

Java Programming

Arthur Hoskey, Ph.D.
Farmingdale State College
Computer Systems Department

- FXML NetBeans

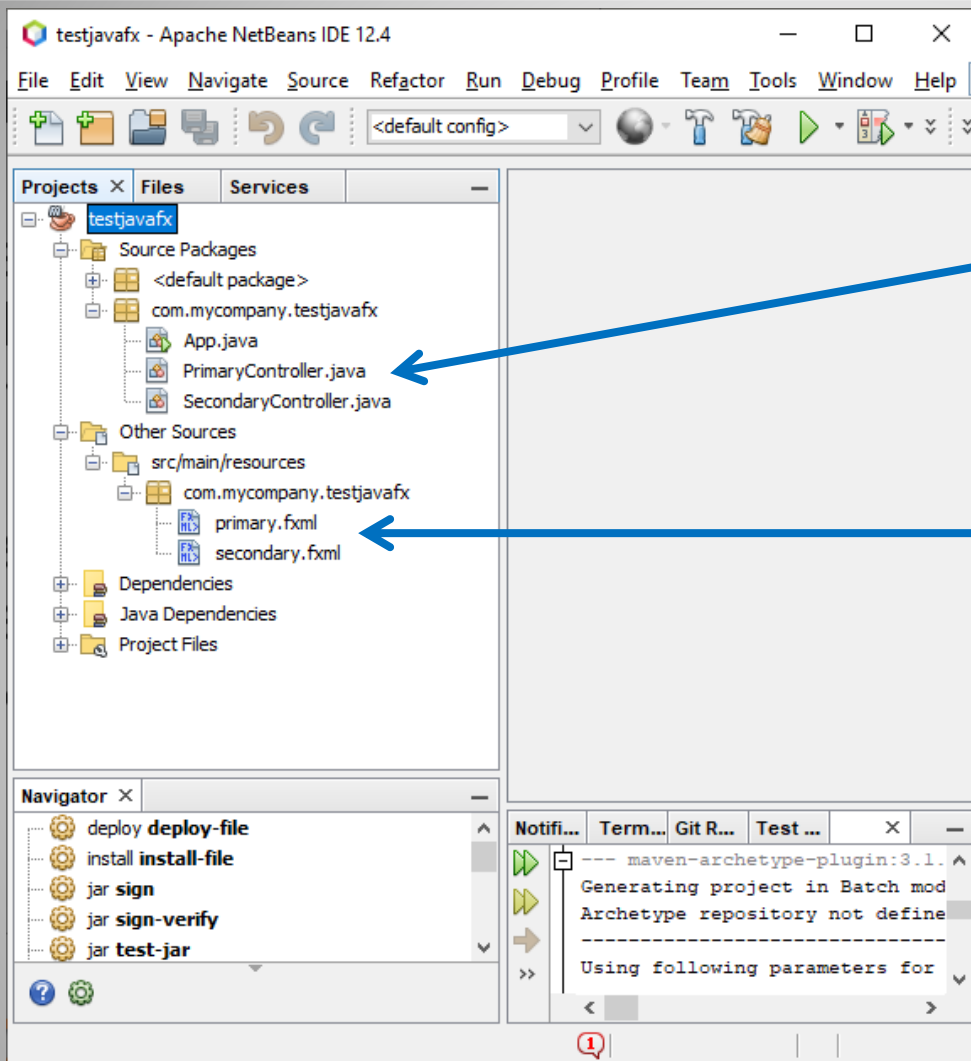
Today's Lecture

- Graphical User Interface (GUI) for Java
 - Create windows type programs
 - You can use buttons, textboxes, labels etc.
- JavaFX is platform independent
 - The code will run on both Windows and Linux platforms
 - Other languages such as Visual Basic can only be run on one platform.
 - A JavaFX program can be written on a Windows machine and run on a Linux machine (assuming the correct Java runtime is installed on both machines).

JavaFX

- Open NetBeans.
- File|New Project|FXML JavaFX Maven Archetype.
- This will create a new project that will display an empty window.

