

## **Oracle® JavaFX**

Getting Started with JavaFX Scene Builder

Release 1.0

**E25448-01**

August 2012

Oracle JavaFX Getting Started with JavaFX Scene Builder, Release 1.0

E25448-01

Copyright © 2012 Oracle and/or its affiliates. All rights reserved.

Primary Author: Cindy Castillo

Contributor: Yves Joan

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle America, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

---

---

# Contents

## 1 Prepare for This Tutorial

## 2 Open the Sample FXML File

Use NetBeans IDE Projects Window .....	2-1
Use JavaFX Scene Builder Open Command.....	2-2

## 3 Create the FXML File and the Base Panes

Use NetBeans IDE New Wizard .....	3-1
Use JavaFX Scene Builder New Command .....	3-3
Set the Style Class File .....	3-4
Create the Base Panes .....	3-4

## 4 Bind the UI to the Application Logic

## 5 Add the List and Table Views

Add a List View .....	5-1
Add a Table View .....	5-2

## 6 Create the Details Section

Add the UI Components for the Details Section .....	6-1
Anchor the UI Elements in the Details Section.....	6-3

## 7 Add the Toolbar

## 8 Use a Style Sheet and Preview the UI

Preview the UI.....	8-1
Use a Style Sheet .....	8-1

## 9 Compile and Run the Application

Use NetBeans IDE .....	9-1
Use the Apache Ant Utility.....	9-2



# Part I

---

## About This Tutorial

This Getting Started tutorial presents the step-by-step creation of a simple issue-tracking application using the JavaFX Scene Builder tool. It shows you how to quickly build the user interface (UI) for a JavaFX application, connect it to the Java source code, and handle the interaction between the data and the user interface. For the purpose of this tutorial, you will use a NetBeans project named `IssueTrackingLite` to illustrate the integration between NetBeans IDE and JavaFX Scene Builder. Note that the use of JavaFX Scene Builder is not dependent on any IDE. This tutorial also includes information that steps you through the creation of the `IssueTrackingLite` FXML layout without the use of NetBeans IDE.

This document contains the following topics:

- [Prepare for This Tutorial](#)
- [Open the Sample FXML File](#)
- [Create the FXML File and the Base Panes](#)
- [Bind the UI to the Application Logic](#)
- [Add the List and Table Views](#)
- [Create the Details Section](#)
- [Add the Toolbar](#)
- [Use a Style Sheet and Preview the UI](#)
- [Compile and Run the Application](#)



---

## Prepare for This Tutorial

JavaFX Scene Builder is a design tool for the JavaFX platform. It allows 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 UI is automatically generated. JavaFX Scene Builder provides a straightforward interface that can help you quickly prototype interactive applications that connect GUI components to the application logic.

---

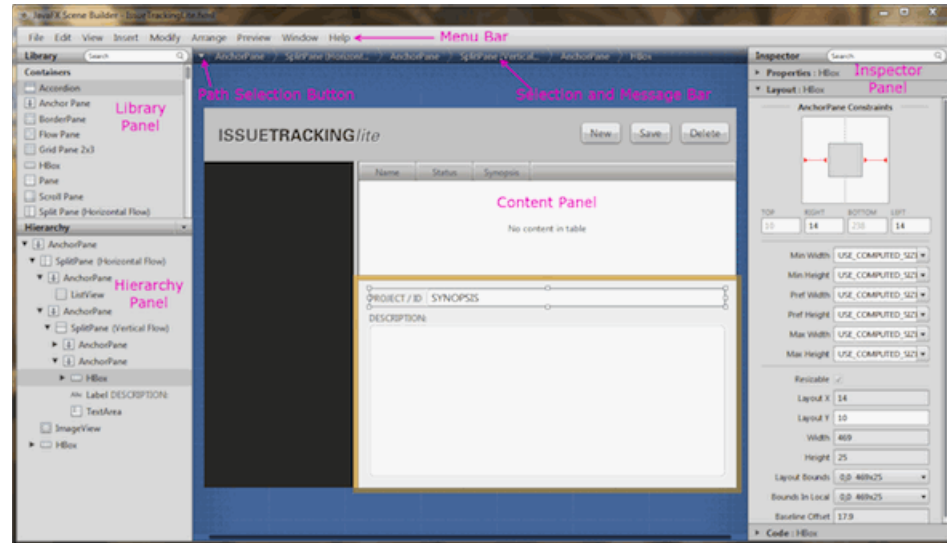
**Note:** JavaFX Scene Builder does not have dependency on any particular IDE. You can run it as a standalone tool to create your UI layout and edit the resulting FXML file using a text editor of your choice. If you choose to work with this tutorial outside of NetBeans IDE, there are sections in this tutorial that highlight what you need to do to connect the layout that you build to the sample application's Java source code, apply the style sheet, and run the sample application using the Apache Ant utility.

---

Use the following requirements and recommendations before continuing with this tutorial:

1. **(Required) Install all the required software** before you use JavaFX Scene Builder. See [JavaFX Scene Builder Installation Guide](#) for more details.
2. **(Required) Download the [IssueTrackingLite.zip](#) sample file** included with this tutorial and extract its contents. The IssueTrackingLite folder you extract contains a completed version of the FXML layout that you will build using this tutorial. The issue-tracking system enables you to query existing project issues, modify them, or add new issue.
3. **(Required) Use [JavaFX Scene Builder User Guide](#)** to familiarize yourself with the JavaFX Scene Builder user interface that is shown in [Figure 1-1](#). Click the image to see a larger version.
4. **(Recommended) Install NetBeans IDE 7.2**, which is used in this tutorial to illustrate the integration between NetBeans IDE and JavaFX Scene Builder. As mentioned earlier, you can also create the IssueTrackingLite UI layout without having to use NetBeans IDE.
5. **(Recommended) Familiarize yourself with the JavaFX concepts** by reading the available articles on the JavaFX Documentation web site at <http://docs.oracle.com/javafx>. In particular, learn about the FXML concepts by reading [Getting Started with FXML](#) and get familiar with layouts using [Working With Layout in JavaFX](#).

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





---

## Open the Sample FXML File

After you have familiarized yourself with the JavaFX concepts and the JavaFX Scene Builder user interface, begin building an FXML layout using JavaFX Scene Builder. Open the completed `IssueTrackingLite.fxml` file to see what the finished `IssueTrackingLite` layout looks like. Use one of the following methods:

- Step through [Use NetBeans IDE Projects Window](#) to open the `IssueTrackingLite` sample NetBeans project and double-click the node for the `IssueTrackingLite.fxml` file to invoke JavaFX Scene Builder.
- Follow [Use JavaFX Scene Builder Open Command](#) to open the `IssueTrackingLite.fxml` sample file.

