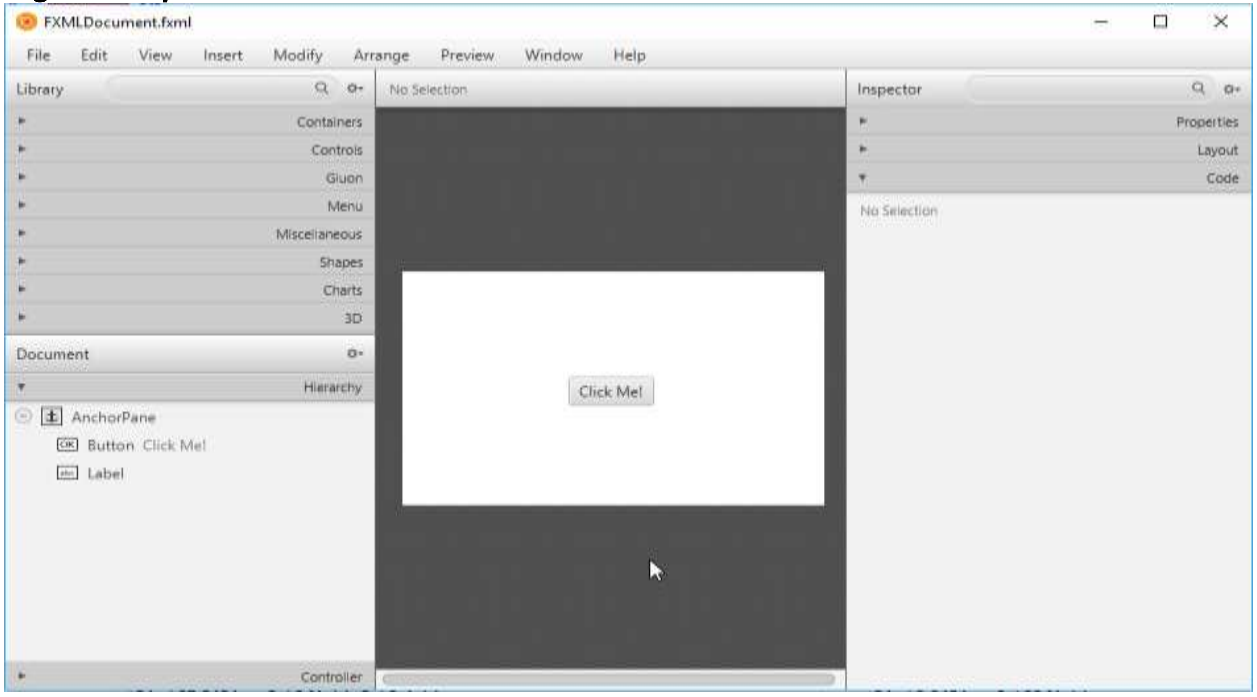
You can modify your FXML file using the NetBeans IDE editor or by opening it using Scene Builder. If you installed Scene Builder in a location other than the default, you need to first ensure that NetBeans IDE is configured with that Scene Builder installation by following Task 2 steps. Right-click on the *FXMLDocument.fxml* and a context menu will appear. As shown in *Figure 3*, you can either choose **Open** to edit the FXML file with the Scene Builder tool or choose **Edit** to edit the FXML file with the NetBeans FXML editor.

***Figure 3 Open FXML File from NetBeans IDE***



When you right-click the node for the FXML file and choose **Open** (or double click on the *FXMLDocument.fxml*) , NetBeans IDE launches the latest installed Scene Builder on your system, and the Scene Builder window appears, as shown in *Figure 4.* Changes you make to your FXML file using Scene Builder will be reflected in the file when you save it.

***Figure 4 Open FXML File in Scene Builder***



By default, the main window of JavaFX Scene Builder includes the following sections:

- **Main Menu Bar**: Provides access to the menu of commands available in JavaFX Scene Builder.

- In the middle is the **Content (GUI) Panel**: This is the scene container for the GUI elements that make up your FXML layout. In the figure above it contains the “Click Me!” Button and invisible Label. By default, a new empty FXML file is opened in JavaFX Scene Builder.

- On the left is the **Library Panel**: It lists all available JavaFX GUI elements or controls, including custom controls, which you can use to build your FXML layout. You can select the GUI elements from this panel and add (drag) them to the Content panel or the Hierarchy panel. You can also use the ***Insert*** option of the **Main Menu** to insert the GUI elements or controls.

- Under the **Library Panel** is the **Document Panel**: it contains the **Hierarchy** and **Controller** sections. The **Hierarchy** section displays a tree view representation of the FXML layout that you are building in the **Content** panel. Elements that are not visible in the **Content** panel (like the Label above) can be placed into focus by selecting it in the **Hierarchy** panel. The Controller section enables you to manage the controller source information and gives information about assigned ***fx:id*** values. The **Document** panel can be resized vertically (up and down).  Sometimes you need to resize it to reveal the hidden library elements.