NetBeans – Create JavaFX Project



Java Source Code Files

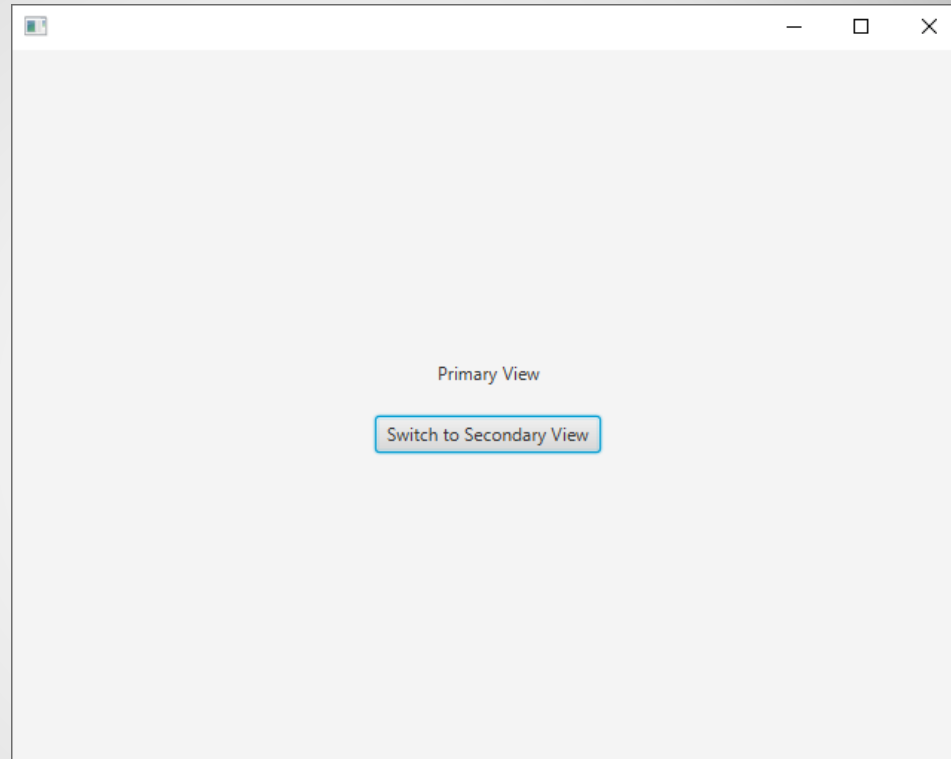
Windows GUI Files
(.fxml files)

NetBeans JavaFX Project

- If you run the project as is, you will see the following:

No title given to the
window

Uses the default
window size



Run the Project – Shows Window

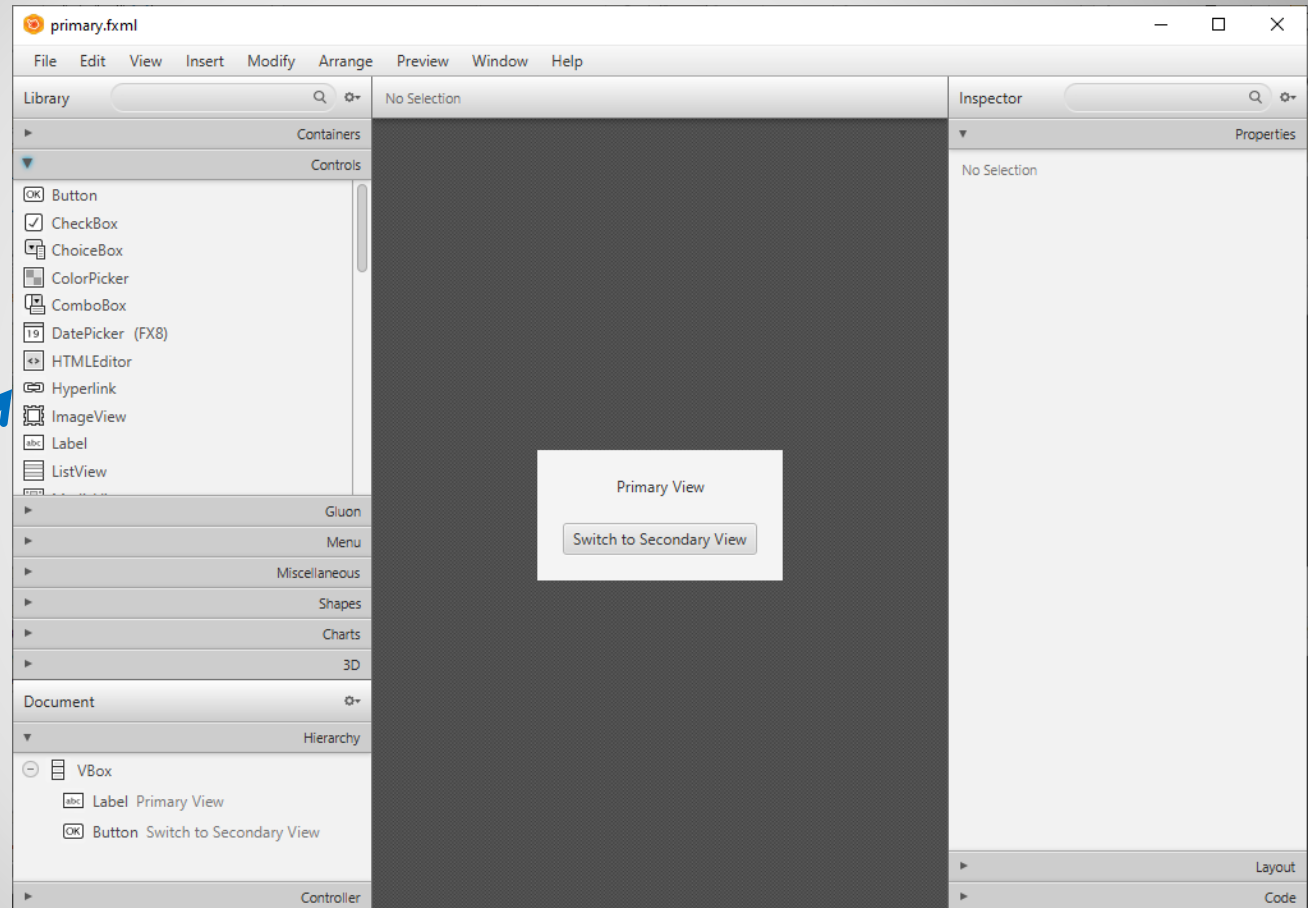
- Scene Builder is an application used to edit .fxml files.
- You can drag and drop controls and make the window look the way you want it to.
- If you double click the primary.fxml file in NetBeans it will automatically open it in Scene Builder.
- For example...