The controller source file and the Cascading Style Sheet (CSS) file used in this tutorial are already provided with the `IssueTrackingLite` sample NetBeans project. These files are in the same project folder that will contain the new FXML file.

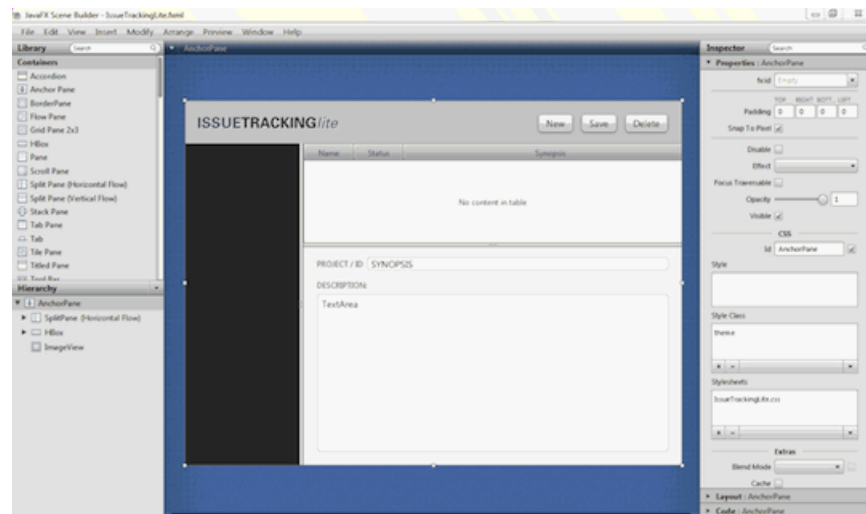
### Use NetBeans IDE Projects Window

Open the completed `IssueTrackingLite.fxml` sample layout file using NetBeans IDE's Projects window:

1. Start NetBeans IDE on your Windows platform by double-clicking the **NetBeans IDE 7.2 shortcut**, or select **Start**, then **Programs**, then **NetBeans**, and finally, **NetBeans IDE 7.2**. On a Mac OS X platform, double-click the **NetBeans IDE 7.2** application icon.
2. From the Menu bar, choose **File** and then **Open Project**.
3. From the Open Project dialog box, go to where you extracted the `IssueTrackingLite.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 2-1](#).

**Figure 2–1** Completed *IssueTrackingLite.fxml* Opened in JavaFX Scene Builder Window  
(Click image to enlarge.)



5. Save the file with a different name so that you can create your own FXML layout file. From the Menu bar, choose **File** and then **Save As**. Enter *IssueTrackingLiteComplete.fxml* in the **File Name** text field and click **Save**. Keep the JavaFX Scene Builder window for this file open so that you can use it to compare with the version of the layout you are about to create.
6. 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**.

## Use JavaFX Scene Builder Open Command

Open the completed sample FXML file directly from the JavaFX Scene Builder window.

1. Start JavaFX Scene Builder on your Windows platform by double-clicking the **JavaFX Scene Builder 1.0 shortcut**, or select **Start**, then **Programs**, then **JavaFX Scene Builder**, and finally, **JavaFX Scene Builder 1.0**. On a Mac OS X platform, double-click the **JavaFX Scene Builder 1.0** application icon.
2. From the Menu bar, choose **File** and then **Open**.
3. From the Open FXML dialog box, go to the folder in which you extracted the *IssueTrackingLite.zip* file and open the **IssueTrackingLite**, **src**, and **issuetrackinglite** folders.
4. Double-click the **IssueTrackingLite.fxml** file.

The *IssueTrackingLite.fxml* file is opened in the Content panel, as shown in [Figure 2–1](#).

5. Save the file with a different name so that you can create your own FXML layout file using the original file name.
  - a. From the Menu bar, choose **File** and then **Save As**.
  - b. Enter *IssueTrackingLiteComplete.fxml* in the **File Name** text field and click **Save**.

---

## Create the FXML File and the Base Panes

The FXML layout that you are about to build for the IssueTrackingLite application is an interface that enables you to query existing project issues, modify them, or add new issues. Build your own IssueTrackingLite user interface:

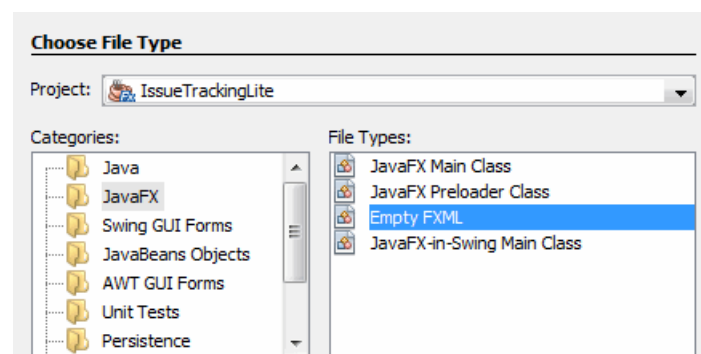
1. Create the new FXML file by following the steps in either [Use NetBeans IDE New Wizard](#) or [Use JavaFX Scene Builder New Command](#).
2. [Set the Style Class File](#) to use for the entire layout.
3. [Create the Base Panes](#) for the layout.

### Use NetBeans IDE New Wizard

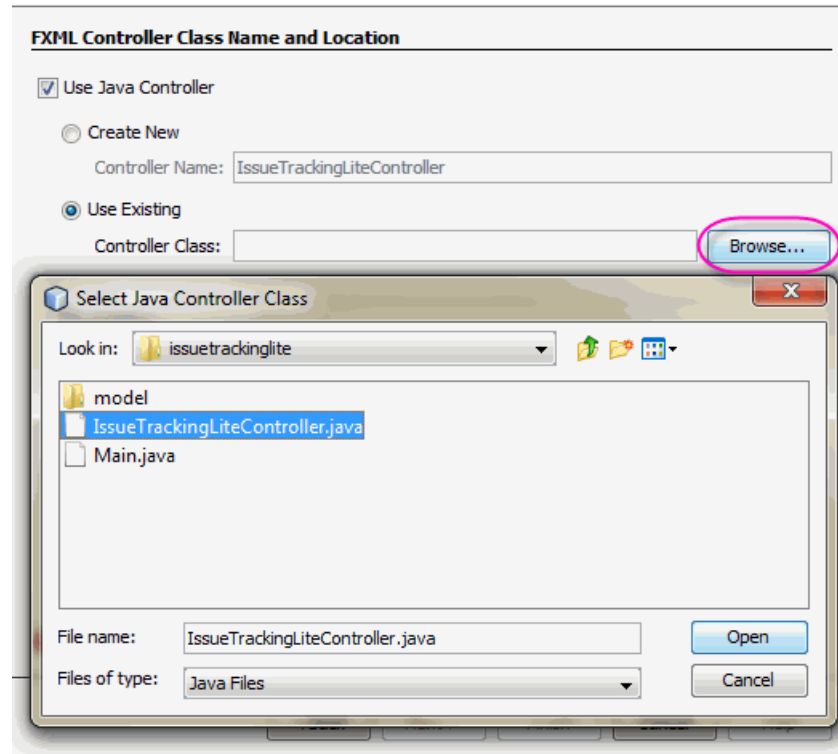
Create a new empty FXML file using the NetBeans IDE New wizard.