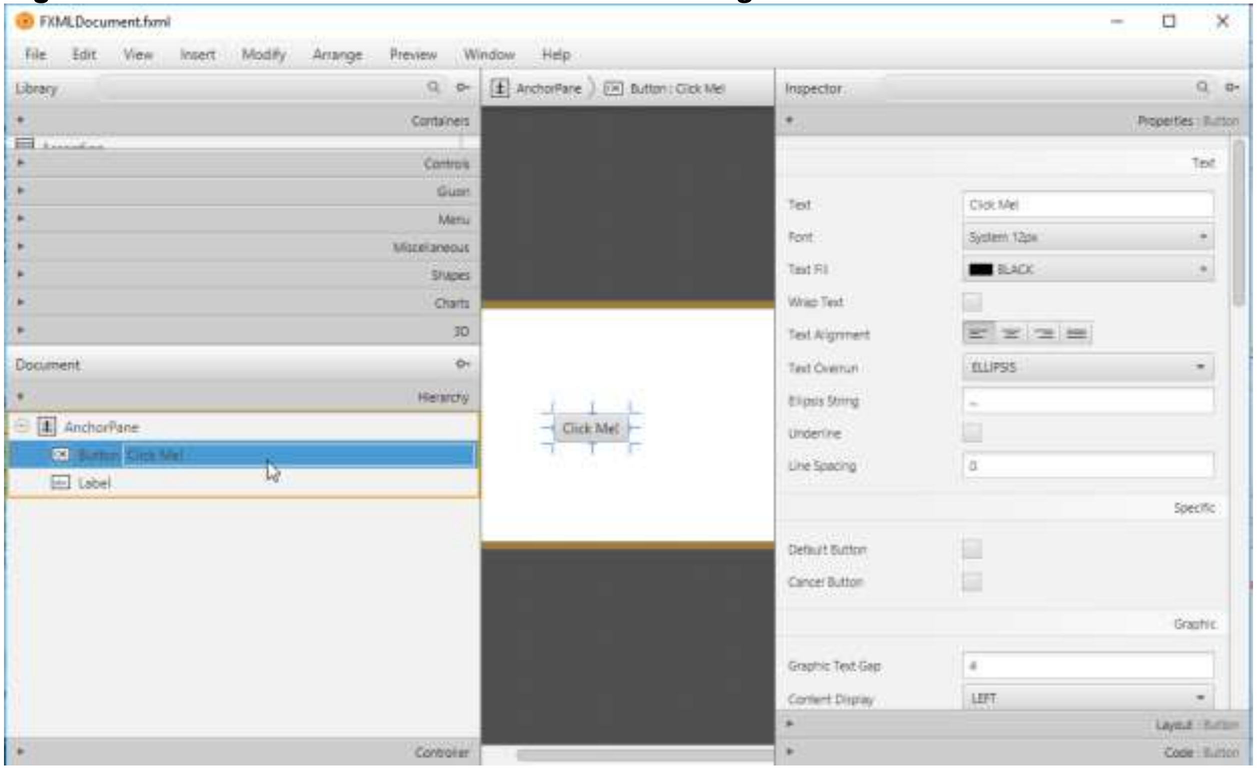
- On the right is the **Inspector Panel**: it contains the **Properties**, **Layout**, and **Code** sections. The **Properties** and **Layout** sections help you manage the properties of the currently selected GUI element in the **Content** panel or in the **Hierarchy** panel. The **Code** section enables you to manage the element ***fx:id*** and the event handling actions to use for the selected GUI element. The Inspector panel also contains a Search text field that enables you to isolate specific properties that you want to modify.

Now that you are familiar with the integration between the Scene Builder tool and NetBeans IDE, you can proceed with the creation of the *MetricConverterFX* application.

**Step 2 – Building the GUI of the MatricConverterFX Aplication**

Select Button from the **Hierarchy** panel or click on the on the "Click Me!" button in the **Content (GUI)** panel.

***Figure 5 Select the "Click Me!" Button and Change the Button Text***

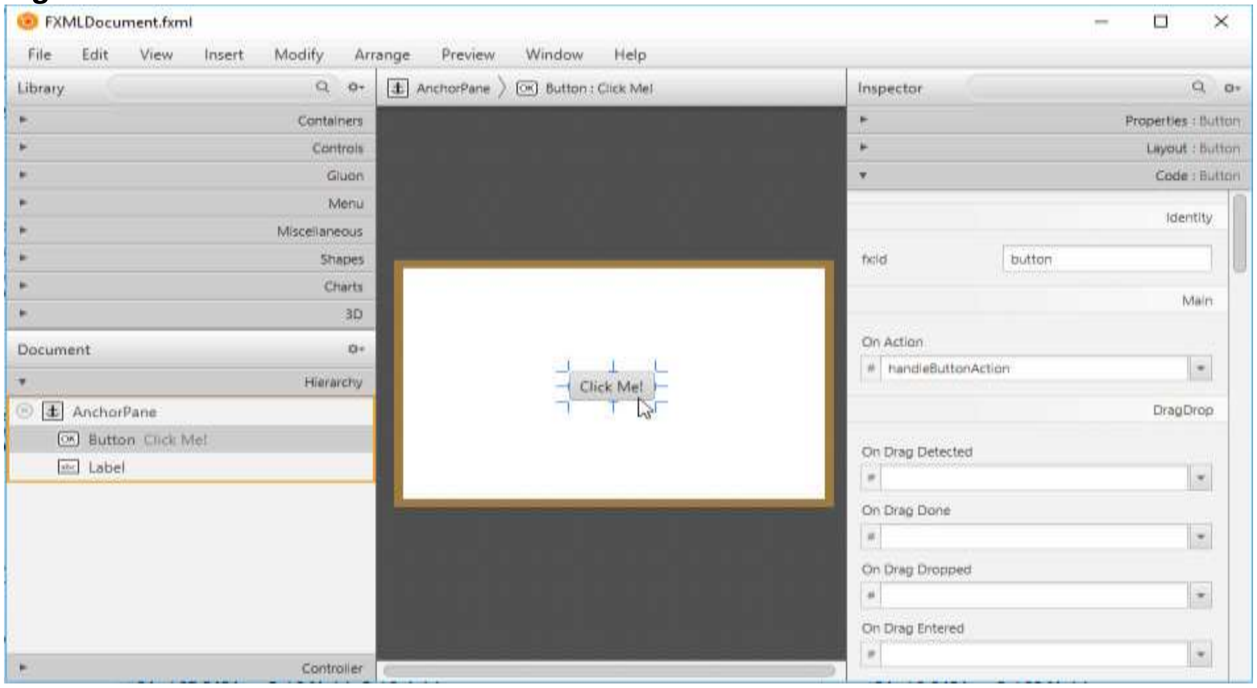


Text: Click Me!
Font: System 12px
Text Fill: BLACK
Wrap Text
Text Alignment
Text Overrun: ELLIPSIS
Ellipsis String: ...
Underline
Line Spacing: 0
Default Button
Cancel Button
Graphic Text Gap: 4
Content Display: LEFT

Use the **Properties** section of the **Inspector** panel to change the button *Text* property from *Click Me!* to *Convert*. Press Enter.

Open the **Code** section of the **Inspector** as shown in *Figure 6.* You can see that *button* is assigned as to ***fx:id***. You can also see that an event handler *handleButtonAction* is assigned to handle the Action event generated by the button.

