JavaFX

Print Version des Blogs  
<https://blog.kneitzel.de>

# Übersicht

Ich habe einen Blog mit diversen IT Themen unter <https://blog.kneitzel.de>. Eine Serie beschäftigt sich mit JavaFX.

Anlass für die Blog Einträge waren Fragen und Problemstellungen, denen ich immer wieder im [Java Forum](https://www.java-forum.org/)[[1]](#footnote-1) begegnet bin.

Kernpunkte sind der Blog Einträge sind:

* Installation von JavaFX, da hier schon einige Anwender in massive Probleme laufen. Dabei ist JavaFX ohne komplexe Installation mit Hilfe von Gradle oder Maven nutzbar.
* Layouts – Eine Übersicht, wie mit einfachen Mitteln eine Oberfläche schnell und einfach im Code erstellt werden kann.
* Scene Builder / MVC Pattern – dieses stelle ich nur kurz vor und zeige ein paar kleine Probleme auf.
* MVVM Pattern mit [mvvmFX](https://github.com/sialcasa/mvvmFX/wiki)[[2]](#footnote-2).

Die Veröffentlichung erfolgt auf mehreren Kanälen:

1. Dieses Dokument / Buch als druckbare Version. Dieses Dokument wird fortlaufend aktualisiert und lebt.
2. Artikel in meinem Blog. Der Blog bildet eine einfache Möglichkeit, neue Beiträge gezielt mit zu bekommen. Änderungen werden aber nur in schwerwiegenden Fällen eingearbeitet.
3. Ich werde versuchsweise ein paar Videos erstellen und über YouTube veröffentlichen. SO ein Video verfügbar ist, werde ich dieses entsprechend im Dokument verlinken.

## Code

Die Code Beispiele finden sich auf GitHub unter <https://github.com/kneitzel/blog-javafx-series>.

Mit Hilfe des Tools git kann eine lokale Kopie erzeugt werden mit dem Befehl:

git clone <https://github.com/kneitzel/blog-javafx-series.git>

## Fragen und Anregungen

Fragen rund um Java werden am besten im Java Forum gestellt. Dort sind viele aktive User, die sehr hilfsbereit auf alle Fragen eingehen.

Spezifische Fragen zum Blog als auch Anregungen können direkt per Email an mich gerichtet werden: [konrad@kneitzel.de](mailto:konrad@kneitzel.de).

## Autor

Ich arbeite als Senior Software Engineer für einen großen IT Konzern. In erster Linie entwickle ich Enterprise Applikationen in Java und C++.

Ich arbeite als Senior Software Engineer für einen großen IT Konzern. In erster Linie entwickle ich Enterprise Applikationen in Java und C++.

Inhalt

[Übersicht 2](#_Toc53682077)

[Code 2](#_Toc53682078)

[Fragen und Anregungen 2](#_Toc53682079)

[Autor 2](#_Toc53682080)

[Installation 4](#_Toc53682081)

[Java / JavaFX 4](#_Toc53682082)

[IDEs Gradle 4](#_Toc53682083)

[Erste Applikation 4](#_Toc53682084)

[Scene Builder 4](#_Toc53682085)

[JavaFX Einführung 4](#_Toc53682086)

[Aufbau 4](#_Toc53682087)

[Layouts 4](#_Toc53682088)

[Model / View / Controller (MVC) 4](#_Toc53682089)

[Übersicht 4](#_Toc53682090)

[Beispiel Applikation 4](#_Toc53682091)

[Probleme 4](#_Toc53682092)

[Model / View / Viewmodel (MVVM) 4](#_Toc53682093)

[Applikation mit MVVM 4](#_Toc53682094)

# Installation

Damit mit JavaFX gearbeitet werden kann, müssen gewisse Voraussetzungen erfüllt sein.

* Betriebssystem: Java / JavaFX und die IDEs sind für die üblichen Betriebssysteme verfügbar. Es spielt keine Rolle, ob Windows, Mac OS oder ein Linux verwendet wird.
* Es wird eine Java Installation benötigt.
* Eine Entwicklungsumgebung ist nicht zwingend notwendig, aber ich empfehle, eine der üblichen IDEs zu nutzen: IntelliJ, Eclipse oder Netbeans.
* Es gibt Build Tools: Gradle / Maven. Diese sind in der Regel nicht manuell zu installieren. Bezüglich Gradle führe ich dies später etwas aus.

