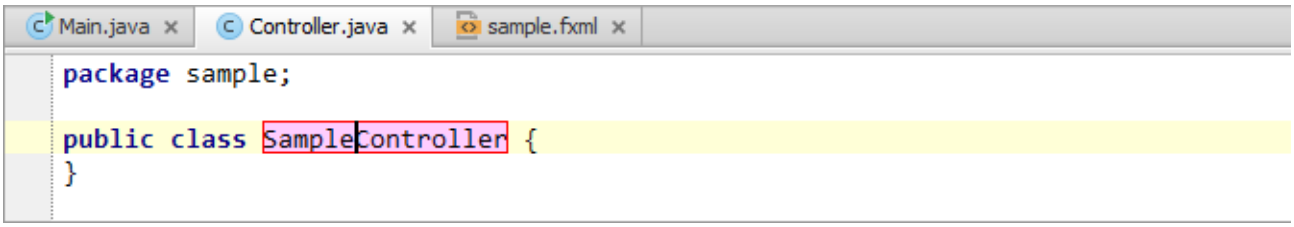
Developing a JavaFX Hello World Application: Coding Examples

In this topic, we transform the sample application created by IntelliJ IDEA into a very basic JavaFX Hello World application. In this way, we show basic coding assistance features provided by the IDE. (The sample application is created by IntelliJ IDEA automatically when you create a project for your JavaFX application development from scratch, see Creating a project for JavaFX application development from scratch.)

Renaming the Controller class

To adjust the sample application to your needs, you may want to start by renaming the files. To see how, let's perform the Rename refactoring for the class `Controller`. We'll rename this class to `SampleController`. You can use a different name if you like.

1. In the editor, place the cursor within the class name and select **Refactor | Rename** (alternatively, press **Shift+F6**).
2. Place the cursor in front of `Controller` and type `Sample`.



3. Press **Enter** to indicate that you have completed the refactoring.

Now, switch to `sample.fxml` in the editor and note that the value of the `GridPanel fx:controller` attribute has changed to `"sample.SampleController"`. (Initially it was `"sample.Controller"`.)

In a similar way you can change the names of other files if necessary.

Developing the user interface

To show you how IntelliJ IDEA can help you write your code, let's implement a kind of Hello World JavaFX application.

In the user interface (UI), we'll define a button which when clicked will display the text `Hello World!` To do that, we'll add the following two elements between the opening and closing `<GridPane>` tags in the file `sample.fxml`:

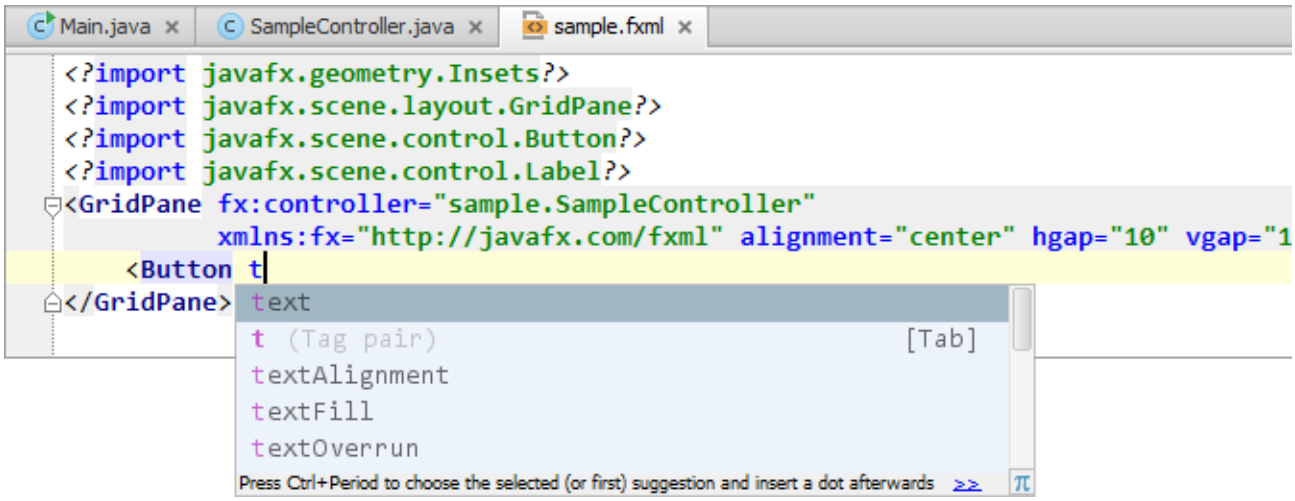
```
<Button text=
        "Say 'Hello World'" onAction=
        "#sayHelloWorld"/>
<Label GridPane.rowIndex=
        "1" fx:id=
        "helloWorld"/>
```

We suggest that you do everything by typing to see how code completion works.

