Object-oriented programming

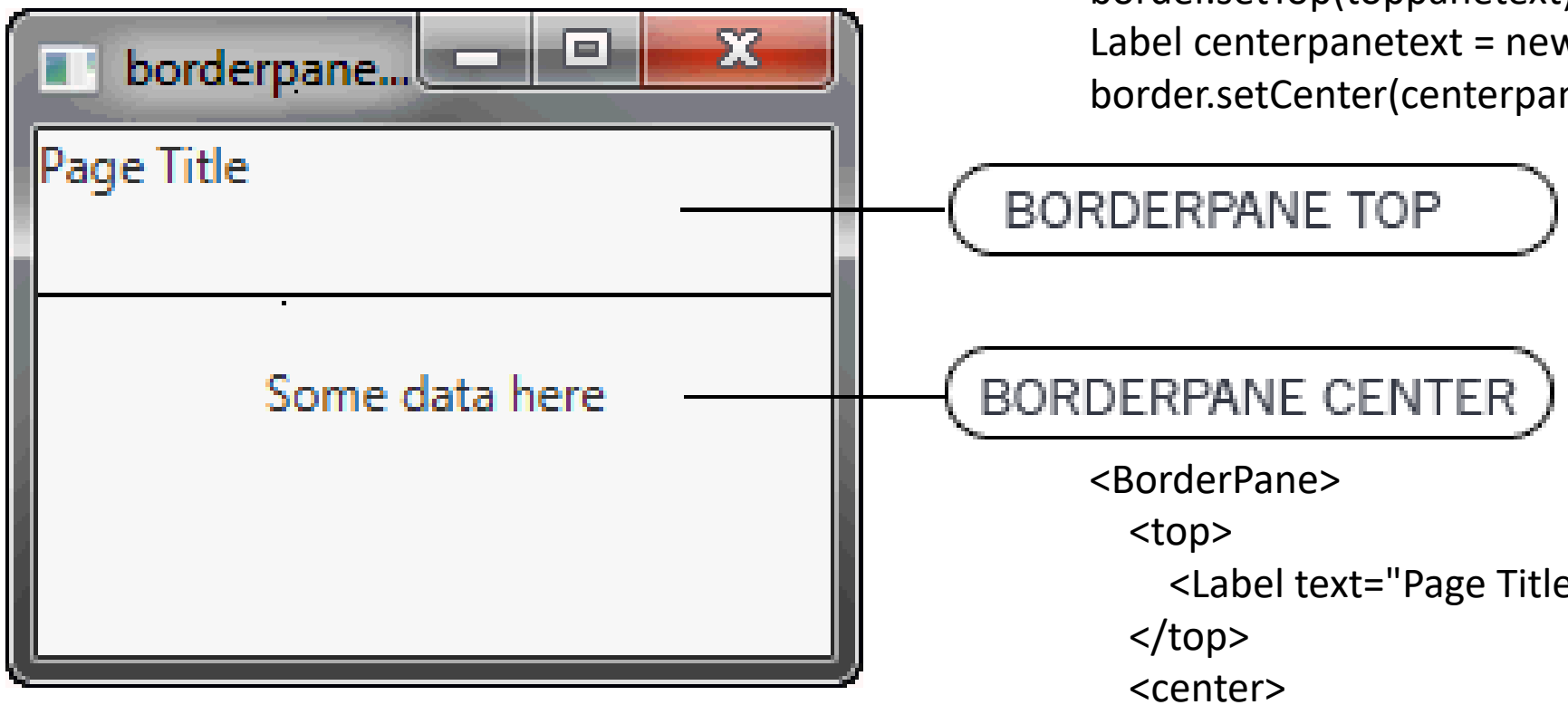
Second semester

Lecture №4

Java FX, FXML, Address Book with FXML

Introduction to FXML

```
BorderPane border = new BorderPane();
Label toppanetext = new Label("Page Title");
border.setTop(toppanetext);
Label centerpanetext = new Label ("Some data here");
border.setCenter(centerpanetext);
```



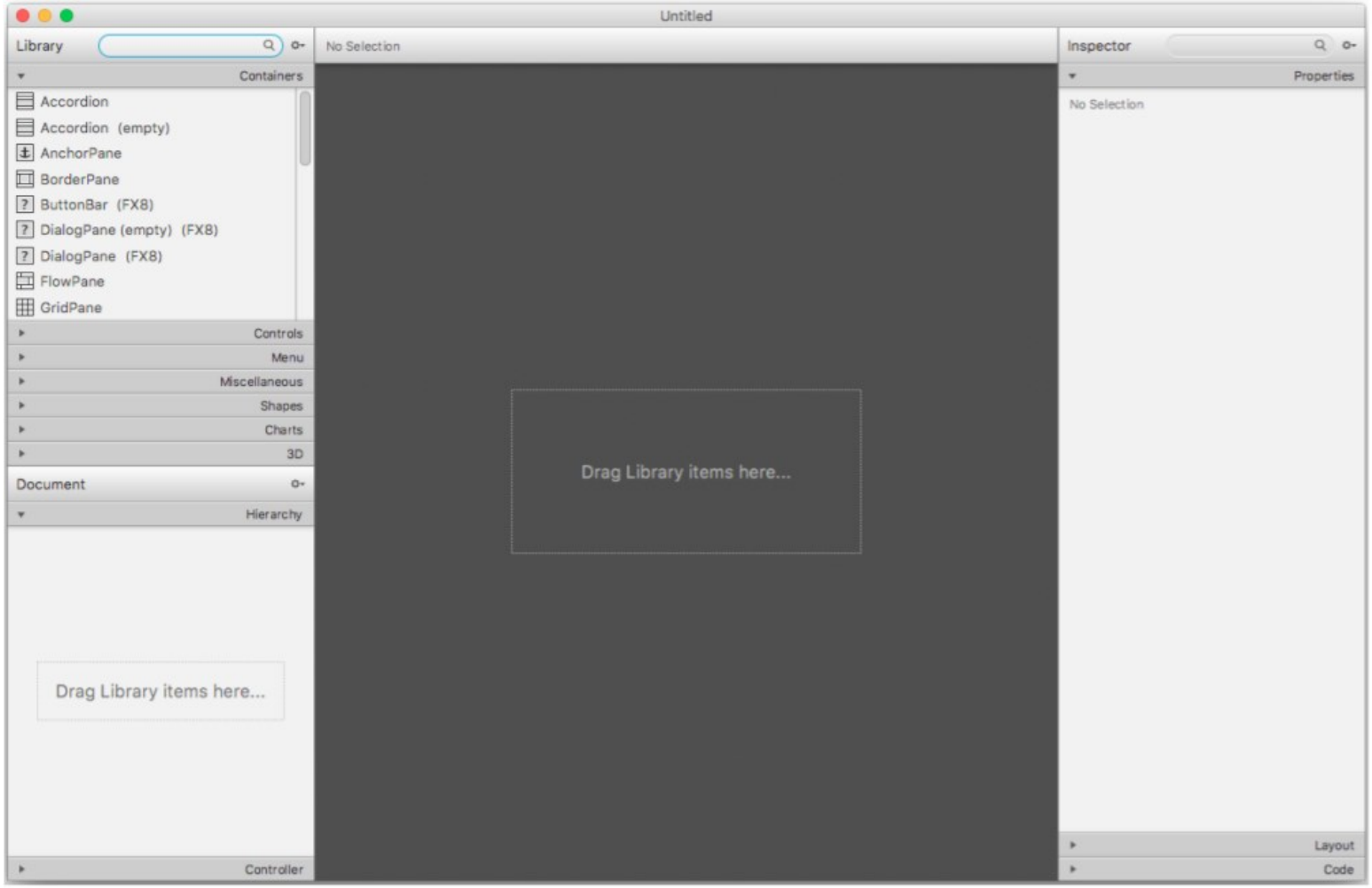

```
<BorderPane>
   <top>
      <Label text="Page Title"/>
   </top>
   <center>
      <Label text="Some data here"/>
   </center>
</BorderPane>
```

