**<http://docs.oracle.com/javafx/index.html>**

**http://docs.oracle.com/javafx/scenebuilder/1/get\_started/jsbpub-get\_started.htm**

**Getting Started with JavaFX Scene Builder**

JavaFX Scene Builder is a design tool for the JavaFX platform. It allows for a simple drag-and-drop positioning of graphical user interface (GUI) components onto a JavaFX scene. As you build the scene, the FXML code for the designed GUI is automatically generated. JavaFX Scene Builder provides a simple yet intuitive interface that can help even nonprogrammers to quickly create prototype interactive applications that connect GUI components to the application logic.

**Introduction**

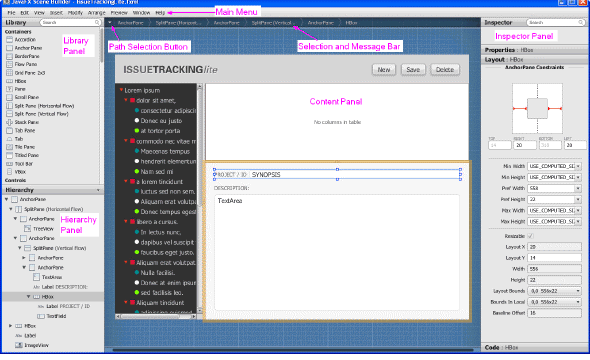
This Getting Started document presents the step-by-step creation of a simple issue-tracking application. It shows how quickly you can build the GUI for a JavaFX application by using JavaFX Scene Builder and connect it to the source code that handles the interaction between the data and the user interface. You will use a NetBeans project named IssueTrackingLite.

**Preparing for This Tutorial**

Ensure that you have made the following preparation necessary before you continue with this tutorial.

1. **Install all the required software** before you can use the JavaFX Scene Builder. You also need to install the NetBeans IDE 7.2 Beta, which is used in this tutorial.
2. **Download the samples file** from the same JavaFX Scene Builder download location at <http://www.oracle.com/technetwork/java/javafx/downloads/devpreview-1429449.html>, and extract its contents. You will find the sample that you need to continue with this tutorial in the <*javafx-scenebuilder-samples-1\_0-beta-install-dir*>/IssueTrackingLite folder. The issue-tracking system enables you to query existing project issues, modify them, or add new issue.
3. **Ensure that you are familiar with how to build a user interface programmatically with JavaFX.** Knowledge of how to work with a scene graph is especially useful, as the hierarchical structure of FXML closely parallels the structure of the JavaFX scene graph. For more information about JavaFX and FXML, visit their respective articles at the [JavaFX Documentation web site](http://docs.oracle.com/javafx/index.html).
4. **Use the** [**JavaFX Scene Builder User Guide**](http://docs.oracle.com/javafx/scenebuilder/1/user_guide/jsbpub-user_guide.htm) to familiarize yourself with the JavaFX Scene Builder user interface that is shown in [Figure 1](http://docs.oracle.com/javafx/scenebuilder/1/get_started/jsbpub-get_started.htm#CIHIFIAC). Click the image to see a larger version.

***Figure 1 Main Window for JavaFX Scene Builder (Click image to enlarge.)***

[](http://docs.oracle.com/javafx/scenebuilder/1/get_started/main-window-big.gif)  
[Description of "Figure 1 Main Window for JavaFX Scene Builder"](http://docs.oracle.com/javafx/scenebuilder/1/get_started/img_text/main-window-small.htm)

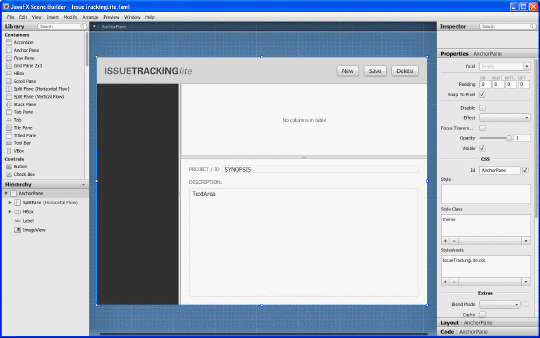
**Start Up**

After you have familiarized yourself with FXML and the JavaFX Scene Builder user interface, begin building an FXML layout using JavaFX Scene Builder. You will use the NetBeans IDE to open a new FXML file, which you will edit using JavaFX Scene Builder. The controller source file and the CSS file used in this tutorial are already provided with the IssueTrackingLite sample. These files are in the same project folder that will contain the new FXML file.

1. Open the IssueTrackingLite sample project using the NetBeans IDE.
   1. Start up the NetBeans IDE on your Windows platform by double-clicking the **NetBeans IDE 7.2 Beta shortcut**, or select **Start,** then **Programs**, then **NetBeans**, and finally, **NetBeans IDE 7.2 Beta**. On a Mac OS X platform, double-click the **NetBeans IDE 7.2 Beta** application icon.
   2. From the Main menu, select **File** and then **Open Project**.
   3. From the Open Project dialog box, navigate to where you extracted the javafx\_scenebuilder\_samples-1\_0.zip file and open the **IssueTrackingLite** project.
   4. In the Projects window, expand the **IssueTrackingLite**, **Source Packages**, and **issuetrackinglite** nodes. Double-click the **IssueTrackingLite.fxml** node to open the file.

