JavaFX

Print Version von Beiträgen 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 sind Fragen und Problemstellungen, denen ich immer wieder im [Java Forum](https://www.java-forum.org/)[[1]](#footnote-1) begegnet bin.

Kernpunkte 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 und diese Option möchte ich vorstellen.
* 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 (Am Ende mit verwerfen des MVC Patterns – daher kann der Teil mit dem MVC Pattern durchaus übersprungen werden).
* MVVM Pattern mit [mvvmFX](https://github.com/sialcasa/mvvmFX/wiki)[[2]](#footnote-2) - die Alternative für ein sauberes Design.

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++.

Inhalt

[Übersicht 2](#_Toc54269816)

[Code 2](#_Toc54269817)

[Fragen und Anregungen 2](#_Toc54269818)

[Autor 2](#_Toc54269819)

[Installation 4](#_Toc54269820)

[Java / JavaFX 4](#_Toc54269821)

[Tools / IDEs 5](#_Toc54269822)

[Git 5](#_Toc54269823)

[Maven / Gradle 5](#_Toc54269824)

[IntelliJ 6](#_Toc54269825)

[Eclipse / Netbeans 6](#_Toc54269826)

[SceneBuilder 6](#_Toc54269827)

[Erste Applikation 6](#_Toc54269828)

[Aufbau des Gradle Projekts (01 helloworld) 7](#_Toc54269829)

[Aufbau des Maven Projekts (01 helloworld – maven) 7](#_Toc54269830)

[Das Programm selbst 8](#_Toc54269831)

[Übersetzen und starten 8](#_Toc54269832)

[JavaFX Einführung 9](#_Toc54269833)

[Aufbau 9](#_Toc54269834)

[Layouts 9](#_Toc54269835)

[Model / View / Controller (MVC) 9](#_Toc54269836)

[Übersicht 9](#_Toc54269837)

[Beispiel Applikation 9](#_Toc54269838)

[Probleme 9](#_Toc54269839)

[Model / View / Viewmodel (MVVM) 9](#_Toc54269840)

[Applikation mit MVVM 9](#_Toc54269841)

# 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 zusätzlich zu dem JRE auch auch alle Tools, die ein Entwickler benötigt, z.B. den Java Compiler.

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

Die Java Software kann direkt von Oracle herunter geladen laden werden oder man nutzt ein OpenJDK von einem anderen Anbieter wie z.B. das AdoptOpenJDK oder Zulu Community von Azul.

|  |  |
| --- | --- |
| Hinweis | Oracle bietet das JRE nur bis Java 8 an. Wenn man ein aktuelles JRE benötigt, dann sollte entweder auf ein JDK zurück gegriffen werden oder man nutzt ein JRE von einem der anderen Anbietern. |

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

* Alles vor Java 8 ist veraltet und bekommt keine Updates mehr. Daher sind Sicherheitsprobleme 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 sich über Google noch keine Lösungen finden lassen.  Mit einer LTS Version hat man den Vorteil, dass man diese Version relativ lange nutzen kann und nur Sicherheitsupdates installieren muss.  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

## Tools / IDEs

### Git

Bei [Git](https://git-scm.com/downloads)[[3]](#footnote-3) handelt es sich um ein Tool zur verteilten Versionsverwaltung von Dateien. Dies bedeutet, dass Dateien Versioniert auf diversen Systemen abgelegt werden. Wir können also den genauen Verlauf mit allen Änderungen an den jeweiligen Dateien nachvollziehen. „Verteilt“ bedeutet, dass es nicht nur einen zentralen Server gibt, sondern es kann auf vielen Systemen das sogenannte Repository geben.

Als erstes installieren wir Git auf unserem System:

* Linux: Ein einfaches sudo apt-get install git installiert das Tool für uns.
* Windows: Wir laden uns den Installer vom oben verlinkten Ort und führen die Installation durch. Bei der Konfiguration können wir erst einmal die vorhandenen Defaults übernehmen.

Die genaue Nutzung von Git soll kein Bestandteil dieser JavaFX Blog Serie sein. Es gibt freie Dokumentation, die ich jedem interessierten Leser ans Herz legen kann. Ansonsten sollte an dieser Stelle ausreichen, dass der befehl, mit dem der Code der JavaFX Serie auf den eigenen Rechner kopiert werden kann, am Anfang des Dokuments erwähnt ist.

Git Dokumentation:

* <https://git-scm.com/book/en/v2> Pro GIT Buch (Englisch)
* <https://git-scm.com/book/de/v2> Pro GIT Buch (Deutsch)

### Maven / Gradle

Im Java Umfeld haben sich mehrere Build Tools etabliert. Derzeit sind die beiden führenden Tools Maven und Gradle. Maven und Gradle benötigen keine lokale Installation, wenn Projekte mit jeweiligem Wrapper oder einer IDE verwendet wird.

Der Wrapper ist ein kleines Tool + Scripte, die die jeweilige Version von Gradle/Maven innerhalb des Projekts herunter laden und von dort nutzen. Dies hat den Vorteil, dass jedes Projekt genau die Version des Build Tools nutzt, die vom Projekt vorgesehen ist.

Dies ist wichtig, da es von Version zu Version zu Inkompatibilitäten kommen kann, so dass es zu Problemen kommen kann, wenn eine zu neue oder zu alte Version in einem Projekt verwendet wird.