Benefits of FXML

In addition to providing web developers a familiar approach to designing user interfaces, FXML offers these benefits:

- Because the scene graph is more transparent in FXML, it is easy for a development team to create and maintain a testable user interface.
- FXML is not a compiled language; you do not need to recompile the code to see the changes.
- The content of an FXML file can be localized as the file is read. For example, if an FXML file is loaded using the en_US locale, then it produces the string "First Name" for a label based on the following resource string:
  - <Label text="%firstName"/>
  - If the locale is changed to ru_RU and the FXML file is reloaded, then the label shows «Первое Имя"
  - The same is not true for Java code, because you must manually update the content of every element of your user interface by obtaining a reference to it and calling the appropriate setter (such as setText()).
- You can use FXML with any Java Virtual Machine (JVM) language, such as Java, Scala, or Clojure.
- FXML is not limited to the view portion of the MVC interface. You can construct services or tasks or domain objects, and you can use JavaScript or other scripting languages in FXML.

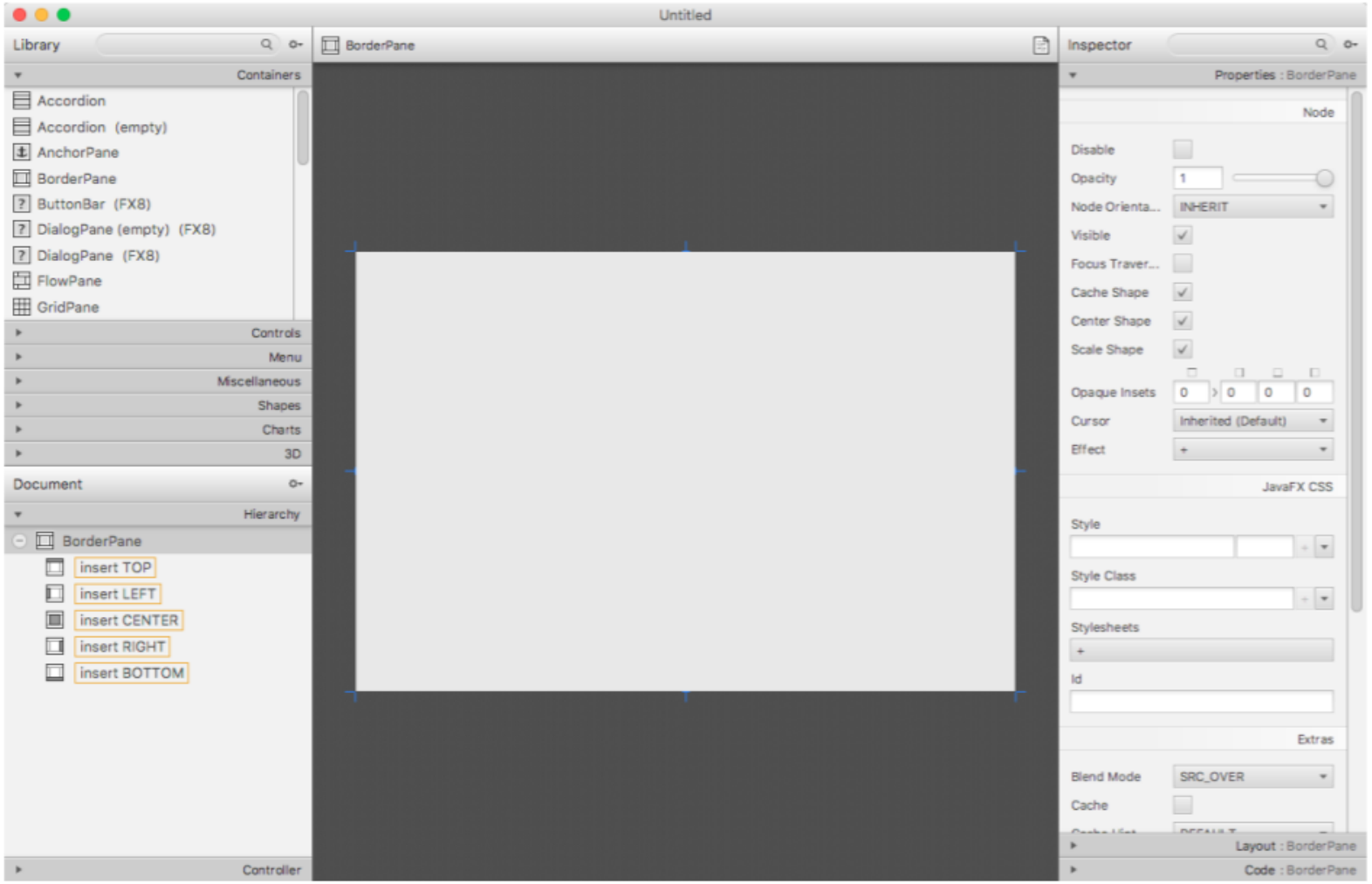
Scene Builder



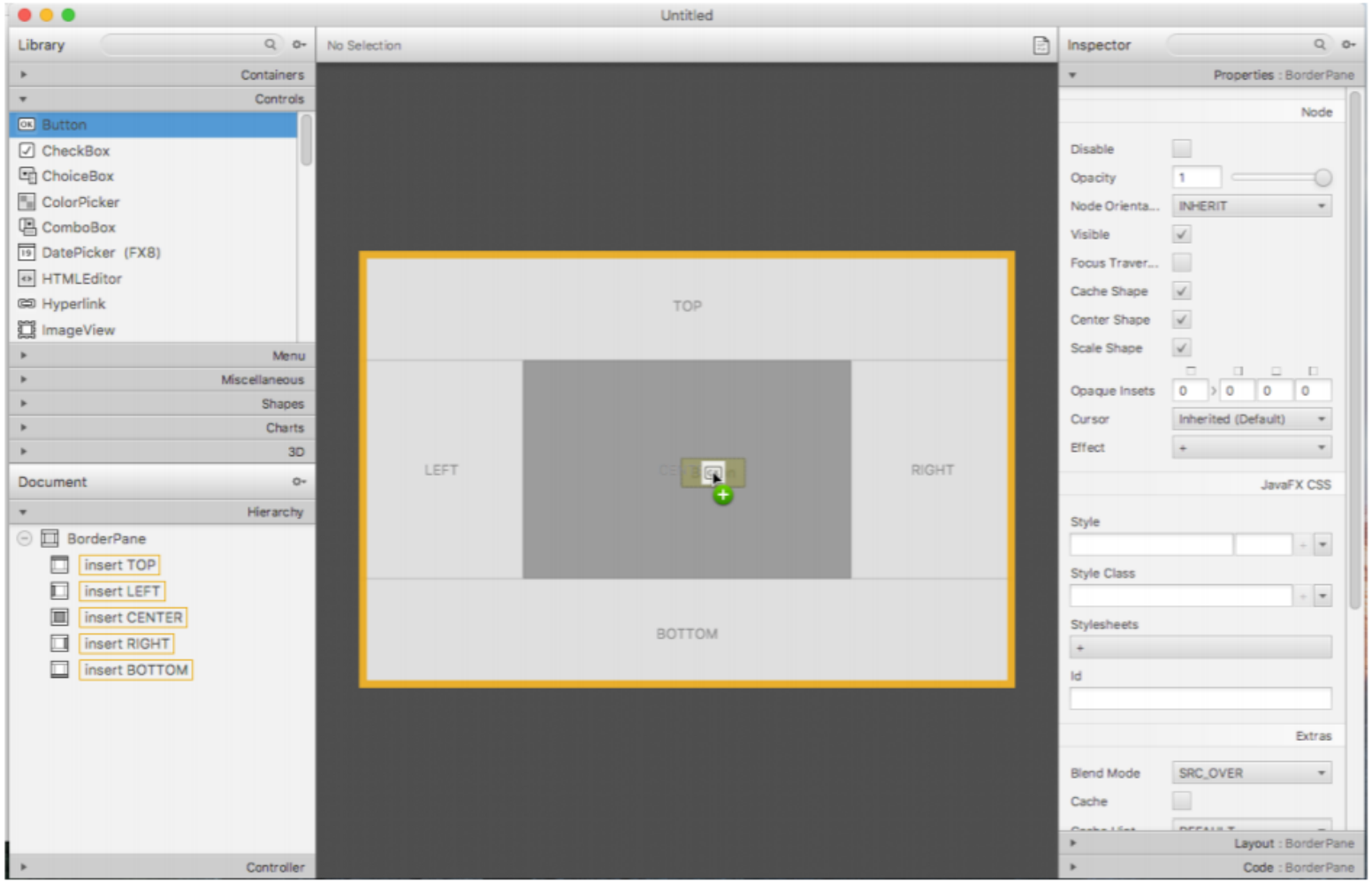
Layouts

- BorderPane - A basic layout that gives you five areas: top, left, right, bottom, and center
- HBox - A basic layout that orders the GUI controls in a horizontal (thus the “H”) line
- VBox - A basic layout that orders the GUI controls in a vertical (thus the “H”) line
- GridPane - A basic layout that orders the GUI controls in a grid. For example, a grid might be 2 row by 2 columns
- StackPane - A basic layout that put all the GUI controls in a stack, in other words, right on top of each other