Scene Builder

- Scene Builder

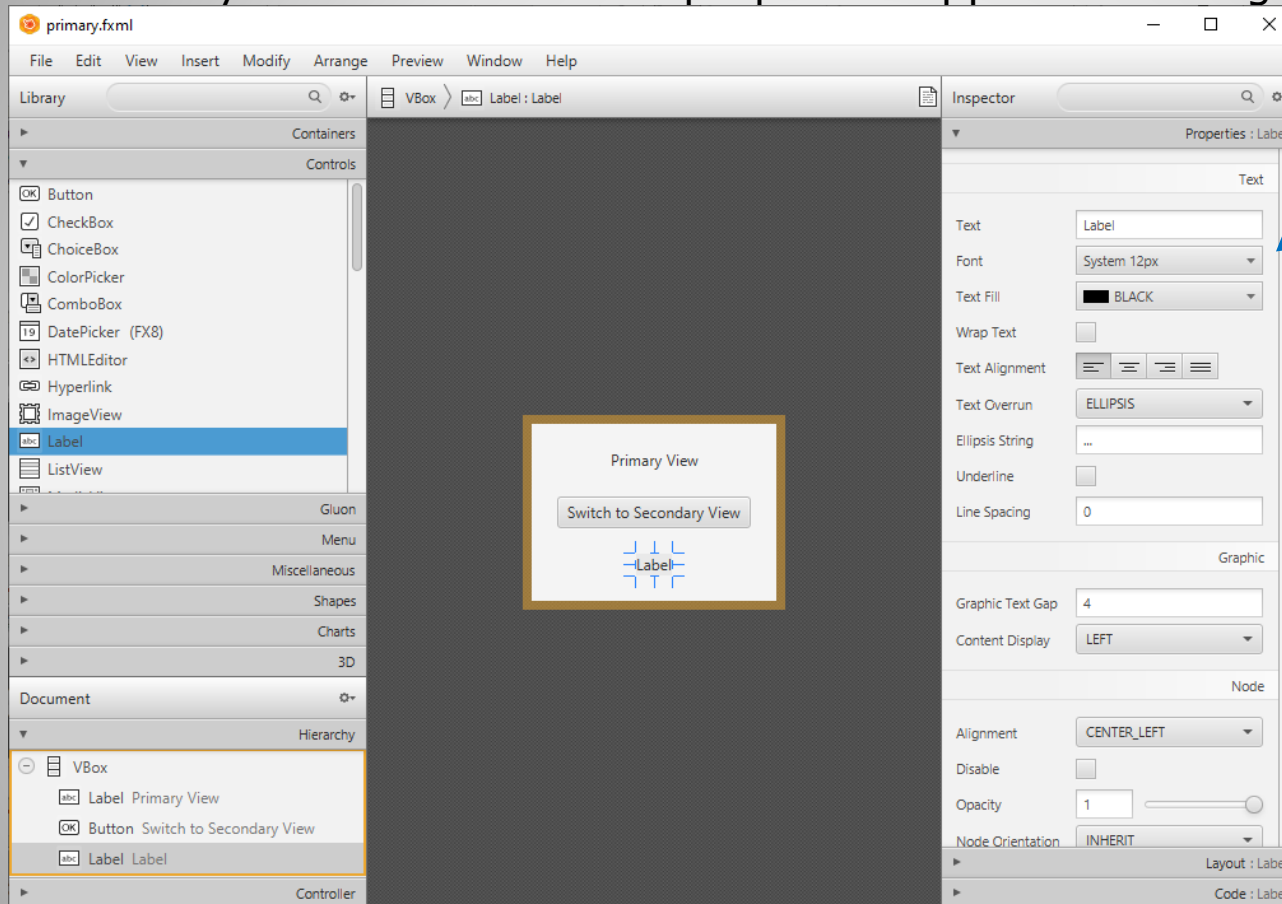
There are multiple tabs of items on the left that you can add to the window

Controls that you can put in the window



Scene Builder

- You can drag a label on to the window.
- When you click the label its properties appear in the right side window.