The main window for the JavaFX Scene Builder tool appears with the IssueTrackingLite.fxml file opened in the Content panel, as shown in [Figure 2](http://docs.oracle.com/javafx/scenebuilder/1/get_started/jsbpub-get_started.htm#BEIFIHCI).

***Figure 2 Completed IssueTrackingLite.fxml Opened in JavaFX Scene Builder Window (Click image to enlarge.)***

[](http://docs.oracle.com/javafx/scenebuilder/1/get_started/itl-ui-complete-big.gif)  
[Description of "Figure 2 Completed IssueTrackingLite.fxml Opened in JavaFX Scene Builder Window"](http://docs.oracle.com/javafx/scenebuilder/1/get_started/img_text/itl-ui-complete.htm)

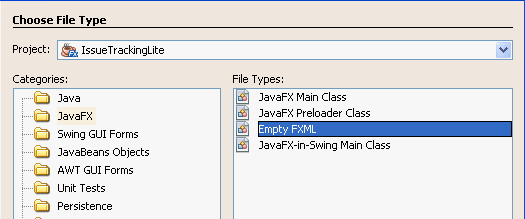
* 1. Save the file with a different name so that you can create your own FXML layout file. From the Main menu, select **File** and then **Save As**. Type IssueTrackingLiteComplete.fxml in the File name text field and click **Save**. You can keep the JavaFX Scene Builder window for this file opened so that you can use it to compare with the version of the layout you are about to create.
  2. In the Projects window of the IDE, right-click the node for the **IssueTrackingLite.fxml** file and select **Delete** so that you can use the same file name for the FXML layout you will build. On the Confirm Object Deletion dialog box, click **Yes**.

## Create the FXML File and the Base Panes

Now you can begin building your own UI interface for the IssueTrackingLite application using JavaFX Scene Builder. The FXML layout that you are about to build is the interface that enables you to query existing project issues, modify them, or add new issues.

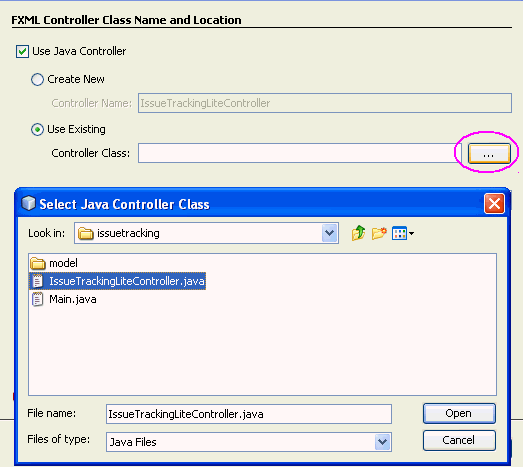
1. Create a new JavaFX FXML file.
   1. From the Projects window of the IDE, right-click the issuetrackinglite node, select New, and then Other.
   2. In the New File dialog box, select the JavaFX category and select Empty FXML file type, as shown in [Figure 3](http://docs.oracle.com/javafx/scenebuilder/1/get_started/jsbpub-get_started.htm#BEICFGII). Click Next.

Figure 3 Choose Empty FXML File Type

  
[Description of "Figure 3 Choose Empty FXML File Type"](http://docs.oracle.com/javafx/scenebuilder/1/get_started/img_text/ide-new-file-dialog.htm)

* 1. In the New Empty FXML dialog box, type IssueTrackingLite in the FXML Name text field. This is the name used in the controller source file. Click Next.
  2. Select the Use Java Controller check box and the Use Existing option.
  3. Click the file browser button next to the Controller Class text field. In the Select Java Controller Class dialog box, open the IssueTrackingLiteController.java, as shown in [Figure 4](http://docs.oracle.com/javafx/scenebuilder/1/get_started/jsbpub-get_started.htm#BEIBEJDB). Click Next

Figure 4 Use the Existing Java Controller Class File

  
[Description of "Figure 4 Use the Existing Java Controller Class File"](http://docs.oracle.com/javafx/scenebuilder/1/get_started/img_text/ide-fxml-controller.htm)

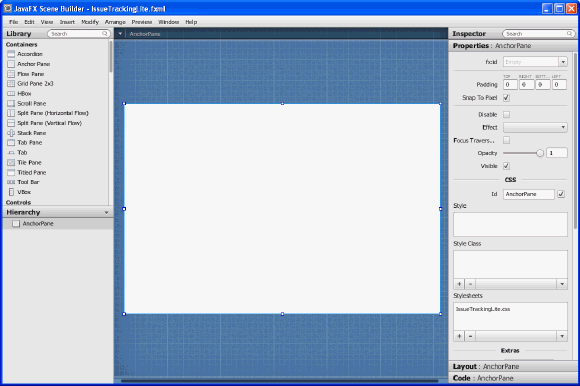
* 1. Select the Use Cascading Style Sheet check box and the Use Existing option.
  2. Click the file browser button next to the CSS File text field. In the Select CSS File dialog box, open the IssueTrackingLite.css file. Click Finish.

The IssueTrackingLite.fxml file is opened in a source editor window. This new FXML file is in the same folder as the provided controller source code, IssueTrackingLiteController.java, that will handle the user interface that you build with this tutorial

* 1. In the Projects window, double-click the IssueTrackingLite.fxml node to open the file in the JavaFX Scene Builder tool.

