

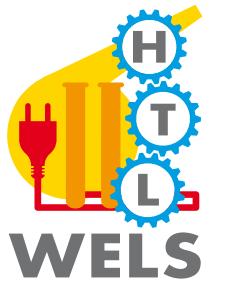
JAVAFX SCENE BUILDER

SEW

DI Thomas Helml



INHALT



- JavaFX Wiederholung
- SceneBuilder
- MVC - Model View Controller
- SceneBuilder How-to
- Ereignisbehandlung
- Projekt FX Chat

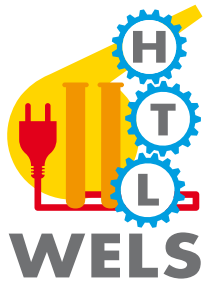


WIEDERHOLUNG





GRUNDGERÜST



```
// HelloFXApp.java

import javafx.application.Application;
import javafx.stage.Stage;

public class HelloFXApp extends Application {

    public static void main(String[] args) {

        // Launch the JavaFX application

        Application.launch(args);

    }

    @Override public void start(Stage stage) {

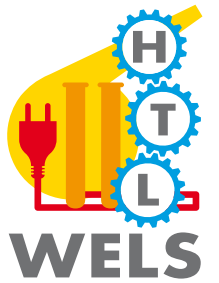
        // Program goes here...

    }

}
```



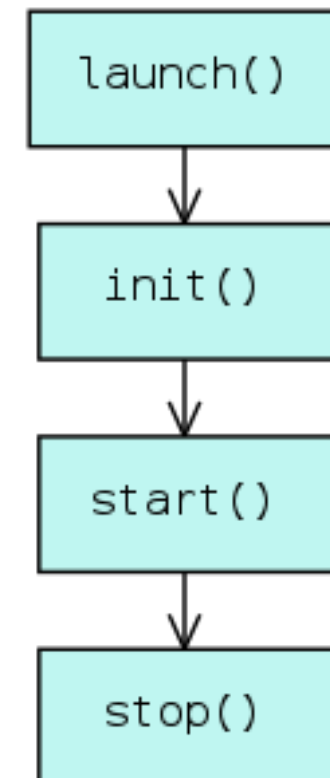
LIFECYCLE



- Jede JavaFX Anwendung ist von `javafx.application.Application` abgeleitet
- JavaFX Runtime erzeugt mehrere Threads, u.a.
 - JavaFX Launcher
 - JavaFX Application Thread
- Methode `launch()` erzeugt diese beiden Threads

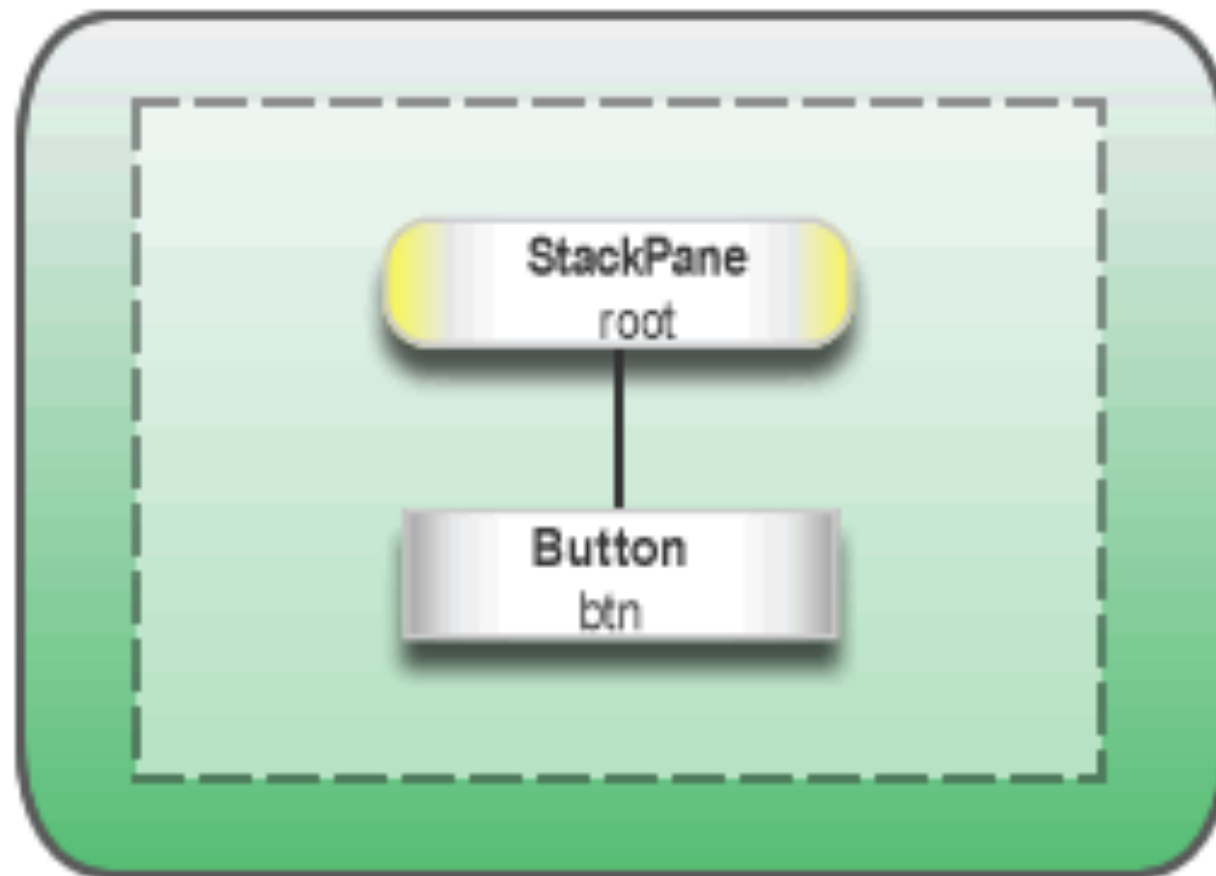
➤ Ablauf

1. **Application.launch()**
2. no-args Konstruktor
3. `init()`
4. **start()**
5. `stop()`