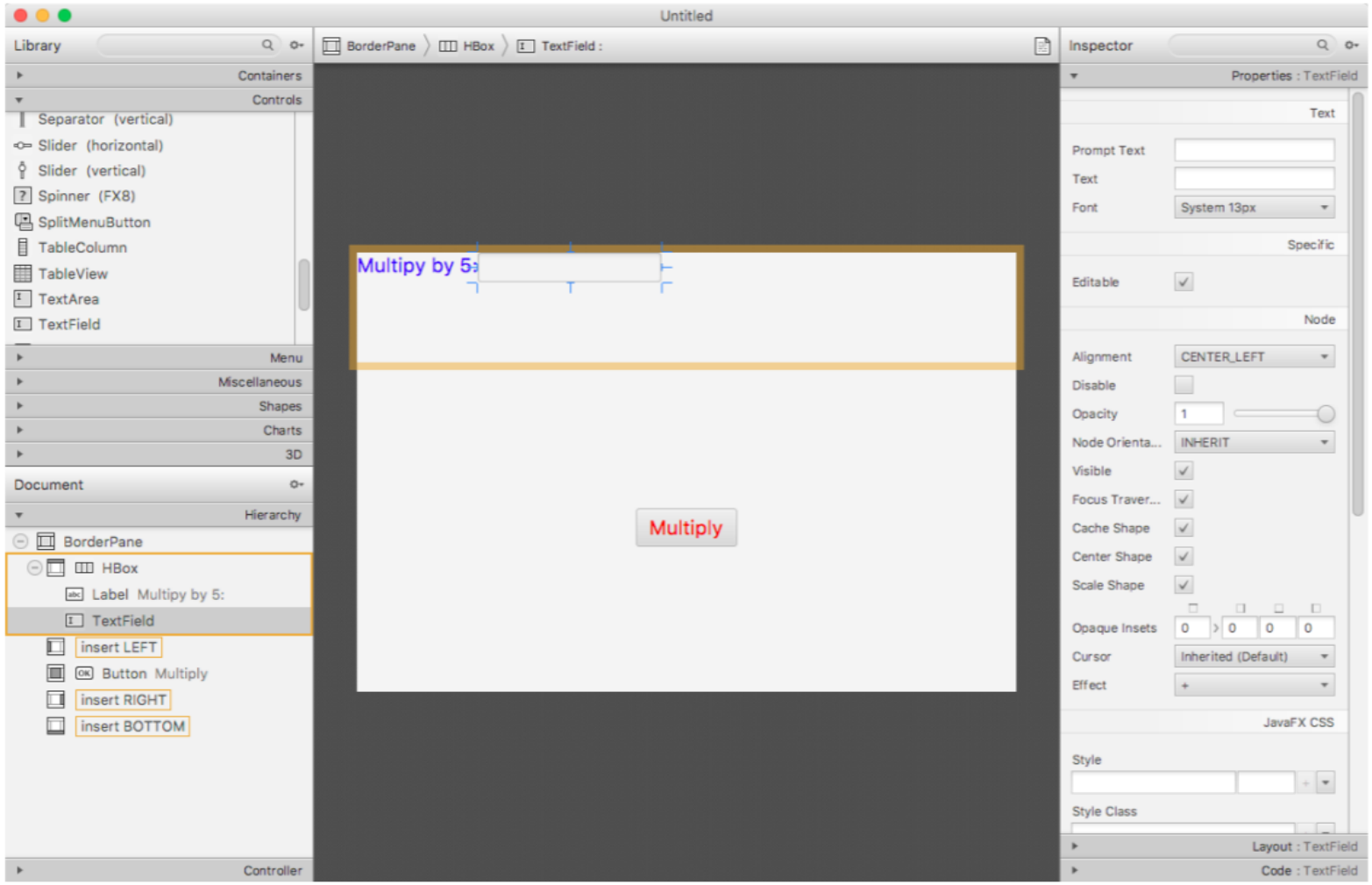
Scene Builder with the Border Layout



Placing a Button in the center of a BorderPane layout in Scene Builder



Placed a TextField to the right of the Label in the HBox layout



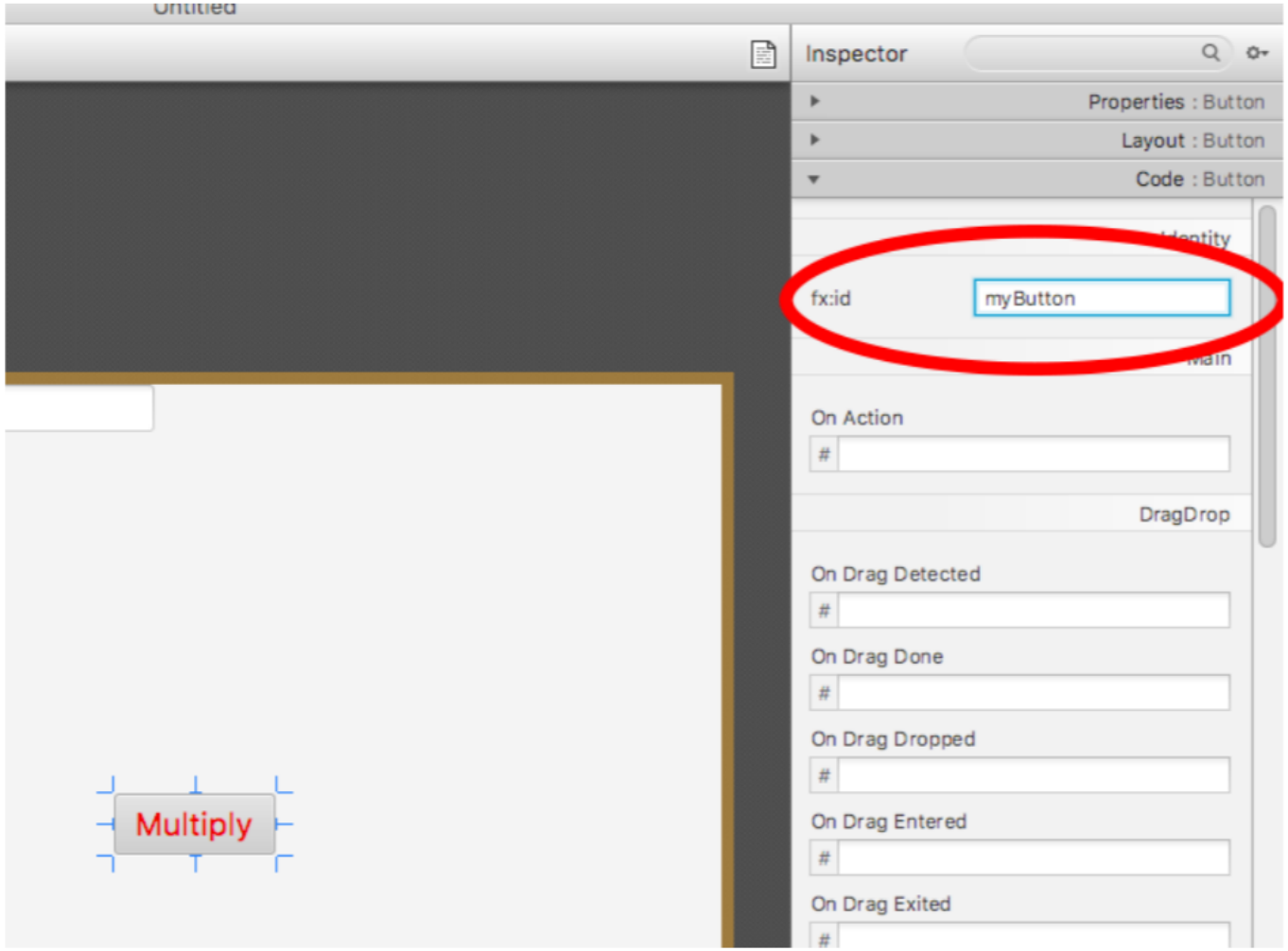
Controller

Before we finish there are two things we need to do:

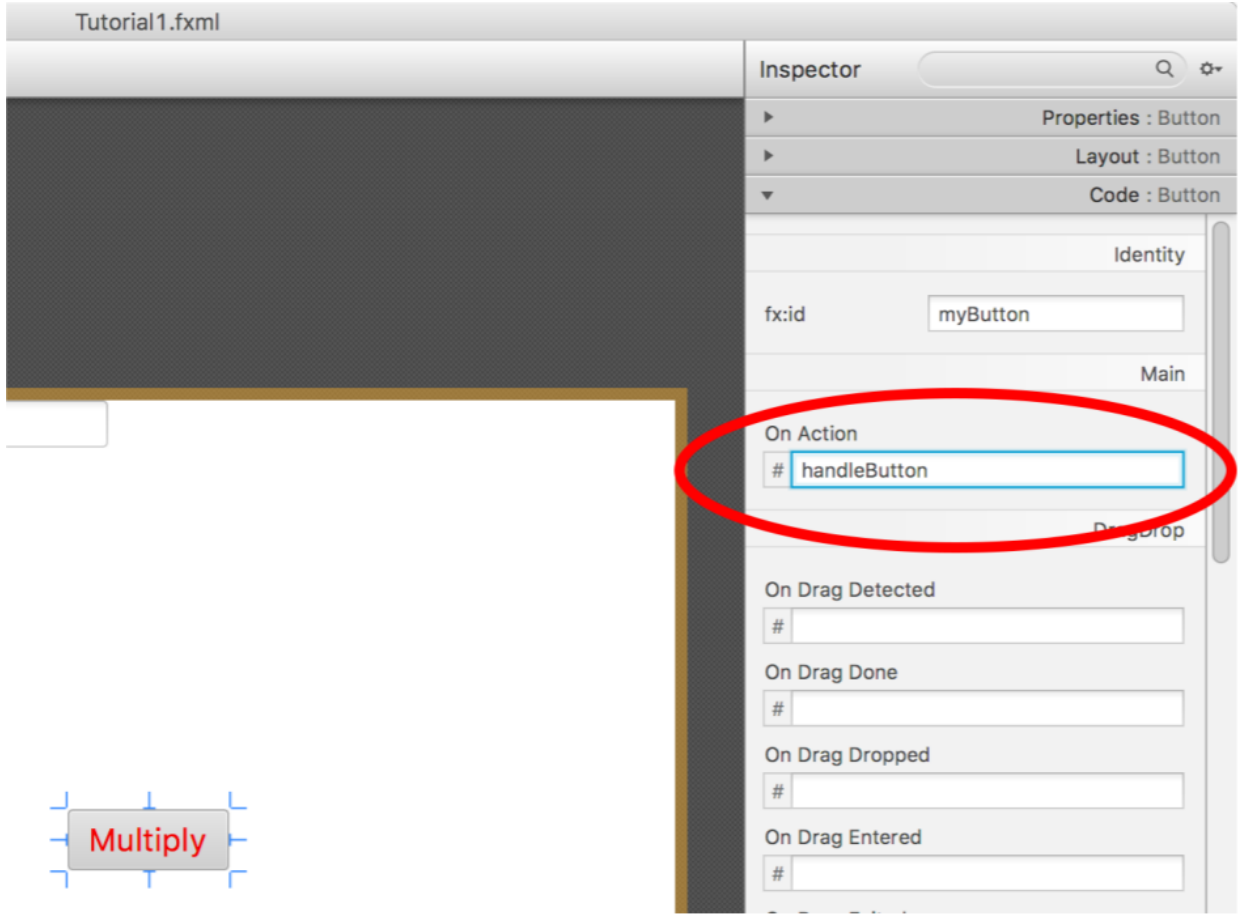
- Give the Button an id so that we can process the button click.
- Give the TextField an id so that we can get input from it.

In order to give the button an id, we need to click on it. The properties will now show the properties of the button. On the bottom of the properties list this is a section called “Code.” Specifically, it should look like “Code : Button.” Click on “Code” and you will be provided with the opportunity to give it an “fx:id.” In the “fx:id” TextField, enter the text, “myButton.”

Showing the fx:id of myButton for the highlighted button



Showing the addition of “handleButton” to handle the button click on the highlighted button.