***Figure 6 Code Section Entries for the Button***

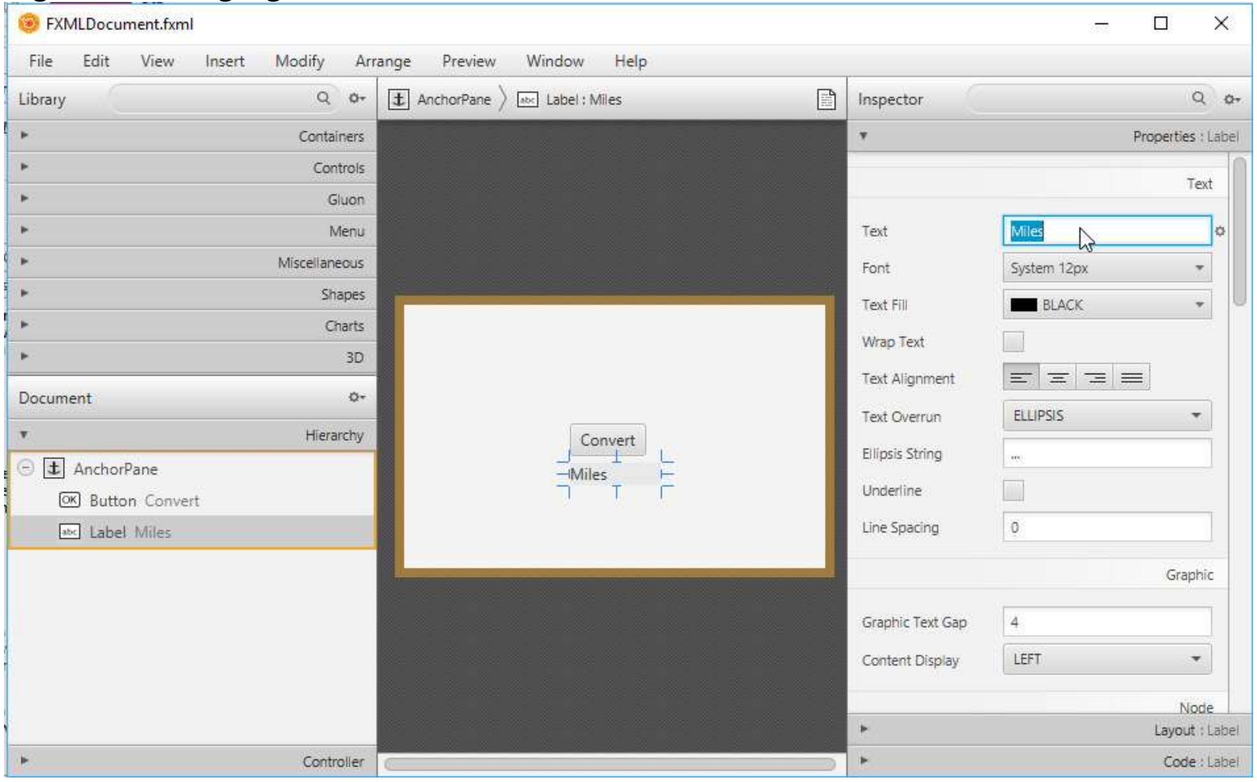


fx:id: button

On Action: # handleButtonAction

Select Label and type *Miles* in the *Text* property text field as shown in *Figure 7*. Press Enter.

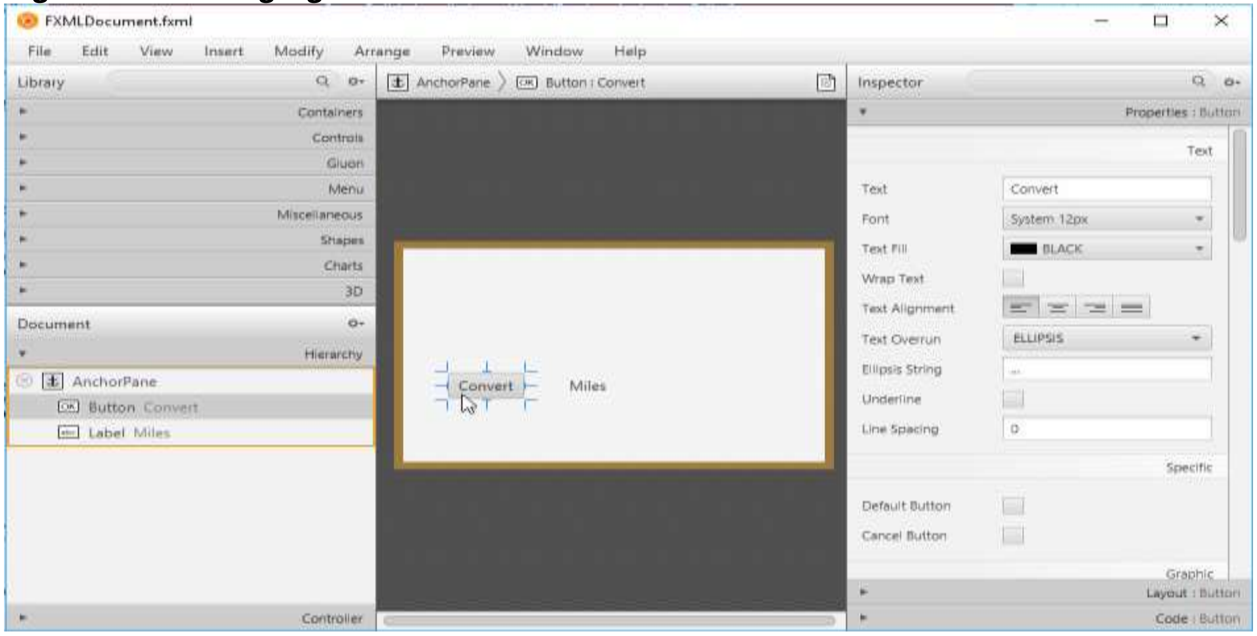
***Figure 7 Changing the Label Text***



Text: Miles
Font: System 12px
Text Fill: BLACK
Wrap Text
Text Alignment
Text Overrun: ELLIPSIS
Ellipsis String: ...
Underline
Line Spacing: 0
Graphic Text Gap: 4
Content Display: LEFT

Now clicking on the Button dragging it around rearrange the GUI elements (controls) so that it looks like in *Figure 8*.