The main window for the JavaFX Scene Builder tool appears with an empty FXML file opened in the Content panel, as shown in [Figure 5](http://docs.oracle.com/javafx/scenebuilder/1/get_started/jsbpub-get_started.htm#BABBGIGE). The initial AnchorPane element shown in the Content panel will contain the GUI layout that you build. Click the image to see a larger version.

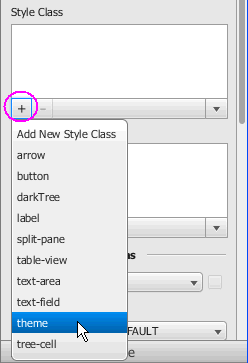
Figure 5 Main Window at Start Up (Click image to enlarge.)

[](http://docs.oracle.com/javafx/scenebuilder/1/get_started/init-window-big.gif)  
[Description of "Figure 5 Main Window at Start Up"](http://docs.oracle.com/javafx/scenebuilder/1/get_started/img_text/init-window.htm)

|  |
| --- |
| Tip:  Enlarge the Content panel to get a larger working area. Select the initial AnchorPane in the Hierarchy panel. In the Layout panel on the right, type 800 in the Pref Width field. Type 600 in the Pref Height and press Enter. The Content panel is resized to 800 by 600 pixels. Alternatively, you can use the handle at the bottom right corner of the panel to enlarge the it to the desired size. Resize the Scene Builder window so you are able to view the larger Content panel in its entirety. |

1. Set the style class to use for the entire layout.
   1. In the Hierarchy panel, select that the row for the top AnchorPane container, if it is not already selected.
   2. In the Properties panel, click the button with the + in the Style Class list. The list of style classes is retrieved from the IssueTrackingLite.css file. Select theme, as shown in [Figure 6](http://docs.oracle.com/javafx/scenebuilder/1/get_started/jsbpub-get_started.htm#BABDFFCJ).

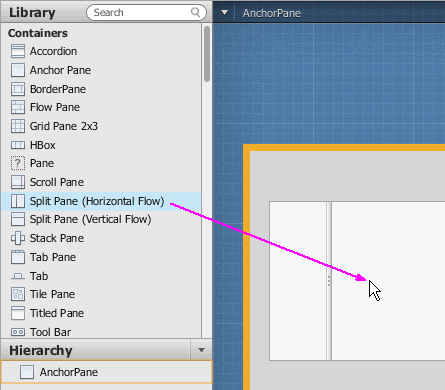
Figure 6 Set the Style Class for the Layout

  
[Description of "Figure 6 Set the Style Class for the Layout"](http://docs.oracle.com/javafx/scenebuilder/1/get_started/img_text/set-styleclass.htm)

1. From the Library panel on the left, select Split Pane (Horizontal Flow) to begin building the GUI for the IssueTrackingLite application. Drag and drop it on the Content panel on the right, as shown in [Figure 7](http://docs.oracle.com/javafx/scenebuilder/1/get_started/jsbpub-get_started.htm#BABJJCHD).

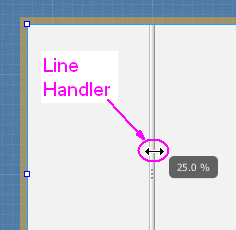
After you drop the Split Pane object, the Hierarchy panel on the bottom left side of the tool is updated to show the Split Pane (Horizontal Flow) element and the two AnchorPane elements that comprise it.

Figure 7 Add Split Pane (Horizontal Flow)

  
[Description of "Figure 7 Add Split Pane (Horizontal Flow)"](http://docs.oracle.com/javafx/scenebuilder/1/get_started/img_text/drag-vsplit-small.htm)

1. With the Split Pane (Horizontal Flow) selected in the Content panel, select Modify and then Fit To Parent from the Main menu. This action adjusts the Split Pane (Horizontal Flow) element to fill the space occupied by its container parent, the topmost AnchorPane shown in the Hierarchy panel.
2. From the Main menu, select File and then Save to save the file for the first time.  
   This action saves your work in the src folder for the IssueTrackingLite sample NetBeans project, <javafx\_scenebuilder\_samples-1\_0-beta-install-dir>/IssueTrackingLite/src/issuetrackinglite/IssueTrackingLite.fxml. Perform this Save action frequently as you build your UI to ensure that your work is intact.
3. With SplitPane (Horizontal Flow) selected in the Hierarchy panel, grab the vertical divider and move it until the percentage displays about 25%, as illustrated in [Figure 8](http://docs.oracle.com/javafx/scenebuilder/1/get_started/jsbpub-get_started.htm#BABGDHCF). Alternatively, you can select the Properties panel on the right-hand side and type 0.25 in the Divider Positions field.

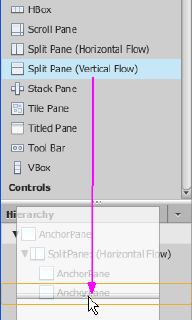
Figure 8 Move Vertical Divider

  
[Description of "Figure 8 Move Vertical Divider"](http://docs.oracle.com/javafx/scenebuilder/1/get_started/img_text/vertical-move.htm)

|  |
| --- |
| Note:  From the Main menu, select Edit and then Undo to undo actions when necessary. |

1. To create the vertical flow split pane sections, drag and drop a Split Pane (Vertical Flow) element from the Library panel at the top to the Hierarchy panel at the bottom. Ensure that you make the drop when the row for the second AnchorPane container of the SplitPane (Horizontal Flow) element is surrounded by the orange highlight, as shown in [Figure 9](http://docs.oracle.com/javafx/scenebuilder/1/get_started/jsbpub-get_started.htm#BABHIFGA).