Scene Builder fxml

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.HBox?>
<?import javafx.scene.text.Font?>
<BorderPane maxHeight="-Infinity" maxWidth="-Infinity“ minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0“
prefWidth="600.0" xmlns=http://javafx.com/javafx/8.0.91 xmlns:fx="http://javafx.com/fxml/1">
  <center>
    <Button fx:id="myButton" mnemonicParsing="false“ onAction="#handleButton" text="Multiply" textFill="RED
BorderPane.alignment="CENTER">
      <font>
        <Font size="18.0" />
      </font>
      </Button>
  </center>
  <top>
    <HBox prefHeight="100.0" prefWidth="200.0“ BorderPane.alignment="CENTER">
      <children>
        <Label text="Multipy by 5:" textFill="#3c00ff">
          <font>
            <Font size="18.0" />
          </font>
        </Label>
        <TextField fx:id="myTextField" />
      </children>
    </HBox>
  </top>
</BorderPane>
```

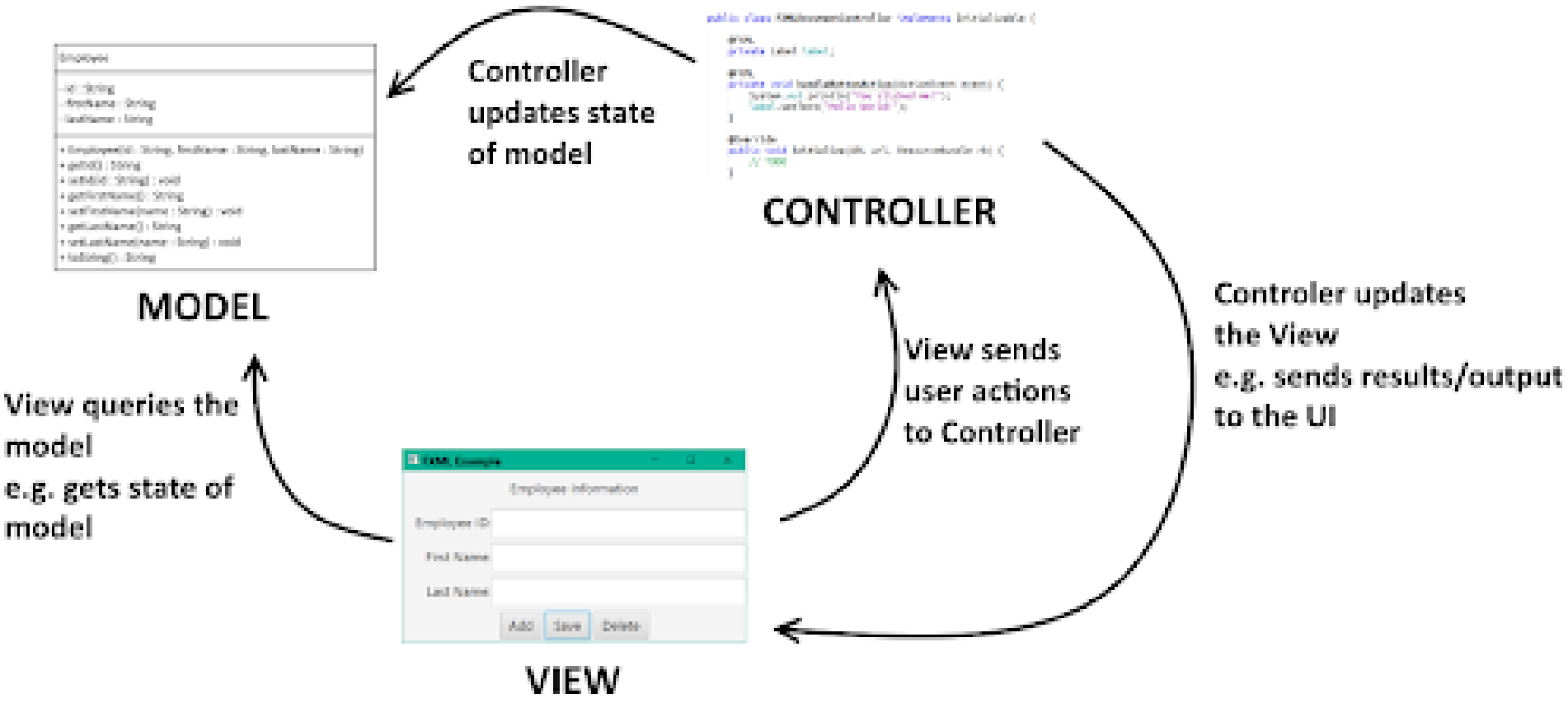
Hand-written code


```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.*;
import javafx.scene.layout.*;
import javafx.scene.control.*;
public class HandWrittenCode extends Application {
  public void start(Stage primaryStage) {
    HBox hbox1 = new HBox();
    Label label1 = new Label("Multipy by 5:"); //show text
    TextField myTextField = new TextField();
    hbox1.getChildren().addAll(label1,myTextField);
    BorderPane borderPane1 = new BorderPane();
    Button myButton = new Button("Add");
    borderPane1.setTop(hbox1);
    borderPane1.setCenter(myButton);
    int width = 300;
    int height = 300;
    Scene scene = new Scene(borderPane1,width,height);
    primaryStage.setScene(scene);
    primaryStage.show(); //show the Stage
  }
  public static void main(String[] args){
    launch(args);
  }
}
```

Controller

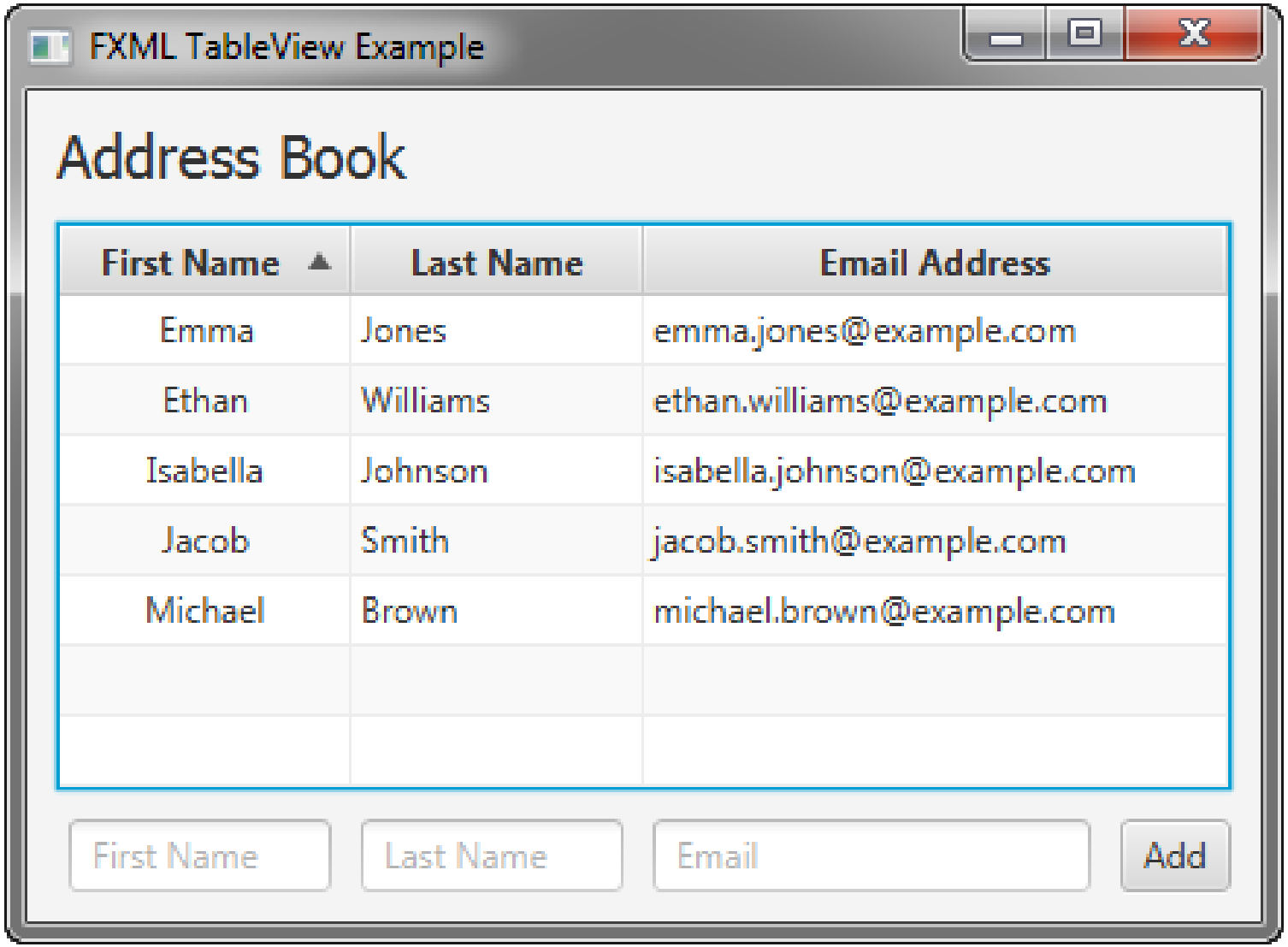
```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.fxml.*;
import javafx.scene.*;
import javafx.stage.Stage;
import javafx.event.*;
import javafx.scene.control.*;

