JavaFX Scene Builder

Getting Started with JavaFX Scene Builder

Release 2.0

E51278-01

April 2014

This document gives an overview of the JavaFX Scene Builder development tool and steps you through the creation of the GUI layout used in a simple JavaFX issue-tracking sample application.

ORACLE

JavaFX Scene Builder Getting Started with JavaFX Scene Builder Release 2.0

E51278-01

Primary Author: Cindy Castillo

Contributor: Yves Joan

Contents

Preface

This preface gives an overview about this tutorial and also describes the document accessibility features and conventions used in this tutorial - *Getting Started with JavaFX Scene Builder*.

About This Tutorial

This Getting Started tutorial is a compilation of two documents that were previously delivered with the JavaFX Scene Builder 1.x documentation set: JavaFX Scene Builder Overview and Getting Started with Scene Builder. The content has been updated with information about the new features and enhancements made in the JavaFX Scene Builder tool.

This document contains the following parts:

- JavaFX Scene Builder Overview
- Building a JavaFX Application Using Scene Builder

Audience

This document is intended for JavaFX developers.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at
`http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit
`http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit
`http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Related Documents

For more information, see the following documents in the JavaFX Scene Builder and JavaFX documentation sets:

- *JavaFX Scene Builder Installation Guide*

- *JavaFX Scene Builder Release Notes*
- *JavaFX Scene Builder User Guide*
- *Using JavaFX Scene Builder with Java IDEs*
- *Mastering FXML*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

What's New

The following list gives a summary of the features and usability improvements made in the JavaFX Scene Builder 2.0 release. See Key Features and the JavaFX Scene Builder User Guide for more information.

- Ability to add custom GUI components to the Library has been added.
- Support for new JavaFX 8 GUI components is now available.
- Support of 3D objects is provided.
- Support for Rich Text is available via the new container, TextFlow.
- Improvements have been made to the Library, Hierarchy, Content, and Inspector panels. Also, the Preview window has been enhanced so that its content is now automatically refreshed as the current FXML document is being edited. See JavaFX Scene Builder User Guide for more information.
- The new JavaFX 8 Modena theme is now used for the JavaFX Scene Builder tool's look and feel.
- JavaFX Scene Builder Kit API is now available, allowing the integration of Scene Builder panels and functionalities directly into the GUI of a larger application, or a Java IDE, such as NetBeans, IntelliJ, and Eclipse. See JavaFX Scene Builder Release Notes for more information.

Part I

JavaFX Scene Builder Overview

Part I contains the following chapter:

1

Overview

This chapter gives an overview of the JavaFX Scene Builder 2.0 development tool, including information about key features, target audience, and download information.

JavaFX Scene Builder provides a visual layout environment that lets you quickly design user interfaces (UI) for JavaFX applications without needing to write any code. It allows simple drag-and-drop positioning of graphical user interface (GUI) components onto a JavaFX scene. As you build the layout of your UI, the FXML code for the layout is automatically generated. JavaFX Scene Builder provides a simple yet intuitive interface that can help even nonprogrammers to quickly prototype interactive applications that connect GUI components to the application logic.

Intended Audience

The target audience for JavaFX Scene Builder includes the following:

- **Java developers**: They can quickly prototype the client application's GUI layout and develop the application logic separately.
- **Designers**: They can quickly prototype the client application's GUI layout without requiring any application code to be written first. They can design and preview the GUI layout and define its look and feel with style sheets.

Key Features

JavaFX Scene Builder includes the following key features:

- **A drag-and-drop WYSIWYG interface** allows you to quickly create a GUI layout without the need to write source code. You can add, combine, and edit JavaFX GUI controls to your layout by using the library of GUI controls and the content panel.
- **Tight integration with the NetBeans IDE** provides optimal development workflow.
- **Integration with any Java IDE is easy** since it is a standalone development tool. See Using JavaFX Scene Builder with Java IDEs for information on how to use Scene Builder with NetBeans IDE, Eclipse, and IntelliJ IDEA.
- **Automatic FXML code generation** occurs as you build and modify your GUI layout. The generated FXML code is stored in a separate file from the application logic source and style sheet files.
- **Live editing and preview features** let you quickly visualize the GUI layout changes that you make without the need to compile. These features help minimize development time for your application. You can also assign Cascading Style Sheets (CSS) to your GUI layout and preview the resulting look and feel that is applied.

- **Access to the complete JavaFX GUI controls library** is provided. To see the full list of supported JavaFX 8 GUI components, type `FX8` in the Library panel's Search text field. The list includes the `TreeTableView`, `DatePicker`, and `SwingNode` components.
- **Ability to add custom GUI components to the Library** is now available. The Library of available GUI components can be extended by importing customized GUI components from third party JAR files, FXML files, or adding them from the Hierarchy or Content panels. See Scene Builder User Guide for more information.
- **3D support** is provided. FXML documents containing 3D objects can now be loaded and saved in the Scene Builder 2.0 tool. You can view and edit properties of the 3D objects using the Inspector panel (Material and Mesh complex properties are not yet supported). You can not, however, create new 3D objects using the Scene Builder tool.
- **Support for Rich Text** has been added. A new container, TextFlow, is now available in the Library of GUI components. You can drag multiple text nodes and other types of nodes, into the a TextFlow container. You can also directly manipulate the text nodes to re-arrange them in the container. Inline and property editing features are also available for each text node.
- **JavaFX Scene Builder Kit** is provided with Scene Builder 2.0. The kit is an API that allows the integration of Scene Builder panels and functionalities directly into the GUI of a larger application, or a Java IDE, such as NetBeans, IntelliJ, and Eclipse. See JavaFX Scene Builder Release Notes for more details.
- **CSS support** enables flexible management of the look and feel of your application's UI.
- **Cross-platform support** is provided on Windows, Linux, and Mac OS X operating systems.

Download Information

Use the following steps to get started using the JavaFX Scene Builder tool to build the GUI layout for your JavaFX application.

1. Go to the Additional Resources section of the Java SE Downloads page at `http://www.oracle.com/technetwork/java/javase/downloads/index.html` to download the JavaFX Scene Builder installer. Use the JavaFX Scene Builder Installation Guide to learn about the system requirements and installation instructions.
2. Read the JavaFX Scene Builder Release Notes to learn about known issues and workarounds.
3. Use the JavaFX Scene Builder User Guide to learn more about the tool's user interface and Building a JavaFX Application Using Scene Builder to create a simple issue tracking application.
4. Read Using JavaFX Scene Builder with Java IDEs to learn about how to use Scene Builder with NetBeans IDE, Eclipse, and IntelliJ IDEA.

Additional Resources

To learn more about the JavaFX technology, see the JavaFX tutorials and articles at `http://docs.oracle.com/javase/8/javase-clienttechnologies.htm`.

Part II

Building a JavaFX Application Using Scene Builder

This chapter 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.

This document contains the following topics:

2

Prepare for This Tutorial

This chapter provides information about requirements and recommendations to prepare your development environment before you create the sample IssueTrackingLite FXML layout that you will build in this tutorial using JavaFX Scene Builder.

As you build the layout, the FXML code for the designed GUI 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. 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. This tutorial also includes information that steps you through the creation of the IssueTrackingLite FXML layout without the use of NetBeans IDE.

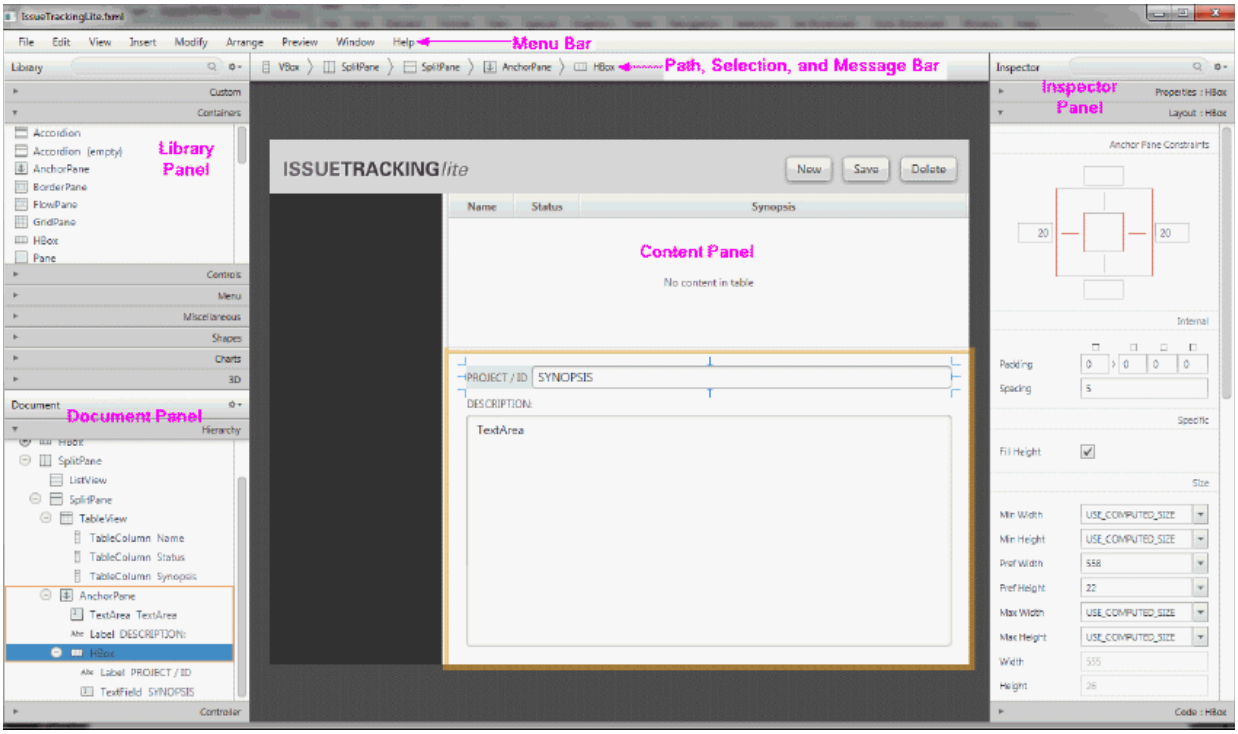
> **Note:** JavaFX Scene Builder does not have dependency on any particular IDE. However, since this tutorial was prepared using NetBeans IDE, it is recommended that you use NetBeans IDE to complete this tutorial. If Eclipse or IntelliJ IDEA is your preferred IDE, see Using JavaFX Scene Builder with 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. If you choose to work with this tutorial outside of any IDEs, there are sections in this tutorial that highlight what you need to do to use the Apache Ant utility to connect the layout that you build to the sample application's Java source code, apply the style sheet, and run the sample application.