1. From the Projects window of the IDE, right-click the **issuetrackinglite** node, choose **New**, and then **Other**.
2. In the New File dialog box, select the **JavaFX** category and select the **Empty FXML** file type, as shown in [Figure 3–1](#). Click **Next**.

**Figure 3–1** Choose Empty FXML File Type



3. In the New Empty FXML dialog box, enter `IssueTrackingLite` in the **FXML Name** text field. This is the controller source file name used in the application when loading the FXML file. Click **Next**.
4. Select the **Use Java Controller** checkbox and the **Use Existing** option.
5. Click **Browse** and in the Select Java Controller Class dialog box, open the `IssueTrackingLiteController.java` file, as shown in [Figure 3–2](#). Click **Next**.

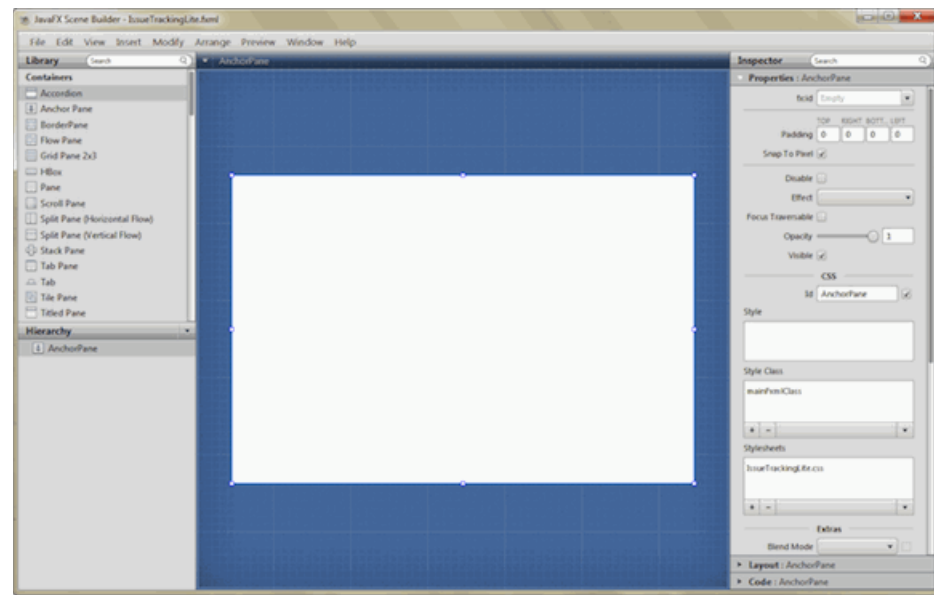
**Figure 3–2 Use the Existing Java Controller Class File**

6. Select the **Use Cascading Style Sheet** checkbox and the **Use Existing** option.
7. Click **Browse** and in the Select CSS File dialog box, open the **IssueTrackingLite.css** file. Click **Finish**.

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

8. 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 3–3](#). 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 3–3** JavaFX Scene Builder Main Window at Startup (Click image to enlarge.)

**Tip:** Enlarge the Content panel to get a larger working area. Select the row for the **AnchorPane** element in the Hierarchy panel. Click the **Layout** section of the Inspector panel and enter 800 in the **Pref Width** field. Enter 600 in the **Pref Height** field and press the **Enter** key. The AnchorPane container is resized to 800 by 600 pixels. Alternatively, you can drag the handle at the bottom right corner of the AnchorPane container to increase its size. Resize the JavaFX Scene Builder window so you are able to view the entire AnchorPane container.

## Use JavaFX Scene Builder New Command

If you chose not to use NetBeans IDE to complete this tutorial, use the following steps to create a new empty FXML file using JavaFX Scene Builder New command.

1. From the Menu bar, choose **File** and then **New**.

A new JavaFX Scene Builder window appears with an empty FXML file opened in the Content panel, as shown in [Figure 3–3](#). 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

2. Set the CSS file to use for your new FXML layout.
  - a. In the Hierarchy panel, select the root **AnchorPane** element and click the **Properties** section of the Inspector panel.
  - b. Click the button with the plus sign (+) symbol in the Stylesheets list view of the CSS subsection.
  - c. In the Add Style Sheet dialog window, go to the **issuetrackinglite** folder and select the **IssueTrackingLite.css** file. Click **Open**.
3. Save the new FXML file before you continue with the rest of the tutorial.
  - a. From the Menu bar, choose **File** and then **Save As**.

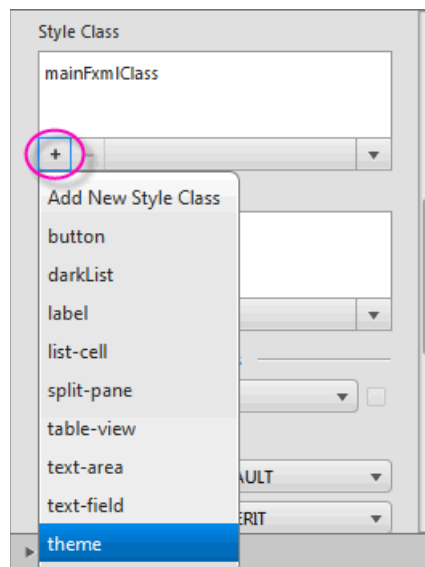
- b. In the Save FXML File As dialog window, go to the IssueTrackingLite folder, and then to the src and issuetrackingle folders.
- c. Enter `IssueTrackingLite.fxml` in the File name text field and click **Save**.
- d. Select **Yes** when asked if you want to replace the existing `IssueTrackingLite.fxml` file.

## Set the Style Class File

Set the style class to use for the entire layout.

1. In the JavaFX Scene Builder Hierarchy panel, select the row for the top **AnchorPane** container, if it is not already selected.
2. In the **Properties** section of the Inspector panel, click the button with the plus sign (+) in the **Style Class** list view. The list of style classes is retrieved from the `IssueTrackingLite.css` file. Select **theme**, as shown in [Figure 3-4](#).