public class Tutorial1 extends Application {
        @FXML TextField myTextField;
        @FXML Button myButton;
        @FXML protected void handleButton(ActionEvent event) {
                String input = myTextField.getText();
                int intInput = Integer.parseInt(input);
                System.out.println(intInput * 5);
        }
        public void start(Stage primaryStage) throws Exception {
                FXMLLoader loader = new          FXMLLoader(getClass().getResource("Tutorial1.fxml"));
                loader.setController(this);
                Parent root = loader.load();
                Scene myScene = new Scene(root,400,400);
                primaryStage.setScene(myScene);
                primaryStage.show();
        }
        public static void main(String[] args) {
                launch(args);
        }
}
```

Model-View-Controller



Model-View-Controller Design Pattern

Creating an Address Book with FXML



Step 1 – Application

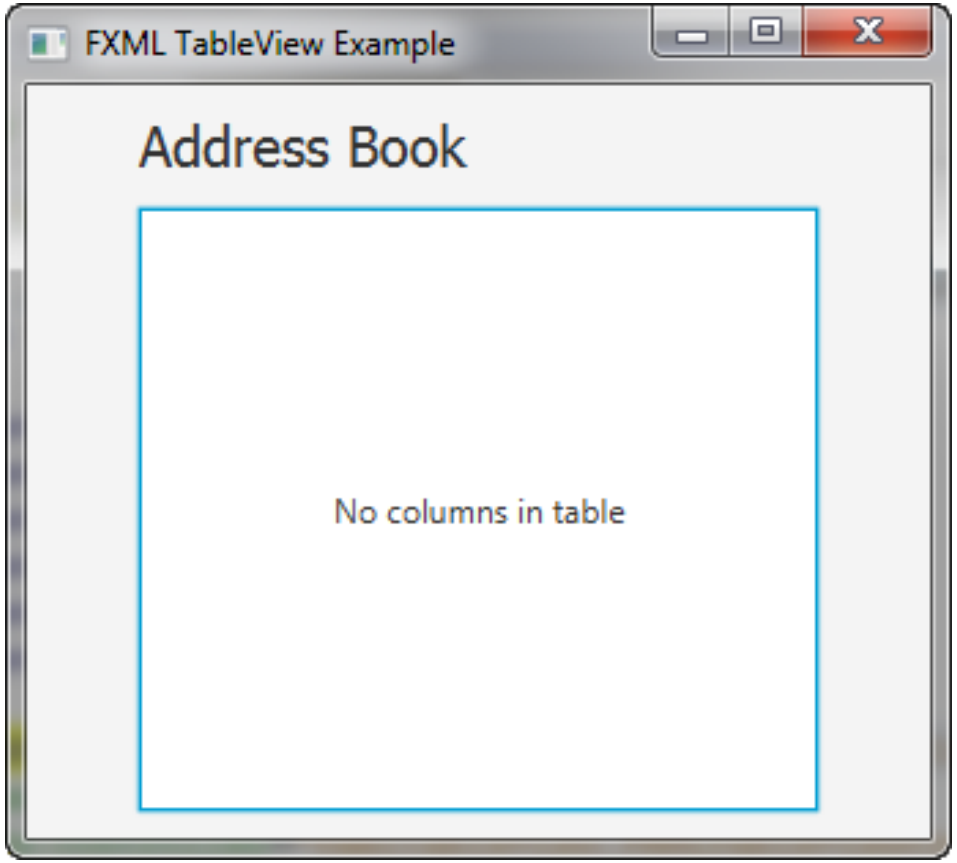
```
public class FXMLTableView extends Application {
   @Override
   public void start(Stage primaryStage) throws Exception {
      primaryStage.setTitle("FXML TableView Example");
      Pane myPane = (Pane)FXMLLoader.load(getClass().getResource("fxml_tableview.fxml"));
      Scene myScene = new Scene(myPane);
      primaryStage.setScene(myScene);
      primaryStage.show();
   }

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

Step 2 - Person

```
package fxmltableview;

import javafx.beans.property.SimpleStringProperty;

public class Person {
  private final SimpleStringProperty firstName = new SimpleStringProperty("");
  private final SimpleStringProperty lastName = new SimpleStringProperty("");
  private final SimpleStringProperty email = new SimpleStringProperty("");
   public Person() {
      this("", "", "");
   }
   public Person(String firstName, String lastName, String email) {
      setFirstName(firstName);
      setLastName(lastName);
      setEmail(email);
   }
   public String getFirstName() {
      return firstName.get();
   }
   public void setFirstName(String fName) {
      firstName.set(fName);
   }
   public String getLastName() {
      return lastName.get();
   }
   public void setLastName(String fName) {
      lastName.set(fName);
   }
   public String getEmail() {
      return email.get();
   }
   public void setEmail(String fName) {
      email.set(fName);
   }
}
```