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 JavaFX Scene Builder samples** from the Additional Resources section of the download page at `http://www.oracle.com/technetwork/java/javase/downloads/index.html`. Extract the contents of the `javafx_scenebuilder_samples-2_0.zip` file, which includes the IssueTrackingLite sample folder. 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 sample project issues, modify them, or add new issue.

3. **(Recommended) Use** JavaFX Scene Builder User Guide to familiarize yourself with the JavaFX Scene Builder user interface that is shown in Figure 2–1.
4. **(Recommended) Install NetBeans IDE 8**, which is used in this tutorial to illustrate the integration between NetBeans IDE and JavaFX Scene Builder. As mentioned in the Note above, you can also create the IssueTrackingLite GUI layout using other Java IDEs, such as Eclipse or IntelliJ IDEA, or a standalone instance of Scene Builder.
5. **(Recommended) Familiarize yourself with the JavaFX concepts** by reading the available documents on the JavaSE Client Technologies page at `http://docs.oracle.com/javase/8/javase-clienttechnologies.htm`. In particular, learn about the FXML concepts by reading Using FXML to Create a User Interface and get familiar with layouts using Working with Layouts in JavaFX.

Figure 2–1 Main Window for JavaFX Scene Builder



3

Open the Sample FXML File

This chapter shows you how to open the `IssueTrackingLite.fxml` sample file in a JavaFX Scene Builder window using the NetBeans IDE Open command or using the Scene Builder Open command.

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 GUI 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 in a standalone Scene Builder tool.

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

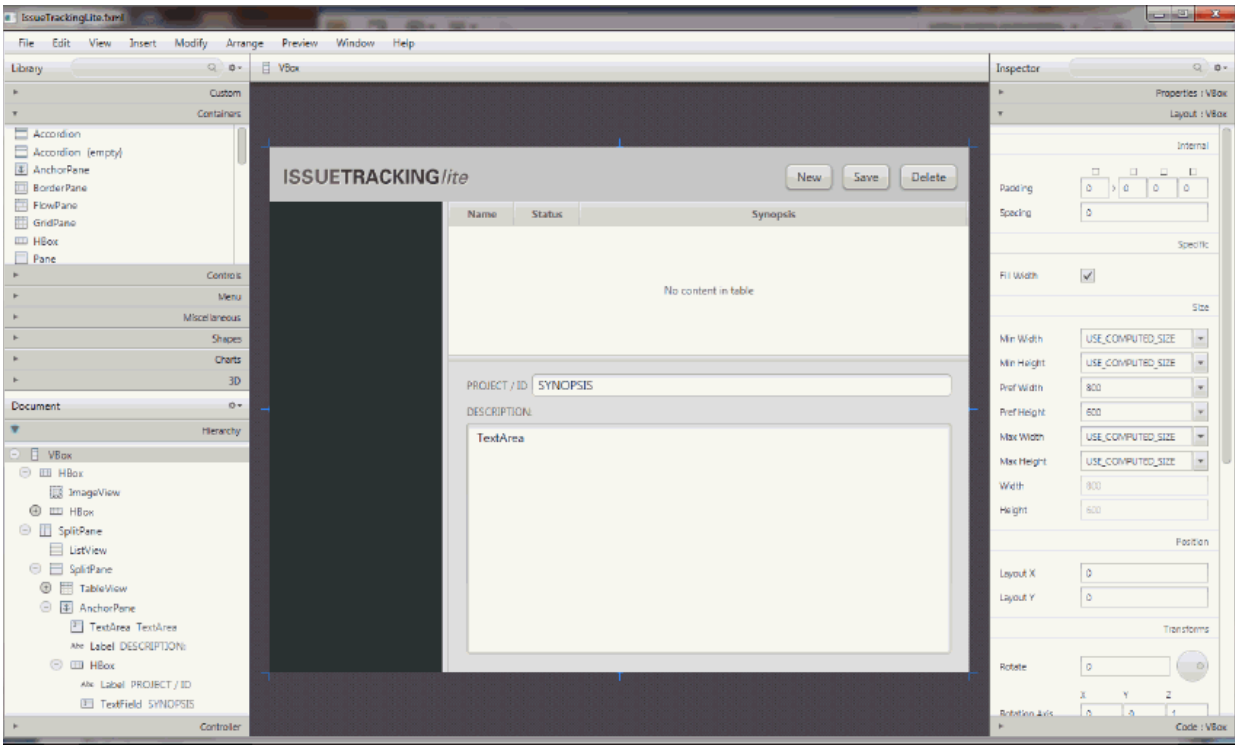
Open the completed `IssueTrackingLite.fxml` sample layout file using NetBeans IDE's Projects window:

1. Start NetBeans IDE on your Windows platform by double-clicking the **NetBeans IDE 8.0 shortcut**, or select **Start,** then **All Programs**, then **NetBeans**, and finally, **NetBeans IDE 8.0**. On a Mac OS X platform, double-click the **NetBeans IDE 8.0** application icon.
2. From the Menu bar, select **File** and then **Open Project**.
3. From the Open Project dialog box, go to the directory where you extracted the `IssueTrackingLite` sample 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 in JavaFX Scene Builder.

Note: If the `IssueTrackingLite.fxml` file opens in the IDE source editor instead of in a Scene Builder window, it may mean that the IDE is not aware of your Scene Builder installation. To verify, select **Tools** from the NetBeans IDE main menu and then **Options**. In the Options window, click the **Java** tab and then the JavaFX tab. The Scene Builder Home directory would be listed if NetBeans is aware of your Scene Builder installation. If it is not listed, see Prepare for This Tutorial chapter for the requirements before continuing with this tutorial.

The main window for the JavaFX Scene Builder tool appears with the `IssueTrackingLite.fxml` file opened in the Content panel, as shown in Figure 3–1.

Figure 3–1 Completed IssueTrackingLite.fxml Opened in JavaFX Scene Builder Window



5. Save the file with a different name so that you can create your own FXML layout file. From the Menu bar, select **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 a standalone JavaFX Scene Builder window.

1. Start JavaFX Scene Builder on your Windows platform by double-clicking the **JavaFX Scene Builder 2.0 shortcut**, or select **Start,** then **All Programs**, then **JavaFX Scene Builder**, and finally, **JavaFX Scene Builder 2.0**. On a Mac OS X platform, double-click the **JavaFX Scene Builder 2.0** application icon.
2. From the Menu bar, select **File** and then **Open**.
3. From the Open FXML dialog box, go to the folder in which you extracted the `IssueTrackingLite` sample 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 3–1.
5. Save the file with a different name so that you can create your own FXML layout file using the original file name that is recognized by the IssueTrackingLite application.
   a. From the Menu bar, select **File** and then **Save As**.
   b. Enter `IssueTrackingLiteComplete.fxml` in the **File Name** text field and click **Save**.

4

Create the FXML File and the Base Panes

In this chapter, you create a new FXML layout file using either the NetBeans IDE New command or the JavaFX Scene Builder New command. You then assign the cascading style sheet (CSS) to use for the FXML layout and create the base panes to start the application's GUI design.

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 by doing the following:

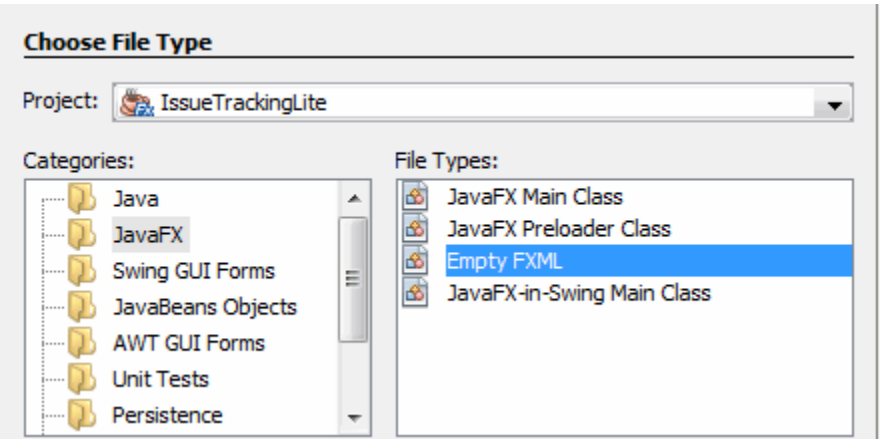
1. Create the new FXML file by following the steps in either Use NetBeans IDE New Wizard or Use JavaFX Scene Builder New Command.
2. Set the Root Container, CSS, and Style Class to use for the entire layout.
3. Create the Base Panes for the layout.

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** folder node under Source Packages, select **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 4–1. Click **Next**.

