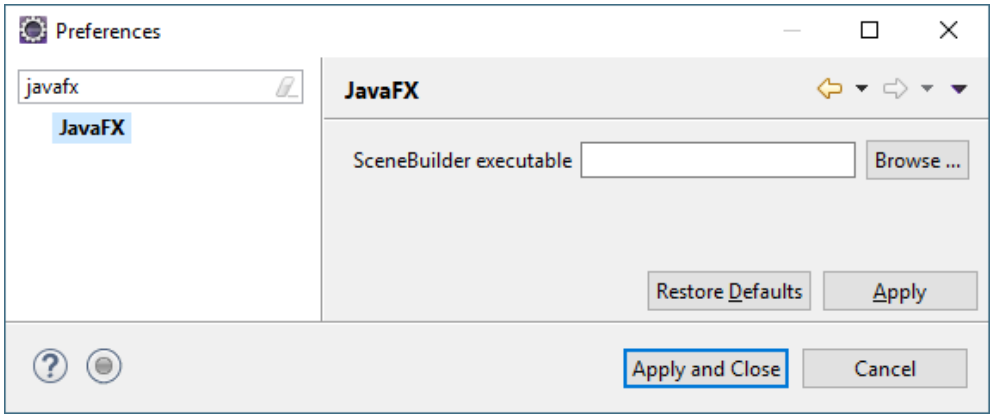
Dieses Tutorial basiert auf einem Tutorial von Andreas Pomarolli [https://examples.javacodegeeks.com/desktop-java/javafx/fxml/javafx-fxml-tutorial/#binding]. Es wurde für unsere Veranstaltung angepasst.

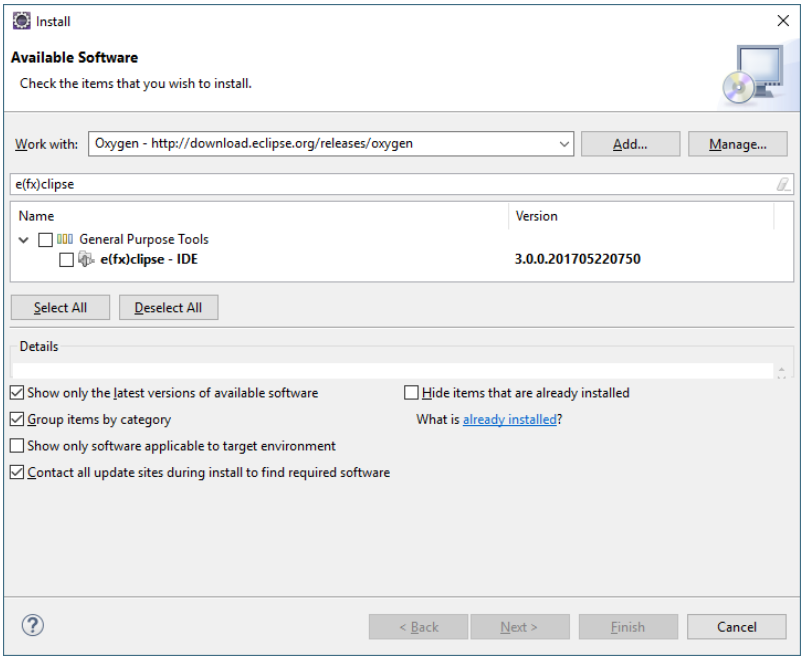
Schritt 1: Notwendige Software installieren:

1. **SceneBuilder** → http://gluonhq.com/products/scene-builder/
   a. Installieren Sie die Version „Scene Builder for Java 8".
   b. Setzen Sie den Pfad zu Ihrer SceneBuilder-Installation in Eclipse (Window → Preferences → Search: JavaFx):



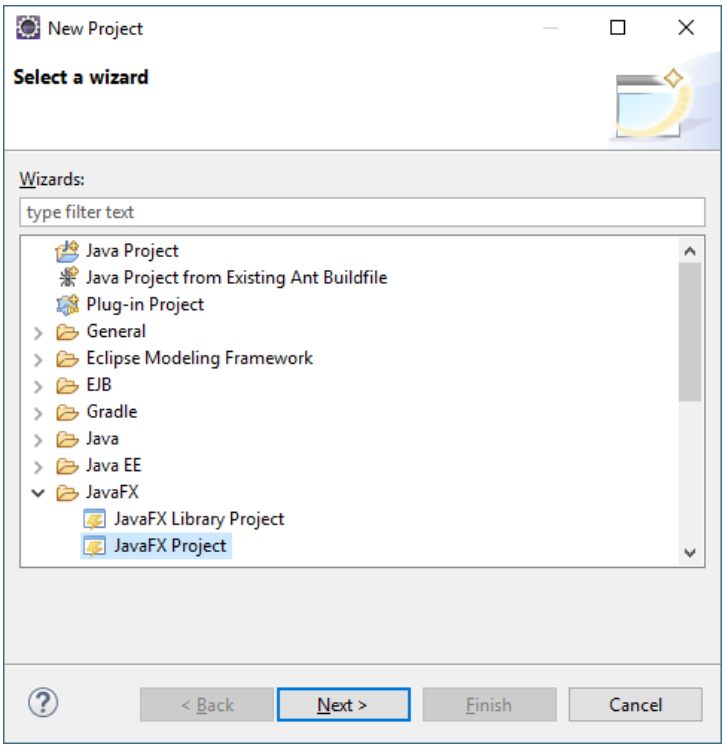
Im Labor 4/319: `C:\Users\student\AppData\Local\SceneBuilder\SceneBuilder.exe`

2. **E(fx)clipse** →in Eclipse: Help → Install New Software:

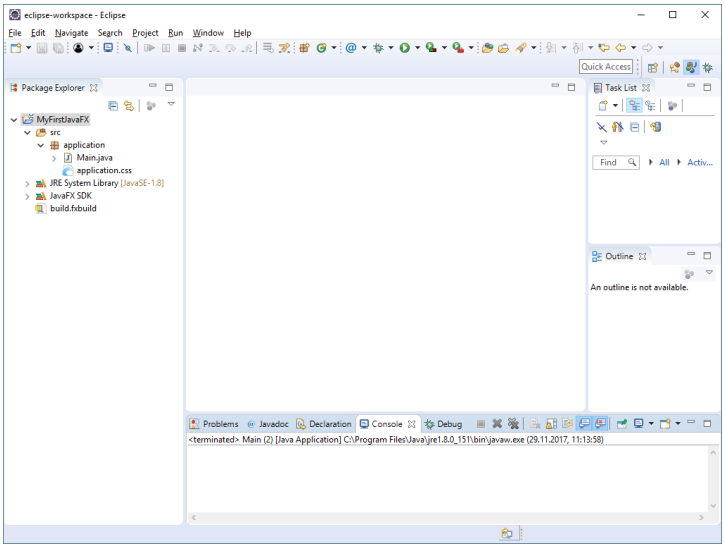


- Falls Sie weitere Unterstützung benötigen: http://www.eclipse.org/efxclipse/install.html

