Einführung

TWS

SEW3

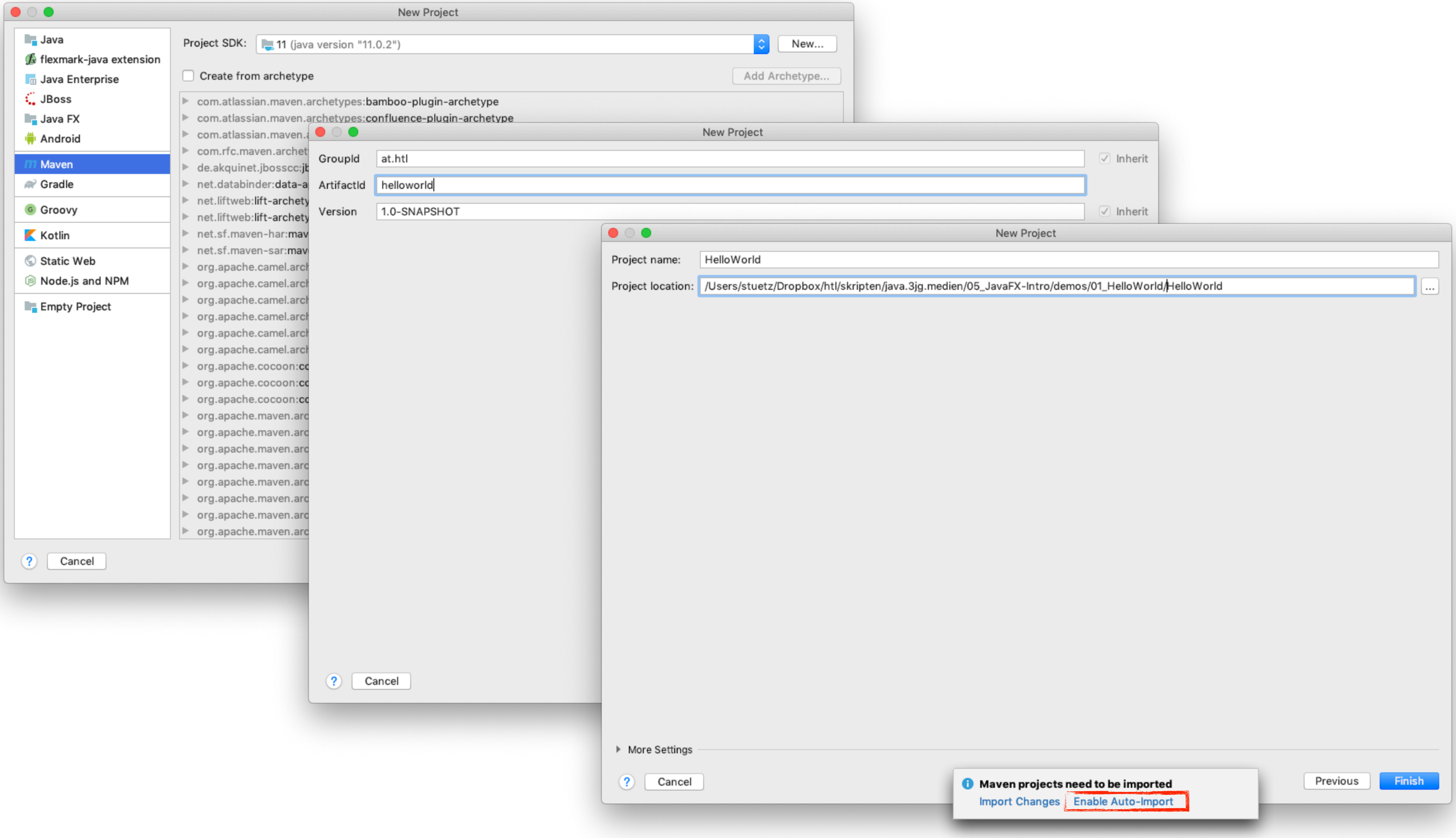
IT-Medientechnik

JavaFX: Eigenschaften

- Ab Java 7 Standard UI-Framework
  - Ersetzt Vorgängerversionen AWT & Swing
- Definition des UI
  - mittels Code
  - oder Markup-Language (FXML) möglich
- Scene Builder für UI-Design
  - Layoutgestaltung mit Cascading Style Sheets (css)
  - Umfangreiche 2D-Funktionalitäten (Schatten, Transformationen, ….), auch 3D-Features
- Im März 2018 kündigte Oracle an, JavaFX als openJFX im Rahmen eines eigenständigen Projektes von der Community weiterführen zu lassen. Es ist derzeit nicht mehr in JavaSE enthalten und muß als Dependency ins Projekt eingefügt werden.