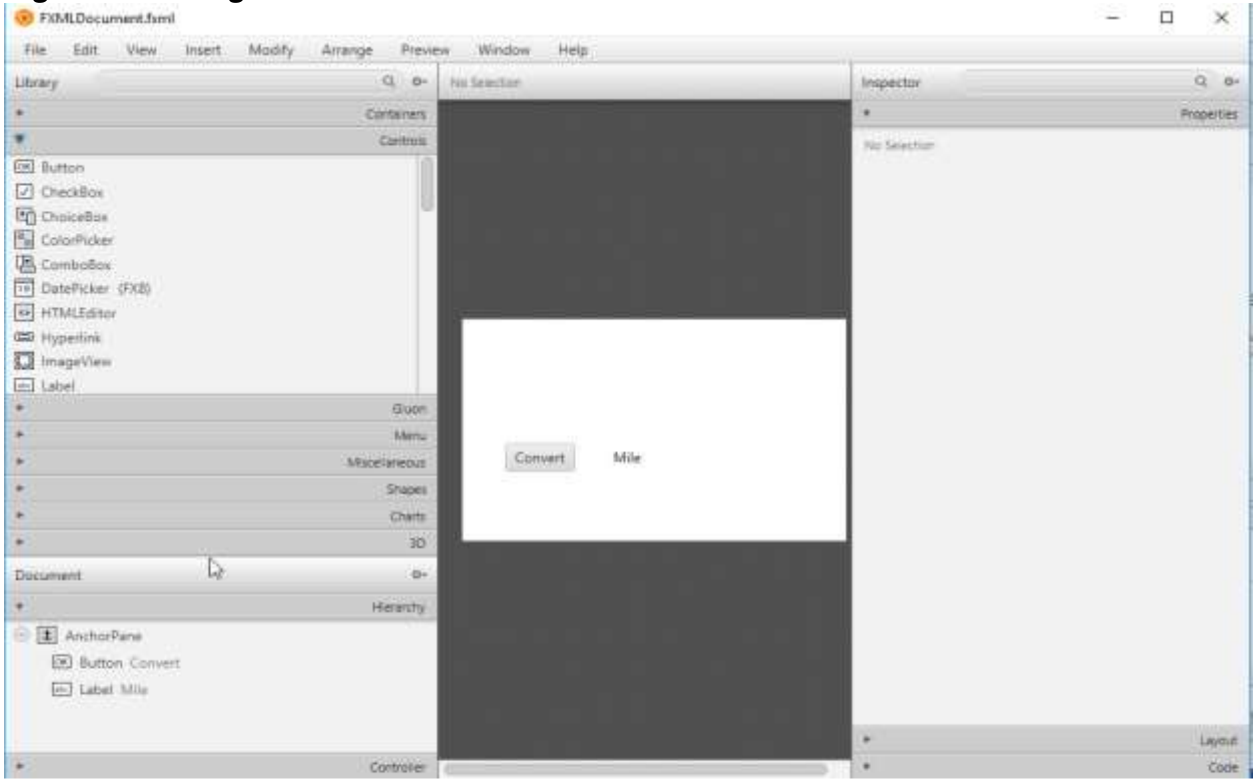
***Figure 8 Rearranging the GUI Elements***



| | |
|---|---|
| Text | Convert |
| Font | System 12px |
| Text Fill | BLACK |
| Wrap Text | |
| Text Alignment | |
| Text Overrun | ELLIPSIS |
| Ellipsis String | ... |
| Underline | |
| Line Spacing | 0 |
| Default Button | |
| Cancel Button | |

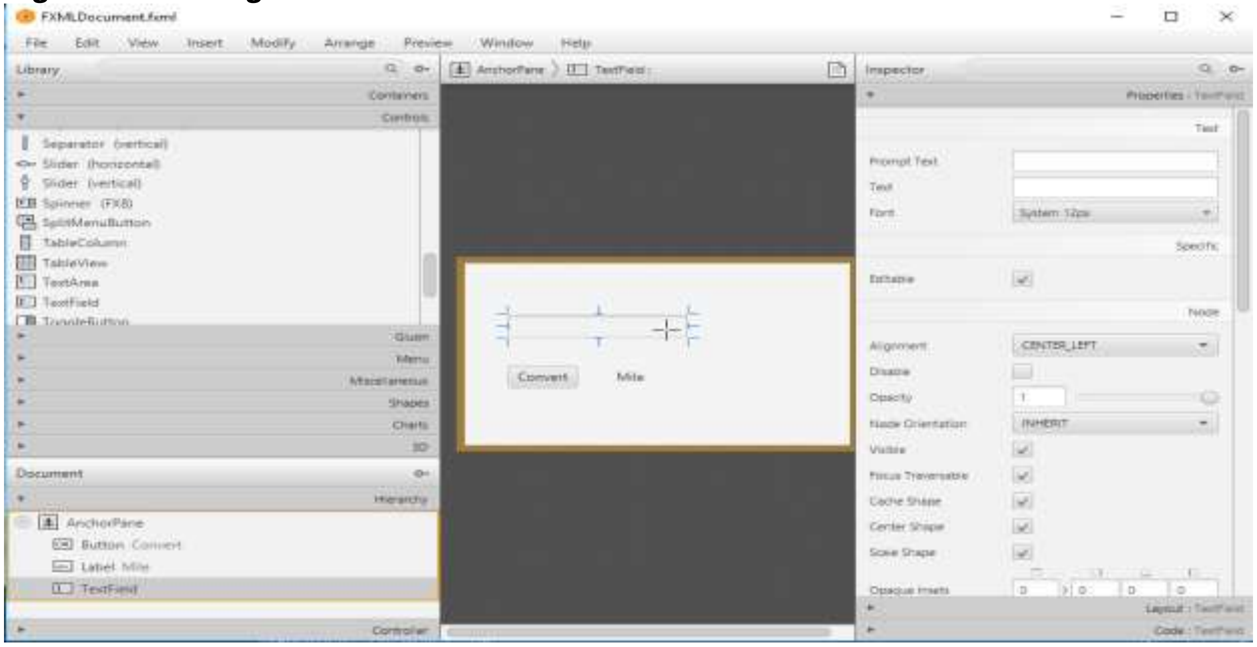
In the **Library** panel click on the Control List. If the controls list does not appear, position the cursor between 3D and Document and the when the resize cursor appears drag down the **Document** panel to make the Controls list visible as shown in *Figure 9*.

***Figure 9 Making the Controls Visible***



Scroll down and find the TextField control. Click on the TextField and drag it into the GUI Area as shown in *Figure 10*.