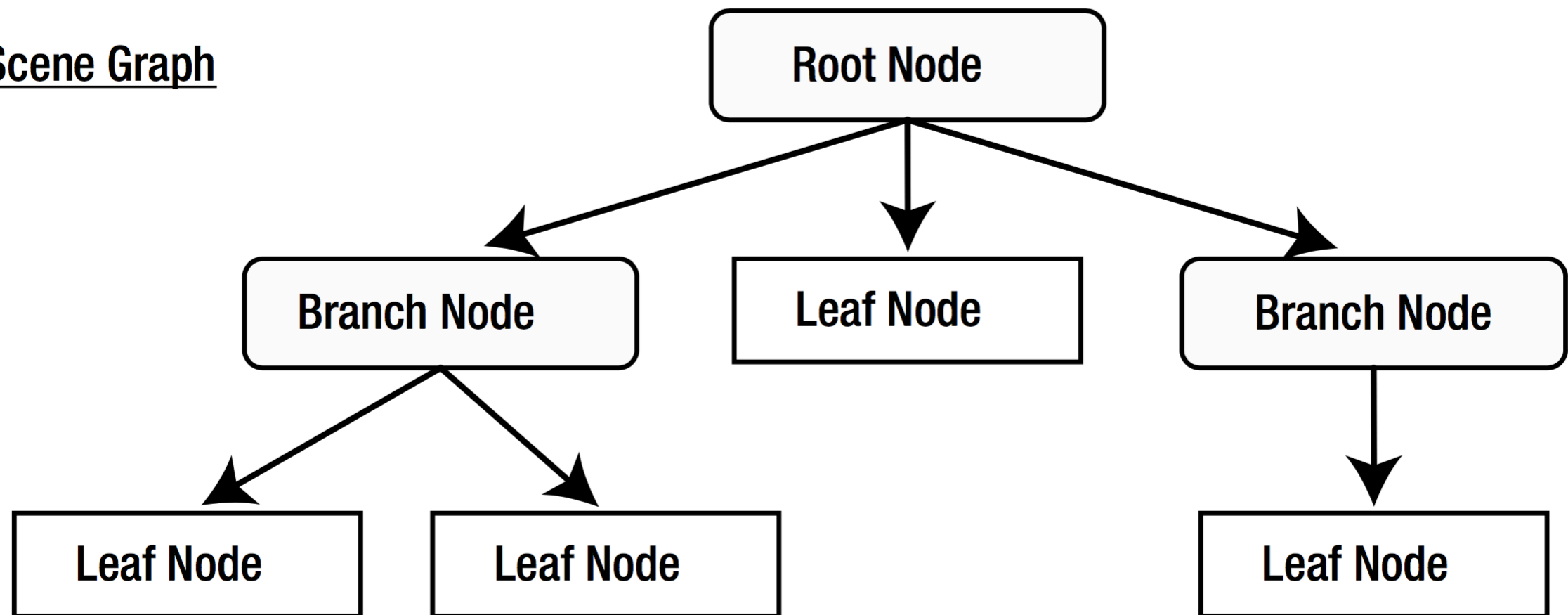


Stage javafx.stage (window)

Scene javafx.scene



A Scene Graph





SCENEBuilder

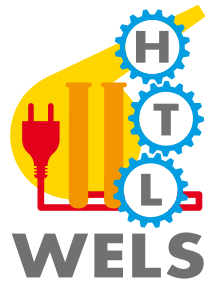




- Was ist SceneBuilder?
 - Grafisches Tool zum Erstellen von GUIs (für JavaFX)
=> GUI Builder
 - erstellt FXML Dateien
 - Integration in IntelliJ
 - Download unter:
<https://gluonhq.com/products/scene-builder/>



FXML



```
<AnchorPane prefHeight="300.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/10.0.1" xmlns:fx="http://javafx.com/fxml/1" fx:controller="com.helml.view.SmurfOverviewController">
```

```
<children>
```

```
<SplitPane dividerPositions="0.29797979797979796" prefHeight="300.0" prefWidth="600.0"
AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">
```

```
<items>
```

```
<AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="160.0" prefWidth="100.0">
```

```
<children>
```

```
<TableView fx:id="smurfTable" layoutX="-13.0" layoutY="34.0" prefHeight="200.0" prefWidth="200.0"
AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">
```

```
<columns>
```

```
<TableColumn fx:id="firstNameColumn" prefWidth="75.0" text="First name" />
```

```
<TableColumn fx:id="lastNameColumn" prefWidth="75.0" text="Last name" />
```

```
</columns>
```

```
<columnResizePolicy>
```

```
<TableView fx:constant="CONSTRAINED_RESIZE_POLICY" />
```

```
</columnResizePolicy>
```

```
</TableView>
```

```
</children>
```

```
</AnchorPane>
```

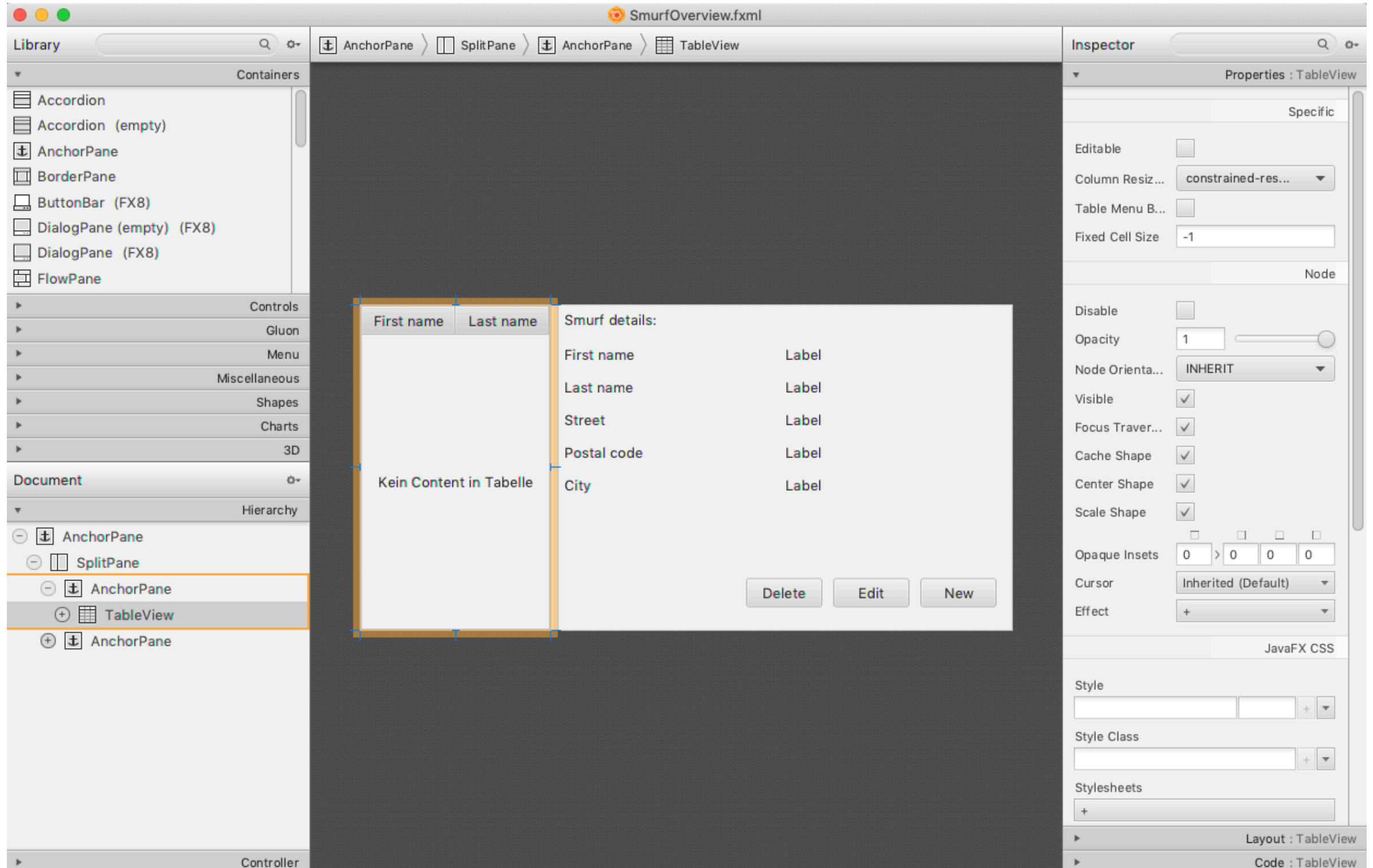
```
<AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="160.0" prefWidth="100.0">
```

```
<children>
```

```
<Label layoutX="25.0" layoutY="14.0" text="Smurf details:" AnchorPane.leftAnchor="5.0"
AnchorPane.topAnchor="5.0" />
```

```
<GridPane layoutX="51.0" layoutY="46.0" AnchorPane.leftAnchor="5.0" AnchorPane.rightAnchor="5.0"
AnchorPane.topAnchor="30.0">
```

```
<columnConstraints>
```