Manuelle Erstellung eines Projekts

ohne maven


New Project
Java
flexmark-java extension
Java Enterprise
JBoss
Java FX
Android
Maven
Gradle
Groovy
Kotlin
Static Web
Node.js and NPM
Empty Project
Project SDK:
11 (java version "11.0.2")
New...
Create from archetype
Add Archetype...
com.atlassian.maven.archetypes:bamboo-plugin-archetype
com.atlassian.maven.archetypes:confluence-plugin-archetype
Cancel
New Project
GroupId
at.htl
Inherit
ArtifactId
helloworld
Version
1.0-SNAPSHOT
Inherit
Cancel
New Project
Project name:
HelloWorld
Project location:
/Users/stuetz/Dropbox/htl/skripten/java.3jg.medien/05_JavaFX-Intro/demos/01_HelloWorld/HelloWorld
More Settings
Cancel
Maven projects need to be imported
Import Changes
Enable Auto-Import
Previous
Finish

pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>at.htl</groupId>
    <artifactId>helloworld</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.openjfx</groupId>
            <artifactId>javafx-controls</artifactId>
            <version>12.0.2</version>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.openjfx</groupId>
                <artifactId>javafx-maven-plugin</artifactId>
                <version>0.0.3</version>
                <configuration>
                    <mainClass>at.htl.helloworld.HelloFX</mainClass>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