Figure 4–1 Choose Empty FXML File Type



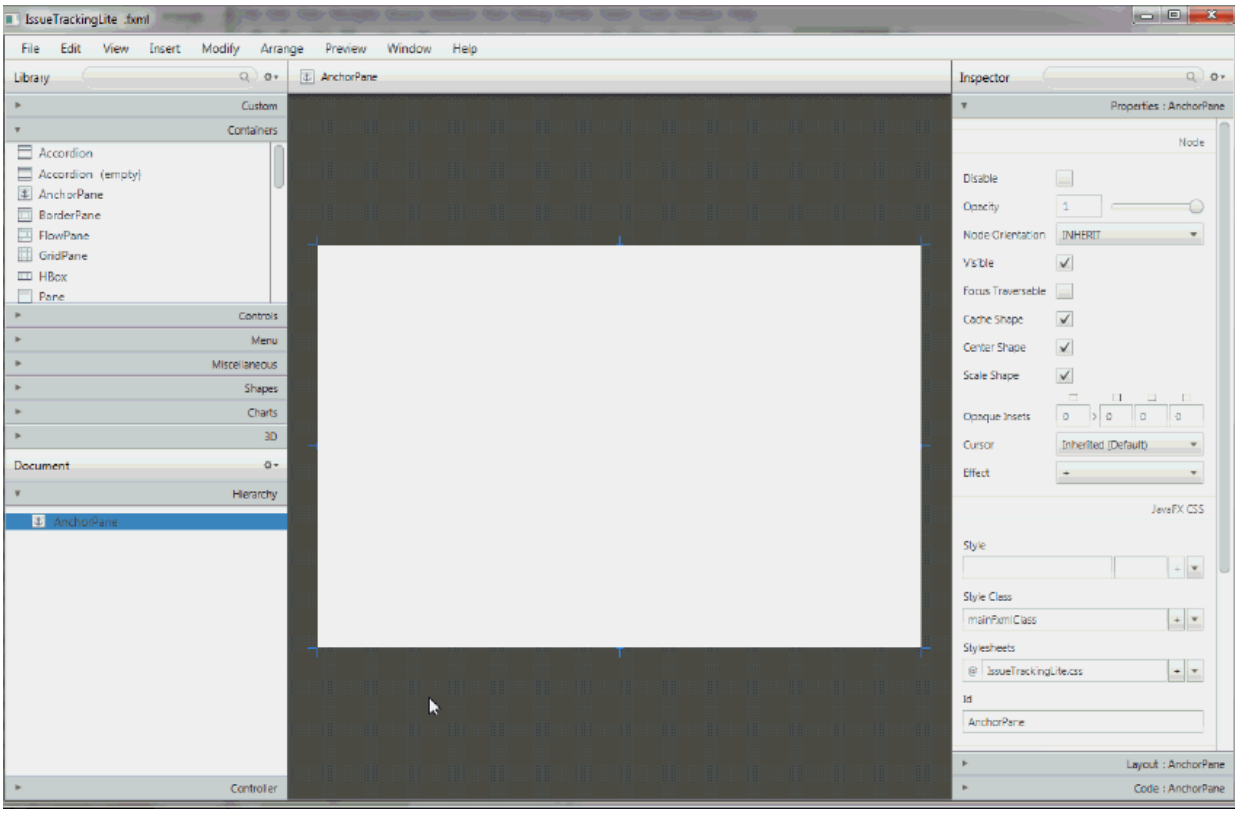
3. In the New Empty FXML dialog box, enter `IssueTrackingLite` in the **FXML Name** text field. Ensure that the **Package** text field has the value `issuetrackinglite`. Click **Finish**

   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.

4. 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 FXML file opened in the Content panel, as shown in Figure 4–2.

Figure 4–2 JavaFX Scene Builder Main Window at Startup from NetBeans IDE



5. In the Scene Builder Hierarchy panel, right-click the **AnchorPane** node and select **Delete**.
   The IssueTrackingLite layout you build in this tutorial uses a different top container which you will add in Set the Root Container, CSS, and Style Class.

6. Select **File** and then **Save** from the JavaFX Scene Builder Menu bar.

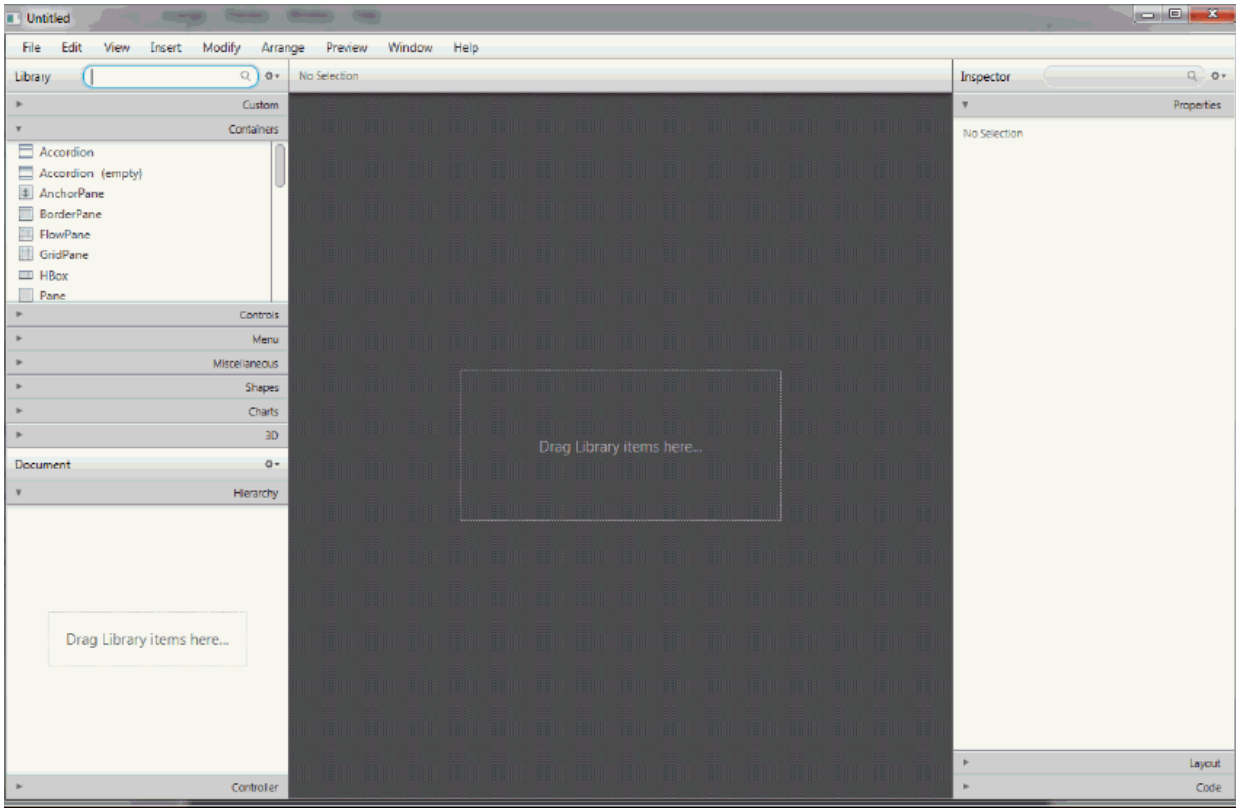
Use JavaFX Scene Builder New Command

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

From the Scene Builder Menu bar, select **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 4–2. The Content panel is initially empty.

Figure 4–3 JavaFX Scene Builder Main Window After New File Creation



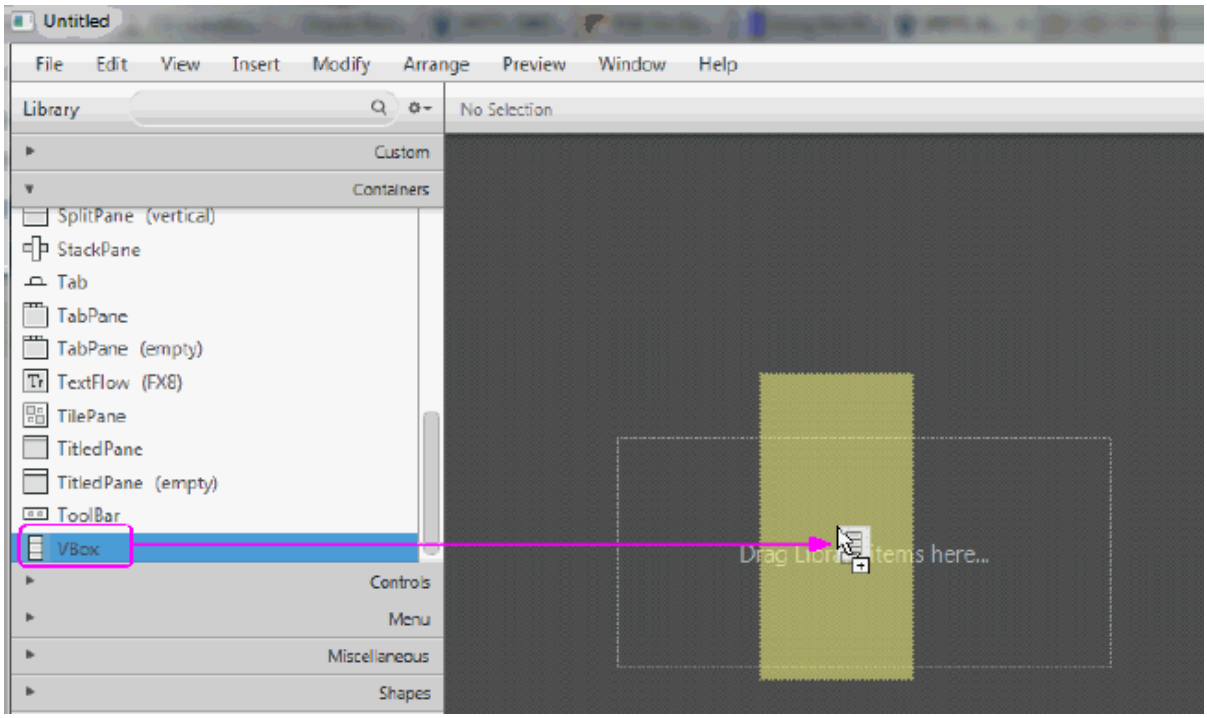
Set the Root Container, CSS, and Style Class

Set the root container, stylesheet, and style class to use for the entire layout.