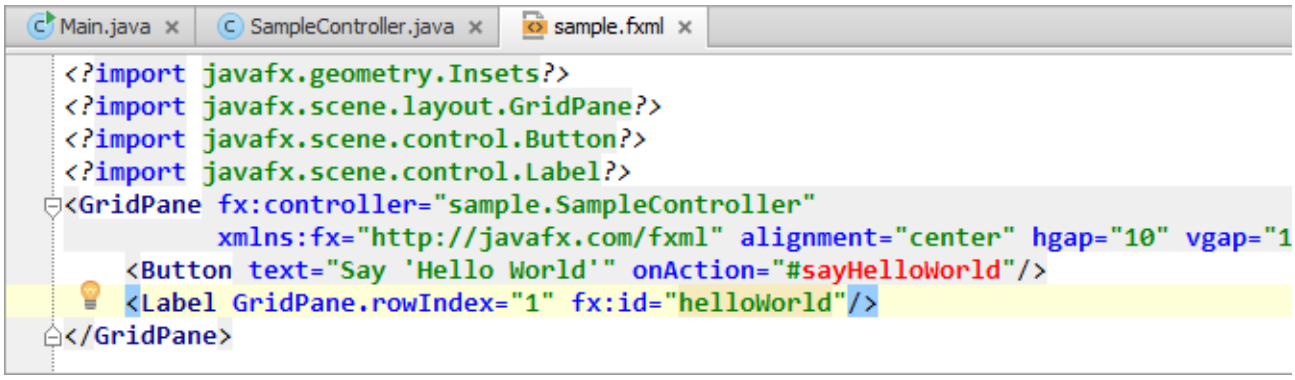
1. Go to the end of the opening `<GridPane>` tag and press **Enter** to start a new line.

2. Type `<B` and select **Button**.

3. Type space, type `t`, and select **text**.



4. In a similar way, add the remaining code fragments. The resulting code will look something similar to this:



As you see, `sayHelloWorld` is shown red and `helloWorld` is also highlighted. This means that IntelliJ IDEA cannot resolve the corresponding references.

To resolve the issues, let's use the quick fixes suggested by IntelliJ IDEA.

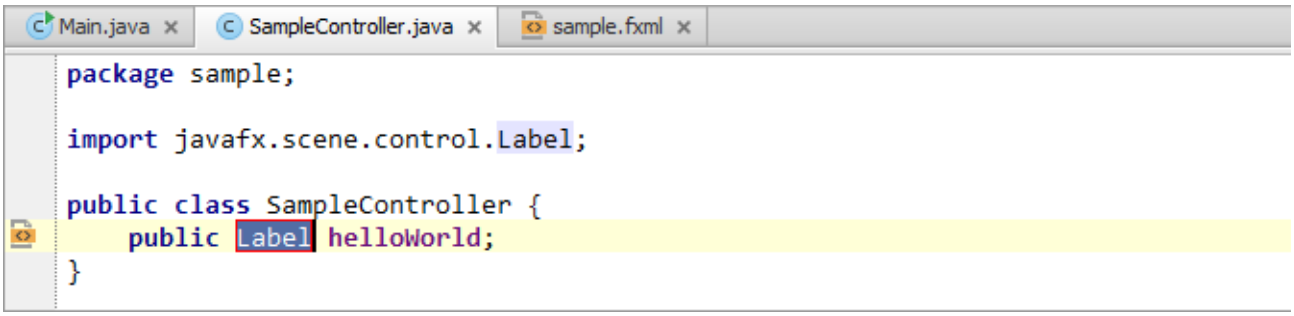
(In IntelliJ IDEA, it's a standard coding practice when you reference a field, method or class that doesn't yet exist and then use a quick fix to create the corresponding field, method or class.)

Completing the code for the SampleController class

Now we are going to define the field `helloWorld` in the `SampleController` class. We will also add the corresponding event handler method (`sayHelloWorld`) that will set the text for the `helloWorld` label. When doing so, as already mentioned, we'll use the quick fixes suggested by IntelliJ IDEA.