**Figure 3-4** Set the Style Class for the Layout

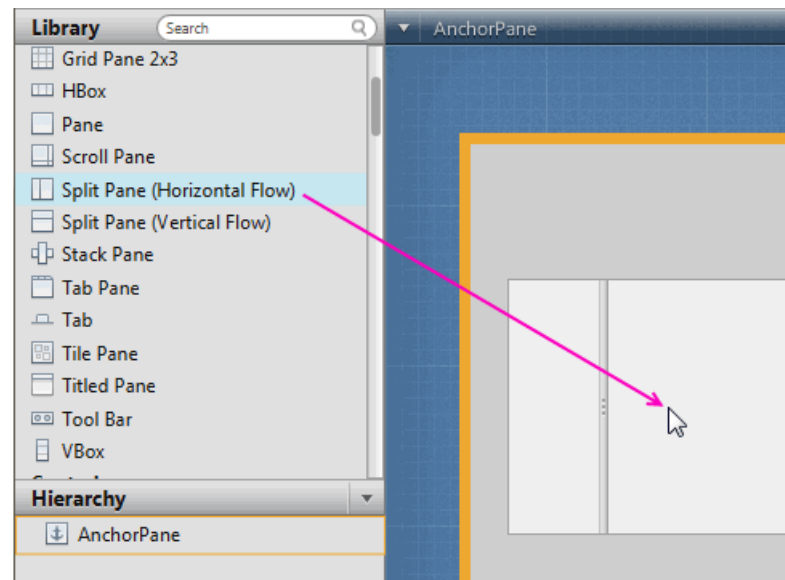


## Create the Base Panes

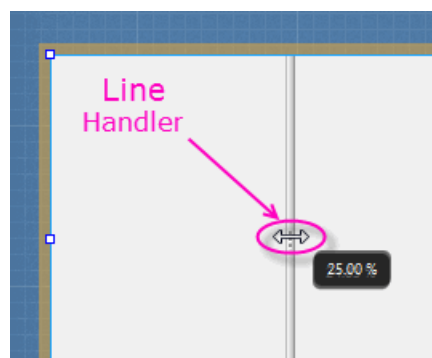
Define the base layout panes to define the different sections of your UI layout.

1. From the Library panel, select **Split Pane (Horizontal Flow)**. Drag it to the Content panel and drop it inside the Anchor Pane element, as shown in [Figure 3-5](#).

After you drop the Split Pane object, the Hierarchy panel is updated to show the Split Pane (Horizontal Flow) element containing the two AnchorPane elements.

**Figure 3–5 Add Split Pane (Horizontal Flow)**

2. With the **Split Pane (Horizontal Flow)** selected in the Content panel, select **Modify** and then **Fit to Parent** from the Menu bar. This action resizes the node to fill the space occupied by its container parent, the topmost **AnchorPane** element.
3. From the Menu bar, choose **File** and then **Save**.  
This action saves your work in the src folder for the IssueTrackingLite sample NetBeans project,  
<install-dir>/IssueTrackingLite/src/issuetrackinglite/IssueTrackingLite.fxml.  
Perform this Save action frequently to preserve your work.
4. With **Split Pane (Horizontal Flow)** selected in the Hierarchy panel, drag the vertical divider until the percentage displays about 25%, as illustrated in [Figure 3–6](#). Alternatively, in the Properties section of the Inspector panel, enter 0.25 in the **Divider Positions** field.

**Figure 3–6 Move Vertical Divider**

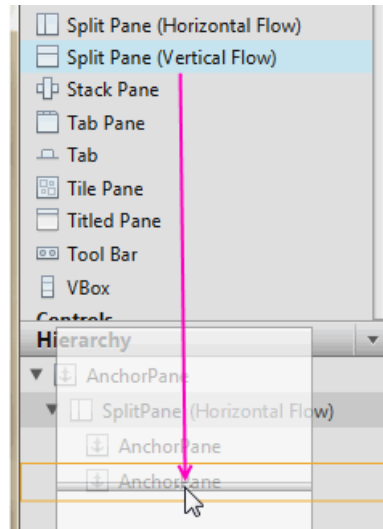

---

**Note:** From the Menu bar, choose **Edit** and then **Undo** to undo actions when necessary.

---

5. To add the Split Pane (Vertical Flow) sections, drag **Split Pane (Vertical Flow)** from the Library panel and drop it to the Hierarchy panel. Ensure that you make the drop when the second AnchorPane element inside the Split Pane (Horizontal Flow) shows an orange highlight, as shown in [Figure 3-7](#).

**Figure 3-7 Add Split Pane (Vertical Flow)**



Notice that after the Split Pane (Vertical Flow) element is dropped, it is added to the Hierarchy panel with its AnchorPane children.

6. In the Hierarchy panel, right-click the newly added row for the **Split Pane (Vertical Flow)** element and choose **Fit to Parent** from the contextual menu.
7. With the **Split Pane: (Vertical Flow)** selected in the Content panel, drag the horizontal divider until the percentage displays about 35%. Alternatively, click the Properties section of the Inspector panel, enter 0.35 in the **Divider Positions** field.



---

## Bind the UI to the Application Logic

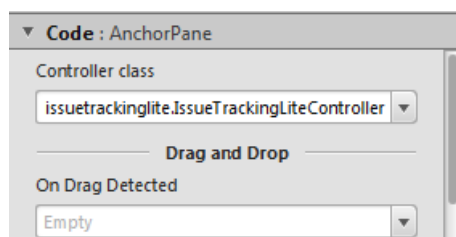
The controller source file, `IssueTrackingLiteController.java`, will manage the events and actions taken on each element you add to the FXML UI layout that you are building.

1. In the Hierarchy panel, select the row for the root (top) **AnchorPane** element.
2. Click the **Code** section of the Inspector panel.  
If you are using NetBeans IDE to complete this tutorial, notice that the Controller class text field already has the value `issuetrackinglite.IssueTrackingLiteController`, as shown in [Figure 4–1](#). When you initially created the FXML file in NetBeans IDE, you specified that file name as the controller source file to bind with the UI layout you are creating.

If you chose not to use NetBeans IDE to complete this tutorial, enter `issuetrackinglite.IssueTrackingLiteController` in the **Controller class** text field.

Setting the controller class file name enables JavaFX Scene Builder to provide you with the names of the event handlers and instance variables that are declared in the controller source file

**Figure 4–1** Add Controller Class



3. Click the drop-down arrow in the **Controller class** field. If there are other controller class files available, it will be listed in a drop-down list and you can choose from the list to replace what is already defined, if necessary.