Schritt 2: Ein JavaFX-Projekt anlegen

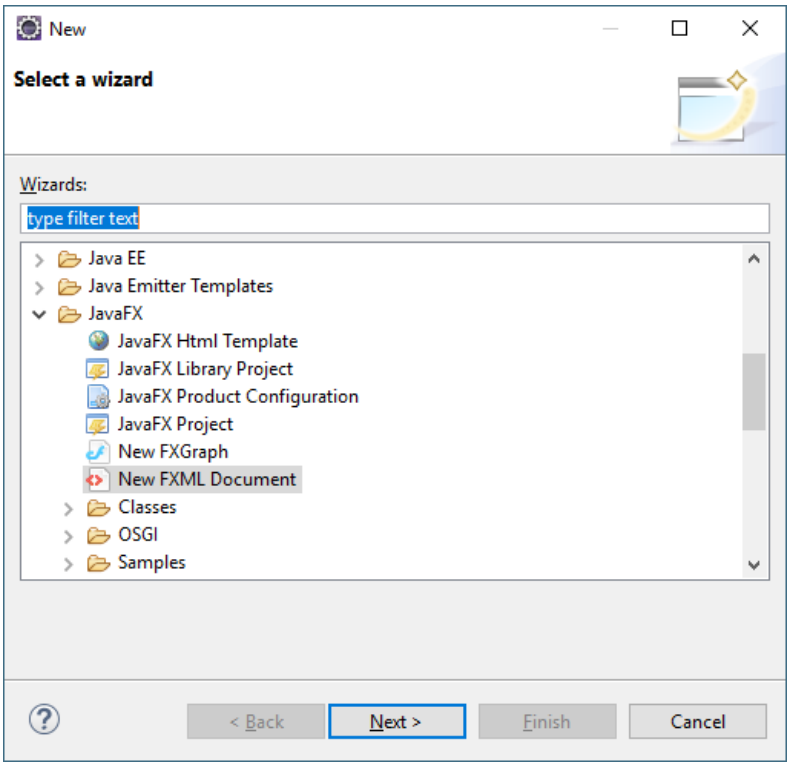


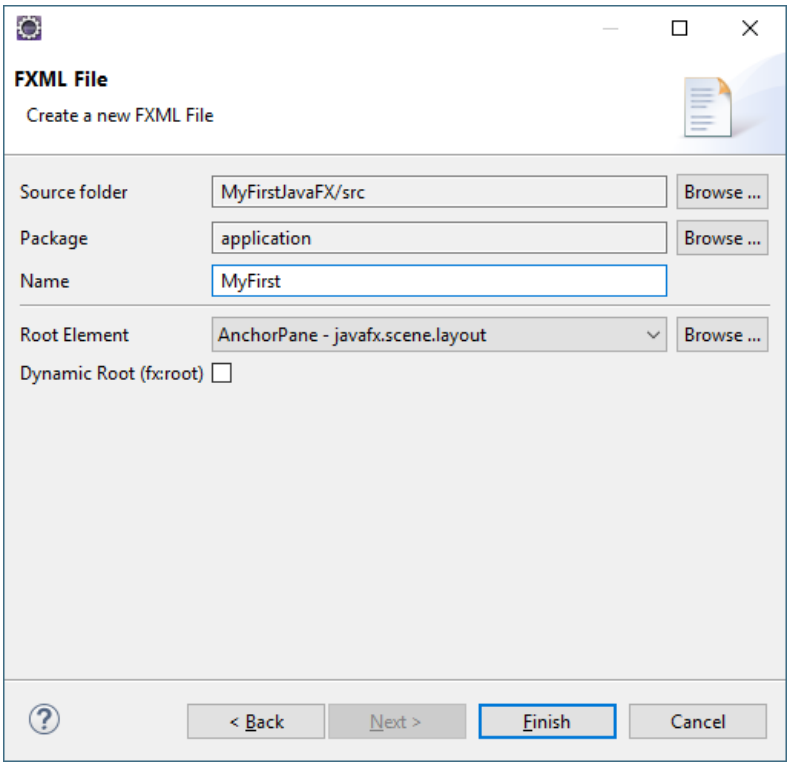
File → New → Project → JavaFX → JavaFX Project



Schritt 3: Eine neue FXML-Datei anlegen

`Rechtsklick auf MyFirstJavaFX/src/application → New → Other → New FXML Document`





Dieses Beispiel nutzt ein AnchorPane-Element als Wurzel. → Finish

Wir könnten die Datei direkt bearbeiten. Komfortabler wird es jedoch mit dem (in Schritt 1 bereits installierten SceneBuilder).

Schritt 4: FXML im SceneBuilder öffnen

```
Rechtsklick auf MyFirstJavaFX/src/application/MyFirst.fxml → Open with SceneBuilder
```