> **Tip:** From the Menu bar, select **File** and then **Save** command often. Also, from the Menu bar, select **Edit** and then **Undo** to undo actions when necessary while you build the IssueTrackingLite application.

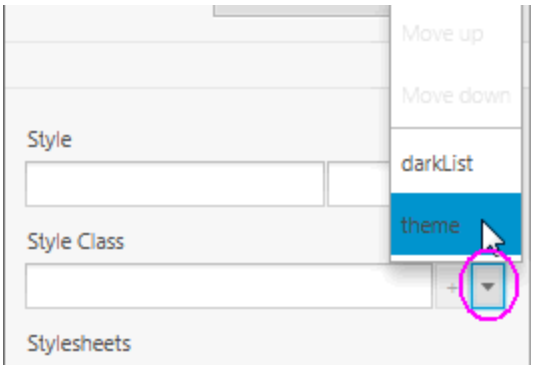
1. Drag a VBox container from the Library panel to the Content panel, as shown in Figure 4–4.

Figure 4–4 Add the Initial VBox Container



2. Set the CSS file to use for your new FXML layout.
   a. In the Hierarchy section of the Document panel, ensure that the root **VBox** container element is selected.
   b. In the Properties section of the Inspector panel, go to the JavaFX CSS section and locate the Stylesheets text field. Click the button with the plus sign (+) symbol.
   c. In the Add Style Sheet dialog box, navigate to where you extracted the `IssueTrackingLite` sample file. Open the `issuetrackinglite` folder and select the **IssueTrackingLite.css** file. Click **Open**.
3. Set the Style Class.
   a. Ensure that the root VBox container is still selected in the Hierarchy panel.
   b. In the **Properties** section of the Inspector panel, locate the Style Class text field and click the drop-down arrow. The list of style classes is retrieved from the `IssueTrackingLite.css` file. Select **theme**, as shown in Figure 4–5

Figure 4–5 Set the Style Class for the Layout



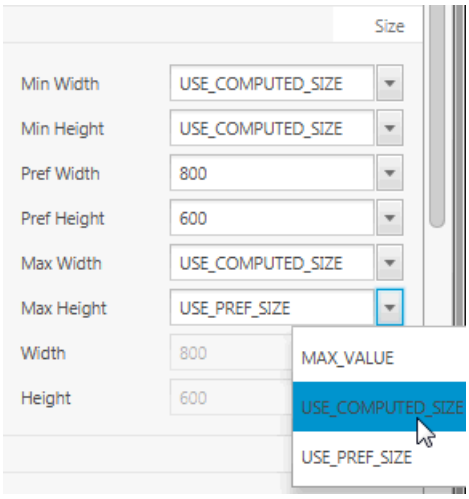
4. Save the new FXML file before you continue with the rest of the tutorial.
    a. From the main menu bar, select **File** and then **Save**, if you created the file via the NetBeans IDE, or **Save As**, if you created the file via Scene Builder File > New command.
    b. In the Save As dialog window, navigate to where you extracted the `IssueTrackingLite` sample file. Expand the `IssueTrackingLite`, `src`, and then `issuetrackinglite` 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.

Resize the Scene and the Scene Builder Window

Resize the scene and the Scene Builder window so that you have a bigger working area.

1. Resize the scene's width and height in the Content panel to get a larger working area.
    a. In the Inspector panel, select the **Layout** section.
    b. In the Size section, change the `Pref Width` property value to **800** and the `Pref Height` property value to **600**.
    c. Change the **Min Width**, **Min Height**, **Max Width**, and **Max Height** property values to **USE_COMPUTED_SIZE**, as shown in Figure 4–6.

Figure 4–6 Resize the Scene



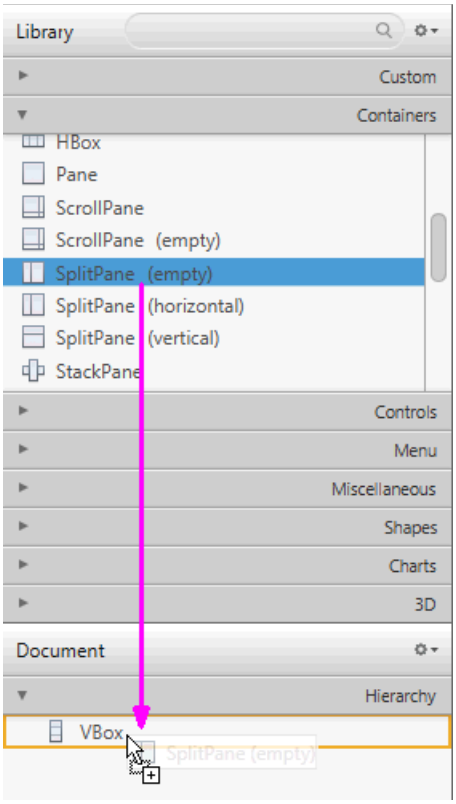
2. Resize the Scene Builder window so you are able to view the entire VBox container.

Create the Base Panes

Create the base layout panes to define the different sections of your GUI layout.

1. Open the Hierarchy section of the Document panel, if not already opened.
2. Add a SplitPane object.
   a. From the Library panel, drag a **SplitPane (empty)** container to the Document panel's Hierarchy section and drop it inside the VBox element, as shown in Figure 4–7.

Figure 4–7 Add Empty SplitPane to the VBox Container



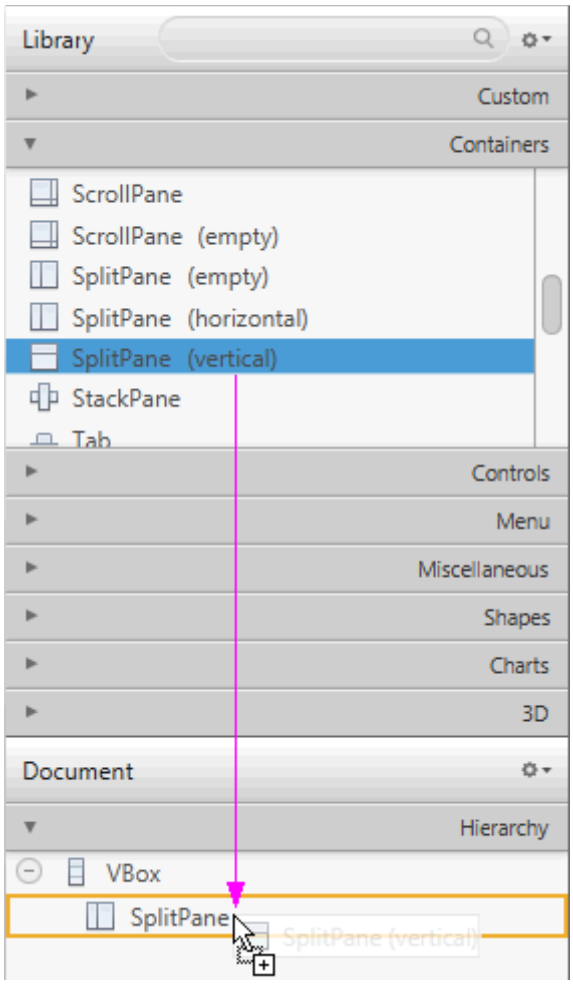
b. Select the **Split Pane** in the Hierarchy panel and from the menu bar select **Modify** and then **Use Computed Sizes**.

c. In the Inspector panel, select the **Layout** section. Set the **Vgrow** property value to **ALWAYS**.

3. Add another `SplitPane` element and modify its contents.

a. From the Library panel's Containers section, drag a **SplitPane (vertical)** element and drop it into the previously added SplitPane in the Hierarchy panel, as shown in Figure 4–8.

Figure 4–8 Add Split Pane (Vertical) Element



Notice that after the SplitPane (Vertical) element is dropped, it is added to the Document panel with its AnchorPane children.

b. From the menu bar, select **Modify** and then **Use Computed Size**. Alternatively, you can press **Ctrl+Shift+K**.

c. In the Hierarchy panel, expand the second SplitPane container element you just added to display its contents. Right-click the first AnchorPane node and select **Delete** from the contextual menu.

d. Select the node for the remaining AnchorPane and from the main menu bar, select **Modify** and then **Use Computed Sizes**.
Notice that in the Layout section of the Inspector panel, the Sizes properties for the AnchorPane is updated to the value of `USE_COMPUTED_SIZE`.

4. From the Menu bar, select **File** and then **Save**. Alternatively, you can press **Ctrl-S** (for Windows or Linux platforms) or **Cmd-S** (for Mac OS platform).
Perform the Save action frequently to preserve your work.

5

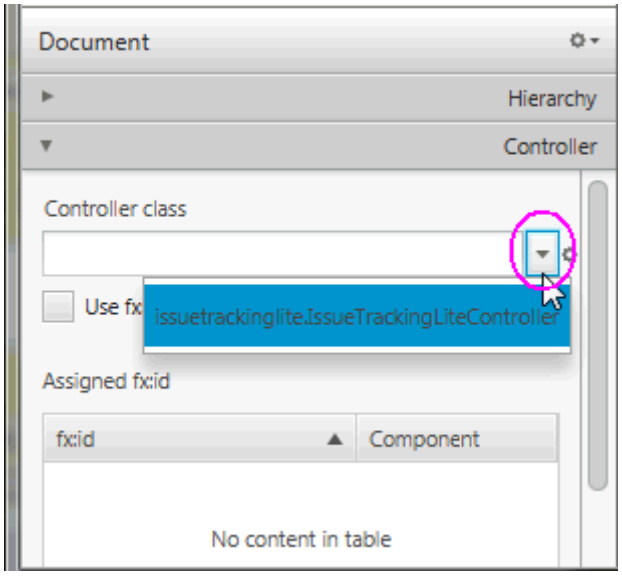
Bind the GUI to the Application Logic

This chapter describes the steps to bind the FXML layout you are building with JavaFX Scene Builder to the source controller file that has been provided with the IssueTrackingLite sample.