---

## Add the List and Table Views

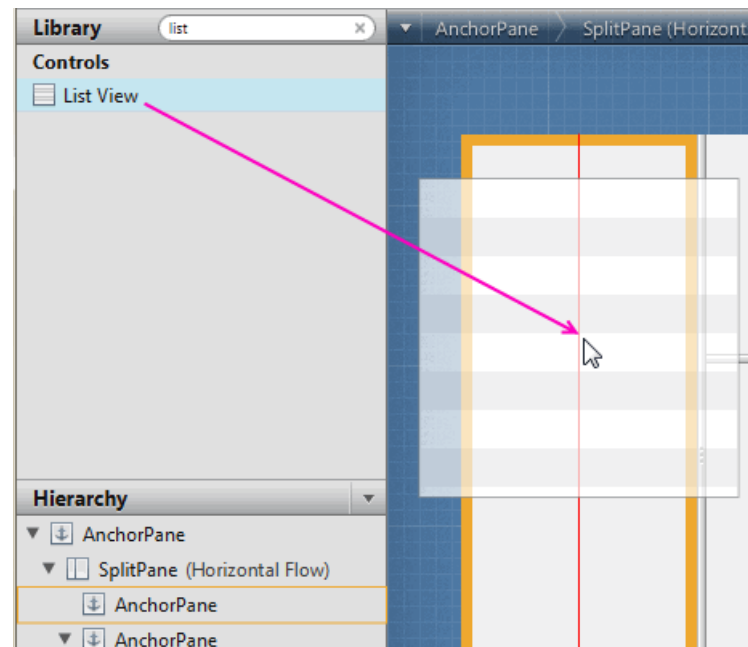
For the next task in this tutorial, you will add List View and Table View components to your UI layout.

### Add a List View

Add the List View section, which will display the issue's project information.

1. In the Library panel's **Search** box, enter `list` to locate the List View control in the list of UI controls.
2. Drag the **List View** control from the Library panel and drop it on the left side of the divider line for Split Pane (Horizontal Flow), as shown in [Figure 5-1](#). Notice that after the listView component is dropped, it is shown in the Hierarchy panel as a child of the first AnchorPane element inside the Split Pane (Horizontal Flow) element.

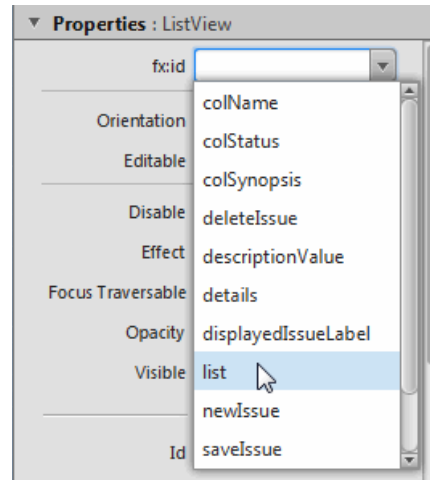
**Figure 5-1** Add List View Control to SplitPane (Horizontal Flow)



3. From the Menu bar, choose **Modify** and then **Fit to Parent**.

- Click the **Properties** section of the Inspector panel. In the **fx:id** field, select the choice button and select **list** from the drop-down list, as illustrated in [Figure 5–2](#).

**Figure 5–2 Set fx:id for ListView Control**




---

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

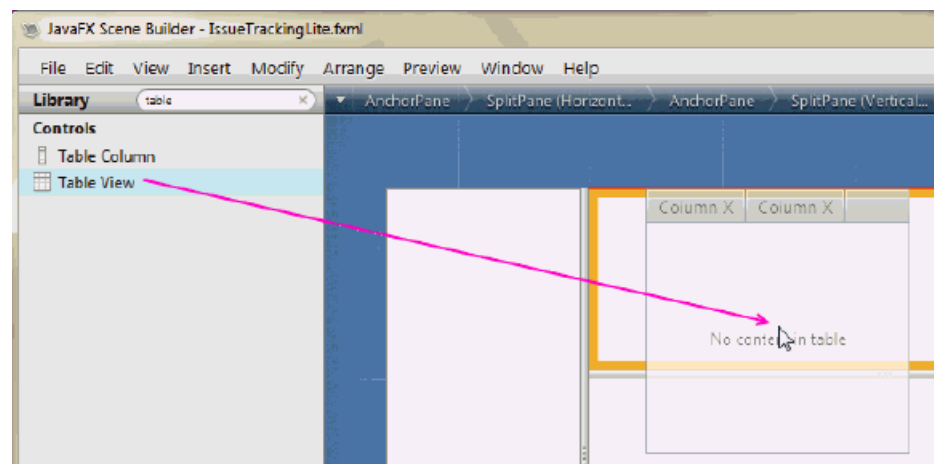
---

## Add a Table View

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

- In the Controls section of the Library panel, select **Table View**. Drag and drop it above the Split Pane (Vertical Flow) element's divider line, as shown in [Figure 5–3](#).

**Figure 5–3 Add the Table View Control**



- From the Menu bar, choose **Modify** and then **Fit to Parent**.

3. Click the **Properties** section of the Inspector panel and select **table** from the **fx:id** field's drop-down list of available instance variables.
4. Set the properties of the two columns in the table view.
  - a. In the Hierarchy panel, select the row for the first **TableColumn** component under the row for the **TableView** component. In the **Properties** section of the Inspector panel, select **colName** from the **fx:id** field's drop-down list and change the default value for the **Text** property to **Name**.
  - b. Back in the Hierarchy panel, select the row for the second **Table Column** component. In the **Properties** section of the Inspector panel, select **colStatus** from the **fx:id** field's drop-down list and set the **Text** property to **Status**
5. Add another column to the table.
  - a. Click the **Controls** section of the Library panel and select the **Table Column** control. Drag and drop it inside the table view in the Content panel. The new column is added to the right of the Status tab.
  - b. In the **Properties** section of the Inspector panel, select **colSynopsis** from the **fx:id** field's drop-down list of available instance variables, and set the **Text** property to **Synopsis**.
6. From the Menu bar, choose **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. Choose **View** and then **Hide Sample Data** from the Menu bar to turn off the display of the sample data.



---

## Create the Details Section

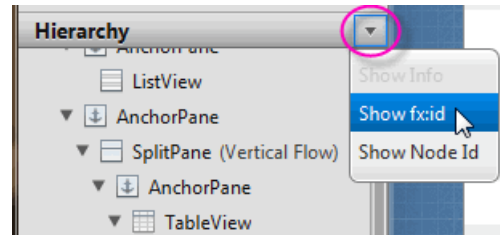
For this part of the tutorial, set up the area where the issue's details will be displayed.

### Add the UI Components for the Details Section

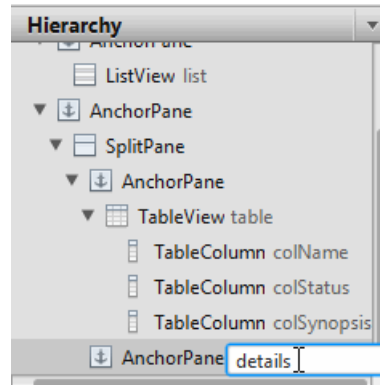
Add the UI components to create the section that will display the details about the issue that you are creating or modifying in the IssueTrackingLite application.

1. Click the menu button on the top right corner of the Hierarchy panel and choose **Show fx:id**, as shown in [Figure 6-1](#). 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.