Figure 9 Add Split Pane (Vertical Flow) Element

  
[Description of "Figure 9 Add Split Pane (Vertical Flow) Element"](http://docs.oracle.com/javafx/scenebuilder/1/get_started/img_text/horiz-split-pane.htm)

Notice that after the Split Pane (Vertical Flow) element has been dropped, nodes for a second SplitPane element and its associated AnchorPane elements are added to the Hierarchy panel.

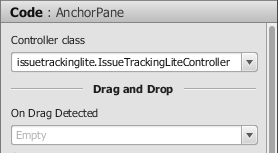
1. In the Hierarchy panel, right-click the newly added row for the SplitPane (Vertical Flow) element and select Fit to Parent from the drop-down list of commands available for the SplitPane element. The vertical flow split pane is adjusted to the size of the right portion of the Content panel.
2. With the SplitPane: (Vertical Flow) selected in the Hierarchy panel, grab the horizontal divider and move it until the percentage displays about 35%. Alternatively, you can select the Properties panel on the right-hand side and type 0.35 in the Divider Positions field

## Bind UI to the Logic

The controller source file, IssueTrackingLiteController.java, will handle the events and actions taken on each element you add to the UI layout that you are building.

1. In the Hierarchy panel, select the top AnchorPane.
2. Click the Code panel in the Inspector panel.  
   Notice that the Controller class text field already has the value issuetrackinglite.IssueTrackingLiteController, as shown in [Figure 10](http://docs.oracle.com/javafx/scenebuilder/1/get_started/jsbpub-get_started.htm#BEIEGCAI). Recall that when you initially created the FXML file in the NetBeans IDE, you had specified that file name as the controller source file to bind with the UI layout you are creating. Setting the controller class file name enables JavaFX Scene Builder to provide you the names of the event handlers and instance variables that are declared in the controller source file.

Figure 10 Add Controller Class

  
[Description of "Figure 10 Add Controller Class"](http://docs.oracle.com/javafx/scenebuilder/1/get_started/img_text/add-controller-class.htm)

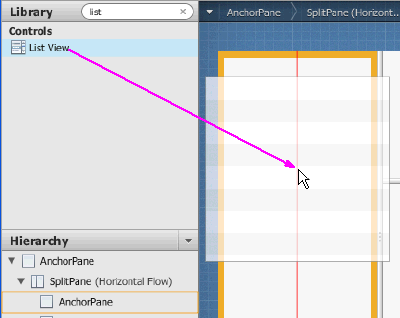
1. Click the drop-down arrow in the Controller class field. If there are other controller class files available, you can simply choose from the list to replace what is already defined, if necessary.

## Add a List View

Now add the List View section, which will display the project information.

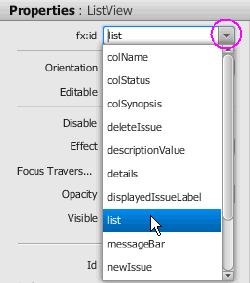
1. In the Library panel's Search box, type list to locate the List View control in the list of UI controls.
2. Drag a List View control from the Library panel and drop it on the left side of the horizontal flow split pane's vertical divider, as shown in [Figure 11](http://docs.oracle.com/javafx/scenebuilder/1/get_started/jsbpub-get_started.htm#BABHFIIH). Notice that in the Hierarchy panel, a row for the listView component is added under the row for the first AnchorPane component of the SplitPane (Horizontal Flow) element.

Figure 11 Add List View Control to SplitPane (Horizontal Flow)

  
[Description of "Figure 11 Add List View Control to SplitPane (Horizontal Flow)"](http://docs.oracle.com/javafx/scenebuilder/1/get_started/img_text/listview-add.htm)

1. Select Modify and then Fit to Parent from the Main menu to adjust the list view element to the size of its parent, the left anchor pane of the horizontal flow split pane.
2. Click the Properties panel in the Inspector panel on the right. In the fx:id field, click the choice button and select list from the drop-down list, as illustrated in [Figure 12](http://docs.oracle.com/javafx/scenebuilder/1/get_started/jsbpub-get_started.htm#BABIADEH).

Figure 12 ListView in Properties Panel

  
[Description of "Figure 12 ListView in Properties Panel"](http://docs.oracle.com/javafx/scenebuilder/1/get_started/img_text/listview-id.htm)

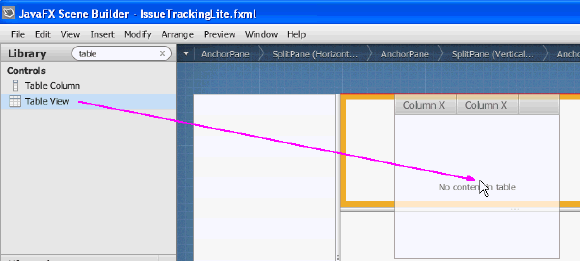
|  |
| --- |
| Note:  The fx:id value corresponds to the name of the controller class' instance variable in which the node will be injected. All the fx:id values must be entered exactly as shown. If they are improperly entered, the IssueTrackingLite sample application will not work correctly. |

## Add a Table View

The Table View control will be used to display the list of issues.