The controller source file, `IssueTrackingLiteController.java`, manages the events and actions taken on each element you add to the FXML GUI layout that you are building. Setting the controller class file name enables Scene Builder to provide you with the names of the event handlers and instance variables that are declared in the controller source file.

1. In the Document panel, select the **Controller** section.
2. Set the value in the **Controller class** text field to `issuetrackinglite.IssueTrackingLiteController` by selecting it from the drop-down list of available values.

Figure 5–1 Add Controller Class



6

Add the List and Table Views

In this chapter, you will continue to use JavaFX Scene Builder to add the JavaFX GUI controls that are used to list the projects and issues assigned to each project in the IssueTrackingLite sample application.

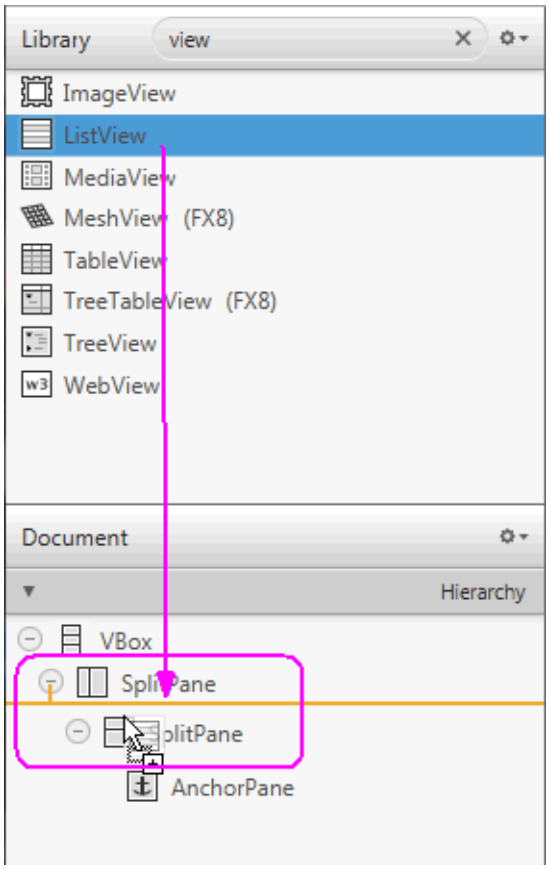
Use the following sections to add List View and Table Views controls to your IssueTrackingLite GUI layout.

Add a List View

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

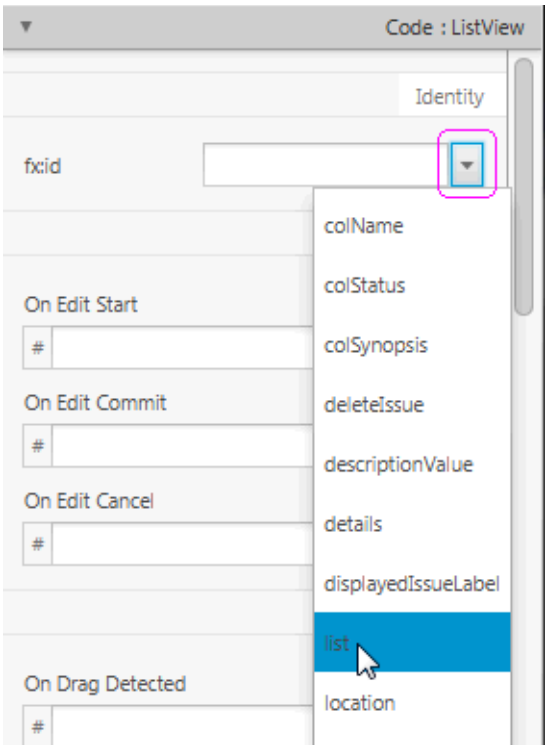
1. Select the Hierarchy section of the Document panel.
2. In the Library panel's **Search** box, clear any existing value and enter `view` to locate the view-styled GUI controls.
3. Drag a **ListView** control from the Library panel and drop into the first SplitPane node in the Hierarchy panel, as shown in Figure 6–1.

Figure 6–1 Add ListView Control to SplitPane



4. From the Menu bar, select **Modify** and then **Use Computed Sizes** or press **Ctrl+Shift+K**.
5. Click the **Code** 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 6–2.

Figure 6–2 Set fx:id for ListView Control



Note: The `fx:id` property 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.

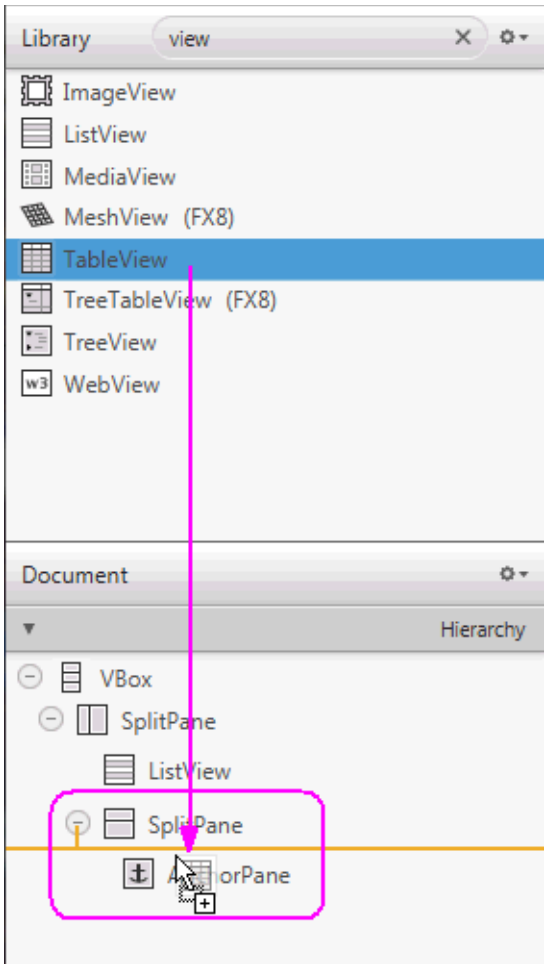
6. Click the **Layout** section of the Inspector panel. Locate the **Split Pane Constraints** subsection and uncheck the **Resizable With Parent** check box.

Add a Table View

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

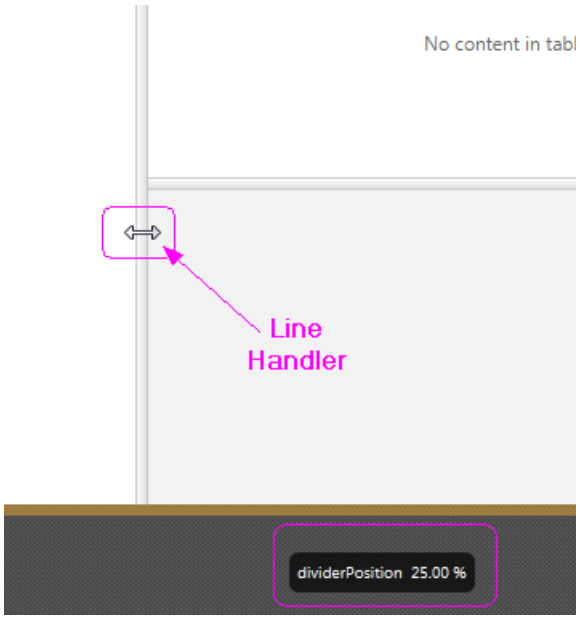
1. From the search results list in the Library panel, select **Table View**.
2. Drag and drop it in the Hierarchy panel, inside the second SplitPane element and just above the AnchorPane, as shown in Figure 6–3.

Figure 6–3 Add the Table View Control



3. From the Menu bar, select **Modify** and then **Use Computed Sizes**.
4. Click the **Code** section of the Inspector panel, type "`t`" in the **fx:id** text field to filter the list of available fx:id values, and select **table**.
5. Tune the positions of the vertical and horizontal splitters shown in the Content panel.
    a. Select the first **Split Pane** node in the Hierarchy panel. In the Content panel, drag the vertical divider to the left until the dividerPosition percentage displays about 25%, as illustrated in Figure 6–4. Alternatively, enter `0.25` in the **Divider Positions** property text field in the Properties section of the Inspector panel.

Figure 6–4 Move Vertical Divider

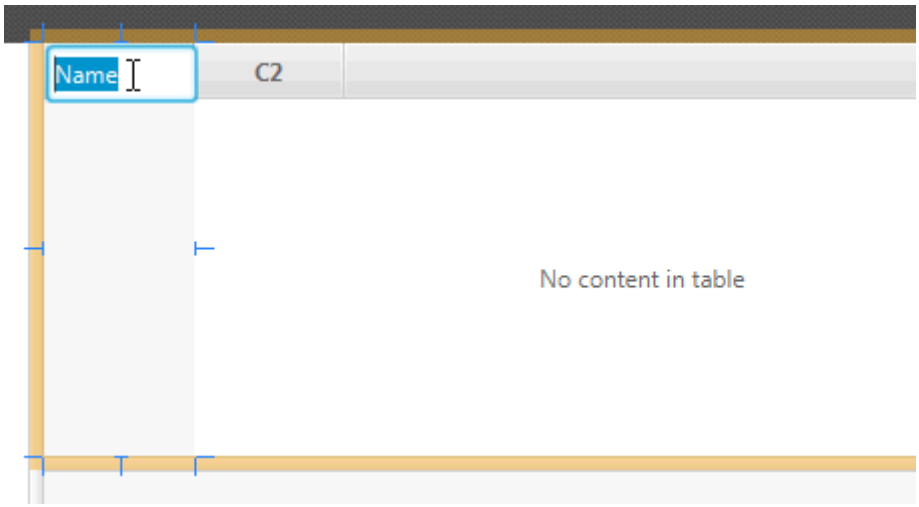


b. Select the second SplitPane node in the Hierarchy panel. In the Content panel, drag the horizontal divider up or down until the dividerPosition percentage displays about 35%. Alternatively, in the Properties section of the Inspector panel, enter `0.35` in the **Divider Positions** property text field.