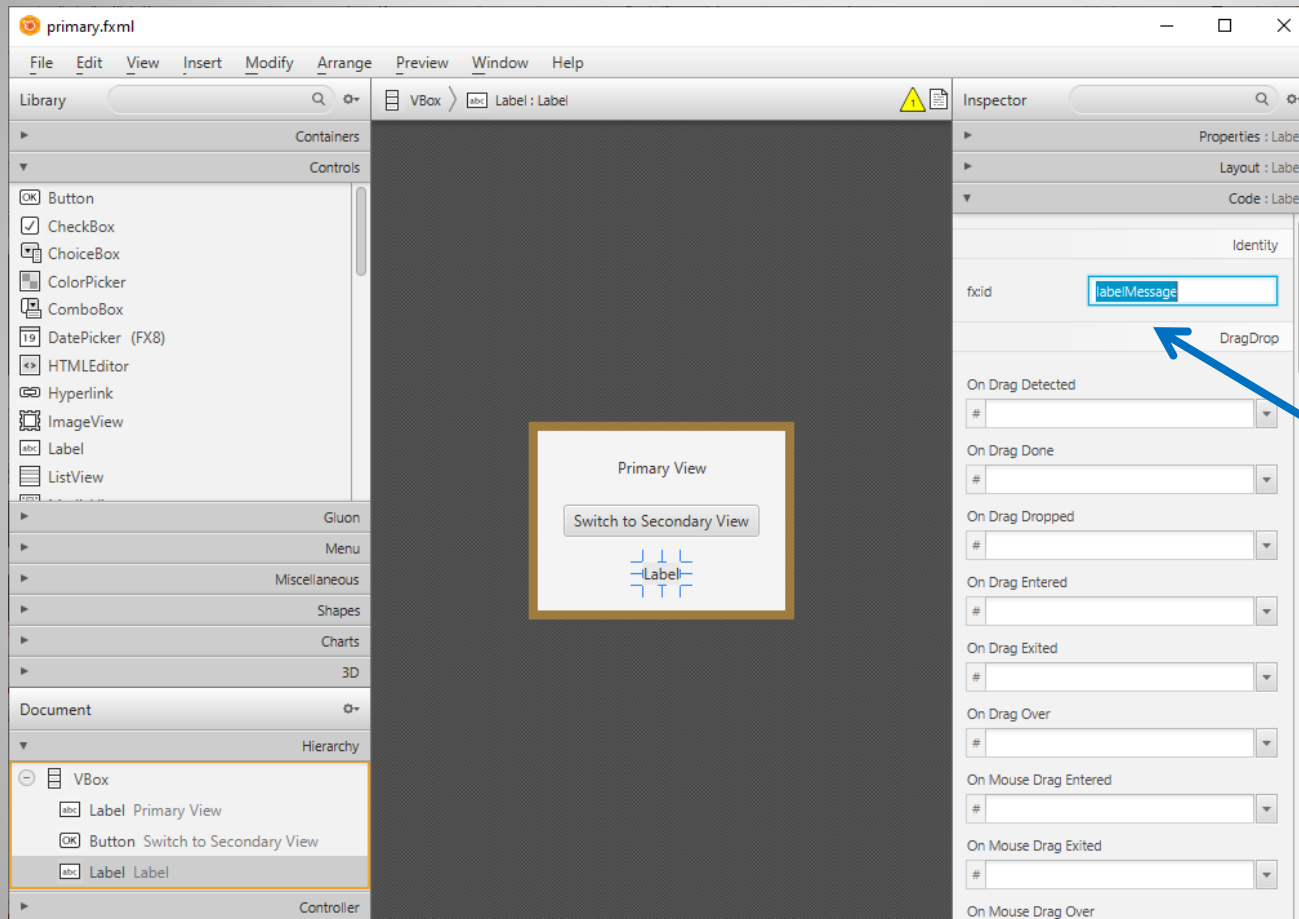


The properties of the selected control appear in the window on the right

There are lots of values that can be updated. For example, the font, color, and text that appears on the label just to name a few

Add Control and Show Properties

- You must set the `fx:id` (basically a variable name) of a control if you want to use it in code.