1. In the Controls section of Library panel, select Table View. Drag and drop it to above the horizontal divider of the vertical flow split pane element, as shown in [Figure 13](http://docs.oracle.com/javafx/scenebuilder/1/get_started/jsbpub-get_started.htm#CIHCGFDI).

Figure 13 Add TableView

  
[Description of "Figure 13 Add TableView"](http://docs.oracle.com/javafx/scenebuilder/1/get_started/img_text/tableview-add.htm)

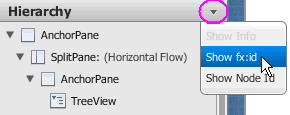
1. Select Modify and then Fit to Parent from the Main menu.
2. In the Properties panel, select table from the fx:id field's drop-down list of available instance variables.
3. Set the properties of the two columns in the table view.
   1. In the Hierarchy panel, select the row for the first TableColumn component under the row for the TableView component. In the Properties panel, select colName from the fx:id field's drop-down list and set the Text property to Name.
   2. Back in the Hierarchy panel, select the row for the second TableColumn component. In the Properties panel, select colStatus from the fx:id field's drop-down list and set the Text property to Status
4. Add another column to the table.
   1. In the Controls section of the Library panel, select the Table Column control. Drag and drop it to the table view in the Content panel. The new column is added to the right of the Status tab.
   2. In the Properties panel, select colSynopsis from the fx:id field's drop-down list of available instance variables, and set the Text property to Synopsis.
5. From the main menu, select View and then Show Sample Data. Notice that the list view and the table view elements in the Content panel are populated with sample data. Select View and then Hide Sample Data to turn off the display of the sample data.

## Add the Details Section

Now set up the area where the details about the issue will be displayed.

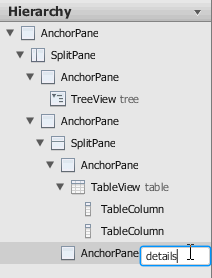
1. In the Hierarchy panel, click the choice button on the top right corner and select Show fx:id, as shown in [Figure 14](http://docs.oracle.com/javafx/scenebuilder/1/get_started/jsbpub-get_started.htm#BEIHFHJF). Notice that the Hierarchy panel now displays the fx:id values next to the elements that have the fx:id property value assigned to them.

Figure 14 Show fx:id Display Mode

  
[Description of "Figure 14 Show fx:id Display Mode"](http://docs.oracle.com/javafx/scenebuilder/1/get_started/img_text/show-fx-id.htm)

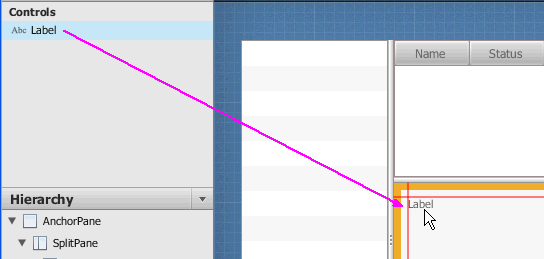
1. Set up the details section.
   1. In the Hierarchy panel, select the AnchorPane component that is currently the last row in the Hierarchy panel. It is right below the row for the TableColumn: colSynopsis element.
   2. Click on the right side of the row for the AnchorPane component to enter the fx:id edit mode. Type details in the inline editor, as shown in [Figure 15](http://docs.oracle.com/javafx/scenebuilder/1/get_started/jsbpub-get_started.htm#BEIIHCEG). This editor is available because the current display mode in the Hierarchy panel is set to Show fx:id. Alternatively, click the Properties panel in the Inspector panel. Select details from the drop-down list of instance variables available for the fx:id field.

Figure 15 Use fx:id Text Editor

  
[Description of "Figure 15 Use fx:id Text Editor"](http://docs.oracle.com/javafx/scenebuilder/1/get_started/img_text/add-fxid.htm)

1. Add a label.
   1. In the Library panel, select Label from the Controls section. Drag and drop it on the upper left corner of the details area, as shown in [Figure 16](http://docs.oracle.com/javafx/scenebuilder/1/get_started/jsbpub-get_started.htm#BABIICJD).

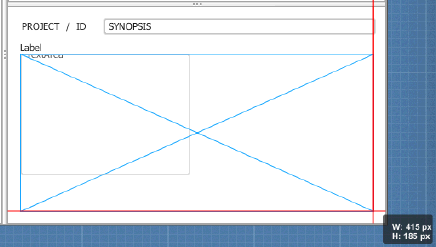
Figure 16 Add Label Element

  
[Description of "Figure 16 Add Label Element"](http://docs.oracle.com/javafx/scenebuilder/1/get_started/img_text/label.htm)

* 1. Click the Properties panel in the Inspector panel. Select displayedIssueLabel in the fx:id field's drop-down list of available instance variables.
  2. Type PROJECT / ID in the Text property field.

