Einstieg in die Programmierung von Computern – Teil VIII

Im Teil 8 beschäftigen wir uns weiter mit dem Thema „GUI-Programmierung mit JavaFX“ und seinen grundlegenden Konzepten. JavaFX bietet eine moderne und zukunftsorientierte technische Plattform für die Entwicklung von grafischen Benutzeroberflächen. Warum und mit welchen Mitteln werden wir in diesem theoretischen Teil erfahren.

```
/**
 * Copyright (c) 2008, 2012 Oracle and/or its affiliates.
 * All rights reserved. Use is subject to license terms.
 */

package com.oemus.itkolumne.data; // package-Anweisung

import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.stage.Stage;
import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.scene.layout.*;
import javafx.scene.web.WebEngine;
import javafx.scene.web.WebView;

/**
 * Klasse WebViewSample demonstriert die Verwendung der webEngine.
 *
 */
public class WebViewSample extends Application {

    public static final String DEFAULT_URL = "http://www.oemus.com";

    /**
     * Initialisierung der JavaFX Webbrowser-Komponenten
     */
    private void init(Stage primaryStage) {

        Group root = new Group();
        primaryStage.setScene(new Scene(root));
        WebView webView = new WebView();
        final WebEngine webEngine = webView.getEngine();

        // Laden der oben definierten Default-URL
        webEngine.load(DEFAULT_URL);

        final TextField locationField = new TextField(DEFAULT_URL);

        /**
         * webEngine special URL-Handling
         */
        webEngine.locationProperty().addListener(new ChangeListener<String>() {
            @Override
            public void changed(ObservableValue<? extends String> observable,
                                String oldValue,
                                String newValue) {
                                locationField.setText(newValue);
            }
        });

        /**
         * Event-Behandlung der Laden-Schaltfläche
         */
        EventHandler<ActionEvent> goAction = new EventHandler<ActionEvent>() {
            @Override public void handle(ActionEvent event) {
```

Kasten 1: JavaFX Webbrowser-Applikation (Fortsetzung S. 9).

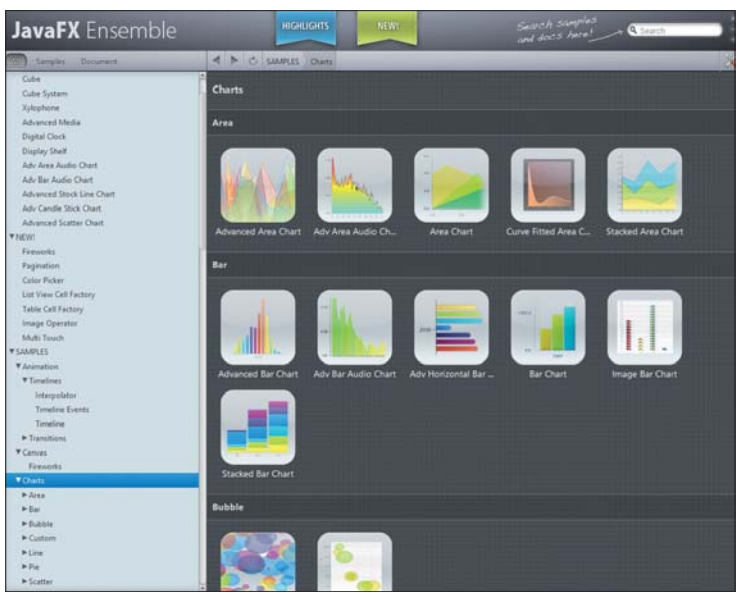

Abb. 1: JavaFX-Ensemble dient zum Kennenlernen der JavaFX-Komponenten.

Die grundlegenden Konzepte von JavaFX

Das Basiskonzept von JavaFX ist der sogenannte „Szenengraph“ und besteht aus insgesamt den drei Bausteinen:

Stage
Stellt das „Top-Level-Window“ (auch als Fenster bezeichnet) auf der jeweiligen Plattform (z. B. Windows) dar. Die „Stage“ besteht wieder aus den Elementen:

- Fenster-Inhalt: das eigentliche Programm
- Dekoration: Rahmen und die Schaltflächen „Minimieren“, „Maximieren“ und „Schließen“

Scene
Jede der oben beschriebenen Stage enthält eine „Scene“, die den Fenster-Inhalt beinhaltet.

Nodes
Eine Scene wiederum besteht aus einer Baumstruktur, die beliebig „Knoten“ *(engl. Nodes)* enthält. Eine „Node“ ist entweder ein „GUI-Element“ oder ein „Container“, der wiederum Nodes enthalten kann.

Wie werden grafische Benutzeroberflächen in JavaFX erstellt?

Der Softwareentwickler hat grundsätzlich zwei unterschiedliche Techniken, um eine grafische Benutzeroberfläche in JavaFX zu erstellen:

- in der Programmiersprache Java oder
- in der Markup-Sprache *(engl. Auszeichnungssprache)* FXML. FXML ist ein proprietärer XML-*(engl. Extensible Markup Language)* Dialekt zur Beschreibung von grafischen Benutzeroberflächen extra für JavaFX.

Die Entwicklung einer grafischen Benutzeroberfläche in JavaFX mittels der Programmiersprache Java selbst (also nicht in FXML) gestaltet sich relativ einfach:

1. Man erzeugt Instanzen der „Komponentenklassen“.
2. Die erzeugte Komponentenklassen-Instanzen werden dann hierarchisch im Scene Graph eines Top-Level-Window (Stage) platziert.