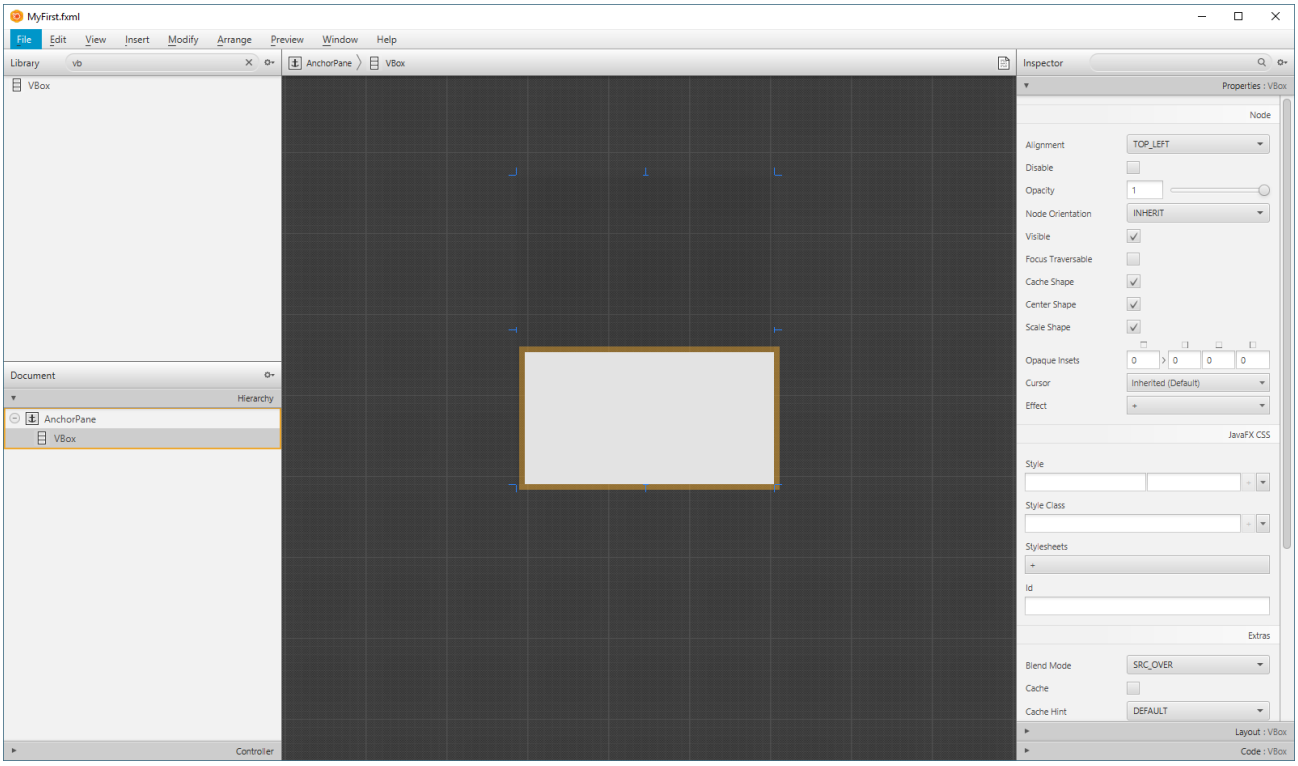
Wie in der Vorlesung erläutert ist FXML ist eine XML-basierte Sprache zur Beschreibung von Benutzeroberflächen unter JavaFX. Mit FXML können die grafischen Elemente Ihrer GUI beschrieben und eine Beziehung zur Ablauflogik hergestellt werden. Die Ablauflogik selbst wird separat mit JavaFX-Packages in Java entwickelt (Trennung von GUI und Logik).

Der SceneBuilder besteht aus folgenden Komponenten:

- Die **Menu Bar** bietet Zugriff alle Funktionen des JavaFX Scene Builder
- Die **Path, Selection and Message Bar** zeigt den Pfad für ausgewählte Elemente und erlaubt die Anwahl eines spezifischen Elements. Zusätzlich werden hier Fehlermeldungen ausgegeben.
- Das **Content Panel** zeigt die aktuelle Scene als Container für GUI Elemente im Rahmen des FXML Layout. Zu Beginn wird hier üblicherweise ein leerer Container dargestellt.
- Das **Library Panel** zeigt verfügbare JavaFX GUI Elemente. Diese können dem existierenden Layout per Drag and Drop hinzugefügt werden. Mit der Suchfunktion kann die Anzeige der Elemente gefiltert werden.
- Das **Document Panel** beinhaltet eine Darstellung der aktuellen (Baum) Struktur des Layouts *(Hierarchy)* und bietet einen Überblick über die eingefügten Komponenten Ihrer GUI einschließlich der vergebenen „fx:id"-Werte *(Controller)*.
- Das **Inspector Panel** beinhaltet „Properties, Layout, und Code" Komponenten. Hiermit können Eigenschaften der GUI Elemente bearbeitet werden. Zusätzlich wird eine Suchfunktion für Eigenschaften geboten. Unter Code kann die fx:id für ein Element gesetzt werden.

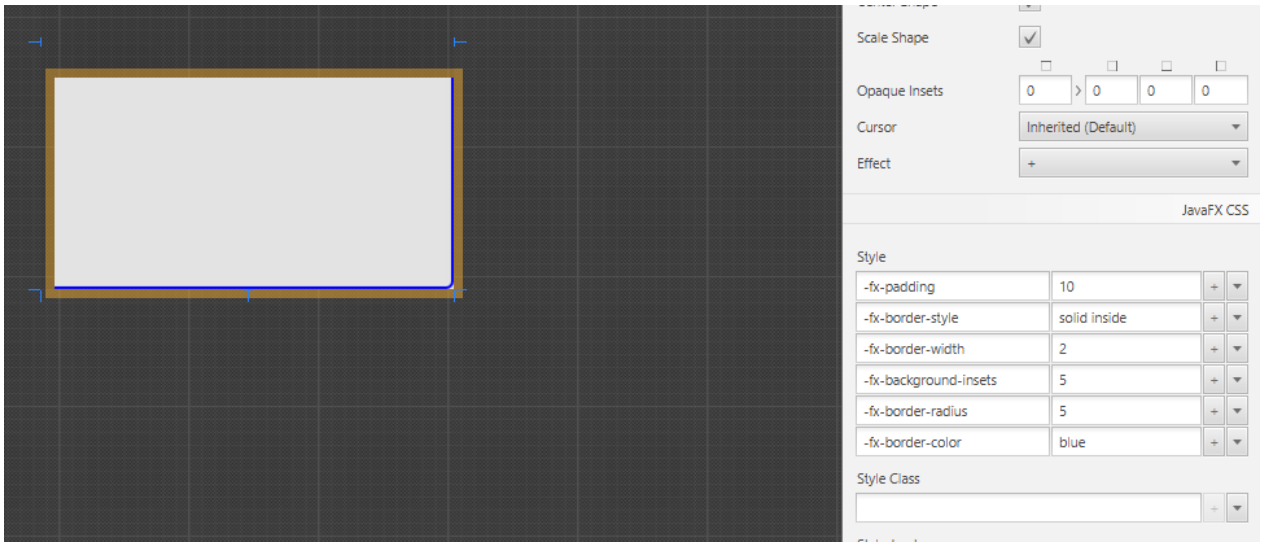
Schritt 5: Oberfläche im SceneBuilder gestalten

Bisher enthält unsere GUI lediglich ein AnchorPane (siehe unten links → Document/Hierarchy). Wir fügen nun einen VBox-Container hinzu. Wählen Sie VBox im Library/Containers Panel und ziehen Sie das Element auf das Content Panel.