1. Add a Text Field control.
   1. From the Library panel, drag and drop a Text Field control to the right side of the Label you just added.
   2. Resize the Text Field UI element so that it occupies the remaining space to the right of the content area.
   3. Double-click the Text Field UI element to get into Edit mode. Type SYNOPSIS in the edit box to replace the default value.
   4. Click the Properties panel in the Inspector panel, select synopsis from the drop-down list of instance variables available for the fx:id field.
2. Group the Label element with the synopsis Text Field element.
   1. In the Hierarchy panel, hold the Ctrl key on the Windows platform or Cmd key on the Mac OS X platform, and select the rows for Label: displayedIssueLabel and TextField: synopsis components.
   2. Select Arrange from the Main menu. Choose Wrap in and then HBox from the menu
3. Add a Label and a TextArea element in the details area.
   1. From the Controls section of the Library panel, drag and drop a Label element to the left side of the details area and below the row occupied by the HBox element you just added. Use the guidelines to position the Label element in line with the HBox element's left side.
   2. Double-click the Label element to get into Edit mode. Type DESCRIPTION: to replace the default value.
   3. Drag and drop a Text Area element below the label that you just added. Grab the handle in the lower right corner of the Text Area element and drag down to enlarge the element so that it takes up the remaining available space in the lower detail area, as shown in [Figure 17](http://docs.oracle.com/javafx/scenebuilder/1/get_started/jsbpub-get_started.htm#CIHGFDGA).
   4. Select the Text Area element in the Content panel. In the Properties panel for TextArea, select descriptionValue from the drop-down list for the fx:id field. Remove the default value of TextArea in the Text box.

Figure 17 Enlarge TextArea

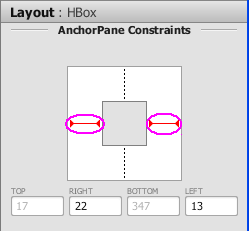
  
[Description of "Figure 17 Enlarge TextArea"](http://docs.oracle.com/javafx/scenebuilder/1/get_started/img_text/textarea-enlarge.htm)

## Anchor the UI Elements in the Details Section

To ensure that the spacing and size of the UI elements that you added adjust properly when the content area is resized, you must modify the anchor pane constraints.

1. In the Hierarchy panel, select the row for the HBox component that contains the Label displayedIssue and the TextField synopsis items. It is directly below the row for the AnchorPane: details element.
2. Click the Layout panel on the right side of the tool window.
3. In the AnchorPane Constraints section, click the left and right dashed anchor lines. After you click the anchor lines, the black dashed lines change into solid red lines, which are circled in [Figure 18](http://docs.oracle.com/javafx/scenebuilder/1/get_started/jsbpub-get_started.htm#BABHEBDF).

Figure 18 Setting the Anchor Lines

  
[Description of "Figure 18 Setting the Anchor Lines"](http://docs.oracle.com/javafx/scenebuilder/1/get_started/img_text/lr-anchors.htm)

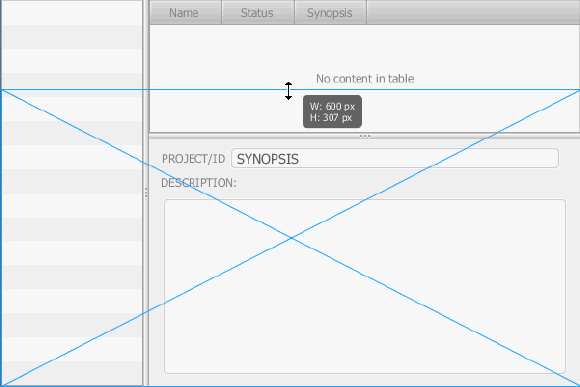
1. In the Hierarchy panel, select the row for the TextField: synopsis component. In the HBox Constraints section of the Layout panel, select ALWAYS from the drop-down list for the Hgrow text field. This setting indicates that the TextField:synopsis element will adjust horizontally when its parent container increases.
2. Select the row for the TextArea: descriptionValue element in the Hierarchy panel. In the Layout panel, click the left, top, right, and bottom dashed anchor lines in the graphical section of the AnchorPane Constraints section so that the black dashed lines become solid red lines.
3. In the Hierarchy panel, select the row for the Label: displayedIssueLabel component. In the Layout panel, set the value of Min Width to USE\_PREF\_SIZE. This setting will keep the labels HBox element still visible when the size of the application window is so reduced that not all of the UI elements can be displayed.

## Create the Toolbar

Add a toolbar to the top portion of the content area. It will contain an image file, three buttons, and a message bar.

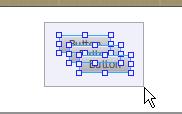
1. In the Hierarchy panel, change the display setting by clicking the choice box in the right corner and selecting Show Info.
2. Select the row for the SplitPane (Horizontal Flow) component. You may have to resize the Hierarchy panel to see the full node Id of the SplitPane.
3. In the Content panel, grab the top middle handle of the split pane and drag it down to make room for a toolbar above the Content area, as illustrated by [Figure 19](http://docs.oracle.com/javafx/scenebuilder/1/get_started/jsbpub-get_started.htm#CIHIFEEC).

Figure 19 Make Room for the Toolbar

  
[Description of "Figure 19 Make Room for the Toolbar"](http://docs.oracle.com/javafx/scenebuilder/1/get_started/img_text/drag-for-toolbar.htm)

1. Select File, Import, and then Media from the Main menu and select IssueTrackingLite.png from <javafx-scenebuilder-samples-1.0-beta-install-dir>/IssueTrackingLite/src/issuetrackinglite folder.
2. Move the image to the top left corner of the toolbar area. Adjust the size of the toolbar area if it is too small for the image being inserted.
3. Add three buttons to the toolbar.
   1. From the Library panel, drag and drop a Button element on the right side of the toolbar.
   2. Duplicate this button by selecting Edit and then Duplicate from the Main menu twice, to add two more new buttons to the toolbar.
4. Select all three button controls by continuously holding down the left mouse button and dragging the cursor over the three buttons, as shown in [Figure 20](http://docs.oracle.com/javafx/scenebuilder/1/get_started/jsbpub-get_started.htm#BABBICBE). When you release the left mouse button, all three buttons should be selected.

