Wieso brauchen wir JavaFX

Niklas von Hirschfeld, Jan Knüpfer

2024-01-18

Contents

Was ist JavaFX?

JavaFX bietet viele möglichkeiten mit Java ein grafische Oberfläche zu Porgrammieren. Es bietet neben bindings für OpenGL eine recht hohe API um auf diese zuzugreifen. Es ist ein Moderner nachfolger von dem veralteten Swing framework.

JavaFX nutzt einen *scenen Graf* um Objekte und Scenen zu sortieren und zugänglich.

Die Basis stellt eine *Stage* dar. Die ist vergleichbar mit einer Bühne eines Theaterstückes, auf der Verschiedene Scenen und Schauspiele auftreten können. Eine Scene beinhaltet die Schauspieler und die Requisieten. In Java wären diese klassisches Objekte, welche durch einfache Formen oder Grafiken dargestellt werden. Diese Objekte beinhalten die Grafische darstellung, position, variablen und funktionen, welche für dieses Objekt wichtig sind.

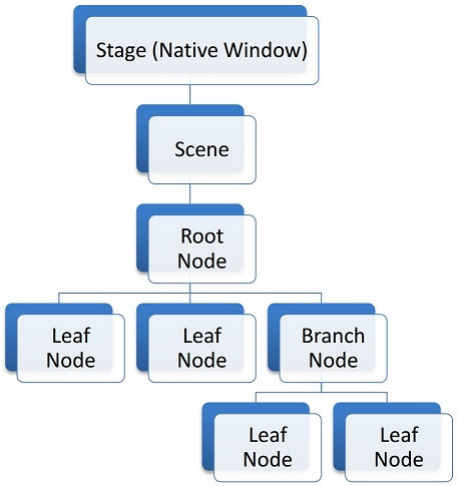

Figure 1: Grafisch darstellung einer *Stage*

Was bietet JavaFX

Neben einer hohen und einfachen API zu OpenGL bietet JavaFX von Haus aus Module für *Text*, *Knöpfe* und Bilder. Es ermöglicht auch die einfache Verwendung von anderen Medien, wie Video- und Audiodatein.

```
public class HelloApp extends Application {

    private Parent createContent() {
        return new StackPane(new Text("Hello World"));
    }

    @Override
    public void start(Stage stage) throws Exception {
        stage.setScene(new Scene(createContent(), 300, 300));
        stage.show();
    }

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

Diese Zeilen allein (plus Import der Libary) reichen aus, um ein Fenster mit dem Text “Hello World” zu erstellen. Anhand des Beispiels lässt sich auch die grundlegende Herangehensweise erkennen. Die Klasse `HelloApp` erweiter die Klasse `Application`, welche von JavaFX gestellt wird. Diese Klasse beinhaltet bereits eine reihe von Funktionen, welche genutzt oder auch überschrieben wer-

den können. `start(Stage stage)` ist eine solche Funktion. Sie wird ausgeführt, nachdem JavaFX initialisiert wurde und stellt den Ort da, an dem die `stage` standardmäßig definiert wird. `privat Parent createContent()` stellt ein Objekt beziehungsweise in dem Fall nur eine einfache Funktion dar, welche eine `StackPane` mit einem Text zurück gibt. `stage.setScene(new Scene(...))` setzt die Szene mit. Das Praktische ist, Szenen können unabhängig und im vorraus erstellt werden.

CSS

JavaFX nutzt eine eigene Version von `Cascading Style Sheets` (CSS), welches auf der W3C CSS Version 2.1 basiert. Dies wird in erster Linie genutzt, um Themen zu erstellen und zu bearbeiten.

JavaFX CSS unterstützt allerdings keine Layout Eigenschaften wie `float`, `position`, `overflow`, etc. Es gibt insgesamt viele Einschränkungen aber eben auch Erweiterungen, speziell für JavaFX.

Event Handlers

In JavaFX werden Event Handler verwendet, um auf Benutzerinteraktionen oder andere ereignisbezogene Aktivitäten zu reagieren.

JavaFX bietet dabei Event-Klassen, zum Beispiel: MouseEvent, KeyEvent, ActionEvent und Event-Quellen, also JavaFX-Steuerelemente wie `Textfeld`, `Knöpfe`, etc.

Auf solche Event kann man mit `EventHandler` reagieren. Diese sind wie folgt aufgebaut:

```
EventHandler<ActionEvent> buttonClickHandler = new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        // Hier steht der Code, der auf den Klick der Schaltfläche reagiert
        System.out.println("Button wurde geklickt!");
    }
};
```

Wenn der `EventHandler` einmal definiert ist, kann dieser Verschiedenen Objekten hinzugefügt werden:

```
Button myButton = new Button("Klick mich!");
myButton.setOnAction(buttonClickHandler);
```

Man muss allerdings nicht im Voraus einen `EventHandler` an sich Definieren, sonder kann auch individuell für ein Objekt direkt beschreiben, was passieren soll:

```
myButton.setOnAction(e -> {
    System.out.println("Button wurde geklickt!");
```

```
});
```

FXML

FXML ist ein ist ein Modul von JavaFX. Diese erlaubt es, Szenen als .xml Datein zu speichern. Dadurch ist es auch möglich, auf grafische Programme zurückzugreifen, um ein Nutzerinterface zu erstellen.

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

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

<AnchorPane id="AnchorPane" prefHeight="200" prefWidth="320" xmlns:fx="http://javafx.com/fx
    <children>
        <Button fx:id="button" layoutX="126.0" layoutY="90" onAction="#handleButtonAction" t
        <Label fx:id="label" layoutX="126" layoutY="120" minHeight="16" minWidth="69" />
    </children>
</AnchorPane>
```

Integrierung in den JavaEditor

JavaFX wird direkt von dem JavaEditor unterstützt und vor allem auch in den Editor integriert.

Alternativen

Java AWT

- https://www.javatpoint.com/java-awt

Java AWT (Abstract Window Toolkit) wurde in den frühen Tagen von Java eingeführt und diente als Grundlage für die GUI-Entwicklung. Allerdings hat es im Vergleich zu JavaFX mehrere Einschränkungen, die es weniger geeignet für die Spieleentwicklung machen.

Quellen:

- Javaeditor.org
- JavaFX Dokumentation von Oracle