1. In `sample.fxml`, place the cursor within `helloWorld`. Click the yellow light bulb or press **`Alt+Enter`**.
2. Select **Create Field 'helloWorld'**.

IntelliJ IDEA switches to `SampleController.java` where the declaration of the field `helloWorld` has been added.

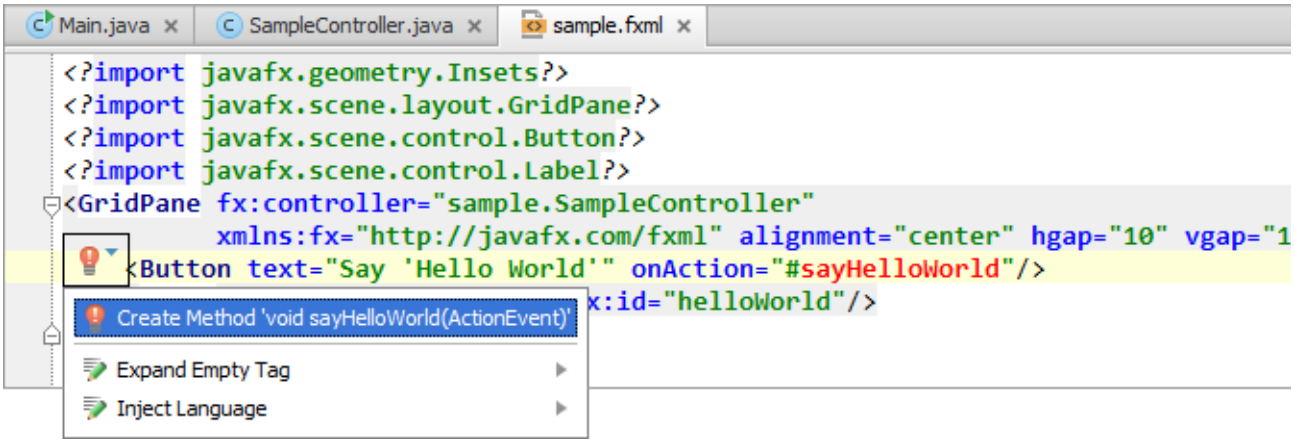


Note the red border around `Label`. You can edit the field type right away. We are not going to do that now, so press **`Enter`** to quit the refactoring mode.

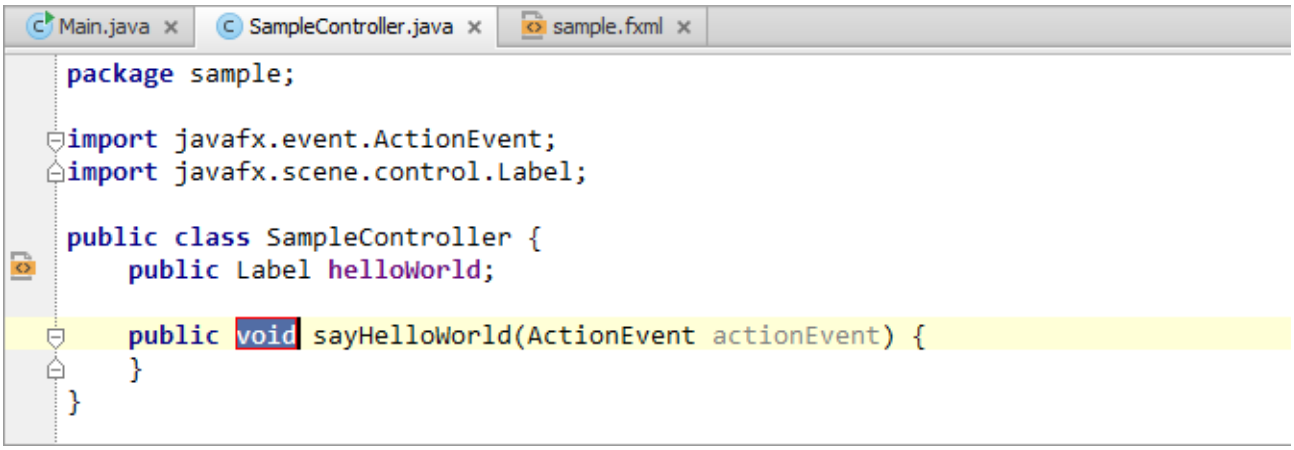
Also note the import statement that has just been added (`import javafx.scene.control.Label;`) and the icon to the left of the field declaration. This is a navigation icon; click it to go back to `sample.fxml`.