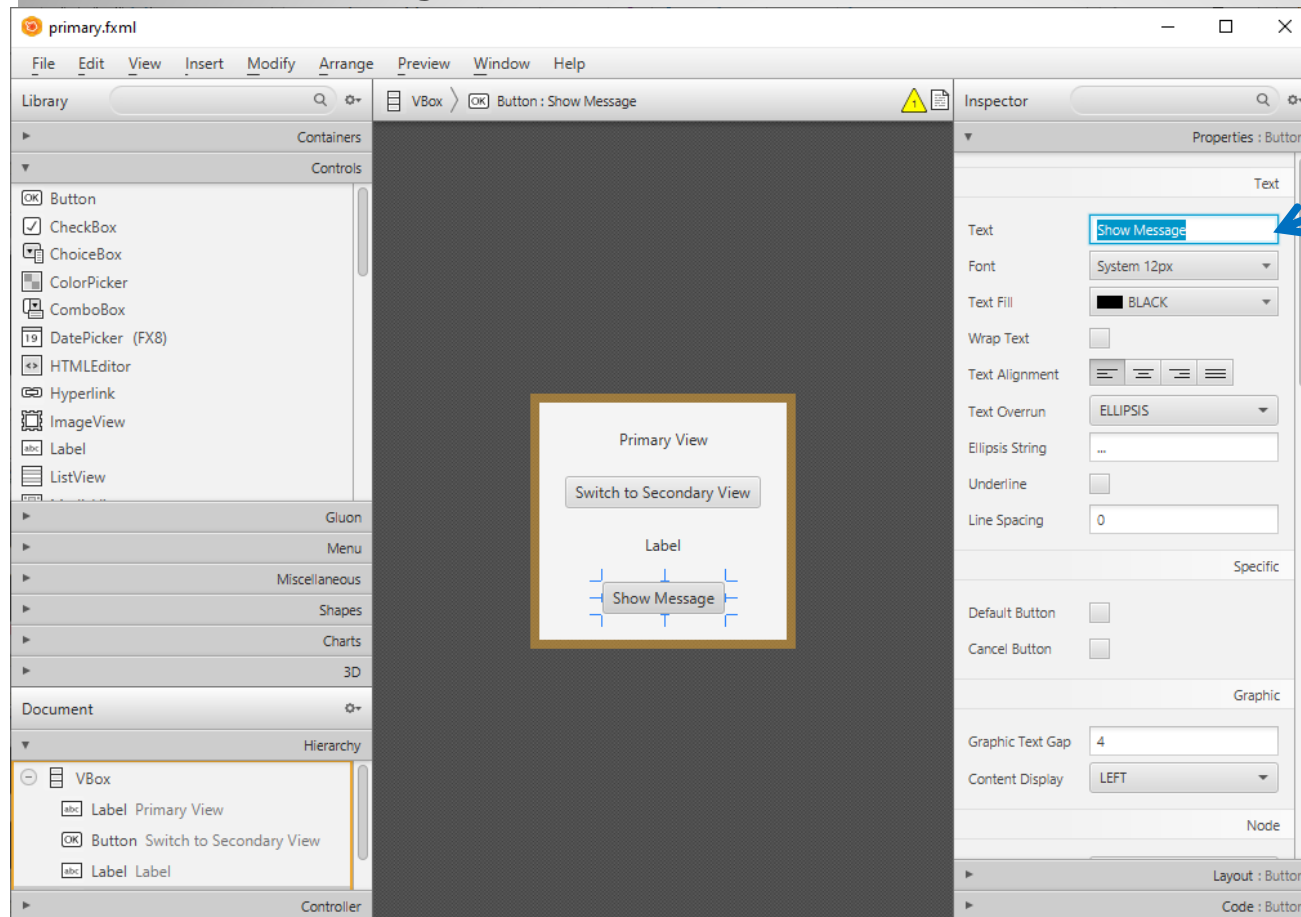


Open the Code Tab by clicking it (it will be at the bottom right before you click it)

Set the `fx:id` property by typing in the edit box. The `fx:id` was set to `labelMessage` in this example

Set `fx:id` of Control

- A button was added below the label. The button's text was changed to "Show Message".



Set button text to "Show Message"

Add a Button

- Add an event handler for the button to the PrimaryController class.

```
public class PrimaryController {
```

```
    @FXML
```

```
    private void handleShowMessageButton()
```

```
{
```


```
    // Button code goes here...
```

```
}
```

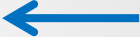
```
    // Other controller code is here...
```

```
}
```

The **@FXML** annotation
indicates that this will be
used in an FXML file



We will setup the
application to run
this method when the
button is pressed
(next slide)