UTF-8

javafx-dependency

javafx-maven-plugin

muss eingetragen werden

Beachte: Die Packages der Klasse sind anzugeben

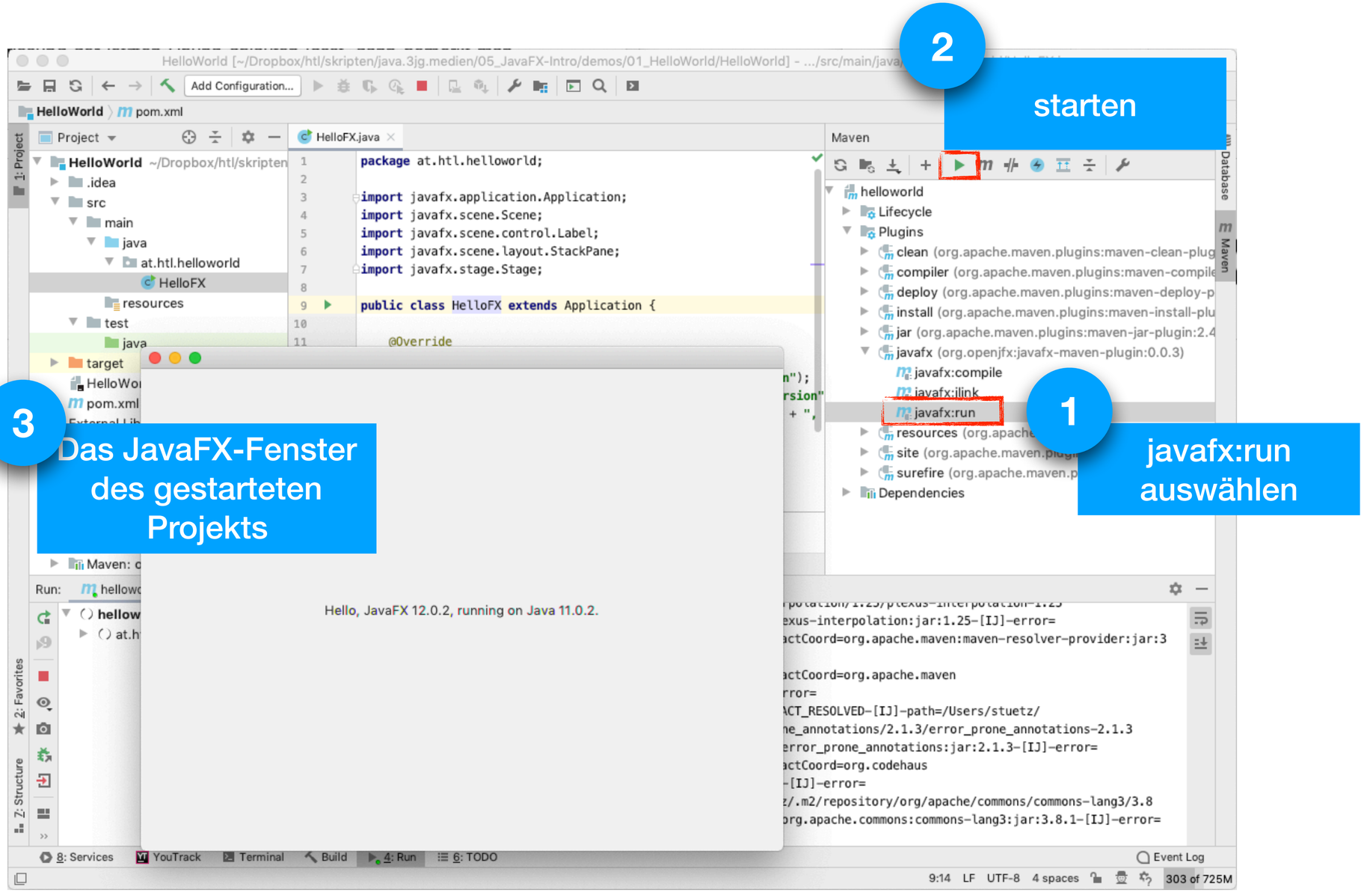
Im configuration-Tag könnte man auch die Java-Version einstellen. Default ist 11.

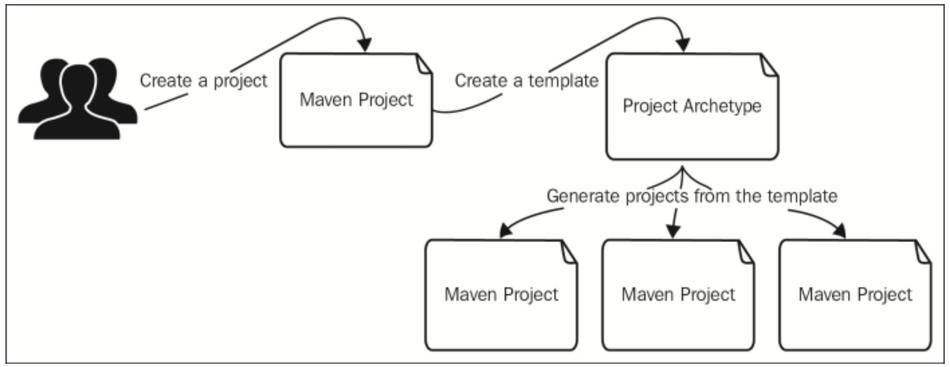
https://github.com/openjfx/samples/blob/master/HelloFX/Maven/hellofx/pom.xml

HelloFX.java

```java
package at.htl.helloworld;

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

public class HelloFX extends Application {

    @Override
    public void start(Stage stage) {
        String javaVersion = System.getProperty("java.version");
        String javafxVersion = System.getProperty("javafx.version");
        Label l = new Label("Hello, JavaFX " + javafxVersion + ", running on Java " + javaVersion + ".");
        Scene scene = new Scene(new StackPane(l), 640, 480);
        stage.setScene(scene);
        stage.show();
    }

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

}
```

https://raw.githubusercontent.com/openjfx/samples/master/HelloFX/Maven/hellofx/src/main/java/HelloFX.java


2
starten
1
javafx:run auswählen
3
Das JavaFX-Fenster des gestarteten Projekts
Hello, JavaFX 12.0.2, running on Java 11.0.2.