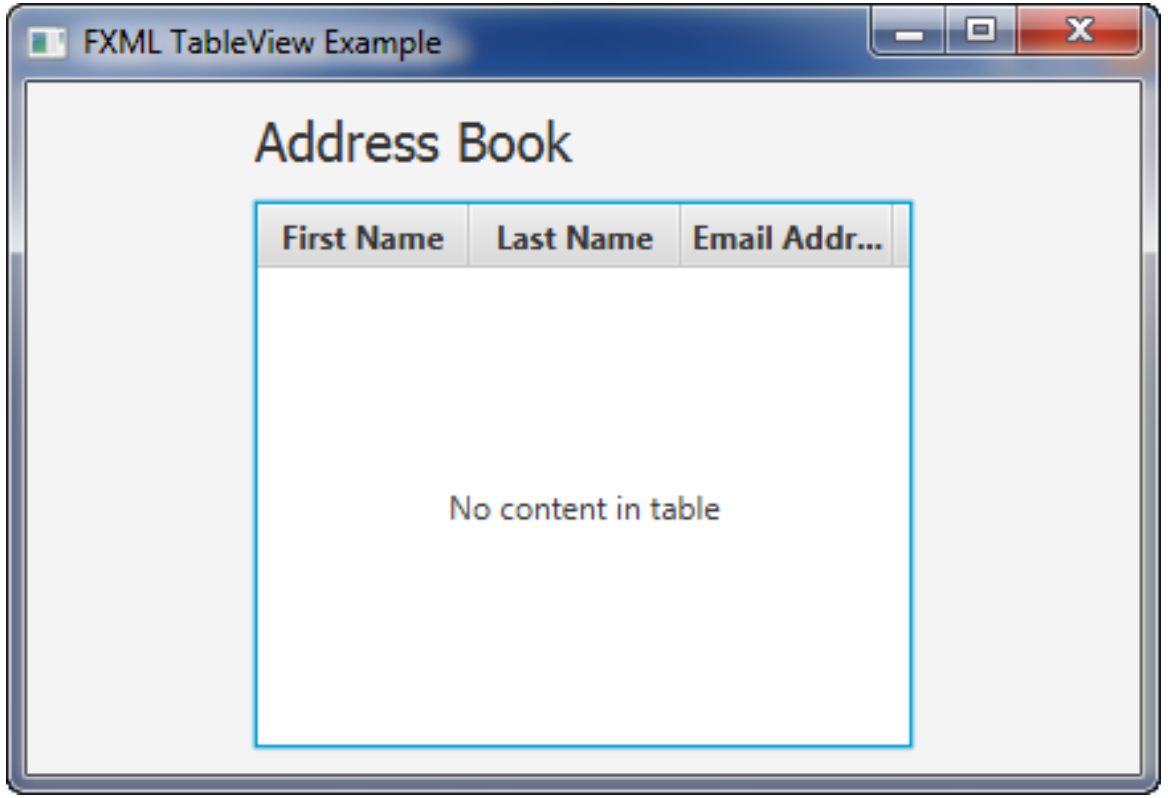
Step 3 – Main window

```
<?import javafx.geometry.Insets?>
<GridPane alignment="CENTER" hgap="10.0" vgap="10.0"
    xmlns:fx="http://javafx.com/fxml"
    fx:controller="fxmltableview.FXMLTableViewController">
    <padding>
        <Insets bottom="10.0" left="10.0" right="10.0" top="10.0"/>
    </padding>
    <Label style="-fx-font: NORMAL 20 Tahoma;" text="Address Book"
        GridPane.columnIndex="0" GridPane.rowIndex="0">
    </Label>
    <TableView fx:id="tableView" GridPane.columnIndex="0"
        GridPane.rowIndex="1">
    </TableView>
</GridPane>
```



Step4 - Add Columns to the Table

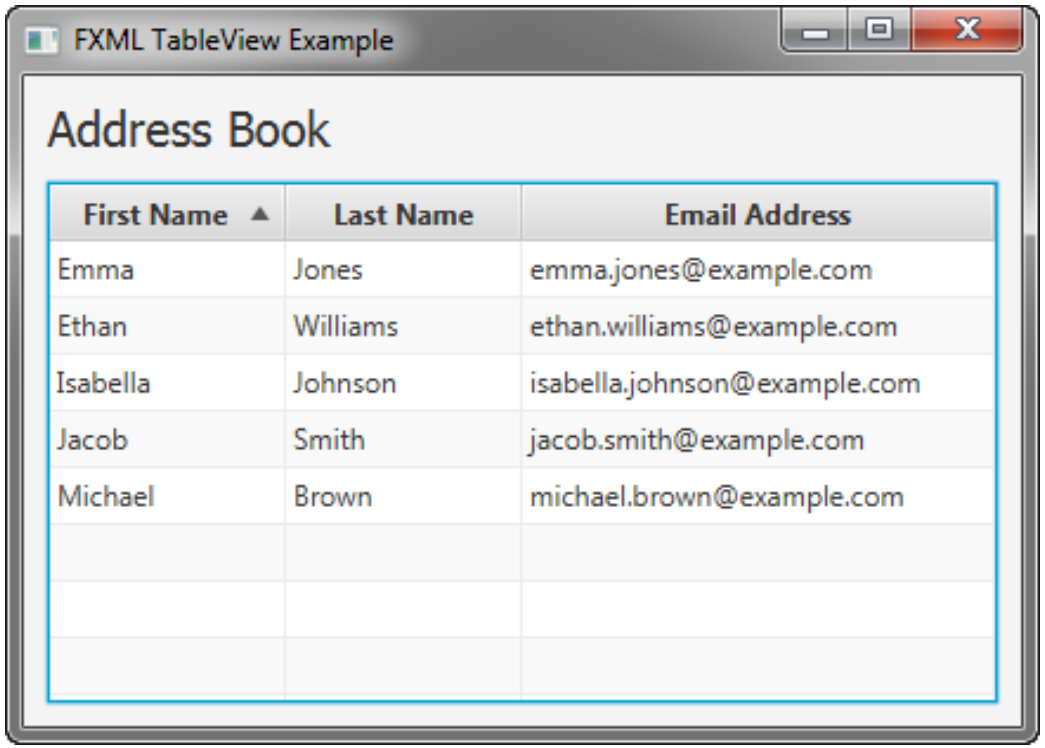
```
<TableView fx:id="tableView" GridPane.columnIndex="0" GridPane.rowIndex="1">
    <columns>
        <TableColumn text="First Name">
        </TableColumn>
        <TableColumn text="Last Name">
        </TableColumn>
        <TableColumn text="Email Address">
        </TableColumn>
    </columns>
</TableView>
```



Step5 – Full FXML

```
<?import fxmltableview.*?>
<?import java.lang.*?>
<?import javafx.collections.*?>
<?import javafx.geometry.*?>
<?import javafx.geometry.Insets?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.control.cell.*?>
<?import javafx.scene.layout.*?>


<GridPane alignment="CENTER" hgap="10.0" vgap="10.0" fx:controller="fxmltableview.FXMLTableViewController" xmlns:fx="http://javafx.c
  <padding>
    <Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />
  </padding>
  <Label style="-fx-font: NORMAL 20 Tahoma;" text="Address Book" GridPane.columnIndex="0" GridPane.rowIndex="0" />
  <TableView fx:id="tableView" GridPane.columnIndex="0" GridPane.rowIndex="1">
  ……..
  </TableView>
  ……
</GridPane>
```