3. Place the cursor within `sayHelloWorld` and press **`Alt+Enter`**.

4. Select **Create Method 'void sayHelloWorld(ActionEvent)'**.

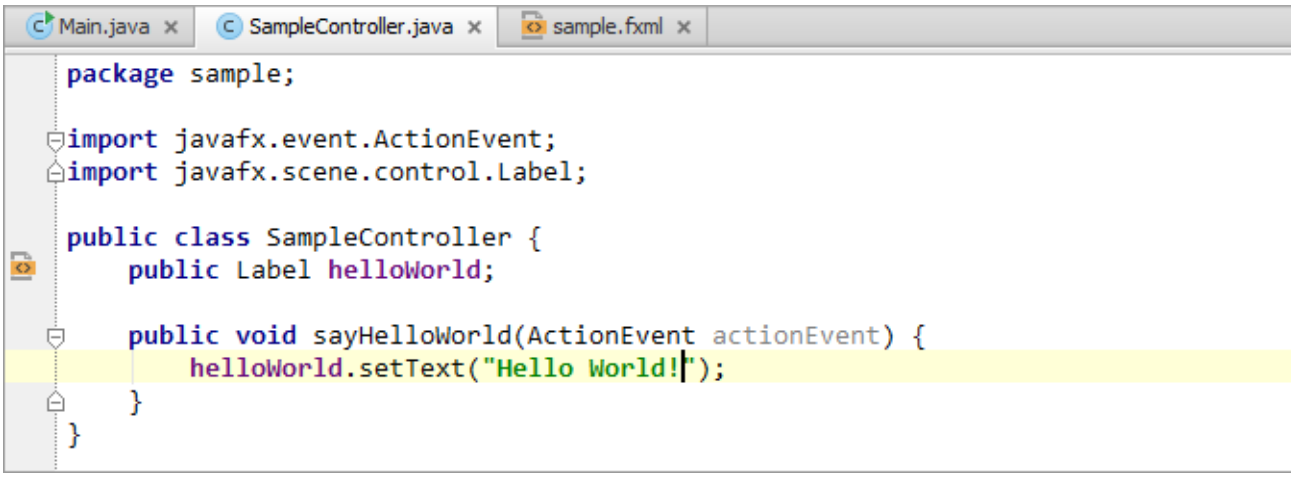


The corresponding method declaration is added to `SampleController.java`.



5. Press `Shift+Enter` to quit the refactoring mode and start a new line.
6. Type the following to set the text for the label:

```
        helloWorld.setText("Hello World!");
```



At this step, the code of the application is ready. Let's run the application to see the result.

Running the application

1. To run the application, click ▶ on the toolbar or press `Shift+F10`.

   The application window now contains the **Say 'Hello World'** button.

2. Click this button to see that the text **Hello World!** is shown.

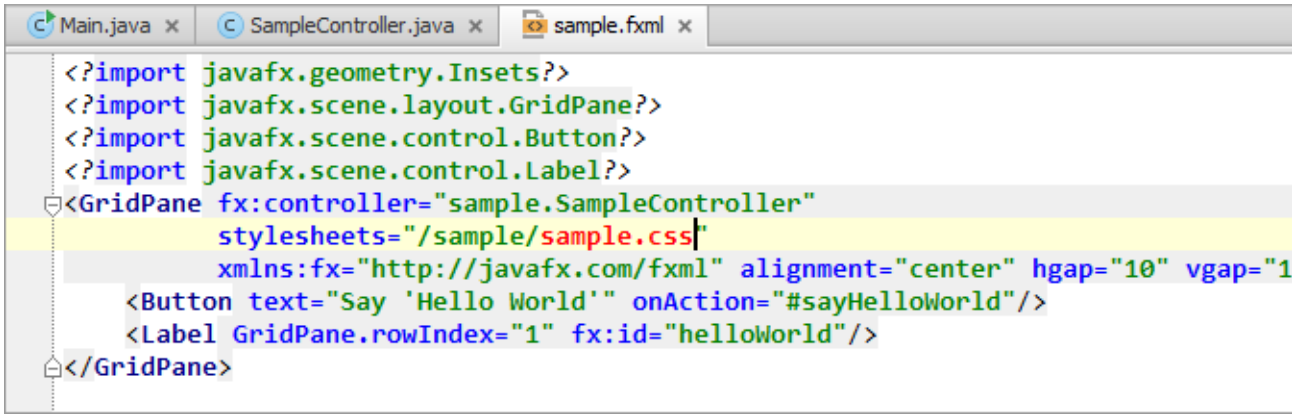
3. Close the application window.

