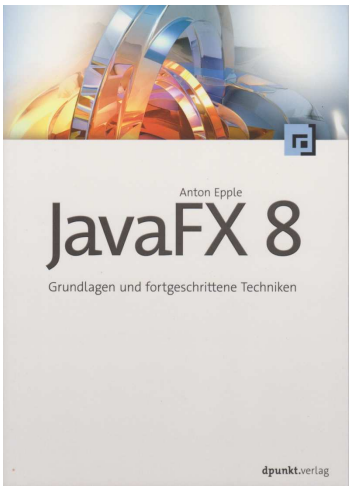
Graphische Nutzerschnittstellen

- Dipl.-Inf., Dipl.-Ing. (FH) Michael Wilhelm
- Hochschule Harz
- FB Automatisierung und Informatik
- mwilhelm@hs-harz.de
- http://www.miwilhelm.de
- Raum 2.202
- Tel. 03943 / 659 338

Inhalt

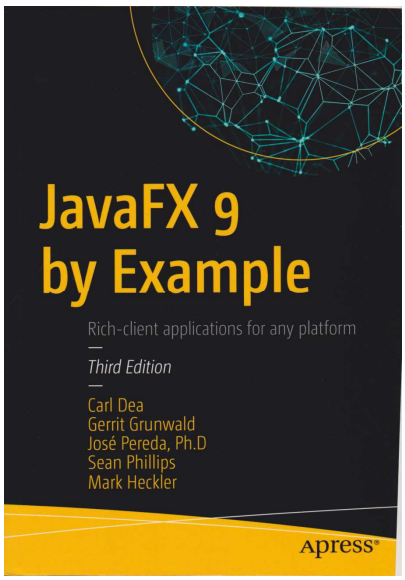
Literatur



- Java FX 8
- Anton Epple
- dpunkt.verlag
- ISBN 978-3-86490-169-0

- Guter Einstieg

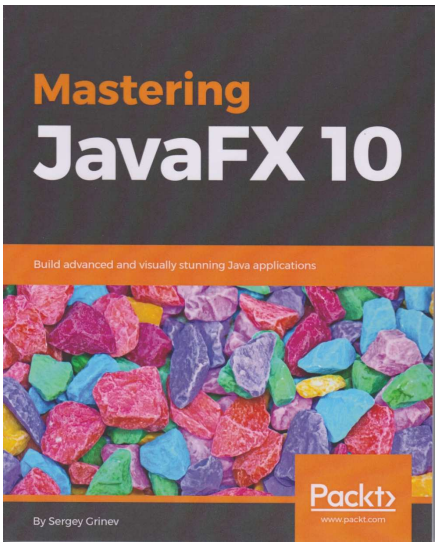
Literatur



- Java FX 9 by Example
- Carl Dea …
- Apress Verlag
- ISBN 13-978-1-4842-1960-7

- Sehr Umfangreich

Literatur



- Mastering Java FX 10
- Sergey Grinev
- packt.verlag
- ISBN 978-1-78829-382-2

- Nicht unbedingt notwendig

Java-Geschichte: User Interface

- **Java AWT**
  - Verwendet Standard-UI-Elemente des BS
  - Unterschiedliches Aussehen
- **Java Swing**
  - Hat eigene UI-Elemente
  - Gleiches Aussehen (Linux, OS, Windows, Sun)
  - Langsam
- **Java SWT**
  - Verwendet Standard-UI-Elemente des BS
  - Unterschiedliches Aussehen (Eclipse)
  - schnell
- **Java FX (ab 2007)**
  - Komplette Neuentwicklung

Java FX

- **Geschichte**
  - Object-orientiertes UI-Framework
  - Version 1,0 wurde 2008 veröffentlicht.
  - Seite JDK 7 Update 6 im Java SDK integriert
  - Ab SDK Version 11 separate jar-Dateien
  - Hat sich noch nicht als Alternative zu Swing durchgesetzt.