MVC – MODEL VIEW CONTROLLER

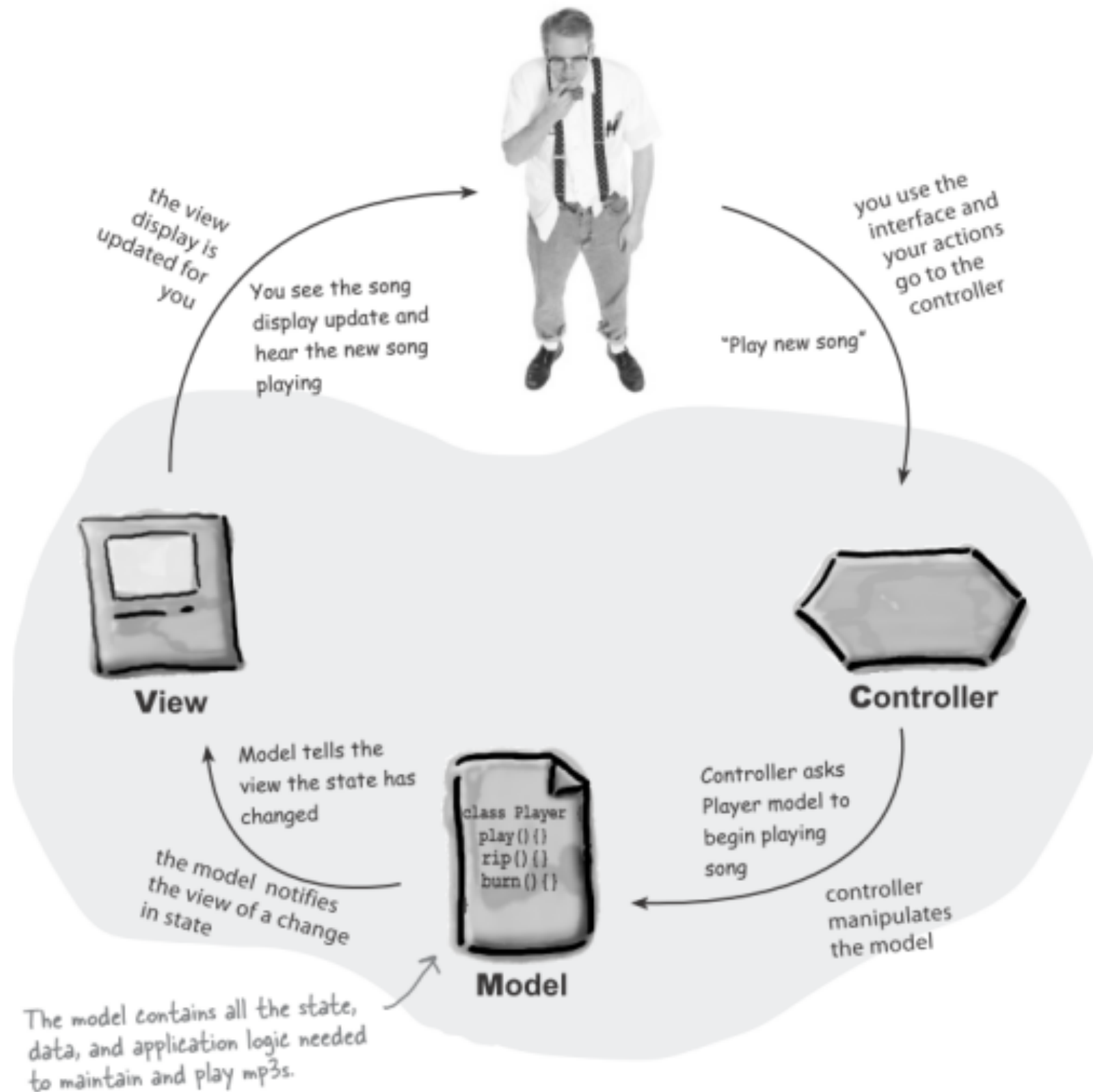




MVC – MODEL VIEW CONTROLLER



MVC – MODEL VIEW CONTROLLER



MVC – MODEL VIEW CONTROLLER

VIEW

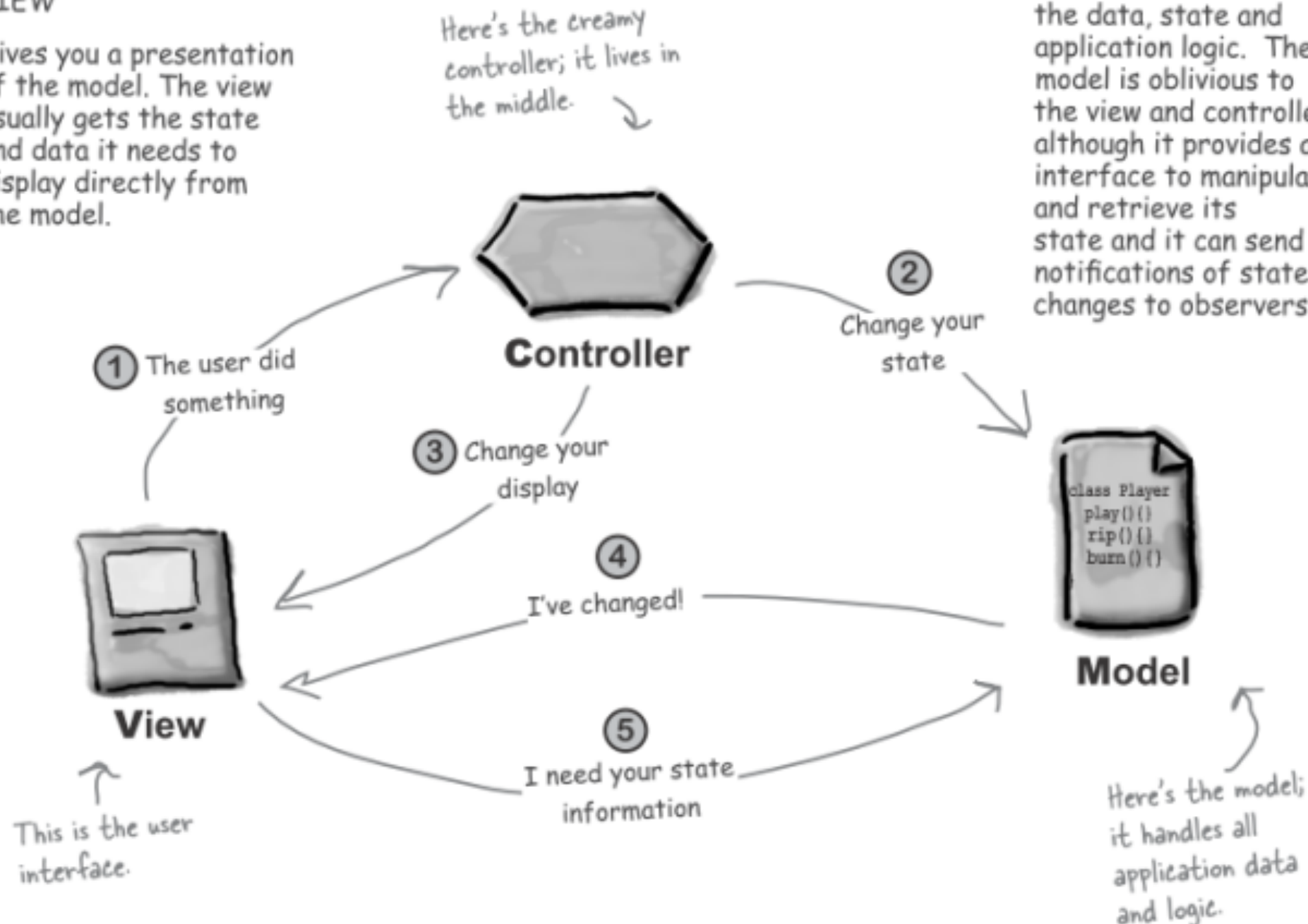
Gives you a presentation of the model. The view usually gets the state and data it needs to display directly from the model.

CONTROLLER

Takes user input and figures out what it means to the model.

MODEL

The model holds all the data, state and application logic. The model is oblivious to the view and controller, although it provides an interface to manipulate and retrieve its state and it can send notifications of state changes to observers.