Styling the UI with CSS

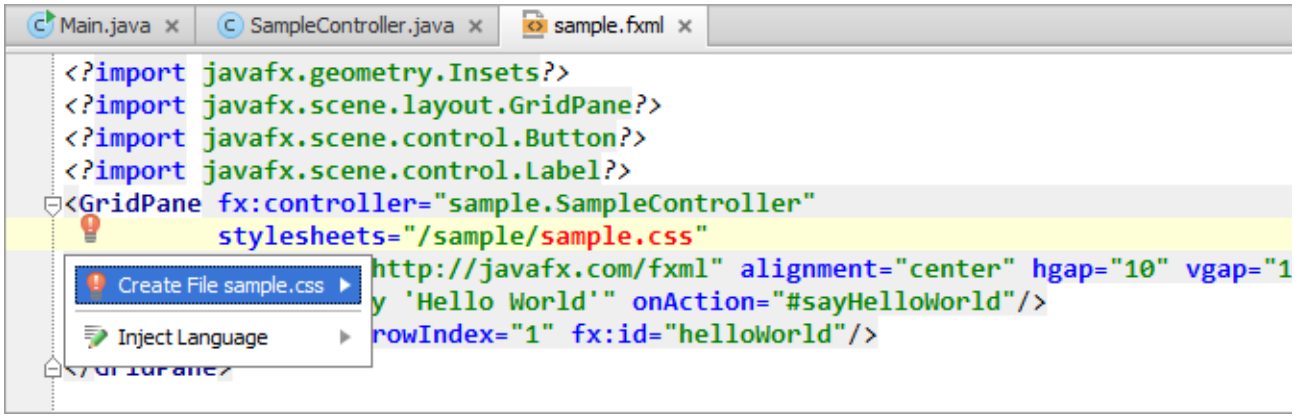
To complete the coding examples, let's change the appearance of the UI by adding a stylesheet and defining a couple of formatting styles in it.

1. In the file `sample.fxml`, add a reference to a (non-existing) CSS file `sample.css`. One way to do that is to add the `stylesheets` attribute within the opening `<GridPane>` tag, e.g.

```
            stylesheets=
            "/sample/sample.css"
```



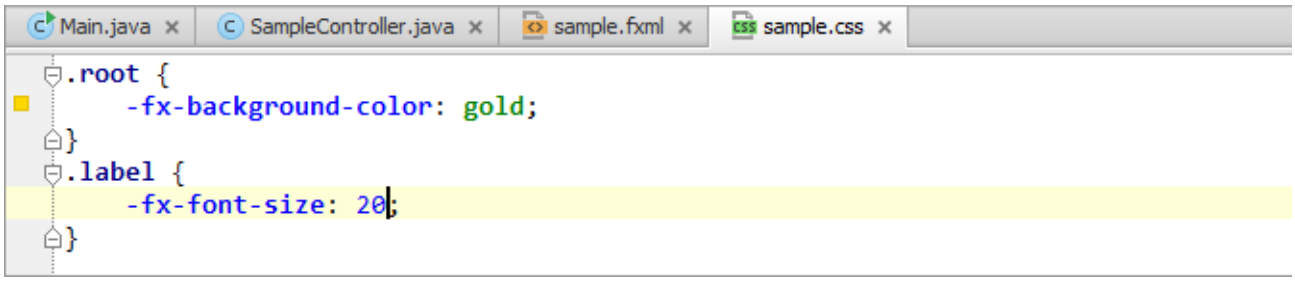
2. As before, use a quick fix to create the CSS file.



3. When the CSS file is created, add the following style definitions into it.

```
.root {
    -fx-background-color: gold;
}
.label {
    -fx-font-size: 20;>
}
```

The first of the styles makes the background in the application window "gold" and the second one - sets the font size for the text `Hello World!` to 20 pixels.



4. Run the application again to see the result (**Shift+F10**).

Now that you've brought the application to a reasonable state, you may want to package it. For corresponding instructions, see Packaging JavaFX Applications.

See Also

Language and Framework-Specific Guidelines:

- JavaFX

Web Resources:

- Developer Community