Figure 20 Marquee Selection of the Three Buttons

  
[Description of "Figure 20 Marquee Selection of the Three Buttons"](http://docs.oracle.com/javafx/scenebuilder/1/get_started/img_text/toolbar-buttons.htm)

1. Select Arrange, Wrap in, and then select HBox from the Main menu. The buttons are arranged in a row with even spacing between them. Move the HBox so that it's right edge is closer to the right side of the Toolbar.
2. Set the anchorLayout and spacing for the buttons.
   1. In the Hierarchy panel, select the row for the HBox component that was just added, if it is not already selected.
   2. Click the Layout panel in the Inspector panel and in the AnchorPane Constraints section, click the top and right dashed anchors lines.
   3. Click the Properties panel and in the Spacing field, type 15 to replace the default value of 5. Notice that the spacings between the buttons are increased.
3. Edit the buttons' details.
   1. Select the button on the right side of the toolbar. In the Properties panel, select deleteIssue from the fx:id property's drop-down list. Type Delete in the Text field to replace the default text.
   2. Double-click the middle button to get into Edit mode. Modify the default text property by typing Save in the edit box. In the Properties panel, select saveIssue in the drop-down list for the fx:id property for the middle button.
   3. In the toolbar, double-click the leftmost button to get into Edit mode and replace the button's default text with New. In the Properties panel, select newIssue in the drop-down list for the fx:id property for the New button.
4. Bind each button to an event handler that will manage the action to take when the button is clicked.
   1. Select the New button. Click the Code panel of the Inspector panel
   2. In the On Action field, select #newIssueFired from the drop-down list of event handlers available in the controller source file. The leading # symbol tells your application to look for the newIssueFired method in the controller source code. The method used must be public, return void, and take an ActionEvent as parameter. Each time the New button is clicked, the public method newIssueFired(ActionEvent):void, which is defined in the controller source code, will be executed.
   3. Select the Save button and in the On Action field, set the value to #saveIssueFired.
   4. Select the Delete button and in the On Action field, set the value to #deleteIssueFired.
5. Add a message bar.
   1. In the Library panel, select Label from the Controls section and drop it in the toolbar area in between the imported image and the New button. Resize and align the Label element so that it fills most of the space between the imported image and the New button. Use the arrow keys to make fine alignment adjustments. Press the arrow keys to move the element by a single pixel and press the Shift key + arrow keys to move the element by ten pixels.
   2. Click the Properties panel and set the fx:id property to messageBar. Remove the default Text value so that the message bar is only visible when a message is displayed.
6. Select File and then Save from the Main menu to save your work.

## Preview the UI

You can preview the UI work that you have done so far.

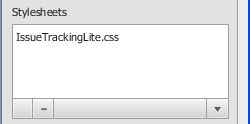
1. From the Main menu, select Preview, and then select Preview in Window.
2. Resize the window multiple times to ensure that the buttons in the toolbar and the text area resize appropriately when the window is resized.
3. To stop viewing the preview, close the window or from the Main menu select Preview and then Hide Preview Window.

## Working with a Style Sheet File

You can apply CSS on the UI that you just built to change the look and feel of the layout. In this example, you use a style sheet file that has been provided with the IssueTrackingLite sample.

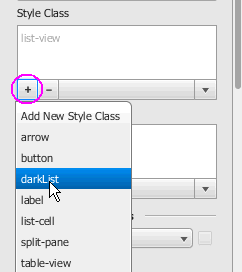
1. Verify that the CSS resource file that is bundled with the IssueTrackingLite sample is already set. In the Hierarchy panel, select the row for the topmost AnchorPane container. Click the Properties panel and in the Stylesheets list view of the CSS section, notice that the IssueTrackingLite.css style sheet is already set, as shown in [Figure 21](http://docs.oracle.com/javafx/scenebuilder/1/get_started/jsbpub-get_started.htm#BEIHBJDF). This is the style sheet that was set when you created the FXML file using the NetBeans IDE.

Figure 21 Adding a Style Sheet File

  
[Description of "Figure 21 Adding a Style Sheet File"](http://docs.oracle.com/javafx/scenebuilder/1/get_started/img_text/stylesheet.htm)

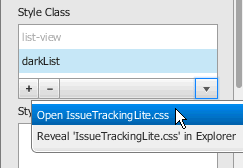
1. Now use a style class for one of the elements in the Content panel.
   1. In the Hierarchy panel, select the row for the ListView component.
   2. In the Properties panel, click the button with the + in the Style Class list. Select darkList as shown in [Figure 22](http://docs.oracle.com/javafx/scenebuilder/1/get_started/jsbpub-get_started.htm#BABDBIAA). Notice that the appearance of the Text View element's background in the Content panel changed to the black color.