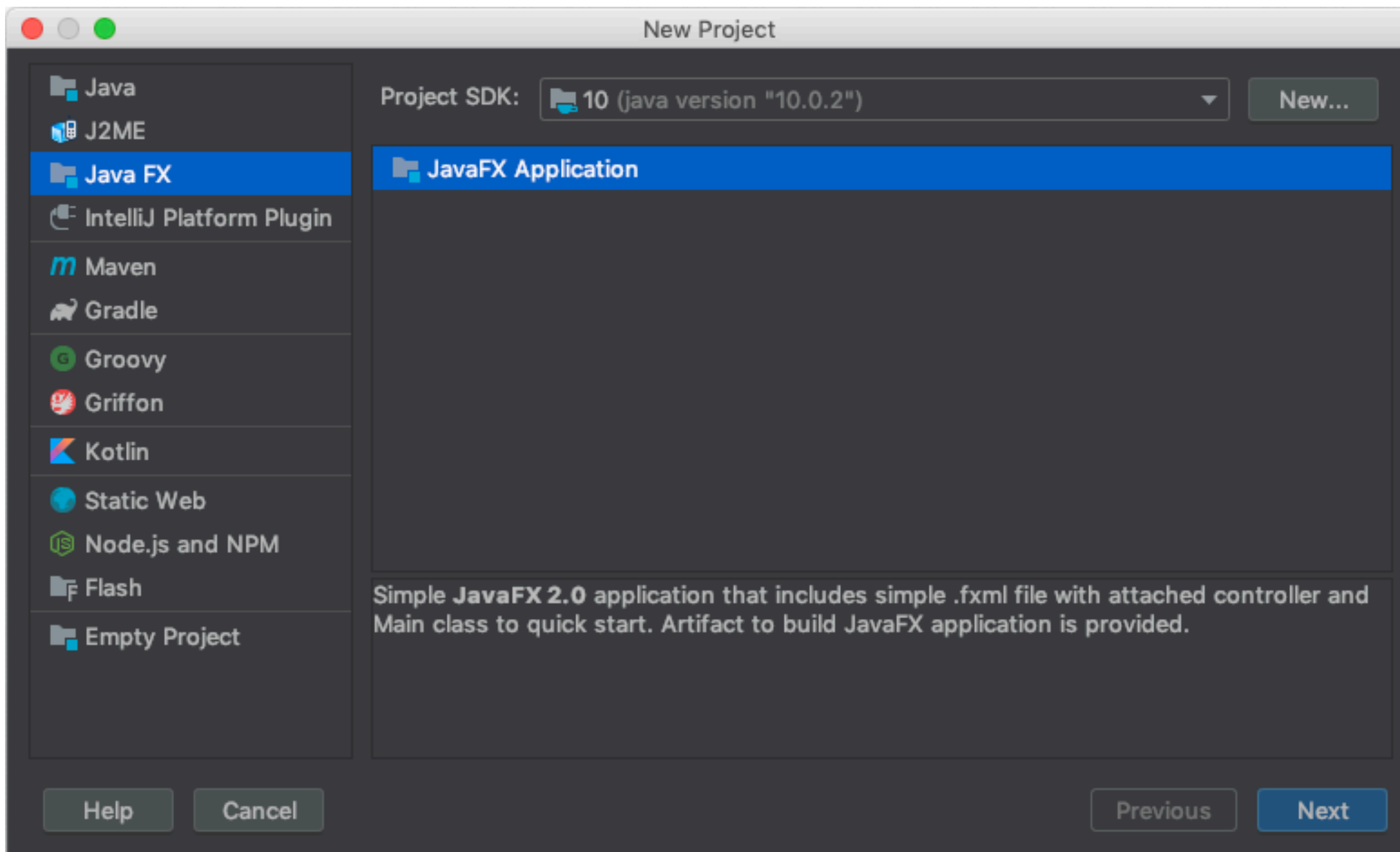


SCENEBuilder HOW-TO





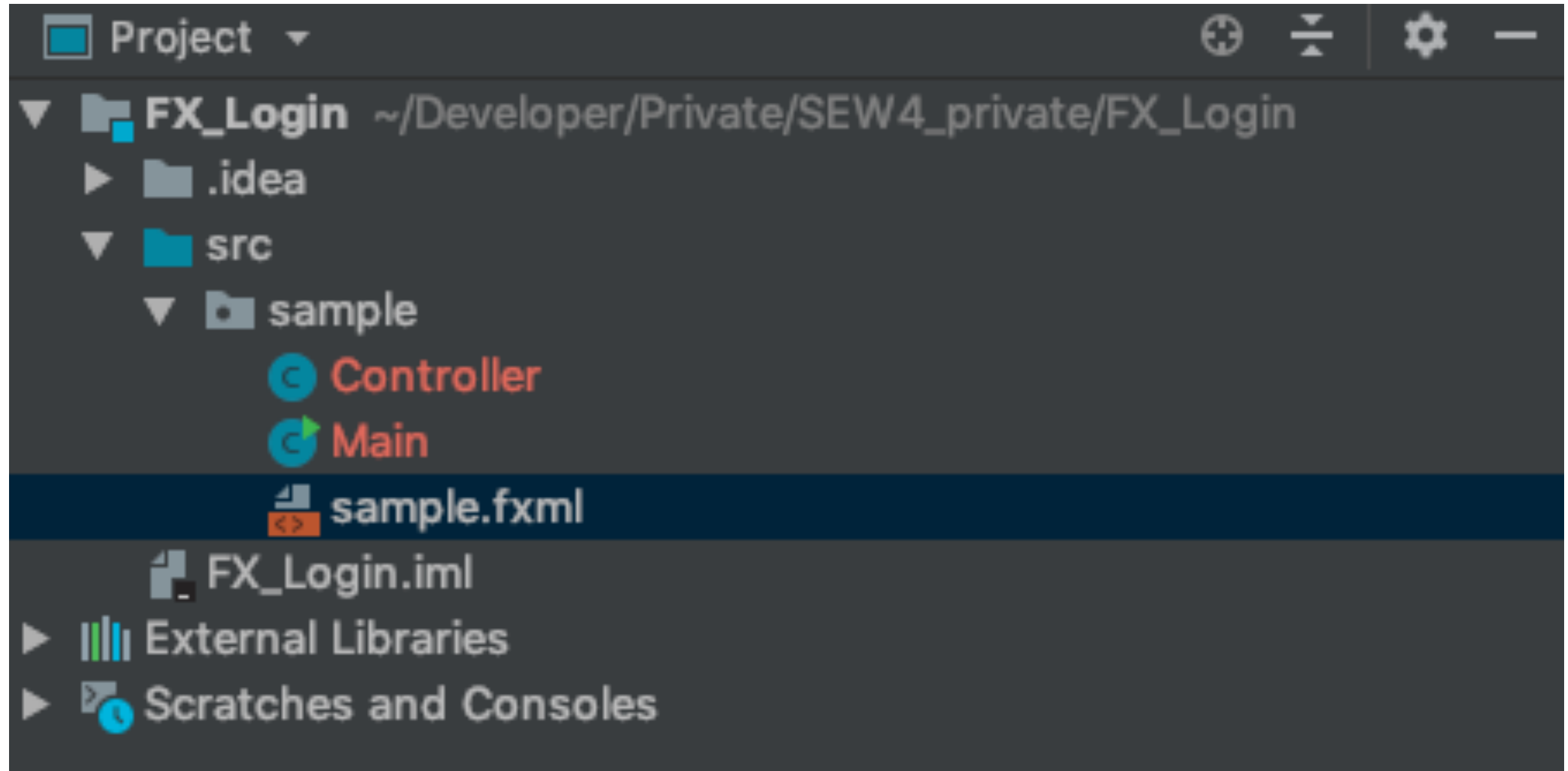
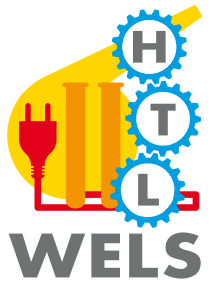
JAVAFX PROJEKT ERSTELLEN



- IntelliJ erstellt automatisch eine Reihe von Dateien samt Verknüpfungen:
 - `Main.java` - Anwendungslogik
 - `Controller.java` - Controller zur Steuerung der Benutzereingaben
 - `sample.fxml` - GUI der Anwendung, mit Scene Builder editierbar

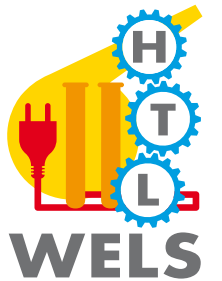


JAVAFX PROJEKT ERSTELLEN





JAVAFX PROJEKT ERSTELLEN



```
public class Main extends Application {
```

```
    @Override
```

```
    public void start(Stage primaryStage) throws Exception{
```

```
        Parent root =
```

```
            FXMLLoader.load(getClass().getResource("sample.fxml"));
```

```
        primaryStage.setTitle("Hello World");
```

```
        primaryStage.setScene(new Scene(root, 300, 275));
```

```
        primaryStage.show();
```

```
    }
```

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

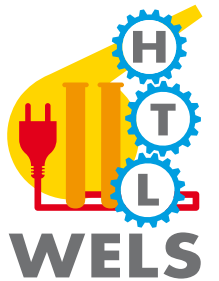
```
        launch(args);
```

```
    }
```

```
}
```



JAVAFX PROJEKT VERKNÜPFUNGEN



➤ sample.fxml:

```
<?import javafx.geometry.Insets?>
```

```
<?import javafx.scene.layout.GridPane?>
```