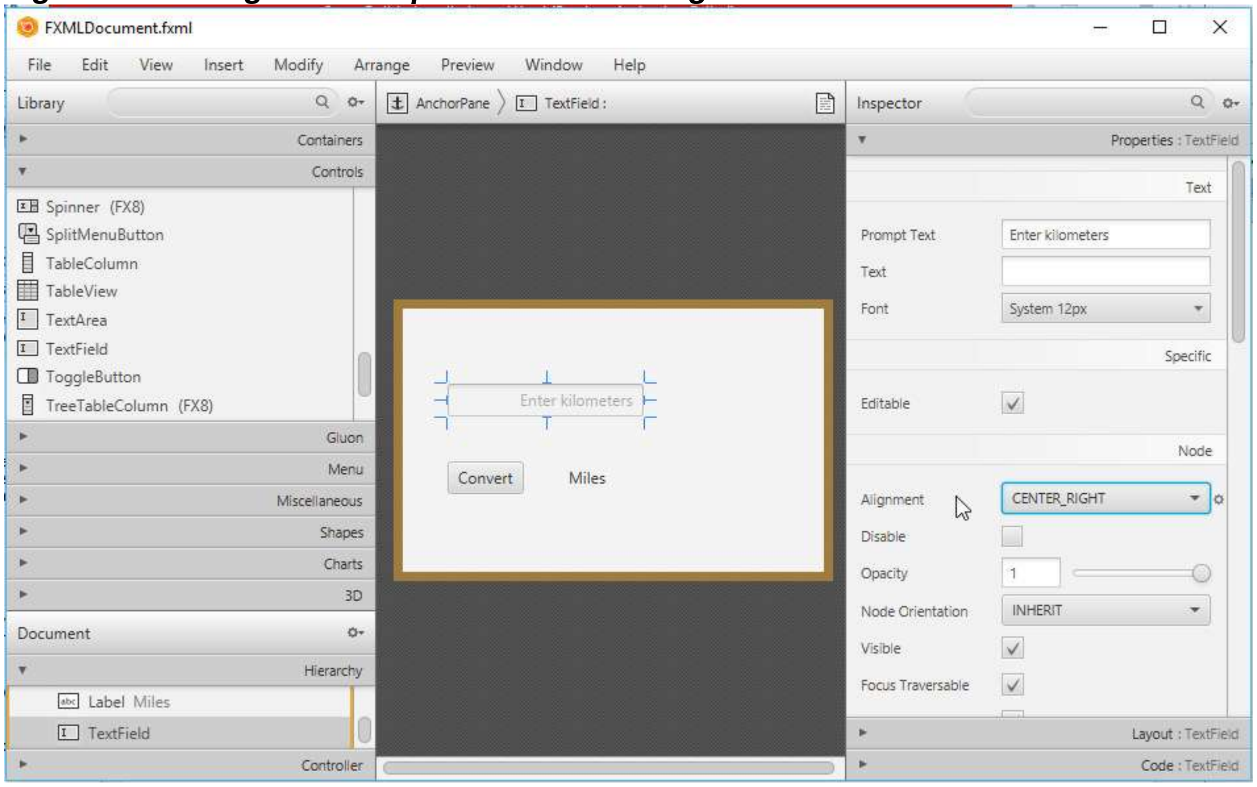
***Figure 10 Adding TextField to the GUI***



Font: System 12px
Alignment: CENTER_LEFT
Opacity: 1
Node Orientation: INHERIT
Opaque Insets: 0 0 0 0

In the Properties section enter *Prompt Text*: *Enter kilometers* and change the *Alignment* to CENTER_RIGHT as shown in Figure 11.

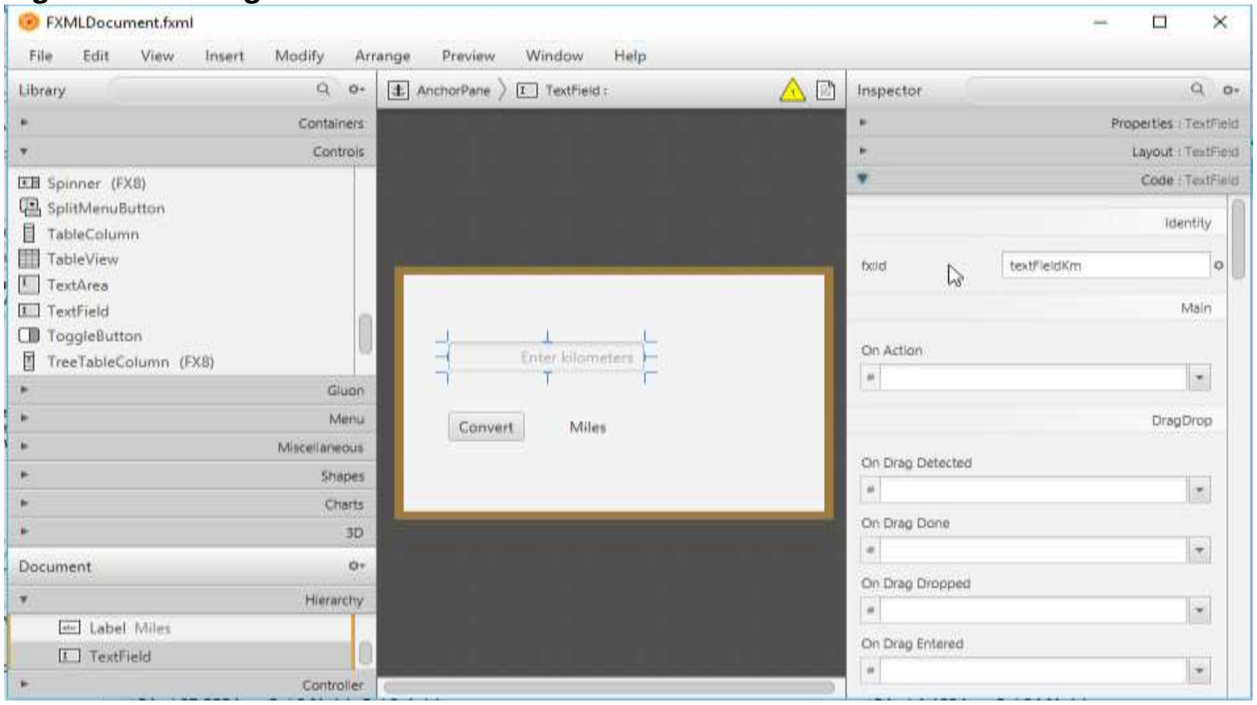
***Figure 11 Setting the Prompt Text and the Alignment***



Prompt Text: Enter kilometers
Text:
Font: System 12px
Editable: ✓
Alignment: CENTER_RIGHT
Disable:
Opacity: 1
Node Orientation: INHERIT
Visible: ✓
Focus Traversable: ✓

In the Code section of the Inspector panel set the *fx:id* to *textFieldKm* as shown in *Figure 12.*