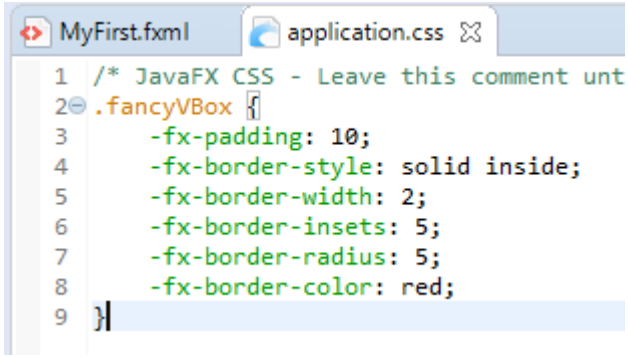
Im Inspector Panel ändern wir nun optische Eigenschaften dieses Elements. Wählen Sie im Document Panel in der Hierarchy das VBox-Element und setzen Sie folgende Werte im Inspector Panel (unter Properties/JavaFX CSS):

```
1 -fx-padding: 10;

2 -fx-border-style: solid inside;

3 -fx-border-width: 2;

4 -fx-border-insets: 5;

5 -fx-border-radius: 5;

6 -fx-border-color: blue;
```



Alternativ könnten wir das Erscheinungsbild unserer Elemente auch in der CSS-Datei unseres JavaFX-Projekts gestalten. Dazu können wir die CSS-Style-Klasse und die CSS-Datei im Inspector-Panel angeben. Die Änderungen aus der CSS-Datei werden dann im SceneBuilder angezeigt.

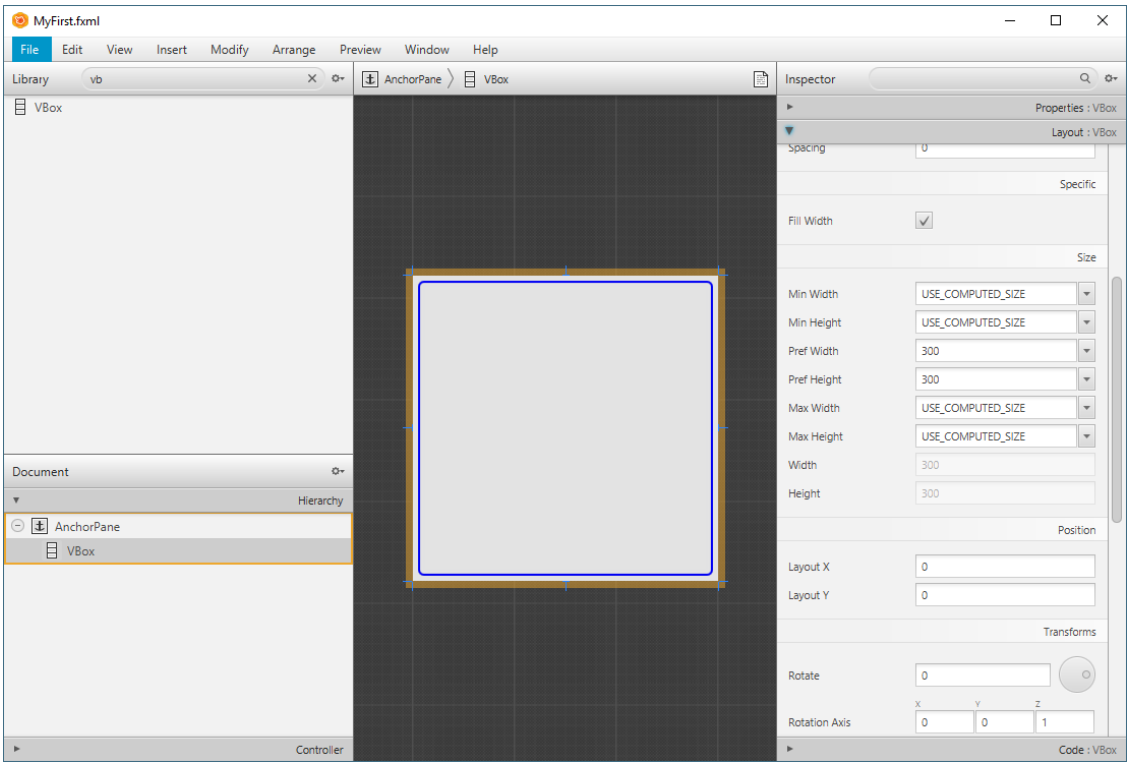
im SceneBuilder



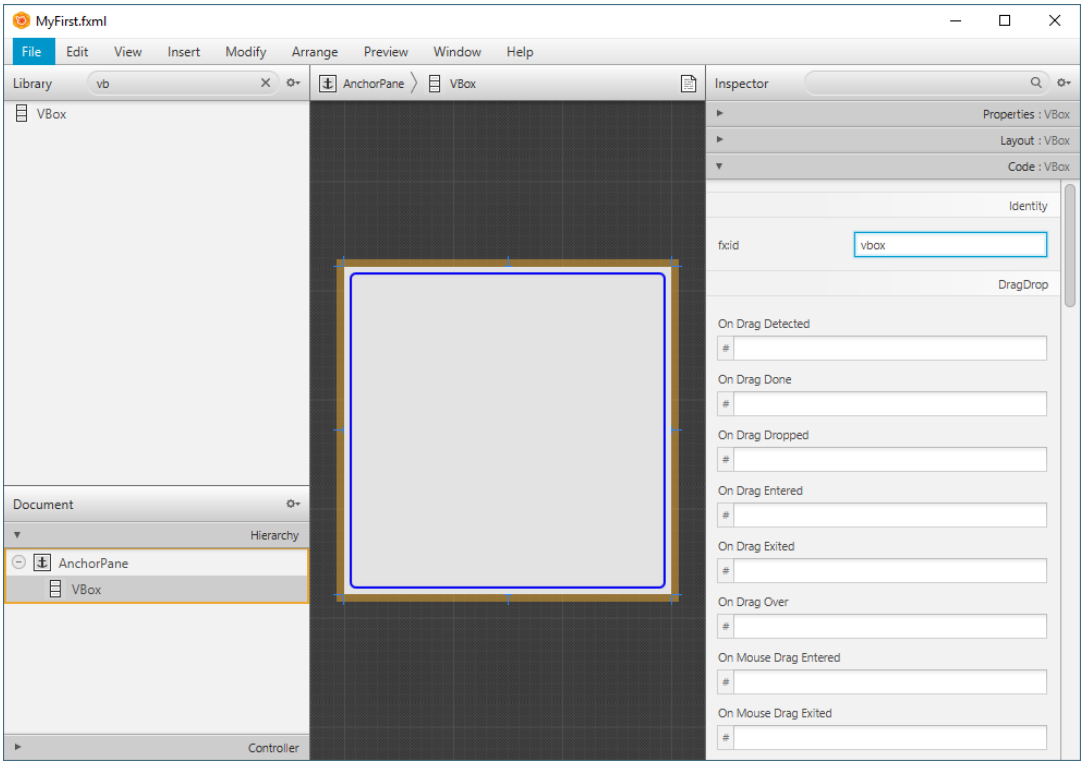
in Eclipse

Weitere Informationen zu CSS in JavaFX finden Sie unter:
https://docs.oracle.com/javase/8/javafx/api/javafx/scene/doc-files/cssref.html#introscenegraph

Im Document/Hierarchy Panel selektieren Sie das `VBox` Element und wählen Sie den setzen Sie im Inspector/Layout-Panel „`Preferred Width`“ und „`Preferred Height`“ auf `300px`.