6. Set the properties of the two columns in the table view.

a. In the Content panel, double-click the C1 column title and type `Name` to replace the default value, as shown in Figure 6–5.

Figure 6–5 Change TableColumn Title



b. In the **Code** section of the Inspector panel, select **colName** from the **fx:id** field's drop-down list.

c. Back in the Hierarchy panel, double-click the C2 text in the row for the second **TableColumn** component. Type in `Status` to replace the default Text property value of C2.

d. In the **Code** section of the Inspector panel, select **colStatus** from the **fx:id** field's drop-down list.

7. Add another column to the table view.

a. In the Library panel's **Search** box, clear any existing value and enter `column`.

b. Select the **TableColumn** control from the Library panel. Drag and drop it anywhere inside the table view in the Content panel. The new `ColumnX` column is added to the right of the Status column

c. Click the **Properties** section of the Inspector panel and set the **Text** property to **Synopsis**.

d. In the **Code** section of the Inspector panel, select **colSynopsis** from the **fx:id** field's drop-down list of available instance variables.

8. From the Menu bar, 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** from the Menu bar to turn off the display of the sample data.

9. Save all the changes you have made.

7

Create the Details Section

This chapter describes how to add the JavaFX GUI components for the Details section of the IssueTrackingLite GUI layout you are building with JavaFX Scene Builder. It also describes how to manage the resizing of the components when the application's window is resized.

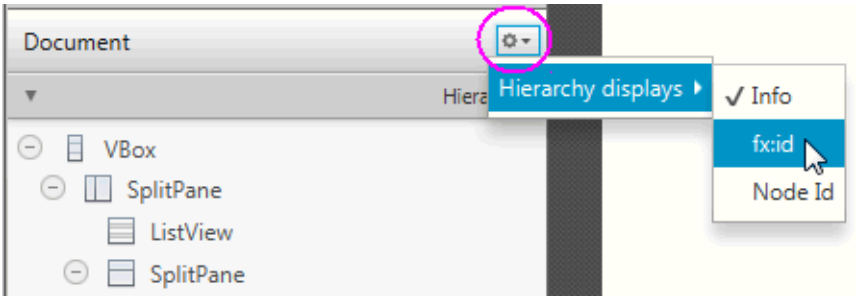
Use the following sections to set up the area where the issue's details will be displayed.

Add the GUI Components for the Details Section

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

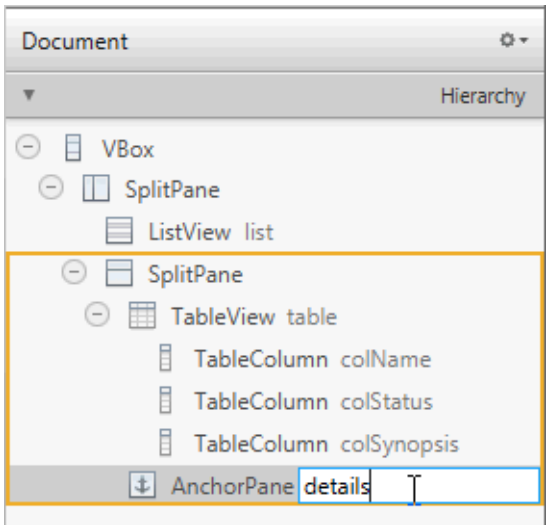
1. Click the menu button on the top right corner of the Hierarchy panel and select **Show fx:id**, as shown in Figure 7–1. By default, the Show info mode is selected. Notice that after you change the display mode to Show fx:id, the Hierarchy panel now displays the fx:id values next to the elements that have the fx:id property value assigned to them, as shown in Figure 7–2.

Figure 7–1 Show fx:id Display Mode



2. Set up the details section.
   a. In the Hierarchy panel, select the node for the only **AnchorPane** element.
   b. Double-click the right side of the row for the AnchorPane element to enter the fx:id inline edit mode. Enter `details` in the inline text editor for the fx:id text property, as shown in Figure 7–2. This editor is available because the current display mode in the Hierarchy panel is set to Show fx:id. Alternatively, click the **Code** section of the Inspector panel and select **details** from the drop-down list of instance variables available for the fx:id field.

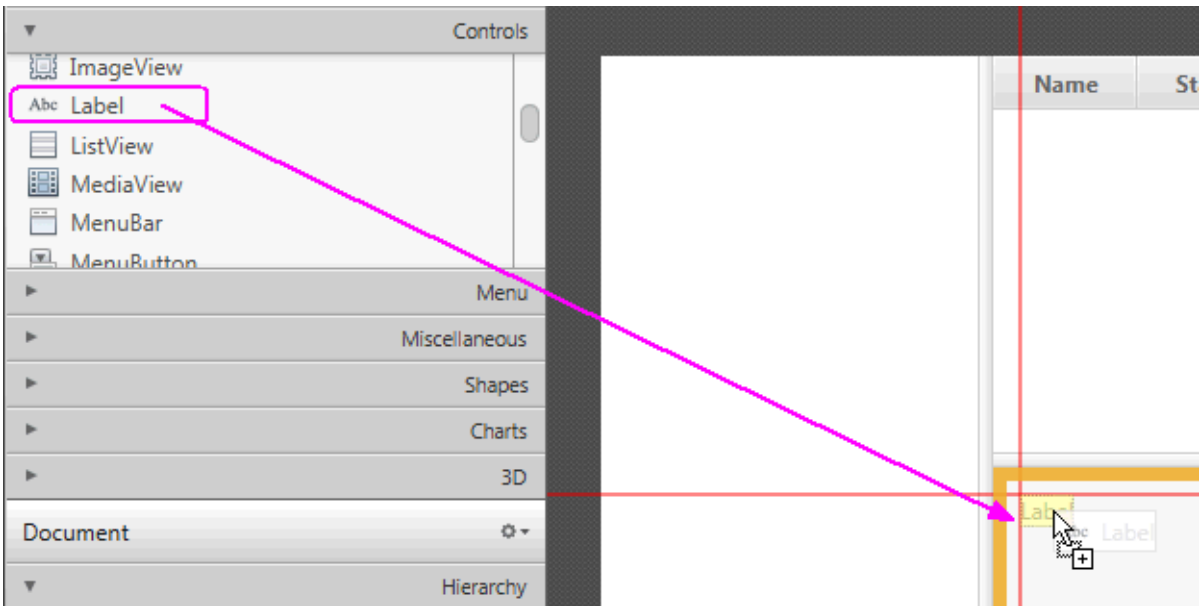
Figure 7–2 Use fx:id Inline Text Editor



3. Add a label.

   a. In the Controls section of the Library panel, drag the **Label** element and drop it on the upper left corner of the details area, as shown in Figure 7–3.

Figure 7–3 Add Label Element



   b. In the Content panel, double-click the new Label element to enter into inline edit mode. Enter `PROJECT / ID` in the **Text** property field to replace the default value.

   c. In the **Code** section of the Inspector panel, select **displayedIssueLabel** in the **fx:id** field's drop-down list of available instance variables.

   d. In the **Layout** section of the Inspector panel, set the value of **Min Width** to `USE_PREF_SIZE`. This setting will keep the labels of the HBox element to be

visible when the size of the application window is so reduced that not all of the GUI elements can be displayed.

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, as shown in Figure 7–4.

Figure 7–4 Resize the TextField Element

   c. Double-click the **Text Field** element in the Content panel to enter Edit mode. Enter `SYNOPSIS` in the editor box.

   d. In the **Code** 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 Content panel, hold the **Ctrl key** (on Windows and Linux platforms) or **Cmd key** (on Mac OS platform) to select the PROJECT/ID label and SYNOPSIS textfield components.

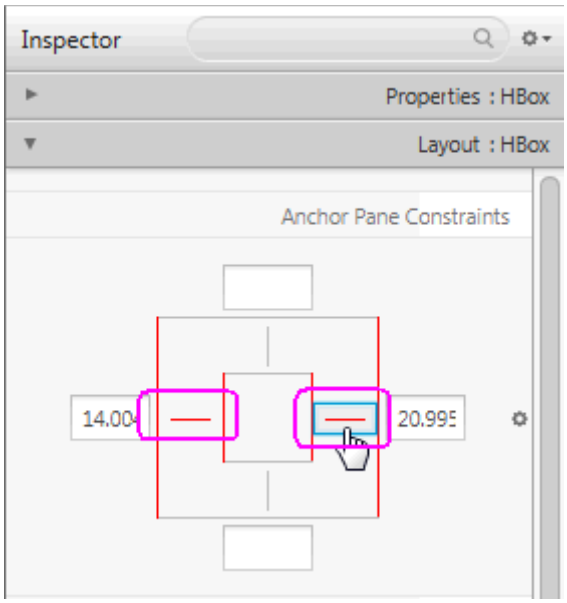
   b. From the Menu bar, select **Arrange**, then **Wrap in**, and then **HBox** from the submenu.

   c. In the Properties section of the Inspector panel, select **CENTER** for the Alignment property value of the HBox element.

   d. Click the **Layout** section and set the **Spacing** property value to **10**.

   e. In the AnchorPane Constraints sub-section, click the left and right black anchor lines. After you click the anchor lines, the black lines change into solid red lines, which are circled in Figure 7–5. This action anchors the HBox element's right and left borders to its container and ensures that when the window is resized, the HBox element is also resized.