Java FX

- **Eigenschaften**
  - FXML-Struktur (optional)
  - **Neue UI-Elemente**
    - Charts
    - ColorPicker
    - ChoiceBox
    - HyperLink
    - Accordion
    - ProgessIndicator
    - TreeTableView
    - WebView
  - **Es fehlen**
    - MDI-Fenster (SplitPane)
  - UI-Element-Anbindung (ListModel vs. ListCollection)
  - Data Binding / Properties
  - Audio / Video
  - CSS-Anbindung

Java FX: FXM Struktur

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.CheckBox?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.layout.AnchorPane?>

<AnchorPane id="AnchorPane" prefHeight="200" prefWidth="320" ..>
   <children>
      <Button fx:id="button" layoutX="126" layoutY="90"
         onAction="#handleButtonAction" text="Click Me!" />

      <Label fx:id="label" layoutX="126" layoutY="120"
         minHeight="16" minWidth="69" />

      <CheckBox layoutX="191.0" layoutY="28.0"
         mnemonicParsing="false" text="CheckBox" />
   </children>
</AnchorPane>
```

Ähnlich WPF XAML

Java FX: FXM Struktur

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<VBox id="Vbox" prefHeight="200" prefWidth="320" ..>
   <children>
      <Button fx:id="button" layoutX="126" layoutY="90"
         onAction="#handleButtonAction" text="Click Me!" />

      <Label fx:id="label" layoutX="126" layoutY="120"
         minHeight="16" minWidth="69" />

      <CheckBox layoutX="191.0" layoutY="28.0"
         mnemonicParsing="false" text="CheckBox" />
   </children>
</AnchorPane>
```

Java FX: FXM Struktur

- **Vorteile**
  - Übersichtlicher als Code Behind (pur Java)
  - Trennung UI vs. Code
  - WYSIWYG Editor (Scene Builder)
  - Man benötigt trotzdem dynamische UI-Programmierung
  - Multitouch Support. (je nach Platform).
  - Hardware-accelerated graphics pipeline (Prism, GPU).
  - High-performance Media engine (GStreamer multimedia framework).
- **Nachteile**
  - Statisch
  - Man benötigt trotzdem dynamische UI-Programmierung
  - Debugging ist schwieriger
  - Refaktoring von Java-Code ändert nicht unbedingt den FXML-Code
  - ID müssen übereinstimmen
    - @FXML

Java FX: Neue UI-Elemente

- **Neue UI-Elemente**
  - ChoiceBox
    - Art einer ComboBox
  - Accordion
    - Mehrere Schalter, automatisches Schließen
  - ToolTip
  - Charts
    - Einfache Charts
  - TreeTableView
    - Tree mit einer Tabelle
  - Pagination
  - ColorPicker
  - HyperLink
  - ProgessIndicator
  - WebView

https://docs.oracle.com/javase/8/javafx/user-interface-tutorial/ui_controls.htm

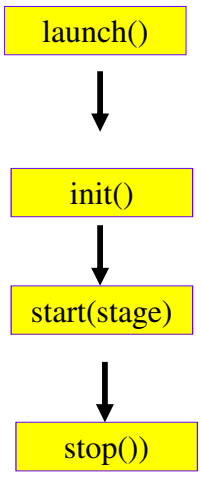
Java FX: Weitere UI-Elemente

- **UI-Elemente**
  - Label
    - Beschriftung für andere UI-Elemente
  - Button / ToggleButton
  - CheckBox
  - ContexMenu
  - ListView
  - MenuBar / Menu / MenuItem
  - PasswordField
  - ProgressBar
  - RadioButton
  - Scrollbar
  - Separator
  - Slider
  - SplitPane

- **UI-Elemente**
  - TabPane / Tab
  - TableView
  - TableColumn
  - TableCell
  - TextArea
  - TextField
  - TitlePane
  - ToolBar
  - TreeView
  - TreeItem

Java FX

- **Grobe Struktur**
  - launch(String[] argv)
    - Ruft init(), start() und stop() auf.
  - init()
    - Ist leer, kann überschrieben werden.
  - start(Stage stage)
    - Diese Methode muss überschrieben werden.
    - Stage ist das erste Fenster.
  - stop()
    - Ist leer, kann überschrieben werden.



Java FX: 1. Beispiel

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;

public class Test2 extends Application {

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


Java FX: 1. Beispiel

```
    @Override
    public void start(Stage stage) {
        StackPane root = new StackPane();
        Label label = new Label("Hello World");
        root.getChildren().add(label);
        Scene scene= new Scene(root, 200, 200);
        stage.setTitle("Hello World Example");
        stage.setScene(scene);
        stage.show();
    }

}
```

Java FX

- **Nomenklatur**
  - Stage — Fenster
  - Scene — Fensterinhalt
  - Nodes — Elemente in der Scene (UI-Elemente)
    - Root-Node