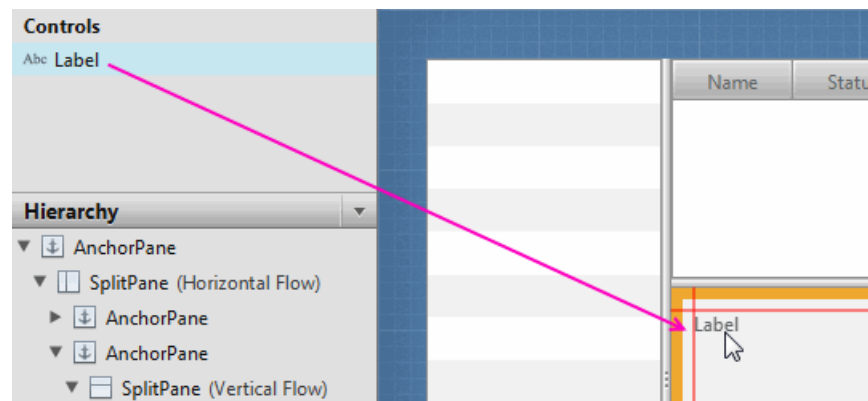
**Figure 6-1** Show fx:id Display Mode



2. Set up the details section.
  - a. In the Hierarchy panel, select **AnchorPane**, shown in the last row. It is right below the row for the TableColumn: colSynopsis element.
  - b. Click the right side of the row for the AnchorPane element to enter the fx:id edit mode. Enter `details` in the inline text editor for the fx:id text, as shown in [Figure 6-2](#). This editor is available because the current display mode in the Hierarchy panel is set to Show fx:id. Alternatively, click the **Properties** section of the Inspector panel. Select **details** from the drop-down list of instance variables available for the fx:id field.

**Figure 6–2 Use fx:id Inline Text Editor**

3. Add a label.
  - a. In the Controls section of the Library panel, drag **Label** and drop it on the upper left corner of the details area, as shown in [Figure 6–3](#).

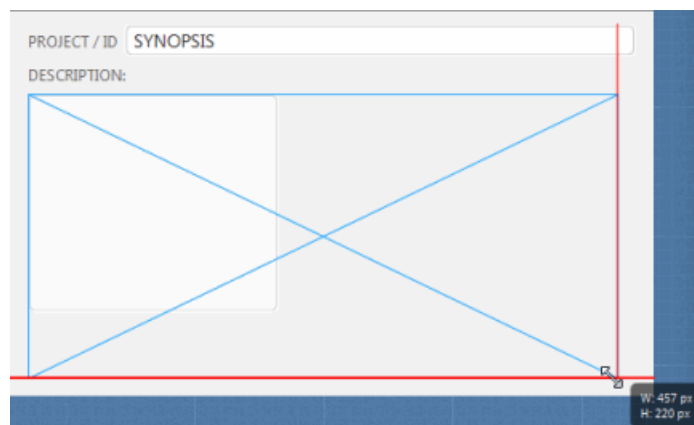
**Figure 6–3 Add Label Element**

- b. In the **Properties** section of the Inspector panel, select **displayedIssueLabel** in the **fx:id** field's drop-down list of available instance variables.
  - c. Enter **PROJECT / ID** in the **Text** property field to replace the default value.
4. Add a Text Field control.
  - a. From the Library panel, drag a **Text Field** control and drop it to the right side of the Label you just added.
  - b. Resize the **Text Field** element so that it occupies the remaining space to the right.
  - c. Double-click the **Text Field** element in the Content panel to enter Edit mode. Enter **SYNOPSIS** in the edit box.
  - d. In the **Properties** section of the Inspector panel, select **synopsis** from the drop-down list of instance variables available for the **fx:id** field.
5. Group the Label element with the synopsis Text Field element.



- a. In the Hierarchy panel, hold the Ctrl key (Windows) or Cmd key (Mac) and select the rows for the **Label displayedIssueLabel** and **TextField synopsis** components.
  - b. From the Menu bar, choose **Arrange**, then **Wrap in**, and then **HBox** from the submenu.
6. Add a Label and a TextArea element in the details area.
  - a. From the Controls section of the Library panel, select **Label**. Drag and drop it to the left side of the details section and below the row occupied by the HBox you just added. Use the guidelines to position the Label element in line with the HBox element's left side.
  - b. Double-click **Label** to enter the edit mode. Enter **DESCRIPTION:** to replace the default value.
  - c. Drag and drop a **Text Area** below the label that you just added. Click and drag the lower right handle of the Text Area to increase its size and fill the remaining space in the details section, as shown in [Figure 6-4](#).
  - d. Select the **Text Area** element in the Content panel. In the **Properties** section of the Inspector panel, select **descriptionValue** from the drop-down list for the fx:id field.

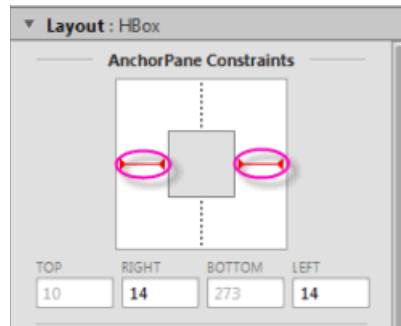
**Figure 6-4 Enlarge the Text Area**



## Anchor the UI Elements in the Details Section

To ensure that the spacing and size of the UI elements adjust properly when the application's window is resized, you must modify their anchor pane constraints.

1. In the Hierarchy panel, select the row for the **HBox** that contains the rows for the **Label displayedIssueLabel** and the **TextField synopsis** elements. It is directly below the row for the **AnchorPane: details** element.
2. Click the **Layout** section of the Inspector panel.
3. In the **AnchorPane Constraints** subsection, 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 6-5](#).