JavaFX-Komponenten

Die umfangreiche Komponenten-Bibliothek von JavaFX bietet dem Entwickler alle Möglichkeiten, moderne und top gestylte Benutzeroberflächen zu gestalten. Was bietet die JavaFX Komponenten-Bibliothek?

- **Window-Komponenten** (z. B. Stage oder Window)
- **Input-Komponenten** (z. B. Textfelder, Schaltflächen, Auswahlboxen, Checkboxen, Kalenderfelder, …)
- **Layout- & Container-Komponenten** (z. B. VBox, HBox, Pane, …)
- **Grafische Formen-Komponenten** wie z.B. Circle und RecTangle
- **Tabellen- & Tree-Komponenten** für die komfortable Anzeige von Daten bzw. Informationen in strukturierten Tabellen und Bäumen *(engl. Trees)*

Das Aussehen der JavaFX-Komponenten

Das wirklich tolle Feature von JavaFX ist die Möglichkeit, mittels „CSS“ (Cascading Style Sheets) das Aussehen individuell anzupassen. Mit CSS kann auch während der Laufzeit die Optik der Komponenten geändert bzw. angepasst werden. Die CSS-Anweisungen werden in einer separaten Style Sheet- bzw. CSS-Datei geschrieben. JavaFX leifert standardmäßig die CSS-Datei „caspian.css“ mit, die in der Tat bereits sehr gut gelungen ist.
Welche CSS-Dateien dann konkret verwendet werden, passiert auf oberster Ebene des Top-Level-Window oder kann auch für jeden Knotenpunkt definiert werden.
Abbildung 1 zeigt die von Oracle zum kostenlosen Download bereitgestellte JavaFX-Ensemble-Applikation zum Kennenlernen der JavaFX-Komponenten.
Für jedes Komponenten-Beispiel kann auch der entsprechende JavaFX-Quellcode betrachtet werden, sodass auch der JavaFX-Anfänger schnell den Umgang mit JavaFX lernen kann. Schauen Sie sich ruhig alle

Komponenten an und Sie werden feststellen, dass JavaFX in puncto GUI-Entwicklung keine Wünsche offenlässt.
Im Kasten 1 ist der JavaFX Sourcecode für einen Webbrowser ab-

gebildet. Eigentlich unglaublich, wie einfach mit wenigen Codezeilen ein Internet-Webbrowser in JavaFX zu erstellen ist.

```
                webEngine.load(locationField.getText().startsWith("http://")
                        ? locationField.getText() : "http://" + locationField.getText());
            }
        };

        locationField.setOnAction(goAction);

        // Start-Schaltfläche zum Laden der Webseite
        Button goButton = new Button("Laden");
        goButton.setDefaultButton(true);
        goButton.setOnAction(goAction);

        // Hier wird das Layout erstellt
        HBox hBox = new HBox(5);
        hBox.getChildren().setAll(locationField, goButton);
        HBox.setHgrow(locationField, Priority.ALWAYS);

        VBox vBox = new VBox(5);
        vBox.getChildren().setAll(hBox, webView);
        VBox.setVgrow(webView, Priority.ALWAYS);

        root.getChildren().add(vBox);
    }

    /**
     * Anzeigen der Stage
     */
    @Override
    public void start(Stage primaryStage) throws Exception {
        init(primaryStage);
        primaryStage.show();
    }

    /**
     * Start der JavaFX-Anwendung
     */
    public static void main(String[] args) {
        launch(args);
    }
}
```

Fortsetzung Kasten 1: JavaFX Webbrowser-Applikation.

Behandlung von Ereignissen in JavaFX

Die Ereignisse *(engl. Events)* werden in JavaFX folgendermaßen behandelt:

Filterphase
In dieser ersten Phase der Event-Behandlung kann der Event Listener (ein Event Listener bekommt ein Event übergeben, wenn er für eine bestimmte Komponente aufgetreten ist) auch entscheiden, dass ein Event nicht verarbeitet wird. Man spricht hier auch von Filterung der Events. Ein Event kann sozusagen weggefiltert werden.

Verarbeitungsphase
In dieser zweiten Phase kann der Event Listener in Java implementiert werden.
In den beiden Phasen wandern die Events im Scene Graph (Knoten-Baum) nach oben. Ein Knoten im Knoten-Baum oder der letzte Knoten im Baum kann sich den Event schnappen und die Event-Behandlung somit abbrechen. Es wird zwischen zwei Arten von Events unterschieden:

- **Echten Events** z. B. Maus-Events (Benutzer zieht mit der Maus) an der Ecke eines Dialogfensters
- und den draus resultierenden **Änderungen**, wie z. B. die Änderung der Fenstergröße.

Ausblick zum Teil 9

Im nächsten Teil werden wir uns mit dem äußerst interessanten Thema Multithreading in Java beschäftigen. Bleiben Sie also dran! ZT

ZT Autor

Infos zum Autor

Thomas Burgard

ZT Adresse

Thomas Burgard Dipl.-Ing. (FH)
Softwareentwicklung & Webdesign
Bavariastraße 18b
80336 München
Tel.: 089 540707-10
info@burgardsoft.de
www.burgardsoft.de
burgardsoft.blogspot.com
twitter.com/burgardsoft