Figure 22 Adding a StyleClass to descriptionValue

  
[Description of "Figure 22 Adding a StyleClass to descriptionValue "](http://docs.oracle.com/javafx/scenebuilder/1/get_started/img_text/styleclass.htm)

* 1. In the Style Class list view again, the darkList should be selected already. Click the choice button with the down arrow and select Open IssueTrackingLite.css from the list, as shown in [Figure 23](http://docs.oracle.com/javafx/scenebuilder/1/get_started/jsbpub-get_started.htm#BABBACGD). The IssueTracking.css file is displayed in the default editor defined for the CSS file type. You can make edits to the file, if you want, and save the file. The changes are immediately applied. Exit out of the editor window.

Figure 23 Editing the Resource File

  
[Description of "Figure 23 Editing the Resource File"](http://docs.oracle.com/javafx/scenebuilder/1/get_started/img_text/edit-styleclass.htm)

1. From the Main menu, select File, and then select Save.

Congratulations! You just completed building the FXML layout for a JavaFX application using JavaFX Scene Builder.

## Compile and Run the Application

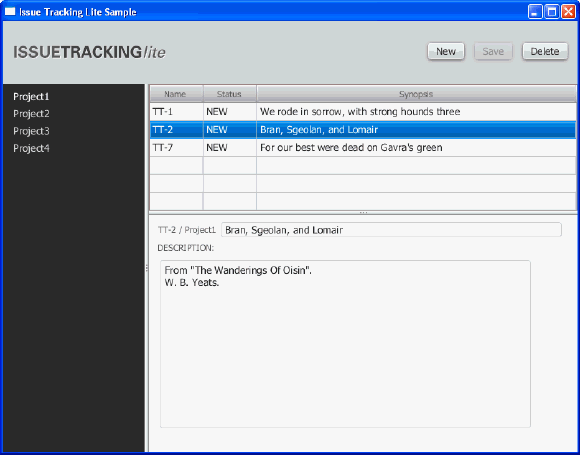
After binding the logic to the UI layout that you just built, you are ready to see your application in action. You can use either the NetBeans IDE or the Ant utility to compile and run the IssueTrackingLite application.

### Use NetBeans IDE

Use the latest NetBeans IDE 7.2 to build and run the NetBeans project in which you saved the IssueTracking.fxml file.

1. Back in the NetBeans IDE 7.2 Beta window, right-click the IssueTrackingLite project node in the Projects window and select Run.  
   The IDE compiles the project and if no errors are encountered, it displays an application similar to [Figure 24](http://docs.oracle.com/javafx/scenebuilder/1/get_started/jsbpub-get_started.htm#BABICGJI). Click the image to enlarge it.

Figure 24 Completed UI for Issue Tracking Sample (Click image to enlarge.)

[](http://docs.oracle.com/javafx/scenebuilder/1/get_started/finished-ui-big.gif)  
[Description of "Figure 24 Completed UI for Issue Tracking Sample"](http://docs.oracle.com/javafx/scenebuilder/1/get_started/img_text/finished-ui.htm)

1. In the list view on the left, select Project1 and in the table view on the right, select the row with the TT-2 in the Name column. Data is displayed in the details area, as shown in [Figure 24](http://docs.oracle.com/javafx/scenebuilder/1/get_started/jsbpub-get_started.htm#BABICGJI).
2. If you encounter any errors, look at the Output window and determine the possible causes of the errors. Some troubleshooting ideas are as follows:
   * Check that all of the fx:id values are entered correctly. The fx:id values in the FXML layout must match the values that the controller source classes expect.
   * Check that you entered the method name correctly in the Event binding section.
   * On the Mac OS X platform, check that you have properly configured a JavaFX 2.2 enabled platform in your NetBeans IDE 7.2 Beta installation. At the present, the JavaFX-enabled platform is not automatically created for you on the Mac OS X platform.

### Use the Ant Utility

If you choose not to run the application in NetBeans IDE, you can use the Ant utility (version 1.8 or above) to build and run the application on the command line. Type a command similar to the one in Example 1.

Example 1 Ant Command to Run the Application

ant -Dplatforms.Default\_JavaFX\_Platform.home=<JAVA\_HOME>

-Djavafx.sdk=/<JAVAFX\_22\_SDK\_HOME>

-Djavafx.runtime=/<JAVAFX\_22\_RUNTIME\_HOME>

-f <JavaFX\_App\_Name>/build.xml

<TARGET>

In this example, the main values for <TARGET> are clean, jar, and run. You can set <TARGET> with the value of -projecthelp to get a list of available targets. For example, to run the IssueTrackingLite application on the Windows platform, type something similar to the command in Example 2.

Example 2 Using Ant to Run IssueTrackingLite on a Windows Platform

ant -Dplatforms.Default\_JavaFX\_Platform.home="C:\Program Files\Java\jdk1.7.0\_03"  
 -Djavafx.sdk="C:\Program Files\Oracle\JavaFX 2.2 SDK"   
 -Djavafx.runtime="C:\Program Files\Oracle\JavaFX 2.2 Runtime"   
 -f IssueTrackingLite\build.xml run

To run the IssueTrackingLite application on a MacOS platform, type something similar to the command shown in Example 3.

Example 3 Using Ant to Run IssueTrackingLite on a MacOS Platform

ant -Dplatforms.Default\_JavaFX\_Platform.home=  
/System/Library/Frameworks/JavaVM.framework/Versions/1.6/Home  
 -Djavafx.sdk=/Applications/javafx-sdk2.2.0-beta  
 -Djavafx.runtime=/Applications/javafx-sdk2.2.0-beta/rt  
 -f IssueTrackingLite/build.xml run