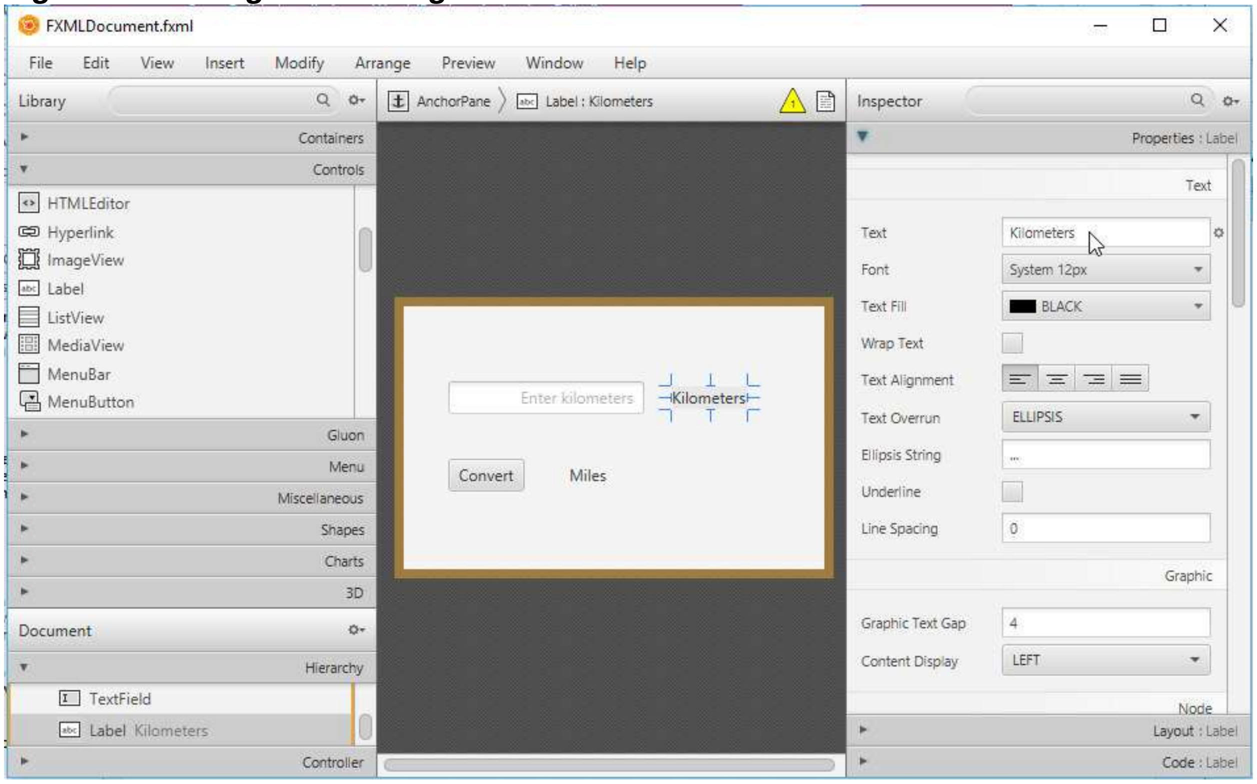
***Figure 12 Setting the fx:id for the TextField***



fx:id: textFieldKm

From the Control list find a Label control, position it on the left of the text filed ant change the *Text* to *Kilometers*, and set the *fx:id* to *labelKm in the Code* section. See the Figure 13 below.

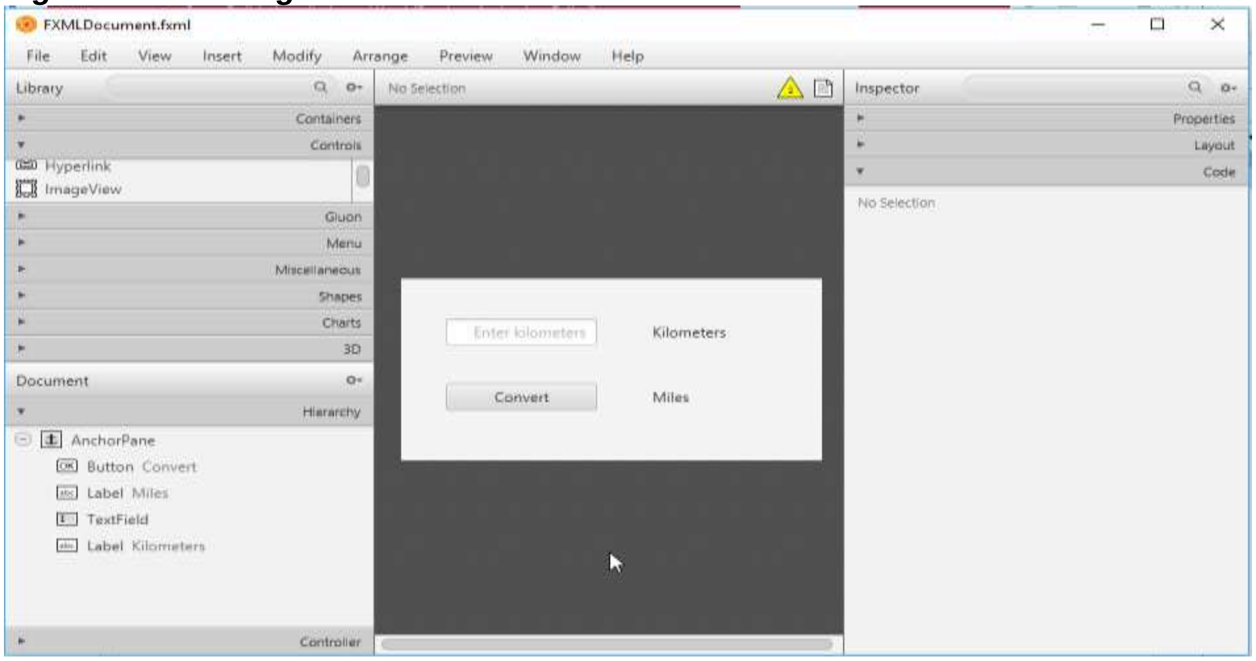
***Figure 13 Adding and Setting the Kilometers Label***



Text: Kilometers
Font: System 12px
Text Fill: BLACK
Wrap Text
Text Alignment
Text Overrun: ELLIPSIS
Ellipsis String: ...
Underline
Line Spacing: 0
Graphic Text Gap: 4
Content Display: LEFT