## Java / JavaFX

Java ist in diversen Versionen verfügbar. Neben der eigentlichen Versionierung (Java 1.4, 1.5, 1.6, 1.7, 8, 9, … 15) ist zwischen JRE (Java Runtime Enviroment) und JDK (Java Development Environment) zu unterscheiden. Die JRE umfasst nur notwendige Teile zur Ausführung von Java Programmen. Das JDK enthält auch alle Tools, die ein Entwickler benötigt und enthält ebenfalls ein JRE.

|  |  |
| --- | --- |
| Achtung | Da wir eigene Java Programme entwickeln wollen, benötigen wir ein JDK. |

Weiterhin kann man die Software direkt von Oracle herunter laden oder man nutzt ein OpenJDK von einem anderen Anbieter wie z.B. AdoptOpenJDK oder Zulu Community von Azul.

|  |  |
| --- | --- |
| Hinweis | Oracle bietet das JRE nur bis Java 8 an. |

Um eine für uns geeignete Version zu finden, benötigen wir noch ein paar Informationen:

* Alles vor Java 8 ist extrem veraltet. Diese Versionen bekommen keine Updates und Sicherheitsprobleme sind vorprogrammiert. Weiterhin sind ggf. Libraries und Tools nicht mehr kompatibel.
* Es gibt Versionen mit Langzeit Support. Dies sind derzeit die Versionen 8 und 11.
* Es kommen regelmäßig neue Versionen heraus mit neuen Features. Wenn man die neuesten Features nutzen möchte, dann kann eine aktuelle Version Sinn machen.

|  |  |
| --- | --- |
| Empfehlung | Um Probleme mit neuen Versionen zu vermeiden, macht es Sinn, frisch erschienene Versionen in den ersten Wochen noch nicht zu nutzen. So ersparen sich speziell Anfänger das lösen von Problemen, für die es ggf. noch keine Lösung gibt.  Eine LTS Version ist angeraten, da diese eine langen Nutzungszeitraum ohne große Updates bereitstellt.  Zum Zeitpunkt dieses Dokuments wäre meine Empfehlung für Anfänger das Java JDK 11. |

Die eigene Installation erfolgt so, wie es auf den Plattformen üblich ist:

* Windows: Die Java Installationsdatei ist als exe oder msi Datei herunter zu laden und kann dann mit einem Doppelklick geöffnet werden. Im Anschluss wird man durch ein paar wenige Installationsfenster geleitet und die Installation ist fertig. Bei der Installation ist in der Regel nicht spezielles zu beachten.
* Mac: Die Installation erfolgt in der Regel über einen Installer (pkg), der in einem Disk Image (DMG) enthalten sein kann. Das Paket kann (ggf. nach mounten der DMG Datei) geöffnet und so die Installation durchgeführt werden.
* Linux: Die Distributionen, die ich so kenne, haben alle ein OpenJDK innerhalb der Repositories. Somit ist die Installation mit den üblichen Tools der Distribution möglich.  
  Unter Ubuntu kann die Installation z.B. auf der Kommandozeile per  
  sudo apt-get install openjdk-11-jdk

## IDEs

### IntelliJ

IntelliJ ist eine IDE von JetBrains und ist in zwei Editionen verfügbar: Die frei verfügbare Community Edition bietet alle notwendigen Funktionen, die ein Entwickler benötigt. Speziell die Integration von Enterprise Funktionen ist der kostenpflichtigen Ultimate Version vorbehalten.

### Eclipse

Bei Eclipse handelt es sich um eine freie Entwicklungsumgebung.

### Netbeans

### SceneBuilder

## Gradle

Im Java Umfeld haben sich mehrere Build Tools etabliert. Derzeit sind die beiden führenden Tools Maven und Gradle. In dieser Dokumentation habe ich mich auf Gradle festgelegt. Dies ist eine rein persönliche Präferenz und soll keine Wertung sein.

### Installation / Gradle Wrapper

Eine Installation von Gradle ist in der Regel auf dem System nicht notwendig. In Projekten wird ein Gradle Wrapper eingesetzt. Dabei handelt es sich um ein kleines Tool, welches die korrekte Version von Gradle innerhalb des Projekts in dem Verzeichnis .gradle entpackt. Dies hat den Vorteil, dass man auf einem Rechner ohne Änderung der Konfiguration in den Projekten unterschiedliche Versionen von gradle nutzen kann.