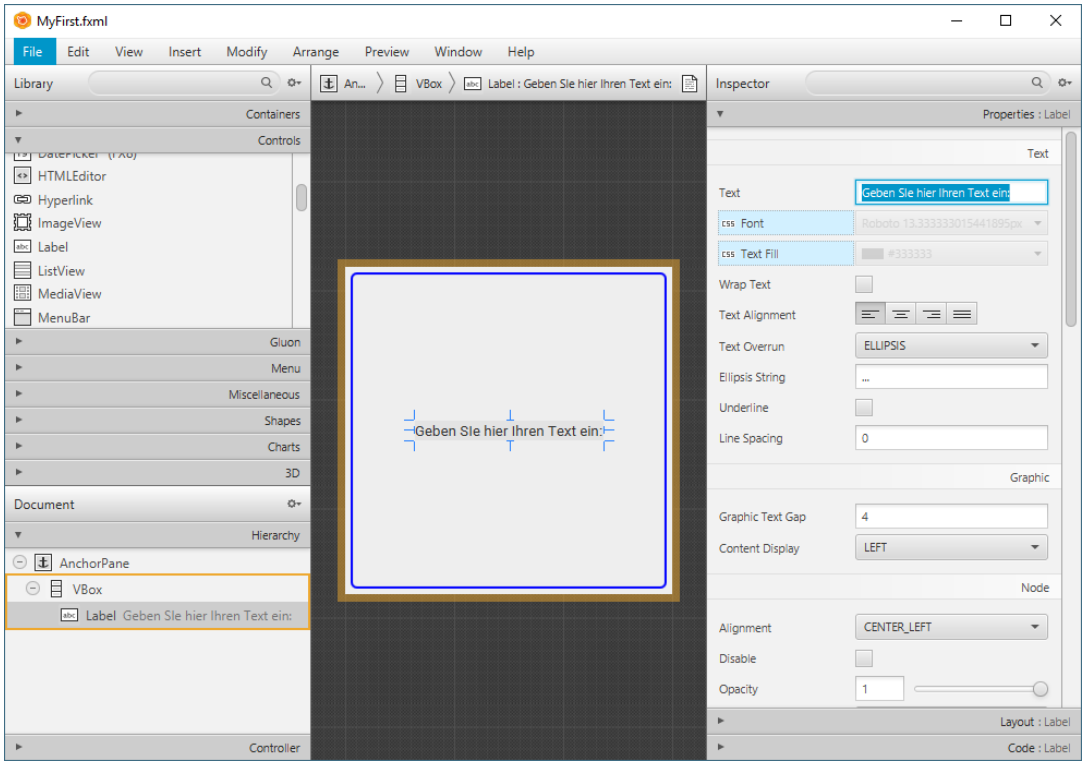


FXML-Elemente werden später im Java-Code über sog. „`fx id`“ referenziert. Diese kann als Attribut „frei“ gewählt werden. Die `fx:id` wird bei der Entwicklung des eigentlichen Verhaltens (Java in Eclipse) benötigt. Für unser Beispiel selektieren Sie das VBox Element und setzen im Inspector/Code-Panel die „`fx:id`“ auf „`vbox`“.

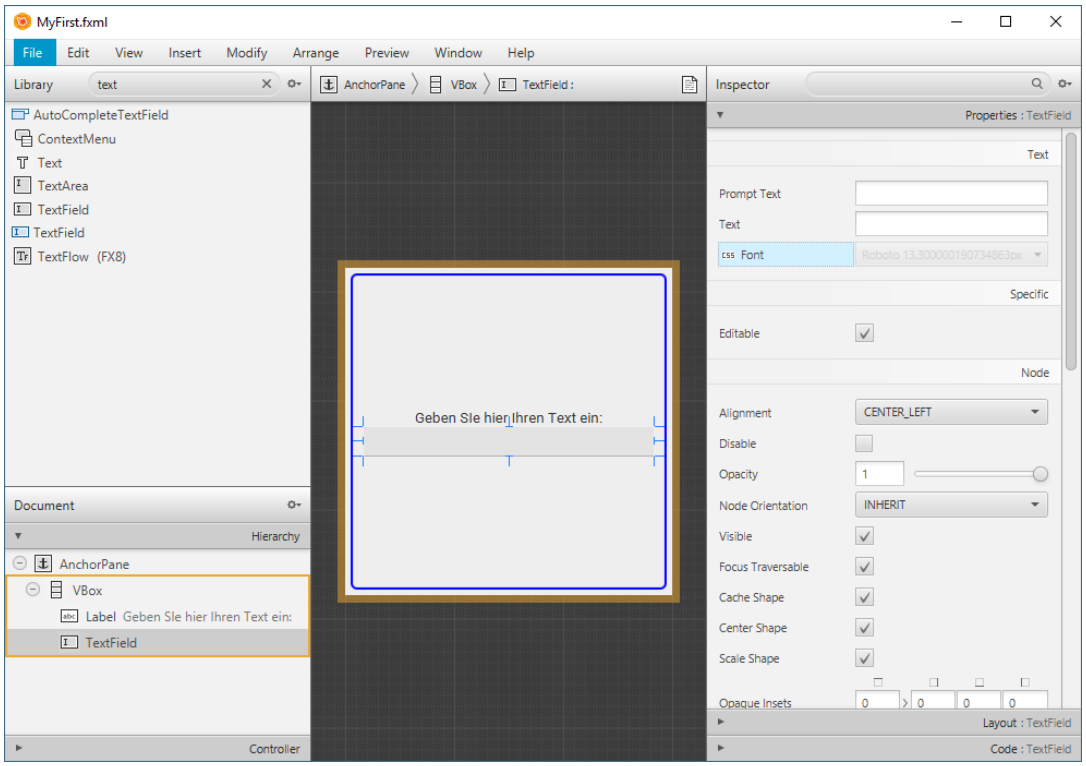


Für unser Beispielprojekt fügen wir der Oberfläche nun weitere Elemente hinzu. Ziehen Sie zunächst ein