empfohlene Vorgehensweise

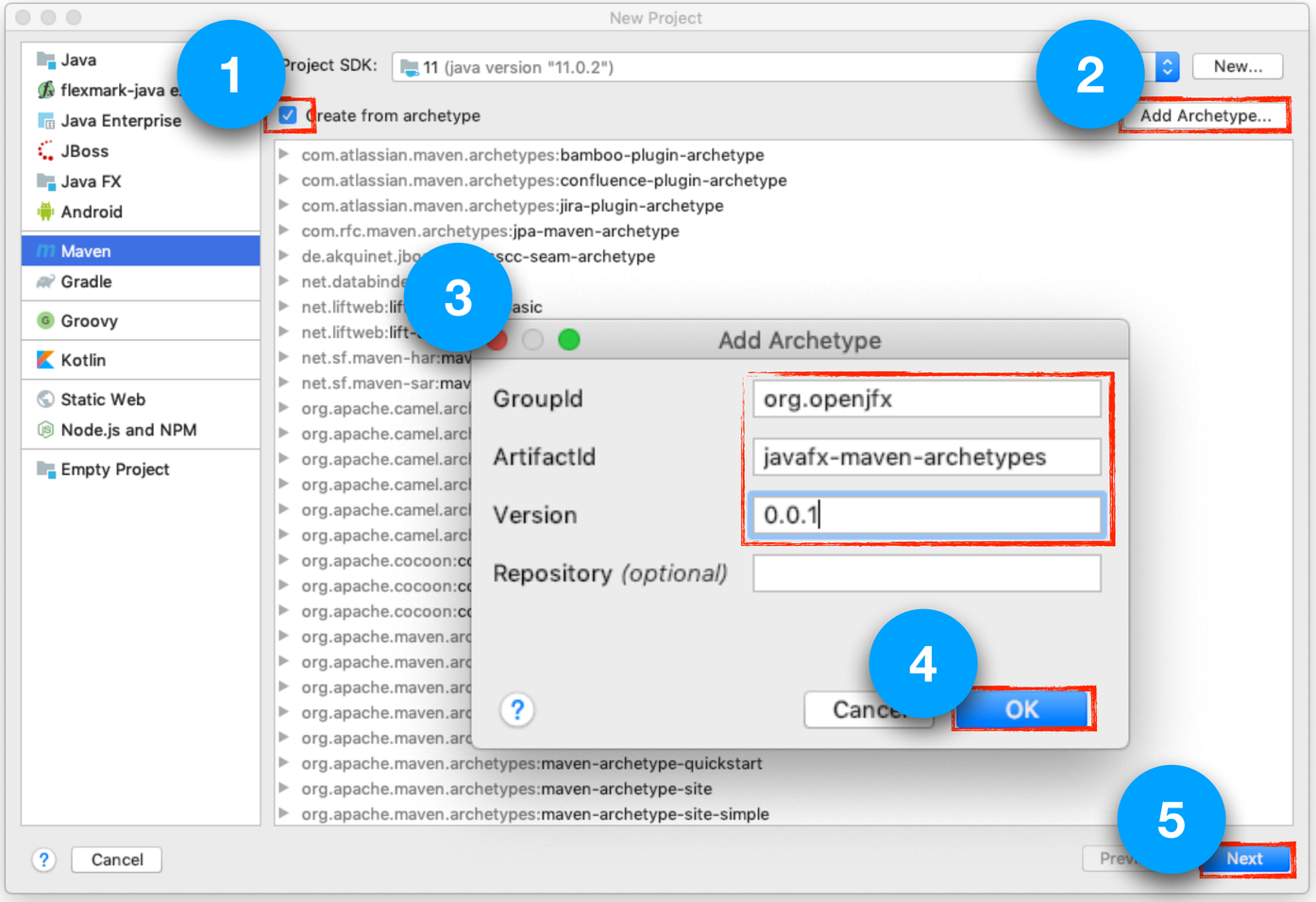
Erstellung eines Projekts mit einem Archetype (Projektvorlage)