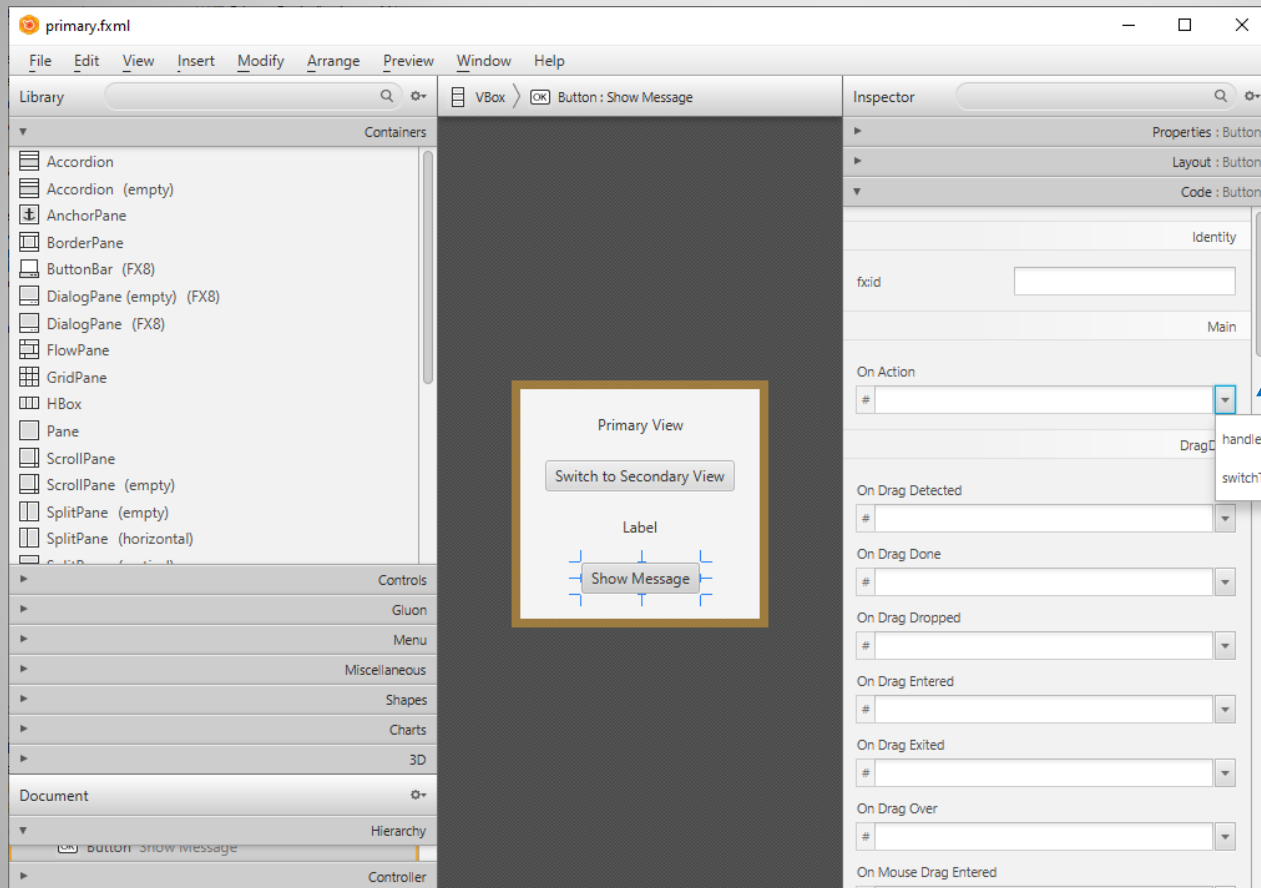


Add Button Event Handler

- The controller class contains the code that will run in the background.
- NetBeans automatically creates a controller class for us that is named PrimaryController.
- The GUI event handlers for one screen go in one controller class that is associated with that screen.

Controller Class

- Set the button's event handler by setting the .



Click the drop down to get a list of methods to choose from (method names were cut off in the screen shot)

Set Button Event Handler

- When a button is pressed the method listed in On Action will be called automatically.
- You can put as much code as you want in the event handler (handleShowMessageButton() in this case).

```
public class PrimaryController {
```

```
    @FXML
```

```
    private void handleShowMessageButton()
```


```
    {
```

```
        System.out.println("I love JavaFX");
```

```
    }
```

```
    // Other controller code is here...
```

```
}
```



The message will be
printed in the console
whenever the button
is pressed.