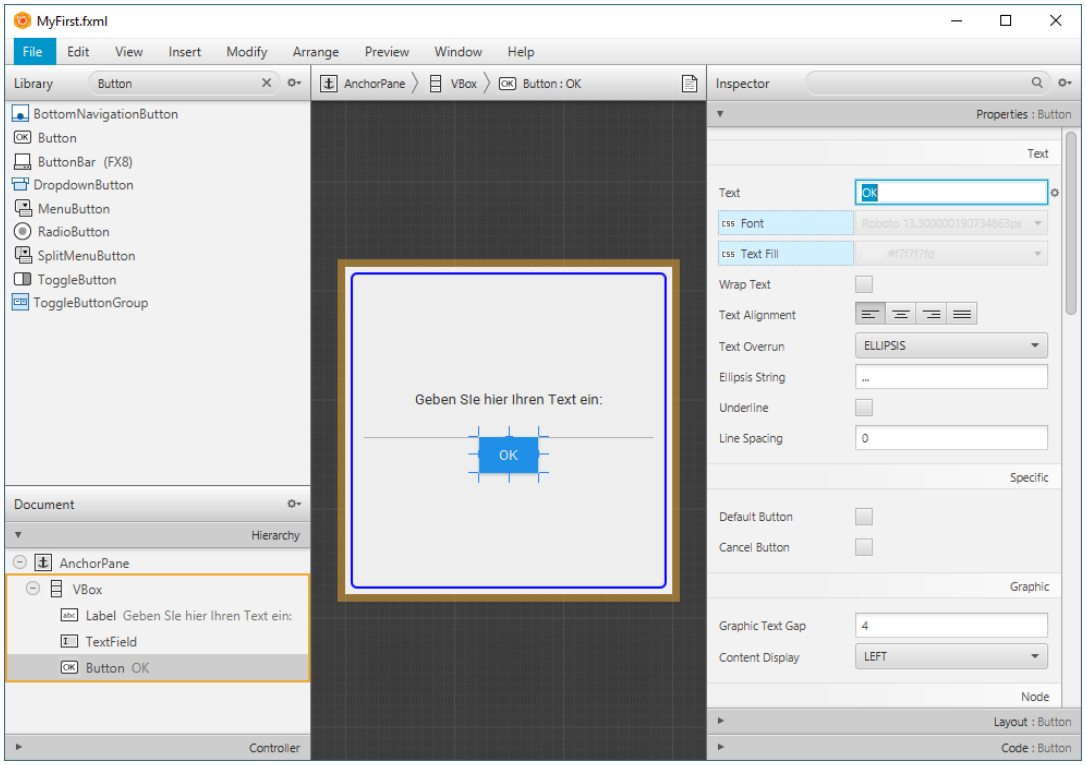
`Label` (Library/Controls Panel) auf die `VBox`. Ändern Sie den Inhalt des `Label`s.



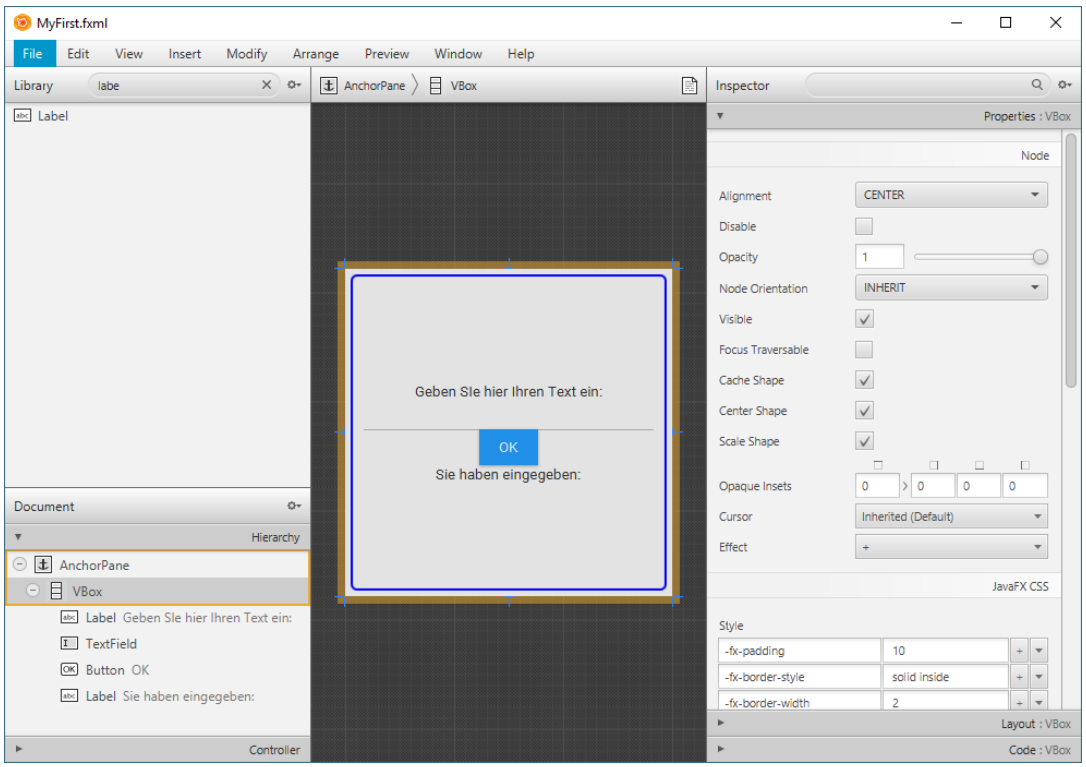
Fügen Sie nun ein `TextField` hinzu. Setzen Sie die `fx:id` dieses Elements auf `textInput`.