Figure 7–5 Setting the Anchor Lines



f. In the Hierarchy panel, select the row for the **TextField SYNOPSIS** element and locate the **HBox Constraints** sub-section. Set the **Hgrow** property to **ALWAYS**.
This setting indicates that the Text Field: synopsis element will adjust horizontally when its parent container increases.

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 Content panel on 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.

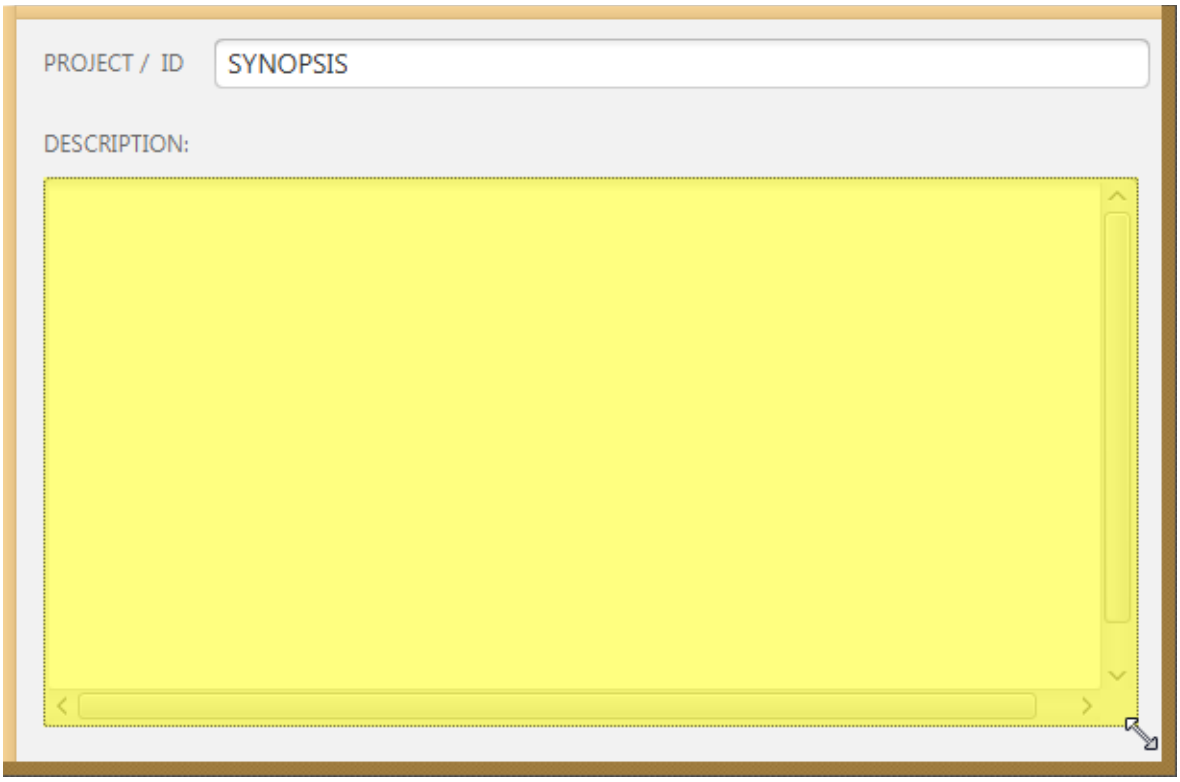
c. Drag and drop a **Text Area** below the label that you just added.

d. In the Layout section of the Inspector panel, locate the Anchor Pane Constraints sub-section and click the left, top, right, and bottom black anchor lines.

e. In the **Code** section of the Inspector panel, select **descriptionValue** from the drop-down list for the fx:id field.

f. In the Content panel, click and drag the lower right handle of the Text Area element to increase its size and fill the remaining space in the details section, as shown in Figure 7–6.

Figure 7–6 Enlarge the Text Area



8

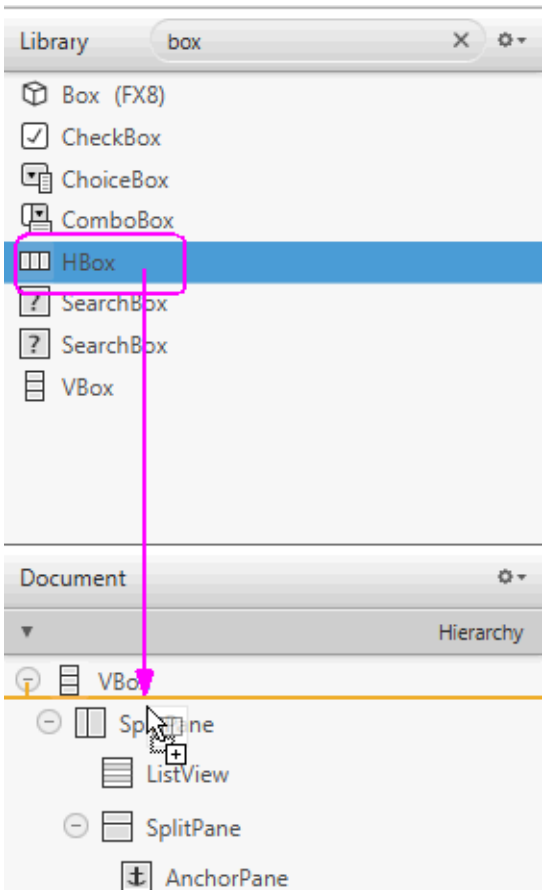
Add the Toolbar

This chapter gives the steps to add a tool bar GUI control to the layout that you are building with JavaFX Scene Builder.

Use the following steps to add a toolbar to the top portion of the IssueTrackingLite's GUI layout. It will contain an image file and three buttons.

1. In the Document panel, change the display setting by clicking the Hierarchy display menu button on the top right corner and choosing **Info**, if it is not already chosen.
2. Drag an **HBox** container element from the Library panel to the Hierarchy section of the Document panel and drop it under the row for the VBox element, as shown in Figure 8–1.

Figure 8–1 Add an HBox Control



3. Add three buttons to the toolbar.
   a. From the Library panel, drag a **Button** element to the Hierarchy panel and drop it in the HBox node that was just added.
   b. In the Content panel, right-click the new **Button** and select **Duplicate** from the contextual menu to add a second button.
   c. Repeat the previous step to add the third button to the toolbar.
4. Edit the button details.
   a. In the Content panel, double-click the leftmost button in the toolbar to get into edit mode and enter `New`. Click the **Code** section of the Inspector panel, select **newIssue** in the drop-down list for the fx:id property for the New button. 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)`, which is defined in the controller source code, will be executed.
   b. Double-click the middle button to get into edit mode. Enter `Save` in the edit box. Click the **Code** section of the Inspector panel, and select **saveIssue** in the drop-down list for the fx:id property for the middle button. In the **On Action** field, set the value to **`#saveIssueFired`**.

c. Select the rightmost button in the toolbar. Click the **Code** section of the Inspector panel, and select **deleteIssue** from the fx:id property's drop-down list. In the Properties section of the Inspector panel, enter `Delete` in the **Text** field. In the On Action field, set the value to **`#deleteIssueFired`**.

5. Put the buttons in a container and adjust the spacing between them.

   a. Hold down the Ctrl-key and select each of the three buttons. From the Menu bar, select **Arrange**, **Wrap in**, and then **HBox**. The buttons are arranged in a row with even spacing between them.

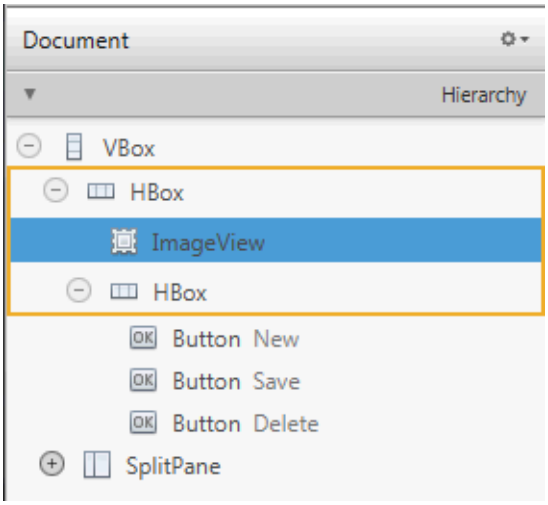
   b. Select the **Layout** section of the Inspector panel and set the value for the **Spacing** property to `15`. Notice that the buttons are adjusted to have more spaces in between them.

6. Add an image to the toolbar.

   a. From the Menu bar, select **File**, **Import,** and then **Media**. Select **IssueTrackingLite.png** from the *<install-dir>*`/IssueTrackingLite/src/issuetrackinglite` folder. The ImageView element is added at the bottom of the Hierarchy panel.

   b. In the Hierarchy section of the Document panel, drag the ImageView element so that it is the first element in the top HBox container, as shown in Figure 8–2. Notice that in the Content panel, the image is moved to the left of the HBox container for the three buttons.

Figure 8–2 Move the ImageView Element to the HBox Container



7. Modify the default values for some of the Layout properties for the first HBox container, so that they match the property values circled in Figure 8–3.