Add Code to Event Handler

- Update a control property from code.

```
public class PrimaryController {  
    @FXML  
    private Label labelMessage;
```

Add a member variable for the label.
This member variable must EXACTLY
match the fx:id it was given in Scene
Builder or it will not work

```
    @FXML  
    private void handleShowMessageButton()  
    {
```

Call setText on the
labelMessage
member variable

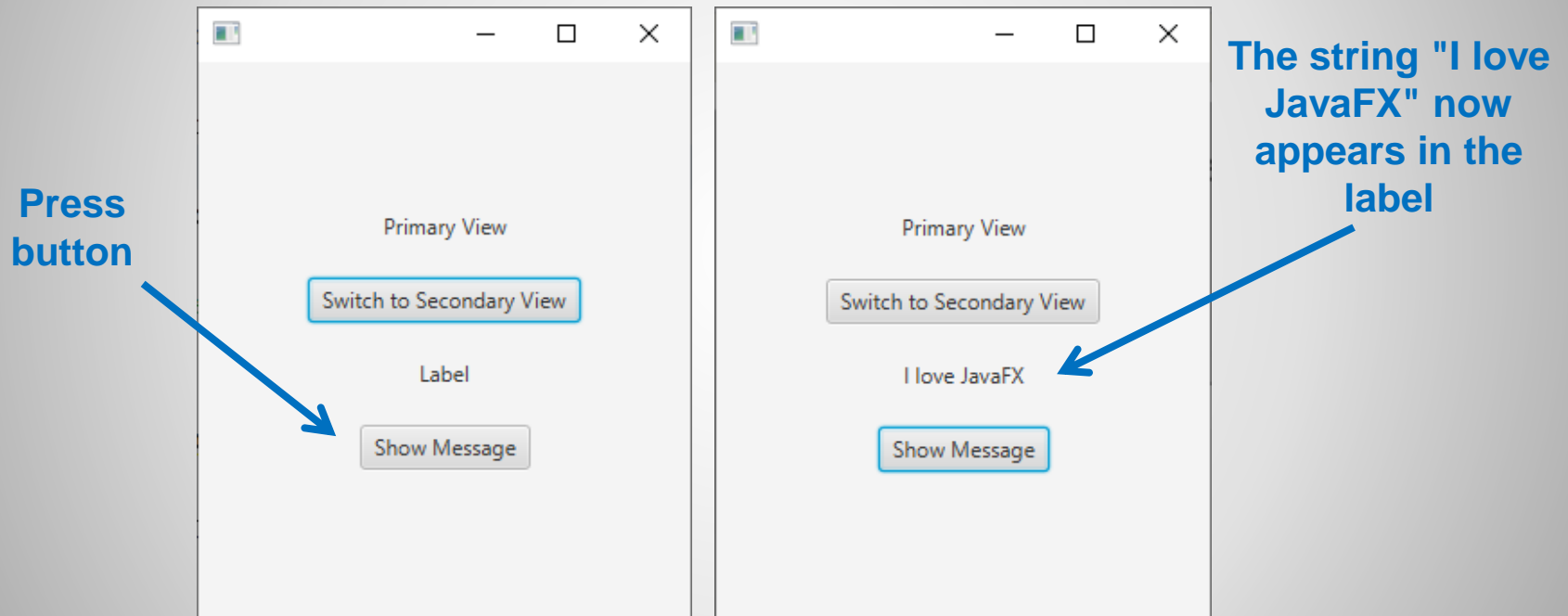
```
        labelMessage.setText("I love JavaFX");
```

Put the string "I love
JavaFX" in the label

```
    }  
  
    // Other controller code is here...  
}
```

Update Control in Code

- The following screenshots show before and after pressing the Show Message button:



Update Control in Code

- End of Slides

End of Slides