**Figure 6–5** *Setting the Anchor Lines*

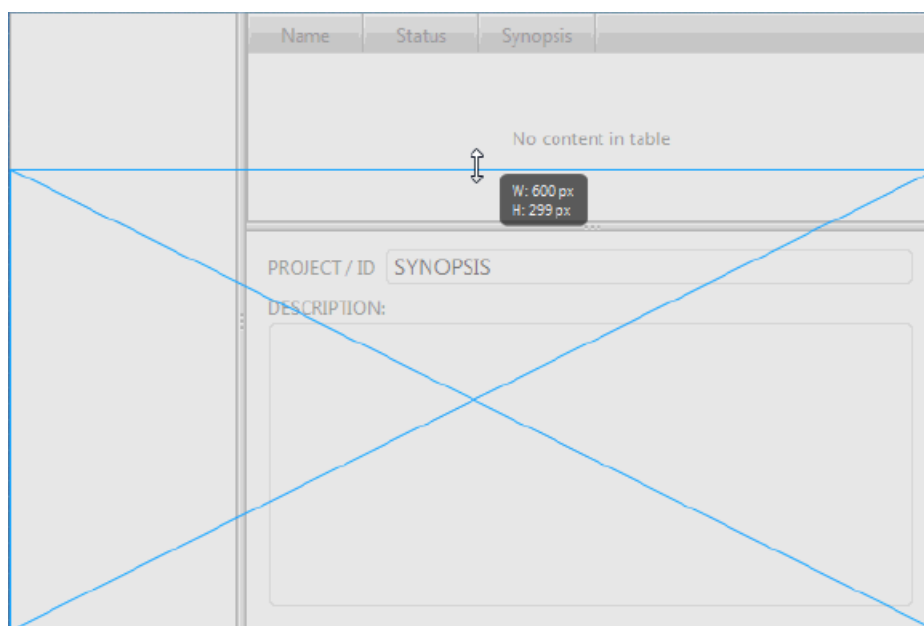
4. In the Hierarchy panel, select the row for the **TextField synopsis** component. Click the **Layout** section of the Inspector panel, and locate the HBox Constraints subsection and select **ALWAYS** from the drop-down list for the **Hgrow** text field. This setting indicates that the Text Field: synopsis element will adjust horizontally when its parent container increases.
5. Select the row for the **TextArea descriptionValue** in the Hierarchy panel. In the Layout section of the Inspector panel, locate the AnchorPane Constraints subsection and click the left, top, right, and bottom dashed anchor lines.
6. In the Hierarchy panel, select the row for the **Label displayedIssueLabel** component. In the **Layout** section of the Inspector 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.

## Add the Toolbar

Add a toolbar to the top portion of the UI. It will contain an image file and three buttons.

1. In the Hierarchy panel, change the display setting by clicking the menu button on the top right corner and choosing **Show Info**, if it is not already chosen.
2. Select the row for the **SplitPane (Horizontal Flow)** element. You might have to resize the Hierarchy panel to see the full node ID of the Split Pane element.
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 panel, as illustrated by [Figure 7-1](#).

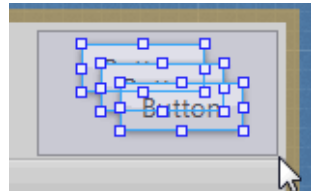
**Figure 7-1** Make Room for the Toolbar



4. From the Menu bar, choose **File, Import**, and then **Media**. Select **IssueTrackingLite.png** from the folder `<install-dir>/IssueTrackingLite/src/issuetrackinglite`
5. Move the image to the top left corner of the toolbar area. Increase the height of the toolbar area if it is too small for the image being inserted.
6. Add three buttons to the toolbar.

- a. From the Library panel, drag a **Button** element and drop it on the right side of the toolbar.
  - b. From the Menu bar, choose **Edit** and then **Duplicate** twice to add two more new buttons to the toolbar.
7. Select all three button controls by holding down the left mouse button and dragging a marquee over the buttons, as shown in [Figure 7-2](#). When you release the left mouse button, all three buttons should be selected.

**Figure 7-2 Marquee Selection of the Three Buttons**



- a. In the Hierarchy panel, select the row for the **HBox** that was just added.
  - b. Click the **Layout** section of the Inspector panel and in the AnchorPane Constraints subsection, click the top and right dashed anchor lines.
  - c. Click the **Properties** section of the Inspector panel and in the **Spacing** text field, enter 15 to replace the default value of 5. Notice that the spacing between the buttons is increased.
9. Define the values for the anchorLayout and spacing for the buttons.
  - a. In the Hierarchy panel, select the row for the **HBox** that was just added.
  - b. Click the **Layout** section of the Inspector panel and in the AnchorPane Constraints subsection, click the top and right dashed anchor lines.
  - c. Click the **Properties** section of the Inspector panel and in the **Spacing** text field, enter 15 to replace the default value of 5. Notice that the spacing between the buttons is increased.
10. Edit the button details.
  - a. Select the button on the right side of the toolbar. Click the **Properties** section of the Inspector panel, and select **deleteIssue** from the fx:id property's drop-down list. Enter `Delete` in the **Text** field.
  - b. Double-click the middle button to get into edit mode. Enter `Save` in the edit box. Click the **Properties** section of the Inspector panel, and select **saveIssue** in the drop-down list for the fx:id property for the middle button.
  - c. In the toolbar, double-click the leftmost button to get into edit mode and enter `New`. Click the **Properties** section of the Inspector panel, select **newIssue** in the drop-down list for the fx:id property for the New button.
11. Bind each button to an event handler that will perform an action when the button is clicked.
  - a. Select the **New** button. Click the **Code** section of the Inspector panel.
  - b. 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 `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.

- 
- c. Select the **Save** button and in the **On Action** field, set the value to **#saveIssueFired**.
  - d. Select the **Delete** button and in the On Action field, set the value to **#deleteIssueFired**.
12. Choose **File** and then **Save** from the Menu bar to save your work.



---

## Use a Style Sheet and Preview the UI

Preview the UI work that you have laid out so far and then change the look and feel of the layout by working with a style sheet.

### Preview the UI

Use the following steps to preview the UI work that you have done so far:

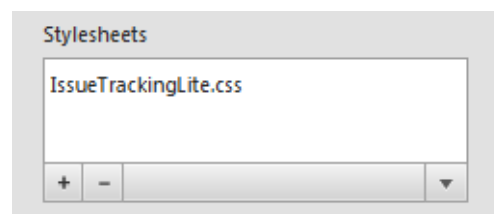
1. From the Menu bar, choose **Preview**, and then select **Preview in Window**.
2. Resize the window multiple times to ensure that the buttons in the toolbar and the text area resize appropriately when the window is resized.
3. To stop viewing the preview, close the Preview window or from the Menu bar,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

### Use a Style Sheet

You can customize the look and feel of your UI by applying style sheets. For this tutorial, you use a style sheet file that has been provided with the IssueTrackingLite sample.

1. Verify that the Cascading Style Sheet (CSS) resource file that is bundled with the IssueTrackingLite sample is already set. In the Hierarchy panel, select the root **AnchorPane** container. Click the **Properties** section of the Inspector panel. In the Stylesheets list view of the CSS subsection, notice that the IssueTrackingLite.css style sheet is already set, as shown in [Figure 8-1](#). This is the style sheet that was set when you created the FXML file.