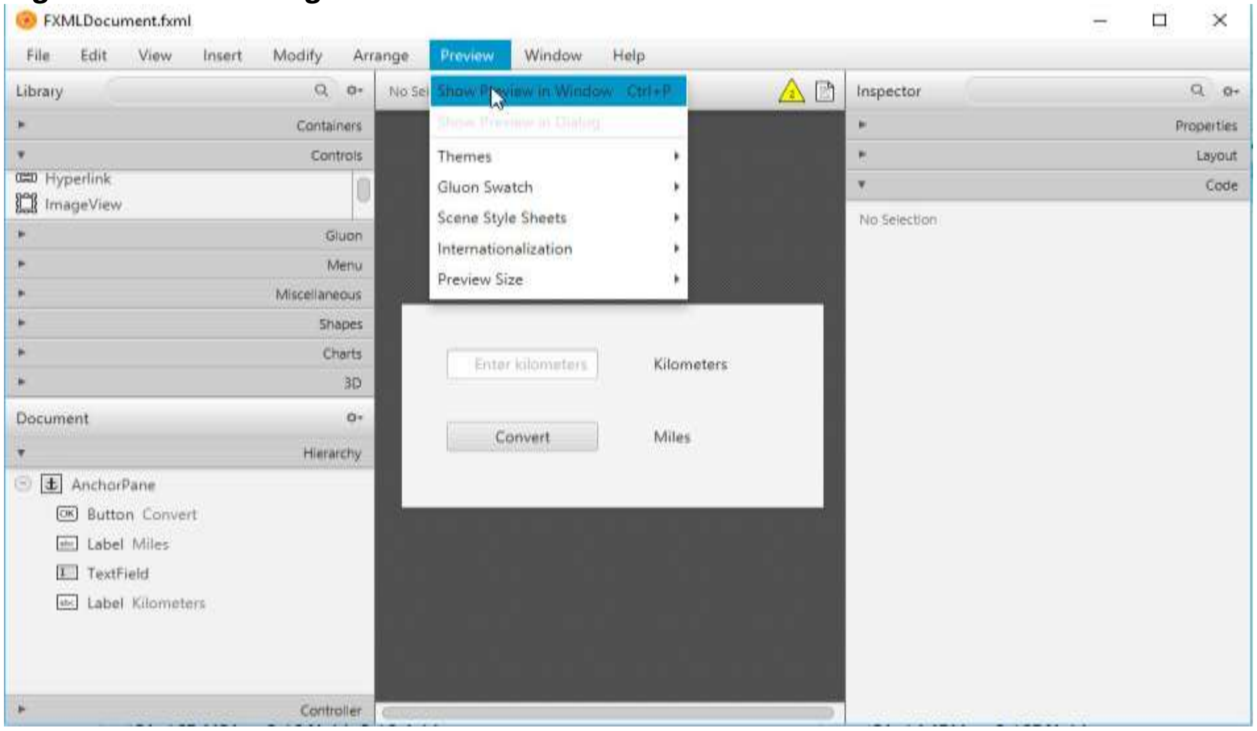
By selecting the controls, resize and rearrange them so that the GUI looks like as in the Figure 14 below.

***Figure 14 Rearranged and Resized GUI***



Now you are ready to see the result from your work.
From the main menu select **Preview, Show Preview in Window**. The GUI will appear on your screen. Close it after previewing it. If you have not saved your work, save it now. Select **File**, **Save** from the main menu.

***Figure 15 Previewing the GUI***



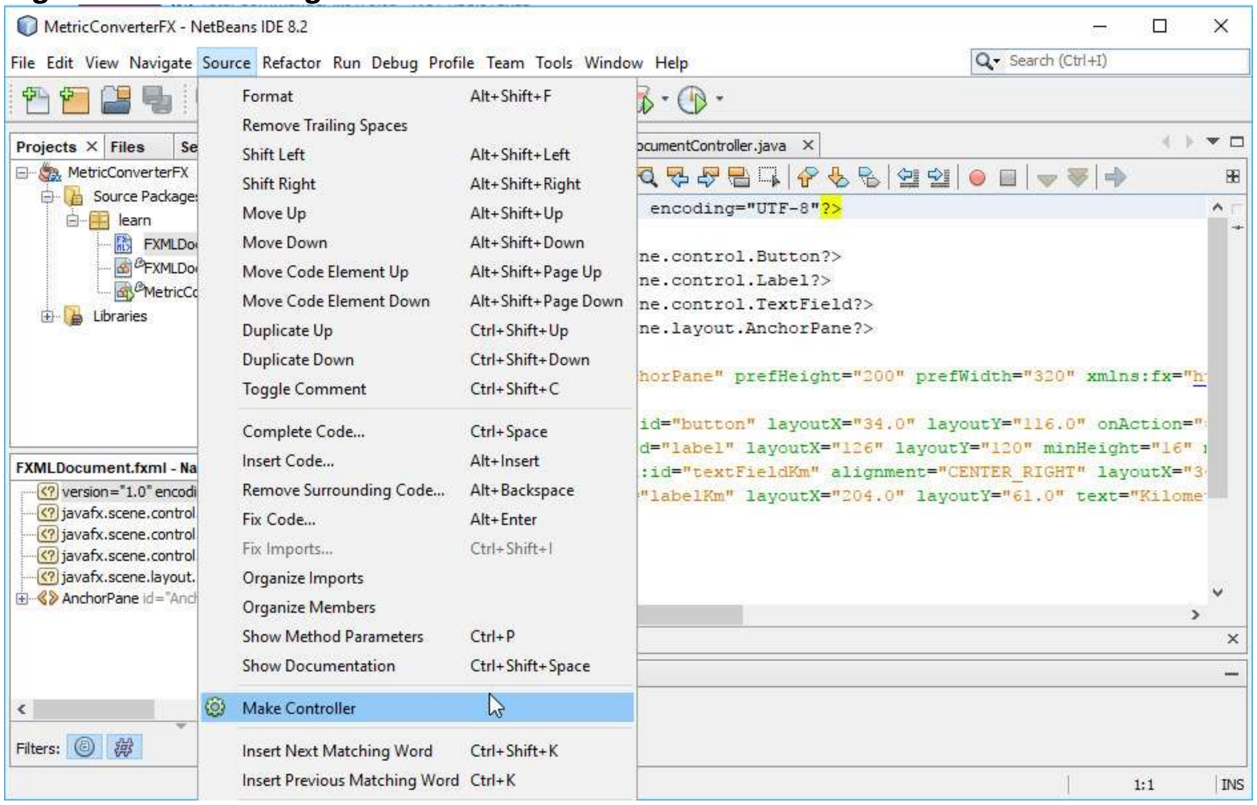
## Step 3 – Generating a Controller Code and Handling Events

### Synchronizing With the Controller Source Code

The NetBeans IDE's **Make Controller** feature allows you to synchronize the modifications you make in the FXML file that is currently opened in Scene Builder and the controller source code opened in NetBeans IDE. Use the **Make Controller** command if you delete or add an element from the Scene Builder's Content panel, or update an fx:id value or a method name in Scene Builder.

Switch to NetBeans IDE. Make sure that the FXMLDocument.fxml is open in the NetBeans Editor. From the main menu select **Source** and then select **Make Controller**.