So Wrapper-Projekte von Hand angelegt werden sollen, so reicht es aus, [Gradle](https://gradle.org/releases/)[[4]](#footnote-4) / [Maven](https://maven.apache.org/download.cgi)[[5]](#footnote-5) in der gewünschten Version herunter zu laden und in einem beliebigen Verzeichnis zu entpacken. Innerhalb der Projektverzeichnisses kann dann der Wrapper erzeugt werden mit (einmal gradle und einmal maven Aufruf – der jeweils gewünschte Aufruf muss gewählt werden, je nachdem ob maven oder gradle verwendet werden soll):

c:\apps\gradle-6.5\bin\gradle wrapper  
c:\apps\apache-maven-3.6.3\bin\mvn -N io.takari:maven:wrapper

Der Aufruf ist jetzt der Windows Aufruf mit dem entpacken der Tools in c:\Apps. Der Aufruf unter Linux kann ähnlich aussehen:  
~/bin/gradle-6.5/bin/gradle wrapper  
~/bin/maven-3.6.3/bin/mvn -N io.takari:maven:wrapper

Mit entpacken der Tools innerhalb eines bin Verzeichnisses im Home Verzeichnis des Users.

### IntelliJ

[IntelliJ](https://www.jetbrains.com/idea/)[[6]](#footnote-6) 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.

Die Installation kann entweder durch einen manuellen Download mit darauffolgender Installation erfolgen oder man installiert die [Toolbox](https://www.jetbrains.com/toolbox-app/)[[7]](#footnote-7) um die Installation / Updates komfortabel durch dieses Tool durchführen zu lassen.

### Eclipse / Netbeans

Neben IntelliJ sind auch [Eclipse](https://www.eclipse.org/)[[8]](#footnote-8) und [Netbeans](https://netbeans.org/)[[9]](#footnote-9) recht weit verbreitet. Aus Zeitgründen gehe ich aber im Video nur auf IntelliJ ein. Ich wollte diese Entwicklungsumgebungen aber nicht unerwähnt lassen, da es sich bei diesen um komplett freie Software handelt.

### SceneBuilder

Bei SceneBuilder[[10]](#footnote-10) handelt es sich um einen visuellen Editor für JavaFX Oberflächen. Diesen werden wir im ersten Video noch nicht nutzen, aber in späteren Videos zu JavaFX wird dieser teilweise eine Rolle spielen. Daher zeige ich die Installation und Einbindung in IntelliJ.

## Erste Applikation

Die erste Applikation, die wir hier kurz umschreiben, findet sich in den Verzeichnissen

01 helloworld

01 helloworld - maven

Im ersten Verzeichnis findet sich die Version mit Gradle, im zweiten mit Maven

### Aufbau des Gradle Projekts (01 helloworld)

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.

#### Gradle Aufrufe

* 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.

### Aufbau des Maven Projekts (01 helloworld – maven)

In dem Maven Projekt finden sich mehrere Dateien

* pom.xml  
  Diese Datei enthält die genaue Beschreibung des Projekts.
* mvnw / mvnw.cmd  
  Start-Scripte mit denen der Wrapper gestartet wird.
* .mvn  
  Verzeichnis mit Maven Dateien. Hier findet sich u.a. die Dateien vom Wrapper.
* 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.
* target  
  Hier finden sich alle Dateien, die von Maven erzeugt werden.

#### Maven Aufrufe

* mvnw package  
  Übersetzt alle Sourcen, lässt Tests laufen und baut das Ziel
* mvnw javafx:run  
  Baut das Projekt und führt die Anwendung aus.

Dies ist nur ein kleiner Ausschnitt der möglichen Aufrufe, aber zur Nachverfolgung der Blog Einträge ist erst einmal nicht mehr notwendig. In späteren Beispielen werde ich in erster Linie Gradle nutzen und nicht weiter auf Maven zurück greifen.

### Das Programm selbst

Das eigentliche Java Programm ist unabhängig von dem verwendeten Build Tool. So findet sich in beiden Projekten unter src/main/java/helloworld die Datei HelloWorld.java:

|  |
| --- |
| **src/main/java/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();  }  } |

Dieser Code macht nicht viel:  
Beim Start der Applikation wird die main Methode aufgerufen. Diese ruft – so wie eigentlich in jedem javaFX Programm, die launch Methode aus der Application Klasse auf.

Es werden diverse Dinge von Application gemacht, auf die ich an dieser Stelle noch nicht näher eingehen werde. Lediglich die start Methode müssen wir beleuchten: Diese Methode wird nach der Initialisierung aufgerufen und bekommt als Parameter die primary Stage – sozusagen das Hauptfenster – übergeben.

Wir setzen den Titel des Fensters und erzeugen einen kleinen Fensterinhalt – eine sogenannte Scene, die wir dem Fenster hinzufügen.

Am Ende machen wir das Fenster noch sichtbar.

### 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.

Hier erkennen wir die große Stärke von gradle: Wir konnten javaFX nutzen, ohne dass wir manuell irgendwas installiert und eingerichtet haben. Das Build Tool hat sich komplett um diese Abhängigkeit gekümmert.

# 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)
3. Git Download: https://git-scm.com/downloads [↑](#footnote-ref-3)
4. Gradle Download: https://gradle.org/releases/ [↑](#footnote-ref-4)
5. Maven Download: https://maven.apache.org/download.cgi [↑](#footnote-ref-5)
6. IntelliJ: https://www.jetbrains.com/idea/ [↑](#footnote-ref-6)
7. Jetbrains Toolbox: https://www.jetbrains.com/toolbox-app/ [↑](#footnote-ref-7)
8. Eclipse: https://www.eclipse.org/ [↑](#footnote-ref-8)
9. Netbeans: https://netbeans.org/ [↑](#footnote-ref-9)
10. SceneBuilder: https://gluonhq.com/products/scene-builder/ [↑](#footnote-ref-10)