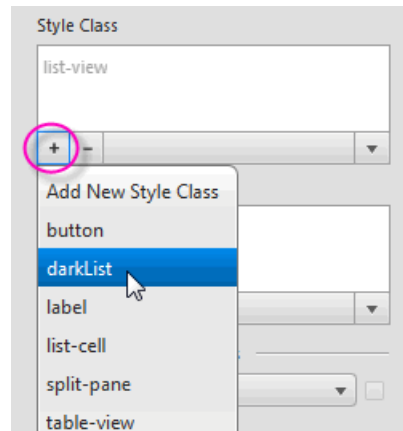
**Figure 8-1** Adding a Style Sheet File



2. Use a style class for one of the elements in the Content panel.
  - a. In the Hierarchy panel, select the row for the **ListView** element.
  - b. Click the **Properties** section of the Inspector panel, and click the button with the plus sign (+) in the **Style Class** list. Select **darkList** as shown in [Figure 8-2](#).

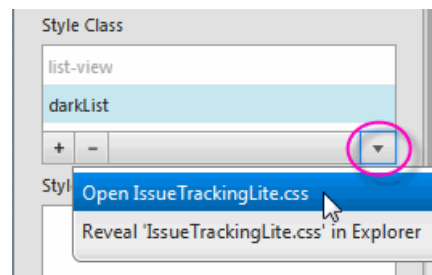
Notice that the appearance of the ListView element in the Content panel has changed to a dark grey color.

**Figure 8–2 Adding a Style Class to the ListView Element**



- c. In the **Style Class** list view, with the `darkList` selected, click the choice button with the down arrow and select **Open IssueTrackingLite.css** from the list, as shown in [Figure 8–3](#). The `IssueTracking.css` file is displayed in the default editor defined for the CSS file type. You can make edits to the file, and then save the file. The changes are immediately applied. Exit from the editor window.

**Figure 8–3 Working with the Style Sheet File**



3. From the Menu bar, choose **File** and then **Save**.

You just completed building the FXML layout for a JavaFX application using JavaFX Scene Builder. Continue with the [Use a Style Sheet and Preview the UI](#) to compile and run the `IssueTrackingLite` application.



---

## Compile and Run the Application

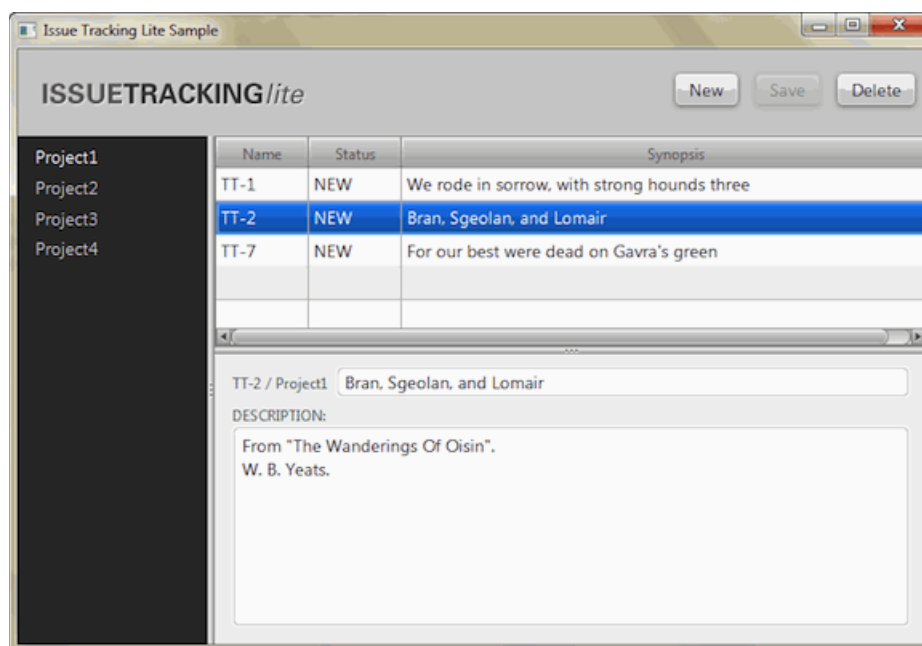
You are ready to see your application in action. You can use either NetBeans IDE or the Apache Ant utility to compile and run the IssueTrackingLite application.

### Use NetBeans IDE

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

1. In the NetBeans IDE 7.2 window, right-click the **IssueTrackingLite** project node in the Projects window and select **Run**. NetBeans IDE compiles the project and if no errors are encountered, it displays an application similar to [Figure 9-1](#). Click the image to enlarge it.

**Figure 9-1** Completed UI for Issue Tracking Sample (Click image to enlarge.)



2. 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 section, as shown in [Figure 9-1](#).

3. 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 were 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 installation. At the present, the JavaFX-enabled platform is not automatically created for you on the Mac OS X platform.

## Use the Apache Ant Utility

If you choose not to run the application in NetBeans IDE, use the Apache Ant utility (version 1.8 or later) to build and run the application on the command line. Enter a command similar to the one in [Example 9–1](#) if you are using JDK 7 or [Example 9–2](#) if you are using JDK 6 on a Windows platform.

### **Example 9–1** Apache Ant Command to Run the Application When Using JDK 7

```
ant -Dplatforms.Default_JavaFX_Platform.home=<JAVA_HOME>
-Dplatforms.Default_JavaFX_Platform.javafx.runtime.home==<JAVA_HOME>/jre
-f <JavaFX_App_Name>/build.xml
<TARGET>
```

### **Example 9–2** Apache Ant Command to Run the Application When Using JDK 6

```
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 the above examples, the main values for <TARGET> are clean, jar, and run. For example, to run the IssueTrackingLite application on the Windows platform, enter something similar to the command in [Example 9–3](#) if you are using JDK 7 and [Example 9–4](#) if you are using JDK 6 on a Windows platform. You can set <TARGET> with the value of -projecthelp to get a list of available targets.

### **Example 9–3** Using Apache Ant to Run IssueTrackingLite on a Windows Platform When Using JDK 7

```
ant -Dplatforms.Default_JavaFX_Platform.home="C:\Program Files\Java\jdk1.7.0_06" \
-Dplatforms.Default_JavaFX_Platform.javafx.runtime.home=\
"C:\Program Files\Java\jdk1.7.0_06\jre" \
-f IssueTrackingLite/build.xml run
```

### **Example 9–4** Using Apache Ant to Run IssueTrackingLite on a Windows Platform When Using JDK 6

```
ant -Dplatforms.Default_JavaFX_Platform.home="C:\Program Files\Java\jdk1.6.0_33" \
-Dkjavafx.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 Mac OS X platform with Java SE 7 update 6 software installed, type the command shown in [Example 9–5](#).

**Example 9-5 Using Apache Ant to Run IssueTrackingLite on a MacOS Platform**

```
ant -Dplatforms.Default_JavaFX_Platform.home=\
    /Library/Java/JavaVirtualMachines/jdk1.7.0_06.jdk/Contents/Home \
-Dplatforms.Default_JavaFX_Platform.javafx.runtime.home=\
    /Library/Java/JavaVirtualMachines/jdk1.7.0_06.jdk/Contents/Home/jre \
-f IssueTrackingLite/build.xml run
```