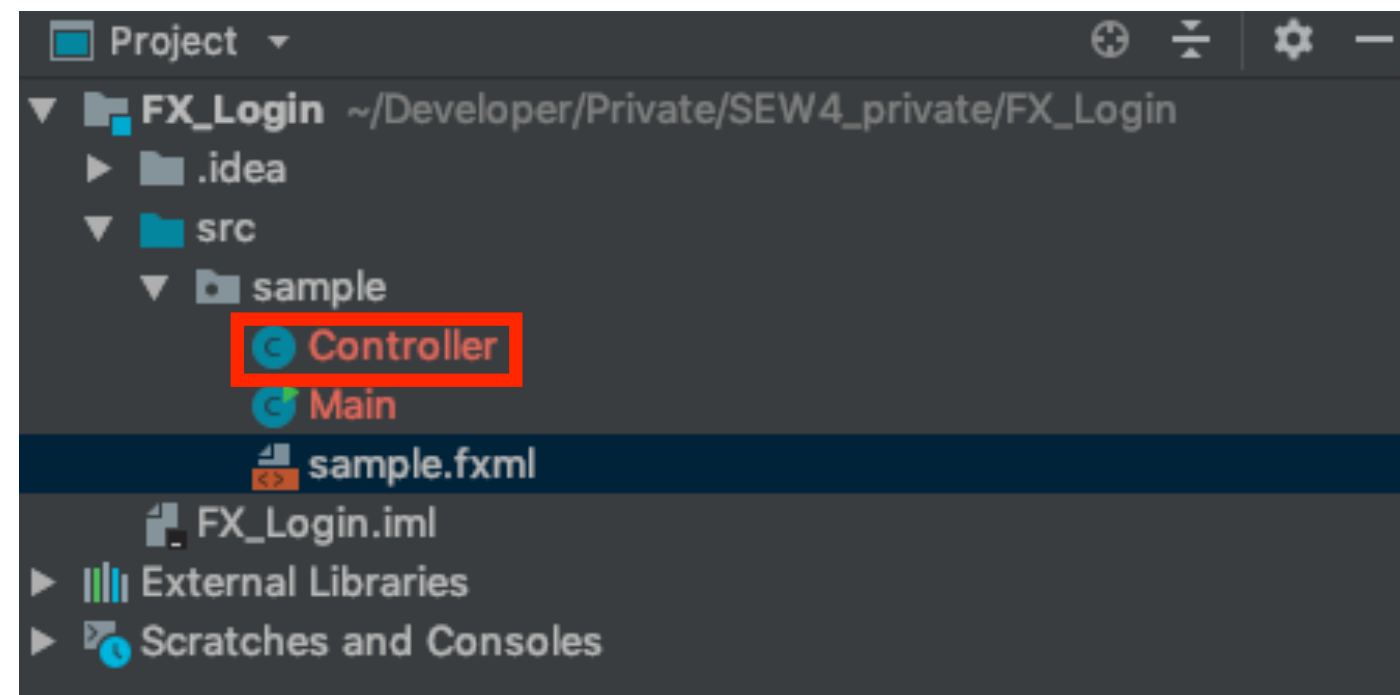
```
<?import javafx.scene.control.Button?>
```

```
<?import javafx.scene.control.Label?>
```

```
<GridPane fx:controller="sample.Controller"
```

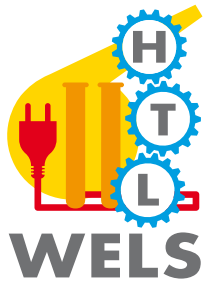
```
xmlns:fx="http://javafx.com/fxml" alignment="center" hgap="10" vgap="10">
```

```
</GridPane>
```





JAVAFX PROJEKT VERKNÜPFUNGEN



sample.fxml

Library

Containers

- Accordion
- Accordion (empty)
- AnchorPane

Controls

- Gluon
- Menu
- Miscellaneous
- Shapes
- Charts
- 3D

Document

Hierarchy

Controller

Controller class

sample.Controller

Use fx:root construct

Assigned fx:id

fx:id	Component
Kein Content in Tabelle	

Inspector

No Selection

Properties

No Selection

Project

- FX_Login ~/Developer/Private/SEW4_private/FX_Login
 - .idea
 - src
 - sample
 - Controller**
 - Main
 - sample.fxml
- FX_Login.iml
- External Libraries
- Scratches and Consoles

Layout

Code

- Der Controller (vgl. MVC) muss mit dem FXML File verknüpft sein:

- Verknüpfung Sourcecode zu FXML:

```
Parent root = FXMLLoader.load(getClass().getResource("sample.fxml"));
```

- Verknüpfung SceneBuilder zu Sourcecode:

Controller class

sample.Controller



EREIGNISBEHANDLUNG



- Definiere in der Controllerklasse ein Attribut für jedes Steuerelement, das angesprochen werden soll
- Verwende die Annotation: @FXML
- Beispiel:

@FXML

```
private TextField userNameTextField;
```

- VORSICHT bei den Imports! Nur javafx.*

- Jetzt kann die Verknüpfung auch im SceneBuilder vorgenommen werden:

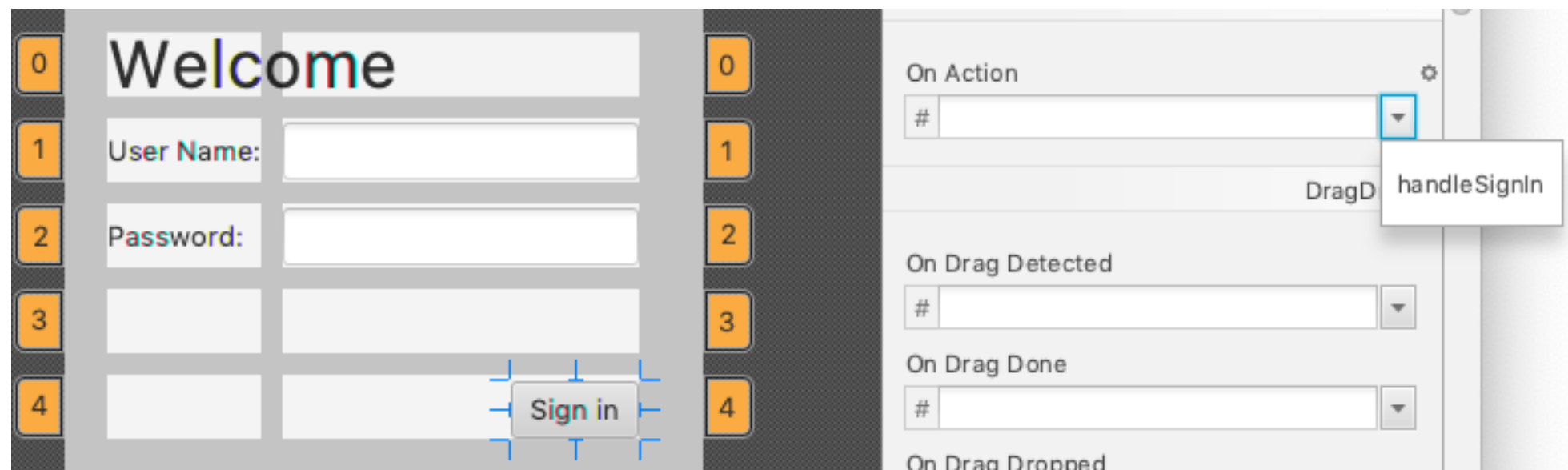


- Vorgabe:
 - Klick auf den Button „**Sign in**“ -> Baue Netzwerkverbindung auf und Prüfe Login Daten
- Dazu muss wiederum in der Controllerklasse eine Methode definiert und mittels **@FXML** annotiert werden:

@FXML

```
public void handleSignIn(){  
    System.out.println(userNameTextField.getText());  
}
```

- Jetzt kann die Verknüpfung im Scene Builder vorgenommen werden:



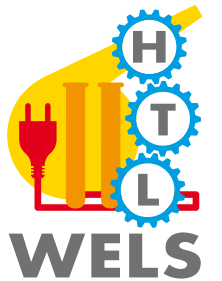


PROJEKT FX CHAT



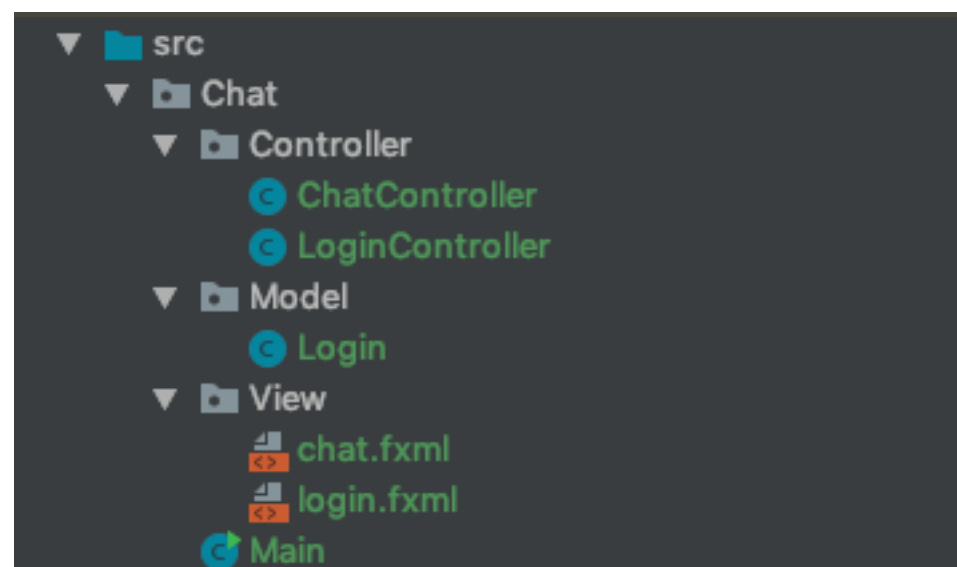


PROJEKT MAGIC 8-BALL

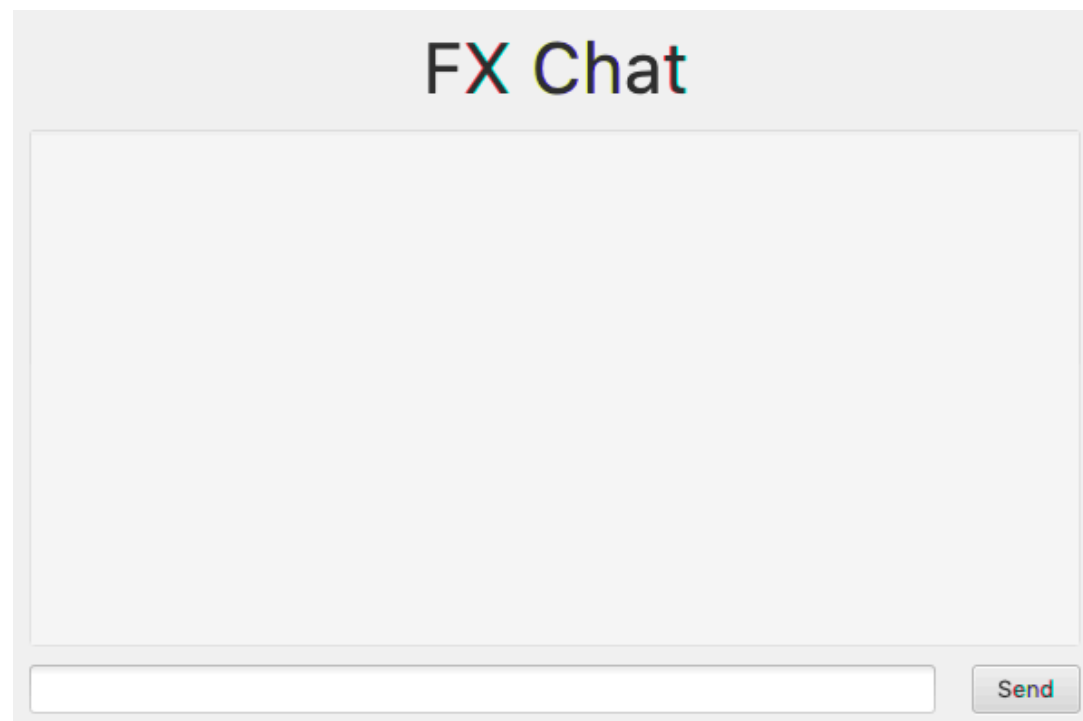


- Wir wollen eine Magic 8-Ball App erstellen
- Dazu sind ein paar Überlegungen notwendig:
 - Die Anwendungslogik kommt in die Klasse `Main`
 - Die Controller müssen mit der `Main` Klasse kommunizieren können
 - Die Anwendung besteht aus mehreren Fenster
 - Login
 - Chat
- Je Fenster ist eine View (FXML) und ein dazugehöriger Controller notwendig
- Gestartet wird im Login, nachdem sich der User eingeloggt hat, kommt er in die „Anwendung“