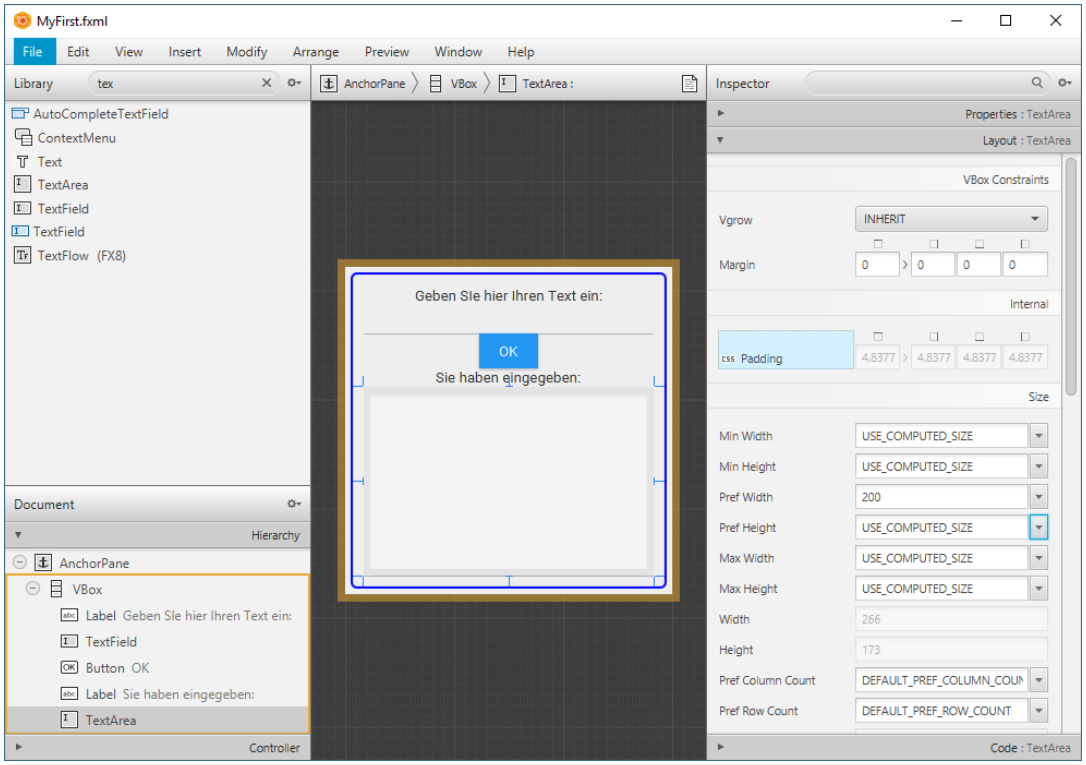
Fügen Sie nun einen Button hinzu. Mit diesem wollen wir später Action Events auslösen. Ändern Sie den Text auf dem Button in „OK“ und setzen Sie die `fx:id` dieses Buttons auf `btn`.



Fügen Sie ein weiteres Label hinzu und ändern Sie dessen Inhalt:



Zum Schluss fügen wir eine `TextArea` hinzu. Diese soll später auf Knopfdruck unsere Eingaben im obigen `TextField` darstellen/duplizieren. Setzen Sie die `fx:id` dieses Elements auf `textOutput`.



Über die MenuBar können Sie sich mit `Preview → Show Preview in Window` einen Eindruck von der erzeugten Oberfläche verschaffen.

Wenn Sie im SceneBuilder Ihre FXML-Datei gespeichert haben, zeigt Ihnen der Editor in Eclipse folgendes:

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TextArea?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.VBox?>

<AnchorPane style="-fx-background-color: #EEE;"
xmlns="http://javafx.com/javafx/8.0.141"
       xmlns:fx="http://javafx.com/fxml/1">
       <children>
              <VBox fx:id="vbox" alignment="CENTER" prefHeight="300.0"
                     prefWidth="300.0"
                     styleClass="fancyVBox"
                     stylesheets="@application.css">
                     <children>
                            <Label text="Geben SIe hier Ihren Text ein:" />
                            <TextField fx:id="textInput" />
                            <Button fx:id="btn" mnemonicParsing="false" text="OK" />
                            <Label text="Sie haben eingegeben:" />
                            <TextArea fx:id="textOutput" prefWidth="200.0" />
                     </children>
              </VBox>
       </children>
</AnchorPane>
```

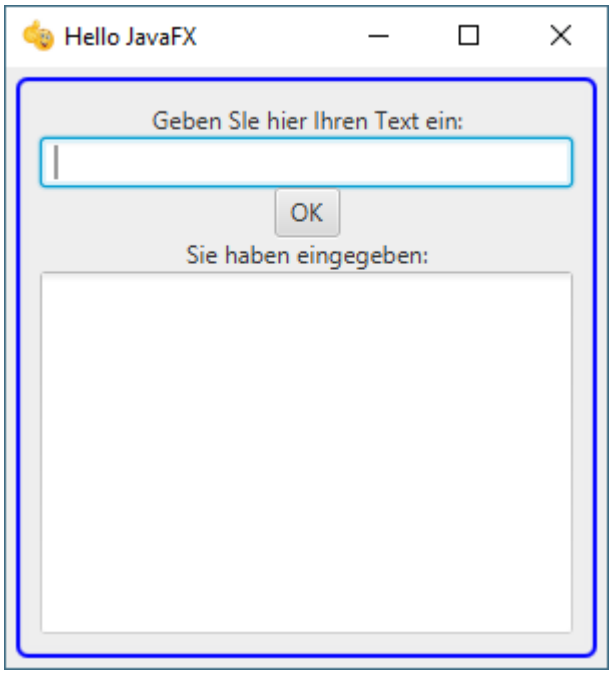
Das abgebildete Code-Beispiel nutzt die CSS-Datei wie oben beschrieben. Wenn Sie diese nicht nutzen, wird Ihr VBox-Element ein weiteres Attribut mit den Stil-Definitionen enthalten.

Schritt 6: Anwendungslogik in Java

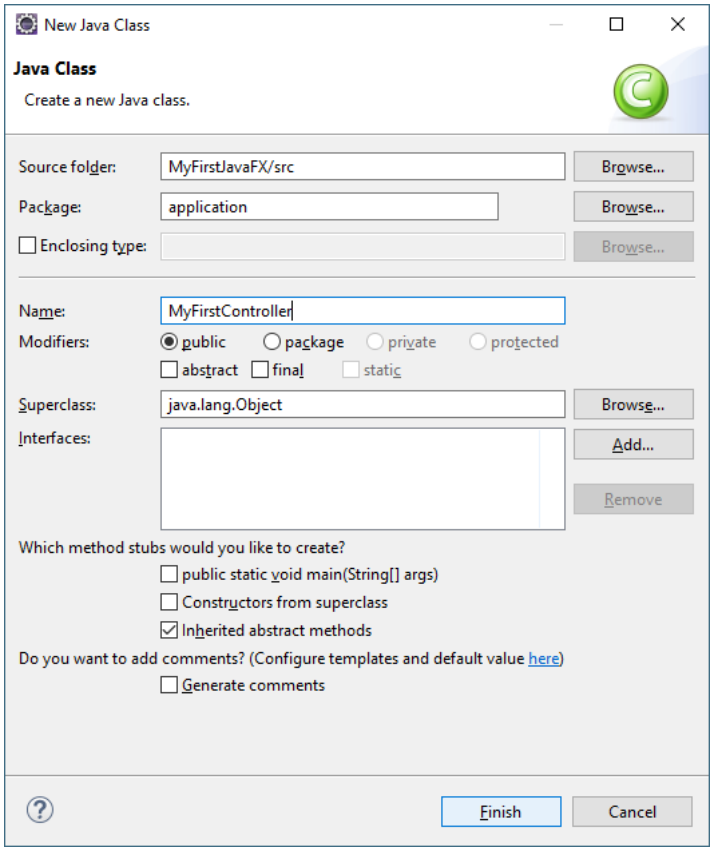
Ändern Sie Ihre Main-Klasse wie folgt, um die im SceneBuilder gestaltete GUI in Ihrem Programm zu verwenden:

```
package application;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.stage.Stage;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.image.Image;
import javafx.scene.layout.BorderPane;

public class Main extends Application {
	@Override
	public void start(Stage primaryStage) {
		try {
			// BorderPane root = new BorderPane();
			Parent root = FXMLLoader
					.load(getClass()
					.getResource("MyFirst.fxml"));

			Scene scene = new Scene(root, 300, 300);
			scene.getStylesheets()
				.add(getClass()
				.getResource("application.css")
				.toExternalForm());

			primaryStage.setTitle("Hello JavaFX");
			primaryStage.getIcons()
				.add(new Image("https://sc.mogicons.com/c/192.jpg"));

			primaryStage.setScene(scene);
			primaryStage.show();
		} catch (Exception e) {
			e.printStackTrace();
		}
	}

	public static void main(String[] args) {
		launch(args);
	}
}
```

Starten Sie Ihr Programm.



Noch ist keine Anwendungslogik implementiert. Das wollen wir nun nachholen. Dazu verbinden wir unser GUI, genauer unsere Scene, (geschrieben in FXML) mit einer Java-Klasse, welche diese Oberfläche steuert. Wir nennen die Klasse in unserem Beispiel `MyFirstController`.

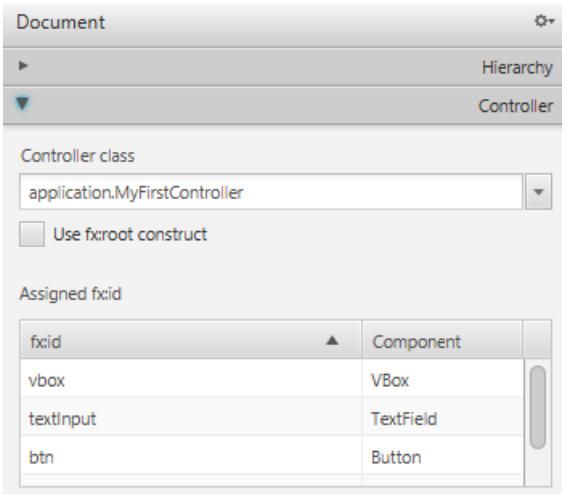


Die Klasse soll folgendes enthalten:

```
package application;

import java.net.URL;
import java.util.ResourceBundle;

import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;

public class MyFirstController
{
	@FXML
	// The reference of inputText will be injected by the FXML loader
	private TextField textInput;

	// The reference of outputText will be injected by the FXML loader
	@FXML
	private TextArea textOutput;

	// location and resources will be automatically injected by the FXML loader
	@FXML
	private URL location;

	@FXML
	private Button btn;

	@FXML
	private ResourceBundle resources;

	// Add a public no-args constructor
	public MyFirstController() {}

	@FXML
	private void initialize() {}
}
```

Diese Klasse stellt die Verbindung zwischen unserem Java Code und der FXML Datei her. Dabei wird die `fx:id` mit einem konkreten Java-Objekt verbunden. Wichtig dabei ist, dass die vergebenen `fx:ids` mit den Attributen übereinstimmen (`textInput`, `textOutput`, `btn`). Der SceneBuilder kann Ihnen den Rahmen einer solchen Controller-Klasse generieren: `MenuBar: Show → Show Sample Controller Skelleton`.

Damit unsere JavaFX-Scene unseren `MyFirstController` kennt, müssen diesen noch (im FXML) bekannt machen. Das funktioniert einfach über den SceneBuilder im Document/Controller-Pane:



Alternativ könnten wir den Controller auch direkt im fxml-Code setzen. Im Ergebnis erhält das Wurzel-Element unserer Scene nun ein weiteres Attribut „`fx:controller`“:

```
<AnchorPane
	style="-fx-background-color: #EEE;"
	xmlns="http://javafx.com/javafx/8.0.141"
	xmlns:fx=http://javafx.com/fxml/1
	fx:controller="application.MyFirstController">
```

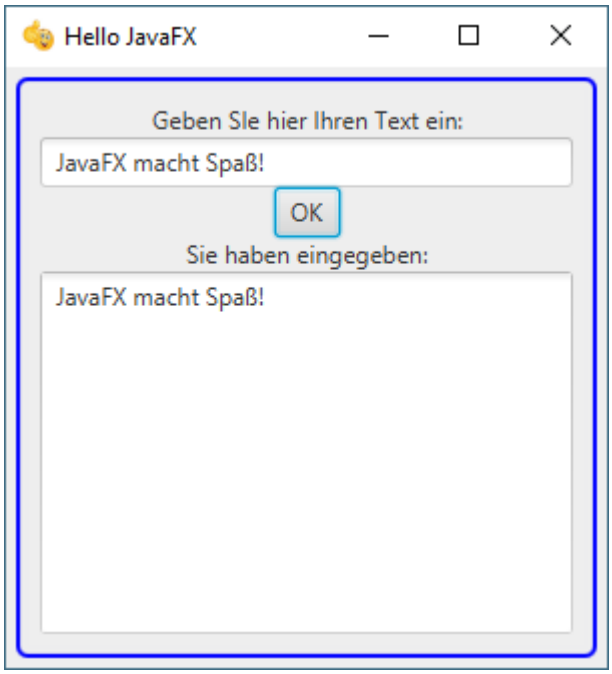
Um dem Button in Ihrer GUI eine Funktionalität zu geben, ergänzen Sie die `initialize()`-Methode des `MyFirstControllers` wie folgt:

```
@FXML
private void initialize() {
	btn.setOnAction(new EventHandler<ActionEvent>() {
		@Override
		public void handle(ActionEvent e) {
			textOutput.setText(textInput.getText());
		}
	});
}
```

Alternativ können Sie die (kürzere) Lambda-Schreibweise verwenden:

```
@FXML
private void initialize() {
  btn.setOnAction( e -> {textOutput.setText(textInput.getText());});
}
```

Nach Start Ihrer Anwendung können Sie einen Text in das obere Feld eingeben. Dieser wird nach dem Klick auf den Button in das untere Textfeld kopiert.



Über die CSS-Datei können Sie Ihre Anwendung sehr leicht „umstylen“, ohne dazu den FXML oder den Java-Code verändern zu müssen.