***Figure 16 Generating a Controller Code***


MetricConverterFX - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Projects Files Se
MetricConverterFX
Source Packages
learn
FXMLDo
FXMLDo
MetricCo
Libraries
FXMLDocument.fxml - Na
version="1.0" encodi
javafx.scene.control
javafx.scene.control
javafx.scene.control
javafx.scene.layout.
AnchorPane id="Anc
Filters:
Format Alt+Shift+F
Remove Trailing Spaces
Shift Left Alt+Shift+Left
Shift Right Alt+Shift+Right
Move Up Alt+Shift+Up
Move Down Alt+Shift+Down
Move Code Element Up Alt+Shift+Page Up
Move Code Element Down Alt+Shift+Page Down
Duplicate Up Ctrl+Shift+Up
Duplicate Down Ctrl+Shift+Down
Toggle Comment Ctrl+Shift+C
Complete Code... Ctrl+Space
Insert Code... Alt+Insert
Remove Surrounding Code... Alt+Backspace
Fix Code... Alt+Enter
Fix Imports... Ctrl+Shift+I
Organize Imports
Organize Members
Show Method Parameters Ctrl+P
Show Documentation Ctrl+Shift+Space
Make Controller
Insert Next Matching Word Ctrl+Shift+K
Insert Previous Matching Word Ctrl+K
ocumentController.java
1:1
INS

```
 encoding="UTF-8"?>

ne.control.Button?>
ne.control.Label?>
ne.control.TextField?>
ne.layout.AnchorPane?>

horPane" prefHeight="200" prefWidth="320" xmlns:fx="h

id="button" layoutX="34.0" layoutY="116.0" onAction="
d="label" layoutX="126" layoutY="120" minHeight="16" 
:id="textFieldKm" alignment="CENTER_RIGHT" layoutX="3
"labelKm" layoutX="204.0" layoutY="61.0" text="Kilome
```

Open the FXMLDocumentController.java in the NetBeans Editor.
Your controller code should look like this:

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package learn;

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;
import javafx.scene.control.TextField;

/**
 *
 * @author user
 */
public class FXMLDocumentController implements Initializable {

    @FXML
    private Label label;
    @FXML
    private TextField textFieldKm;

    @FXML
    private Button button;

 @FXML
    private Label labelKm;

    @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
    }

}
```

NOTE: You can also generate the Controller code using the Scene Builder. From the main menu of the Scene Builder select **View** then select **Show Sample Controller Skeleton**. Copy the text, switch to NetBeans, open the FXMLDocumentController.java in your editor and replace the code with the copied one. Add the necessary imports (for example, import javafx.event.ActionEvent;)

Now you have to write the body of the event handler *handleButtonAction(ActionEvent event).*

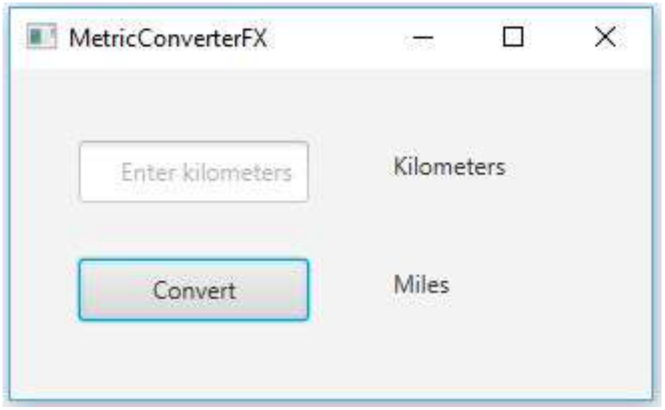
Replace the existing body with the following implementation:

- Get the text from the *textFieldKm*.
- Safely convert the string to a floating-point number. If it cannot be converted, display the message **Enter number** in the *textFieldKm*.
- To convert from kilometers to miles multiply the number by the following coefficient: 0.621371.
- Change the text of the *label* so that it display the conversion.
- Finally, request the focus to the *textFieldKm.*

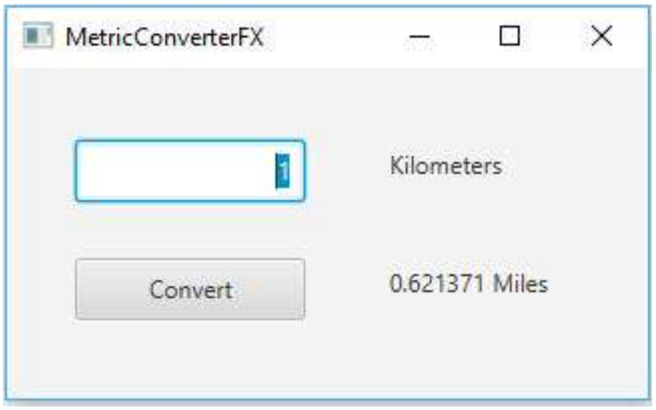
Leave the body of the *initialize()* method empty.

The final step is to set the title of your application. Open the *MetricConverterFX.java* in your NetBeans IDE editor and set the stage title to **MetricConverterFX**.

Run the application from NetBeans. This is how your final result should look like.



Enter 1 in the text field and click the *Convert* button. The result of the conversion is shown below.



A working application jar file is provided for you on Blackboard. If you want to earn a full credit for your work, your application must look and behave as the provided application.

## *Submission*

No submission is required for this Hybrid Activity but if you want to earn some marks, you have to demonstrate your work by the end of your lab period in week 10 of the semester

.

**Marks:** **3% of your maximum course mark**

**3%** of your total course marks are allocated for this activity. In order to receive full marks your application **must** look and behave exactly as the provided application.

## Marking Scheme

**Maximum mark – 3% of the total course marks**

| **Deduction Event** | **ActionPerformed (Deduction (%))** |
|---|---|
| Missing or late demonstration | **100** |
| Missing NetBeans ***MetricConverterFX*** project folder and package | **100** |
| **Application does not work properly** | **up to 100** |
| Application does not compile and/or run | **100** |
| Wrong title or package (learn) | **10** |
| Components are not aligned properly | **20** |
| Missing component | **20** |
| Converter does not convert safely and correctly | **50** |

## *Questions*

Q1. Can you use Scene Builder as a stand-alone application not integrated with any IDE?
Q2. Can you generate a controller code using Scene Builder?
Q3. Are NetBeans IDE and Scene Builder synchronized?
Q4. Does Scene Builder generate FXML describing the GUI?

And do not forget that:

“In the entire history of the human species, every tool we've invented has been to expand muscle power. All except one. The integrated circuit, the computer. That lets us use our brain power.” David Gerrold

but also never forget that*:*

“Technology is just a tool. In terms of getting the kids working together and motivating them, the teacher is the most important.” Bill Gates

and also:

“Technology is a bit of a double-edged sword. Used right, it's a wonderful tool, but unfortunately, it makes it easier for a lot of mediocre people to get really crappy ideas out.” Martin Gore