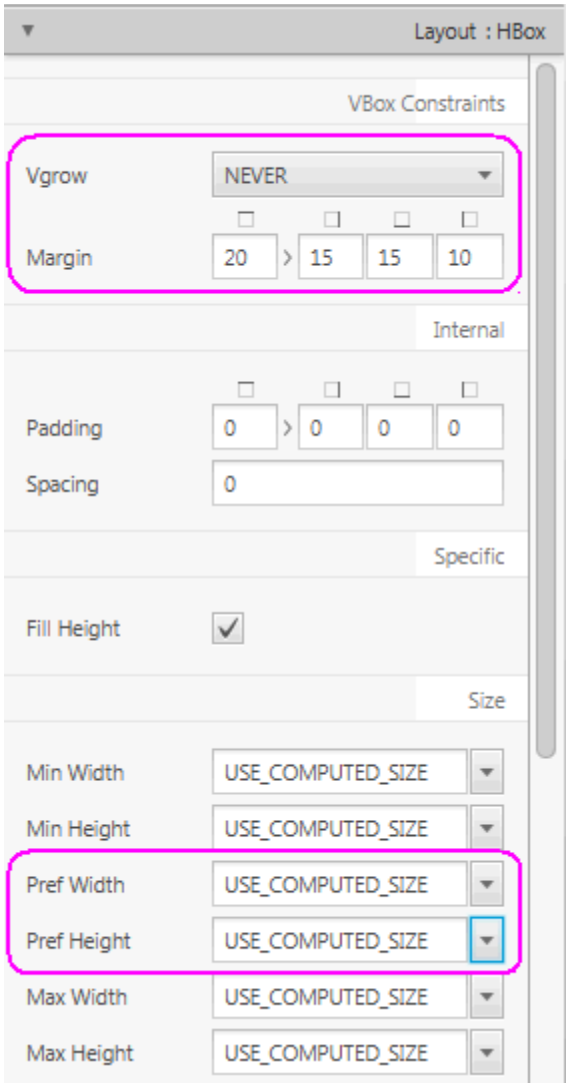
   a. In the Hierarchy panel, select the row for the **top HBox** container.

   b. Click the **Layout** section of the Inspector panel and in the **VBox Constraints** sub-section, set the `Vgrow` property value to `NEVER`.

   c. Change the default values for the Margin property to `20`, `15`, `15`, and `10`, respectively.

   d. In the Size sub-section, change the **prefWidth** and **prefHeight** default values to `USE_COMPUTED_SIZE`.

Figure 8–3 Modify the HBox Layout Default Properties



8. Modify some of the default property values for the second HBox.
   a. In the Hierarchy panel, select the **HBox** element that contains the three buttons.
   b. Select the **Layout** section of the Inspector panel and change the value of the Hgrow property to `ALWAYS`.
   c. Select the **Properties** section of the Inspector panel and in the Node sub-section, change Alignment property value to `CENTER_RIGHT`. The HBox with the three buttons are shifted to the right side of the Content panel.
9. Select **File** and then **Save** from the Menu bar to save your work.

9

Use a Style Sheet and Preview the UI

This chapter describes how you can use the JavaFX Scene Builder to preview the FXML layout that you just created and also how to apply a style sheet to customize the look and feel of the IssueTrackingLite application.

Use the following sections to preview the GUI layout that you have created and then change its look and feel by using a style sheet.

Preview the UI

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

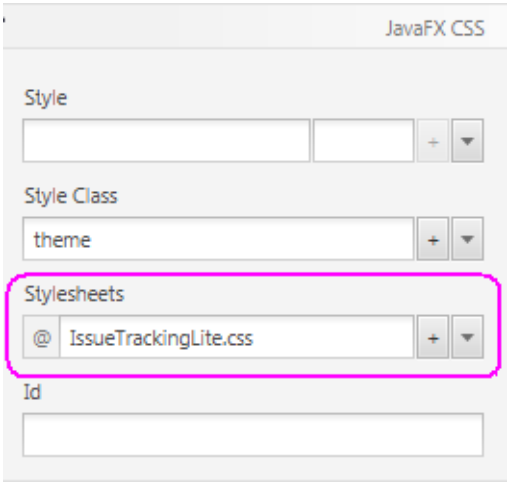
1. From the Menu bar, select **Preview**, and then select **Show Preview in Window**.
2. Resize the window multiple times to ensure that the buttons in the toolbar and the text area resize appropriately when the window is resized. You can also make modification to the layout and the Preview window is updated
3. To stop viewing the preview, close the Preview window.

Use a Style Sheet

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

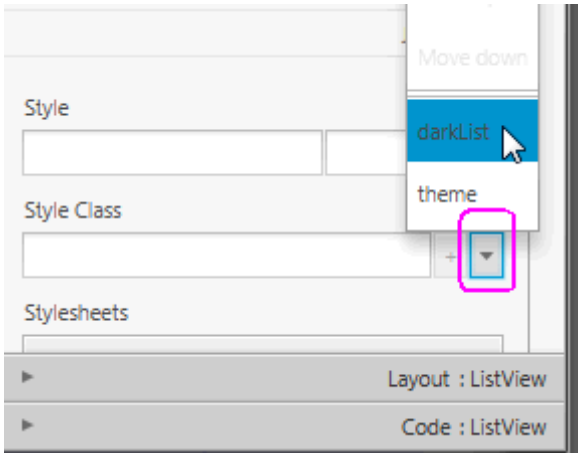
1. Verify that the Cascading Style Sheet (CSS) resource file that is bundled with the IssueTrackingLite sample is already set. In the Hierarchy panel, select the root **VBox** container. Click the **Properties** section of the Inspector panel. In the Stylesheets text field of the JavaFX CSS subsection, notice that the `IssueTrackingLite.css` style sheet is already set, as shown in Figure 9–1. This is the style sheet that was set when you created the FXML file.

Figure 9–1 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 downward arrow in the **Style Class** list. Select **darkList** as shown in Figure 9–2. Notice that the appearance of the ListView element in the Content panel has changed to a dark grey color.

Figure 9–2 Adding a Style Class to the ListView Element



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

You just completed building the FXML layout for a JavaFX application using JavaFX Scene Builder. Continue with the Compile and Run the Application to compile and run the IssueTrackingLite application.

10

Compile and Run the Application

This chapter steps you through using NetBeans IDE or the Apache Ant utility to compile and run the IssueTrackingLite application for which you build the FXML layout using JavaFX Scene Builder.

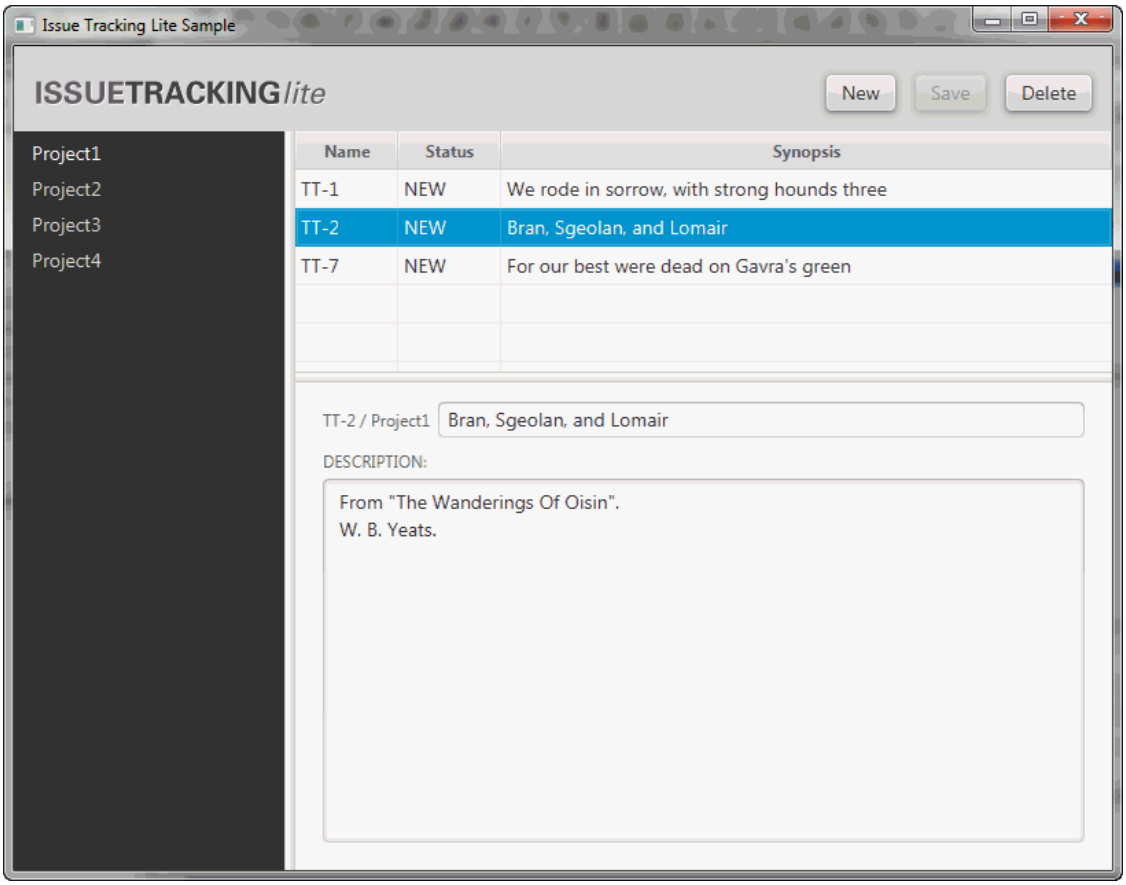
Use either of the sections below to see the IssueTrackingLite application in action.

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 8 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 10–1. Click the image to enlarge it.

Figure 10–1 Completed GUI Layout for Issue Tracking Sample



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 10–1.
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.

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 10–1. Note that the examples shown use JDK 8.

Example 10–1 Apache Ant Command to Run the Application

```
ant -f <JavaFX_App_Name>/build.xml <TARGET>
```

In the above example, the main values for <*TARGET*> are `clean`, `jar`, and `run`. For example, to run the IssueTrackingLite application on the Windows or Mac OS

platform, enter something similar to the command in Example 10–2. You can set <*TARGET*> with the value of `-projecthelp` to get a list of available targets.

Example 10–2 Using Apache Ant to Run IssueTrackingLite

```
ant -f IssueTrackingLite/build.xml run
```