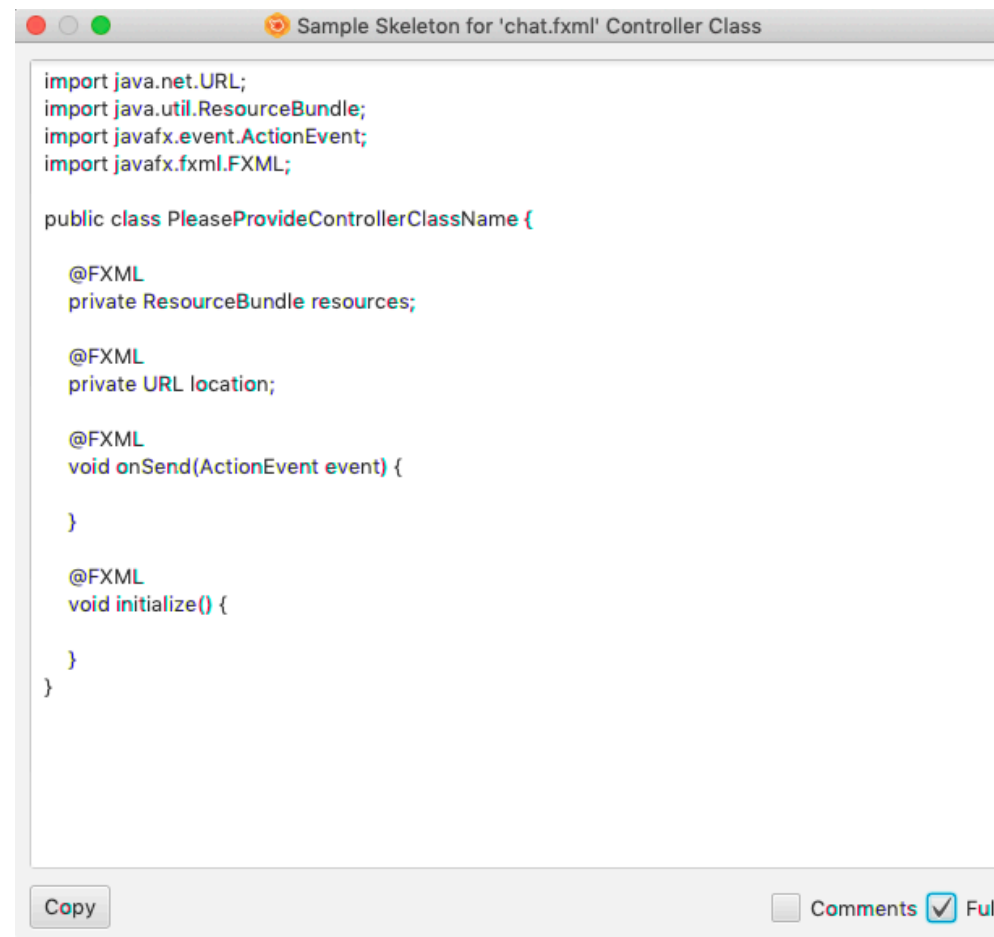
- Schritt 1:
 - Packages für Struktur
 - Klasse für ChatController, LoginController
 - neues FXML für Chat
 - Model für Login (Datenklasse für Logindaten)



- Schritt 2:
 - GUI für Chat erstellen
 - TextArea für Chatverlauf, TextField und Button für Eingabe
 - alle GUI Elemente benennen + Action für Send



- Schritt 3:
 - ChatController.java
 - im SceneBuilder auf View-Show Sample Controller Skeleton



```
import java.net.URL;
import java.util.ResourceBundle;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;

public class PleaseProvideControllerClassName {

    @FXML
    private ResourceBundle resources;

    @FXML
    private URL location;

    @FXML
    void onSend(ActionEvent event) {

    }

    @FXML
    void initialize() {

    }

}
```

Copy ☐ Comments ☒ Full



- Schritt 4:
 - Refactoring der Klasse `Main` und `LoginController`
 - `LoginController` muss mit Klasse `Main` kommunizieren



➤ LoginController.java:

...

```
// Reference to Main for communication
```

```
private Main mainApp;
```

```
public void setMainApp(Main mainApp) {
```

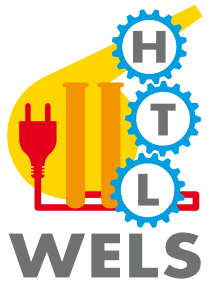
```
    this.mainApp = mainApp;
```

```
}
```

...



PROJEKT FX CHAT



➤ Main.java:

...

```
private Parent root;
```

```
@Override
```

```
public void start(Stage primaryStage) throws Exception{  
    initRootLayout(primaryStage);  
}
```

```
}
```

```
private void initRootLayout(Stage primaryStage) throws IOException {
```

```
    FXMLLoader loader = new FXMLLoader();
```

```
    loader.setLocation(getClass().getResource("view/login.fxml"));
```

```
    root = loader.load();
```

```
    primaryStage.setTitle("Login");
```

```
    primaryStage.setScene(new Scene(root, 300, 275));
```

```
    // Give the controller access to the main app.
```

```
    LoginController controller = loader.getController();
```

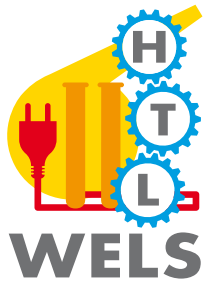
```
    controller.setMainApp(this);
```

```
    primaryStage.show();
```

```
}
```



PROJEKT FX CHAT



➤ Main.java:

...

```
public void showChatWindow() throws IOException{  
    try {  
        // Load the fxml file and create a new stage  
        Parent root = FXMLLoader.load(getClass().getResource("view/chat.fxml"));  
        ...  
    }  
}
```



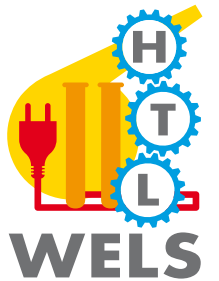
➤ Schritt 5:

➤ Datenklasse Login.java:

```
public class Login {  
    private String userName;  
    private String password;  
    ...  
}
```



PROJEKT FX CHAT



➤ Schritt 6:

➤ Login überprüfen:

➤ LoginController.java:

```
@FXML
```

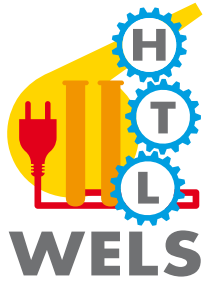
```
public void handleSignIn(){  
    System.out.println(userNameTextField.getText());  
    System.out.println(passwordField.getText());  
    mainApp.doLogin(new Login(userNameTextField.getText(), passwordField.getText()));  
}
```

➤ Main.java:

```
public void doLogin(Login loginData){  
    System.out.println("User hat sich eingeloggt");  
    // Check User + Password  
    // if ok, show Chat Window  
    // ...  
}
```



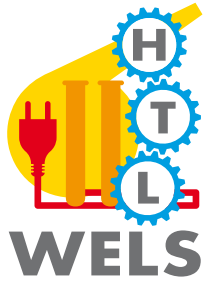

PROJEKT FX CHAT



- Schritt 7:
 - Was sonst noch fehlt
 - Logik, Antworten, ...



PROJEKT FX CHAT



- Implementiere auf Basis der letzten Folien das Magic 8-Ball
 - 2 User oder mehr
 - User/Passwort sind in einem Login-Array gespeichert