Die Entwicklungsumgebungen können neue Gradle Projekte erzeugen können, die den gradle Wrapper beinhaltet. Ohne Entwicklungsumgebung kann man Gradle herunter laden und für die Erstellung eines Wrappers:

* Von <https://gradle.org/releases/> kann Gradle in einer ZIP Datei herunter geladen werden.
* Die ZIP Datei in einem beliebigen Verzeichnis entpacken.
* In dem entpackten bin Verzeichnis findet sich ein gradle Script, mit dem sich Gradle starten lässt. Um in einem Projektverzeichnis den Gradle Wrapper zu erstellen wechselt man in das Zielverzeichnis und startet gradle mit dem Parameter Wrapper, also z.B.:  
  c:\Apps\gradle-6.7\bin\gradle wrapper  
  /home/konrad/bin/gradle-6.7/bin/gradle wrapper

### Aufbau des Gradle Projekts

In einem Gradle Projekt finden sich mehrere Dateien

* build.gradle  
  Die build.gradle Datei enthält die eigentliche Projektbeschreibung.
* settings.gradle  
  In der settings.gradle Datei finden sich Konfigurationseinstellungen des Projektes
* gradlew / gradlew.bat  
  Start Scripte des gradle Wrappers.
* gradle  
  Verzeichnis mit Versionierten Dateien des Wrappers
* .gradle  
  Temporäres Verzeichnis mit Dateien, die der Gradle Wrapper / Gradle benötigt. Dieses Verzeichnis kann jederzeit gelöscht werden und der Gradle Wrapper bzw. Gradle laden die Dateien selbständig neu herunter.
* src  
  Verzeichnis für die Sourcen des Projektes. Die Java Dateien finden sich unter src/main/java, die Unit Test finden sich unter src/test/java.
* build  
  Die beim bauen des Projekts erzeugten Dateien finden sich innerhalb des build Verzeichnisses.

### Nutzung

* gradlew  
  Beim Aufruf von gradlew ohne Parameter lädt der Wrapper Gradle herunter so dies noch nicht erfolgt ist und es wird eine kleine Hilfe ausgegeben.
* gradlew tasks  
  Auflistung der verfügbaren Tasks. Da Tasks beliebig erstellt werden können, kann hier nicht eine immer gültige Auflistung gegeben werden.
* gradlew build  
  Das Projekt wird übersetzt.
* gradlew run  
  Das Projekt wird übersetzt und dann wird die Applikation gestartet
* gradlew jar  
  Es wird eine jar Datei gebaut mit dem übersetzten Projekt.

## Erste Applikation

Die erste Applikation, die wir hier kurz umschreiben, findet sich im Verzeichnis

01 helloworld

|  |
| --- |
| **HelloWorld.java** |
| package helloworld;  import javafx.application.Application;  import javafx.scene.Scene;  import javafx.scene.layout.StackPane;  import javafx.stage.Stage;  public class HelloWorld extends Application {  public static void main(String[] args) {  launch(args);  }  @Override  public void start(Stage primaryStage) {  primaryStage.setTitle("Hello World!");  StackPane root = new StackPane();  primaryStage.setScene(new Scene(root, 300, 250));  primaryStage.show();  }  } |

In der Applikation öffnen wir nur ein leeres Fenster mit Titel „Hello World!“ und zeigen dies an.

### Übersetzen und starten

Im Verzeichnis „01 helloworld“ rufen wir einfach auf

|  |
| --- |
| gradlew run |

Dadurch wird ggf. Gradle sowie alle notwendigen Abhängigkeiten (JavaFX) werden herunter geladen, der Source wird übersetzt und am Ende die Applikation gestartet.

# Scene Builder

# JavaFX Einführung

## Aufbau

## Layouts

# Model / View / Controller (MVC)

## Übersicht

## Beispiel Applikation

## Probleme

# Model / View / Viewmodel (MVVM)

# Applikation mit MVVM

1. Java Forum: https://www.java-forum.org/ [↑](#footnote-ref-1)
2. mvvmFX: https://github.com/sialcasa/mvvmFX/wiki [↑](#footnote-ref-2)