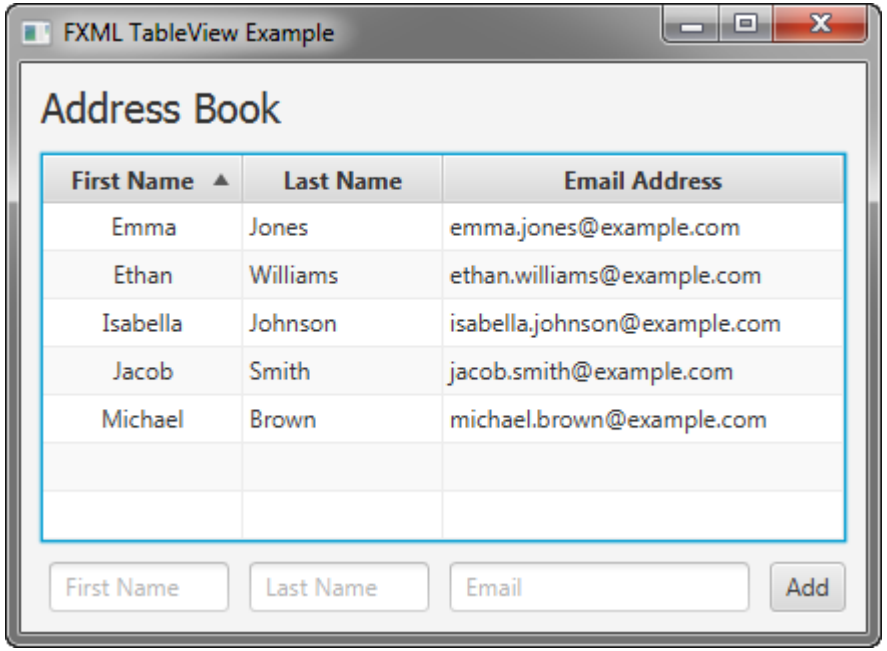
TableView




```
<TableView fx:id="tableView" GridPane.columnIndex="0" GridPane.rowIndex="1">
  <columns>
    <TableColumn prefWidth="100.0" text="First Name" fx:id="firstNameColumn">
      <cellFactory>
        <FormattedTableCellFactory alignment="CENTER" />
      </cellFactory>
      <cellValueFactory>
        <PropertyValueFactory property="firstName" />
      </cellValueFactory>
    </TableColumn>
    <TableColumn prefWidth="100.0" text="Last Name">
      <cellValueFactory>
        <PropertyValueFactory property="lastName" />
      </cellValueFactory>
    </TableColumn>
    <TableColumn prefWidth="200.0" text="Email Address">
      <cellValueFactory>
        <PropertyValueFactory property="email" />
      </cellValueFactory>
    </TableColumn>
  </columns>
  <items>
    <FXCollections fx:factory="observableArrayList">
      <Person email="jacob.smith@example.com" firstName="Jacob" lastName="Smith" />
      <Person email="isabella.johnson@example.com" firstName="Isabella" lastName="Johnson" />
      <Person email="ethan.williams@example.com" firstName="Ethan" lastName="Williams" />
      <Person email="emma.jones@example.com" firstName="Emma" lastName="Jones" />
      <Person email="michael.brown@example.com" firstName="Michael" lastName="Brown" />
    </FXCollections>
  </items>
  <sortOrder>
    <fx:reference source="firstNameColumn" />
  </sortOrder>
</TableView>
```

Add record

```
<HBox alignment="BOTTOM_RIGHT" spacing="10.0" GridPane.columnIndex="0" GridPane.rowIndex="2">
  <TextField fx:id="firstNameField" prefWidth="90.0" promptText="First Name" />
  <TextField fx:id="lastNameField" prefWidth="90.0" promptText="Last Name" />
  <TextField fx:id="emailField" prefWidth="150.0" promptText="Email" />
  <Button onAction="#addPerson" text="Add" />
</HBox>
```

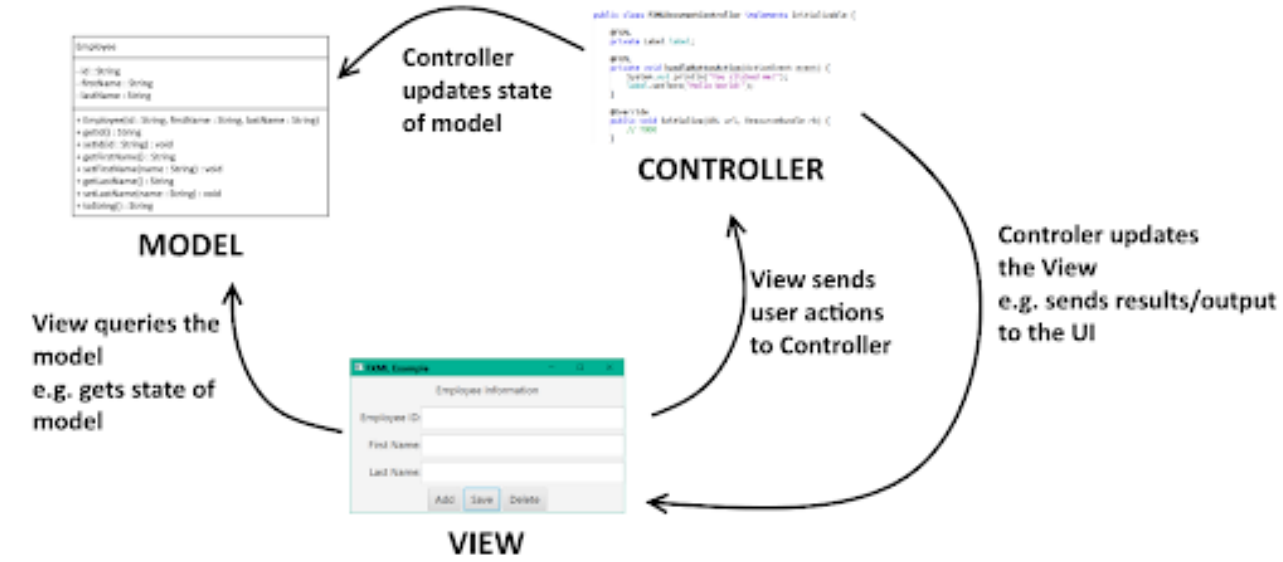


Step 6 - Cell

```
package fxmltableview;

import java.text.Format;
import javafx.geometry.Pos;
import javafx.scene.Node;
import javafx.scene.control.TableCell;
import javafx.scene.control.TableColumn;
import javafx.scene.text.TextAlignment;
import javafx.util.Callback;

public class FormattedTableCellFactory<S, T> implements
Callback<TableColumn<S, T>, TableCell<S, T>> {
   private TextAlignment alignment;
   private Format format;
   public TextAlignment getAlignment() {
      return alignment;
   }
   public void setAlignment(TextAlignment alignment) {
      this.alignment = alignment;
   }
   public Format getFormat() {
      return format;
   }
   public void setFormat(Format format) {
      this.format = format;
   }
```

```
@Override
   public TableCell<S, T> call(TableColumn<S, T> p) {
      TableCell<S, T> cell = new TableCell<S, T>() {
         @Override
         public void updateItem(Object item, boolean empty) {
            if (item == getItem()) {
               return;
            }
            super.updateItem((T) item, empty);
            if (item == null) {
               super.setText(null);
               super.setGraphic(null);
            } else if (format != null) {
               super.setText(format.format(item));
            } else if (item instanceof Node) {
               super.setText(null);
               super.setGraphic((Node) item);
            } else {
               super.setText(item.toString());
               super.setGraphic(null);
            }
         }
      };
      cell.setTextAlignment(alignment);
      switch (alignment) {
         case CENTER:
            cell.setAlignment(Pos.CENTER);
            break;
         case RIGHT:
            cell.setAlignment(Pos.CENTER_RIGHT);
            break;
         default:
            cell.setAlignment(Pos.CENTER_LEFT);
            break;
      }
      return cell;
   }
```

Step 7 - Controller



Model-View-Controller Design Pattern

```
package fxmltableview;

import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;

public class FXMLTableViewController {
   @FXML private TableView<Person> tableView;
   @FXML private TextField firstNameField;
   @FXML private TextField lastNameField;
   @FXML private TextField emailField;

   @FXML
   protected void addPerson(ActionEvent event) {
      ObservableList<Person> data = tableView.getItems();
      data.add(new Person(firstNameField.getText(),
         lastNameField.getText(),
         emailField.getText()
      ));

      firstNameField.setText("");
      lastNameField.setText("");
      emailField.setText("");
   }
}
```

Databinding