https://openjfx.io/openjfx-docs/#maven

https://youtu.be/r_tdK8zWr_w

https://youtu.be/Ri6No63fl-A

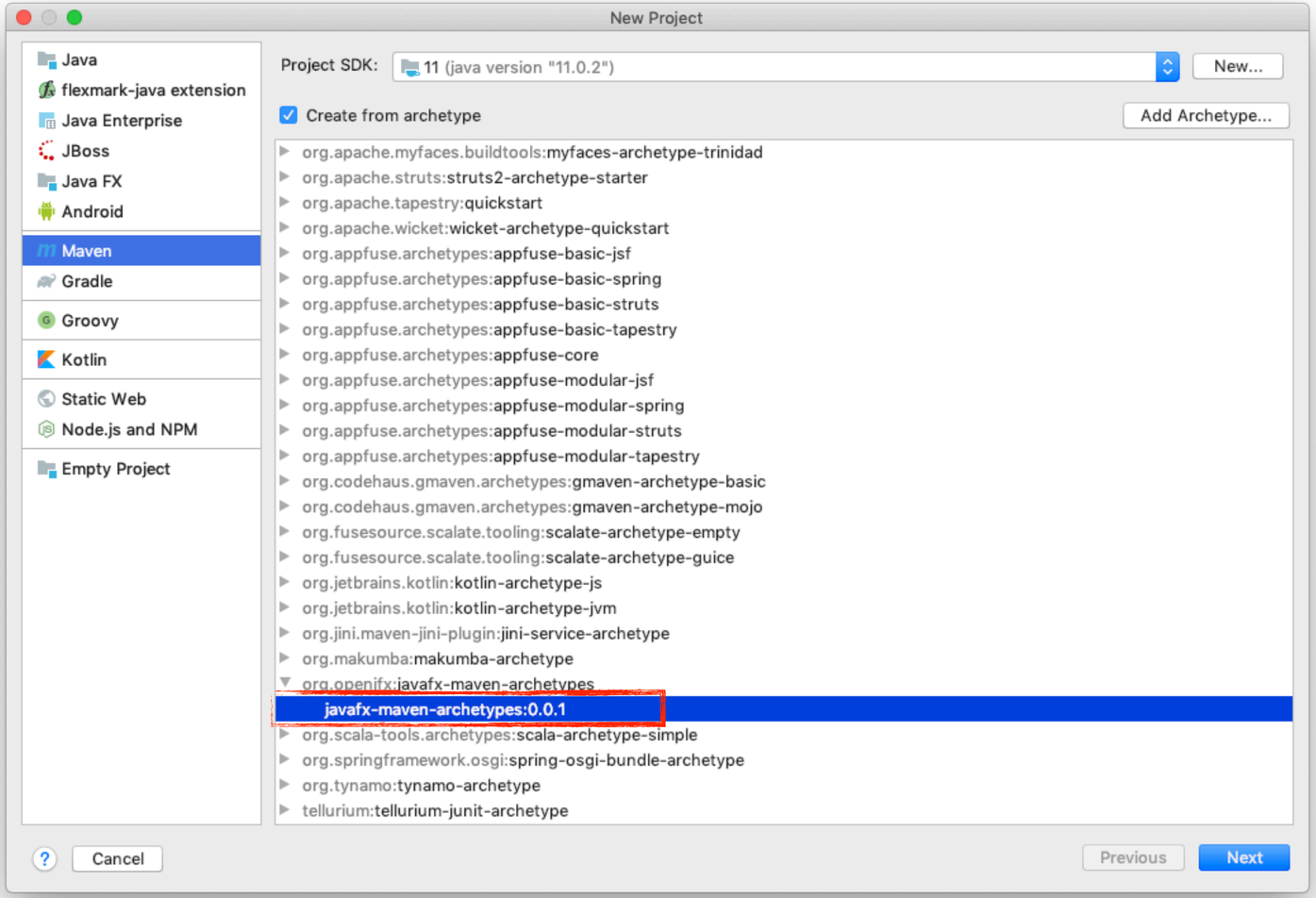
Download des Archetypes in das lokale Repo



Das lokale maven-Repository befindet sich im User-Home im Ordner .m2

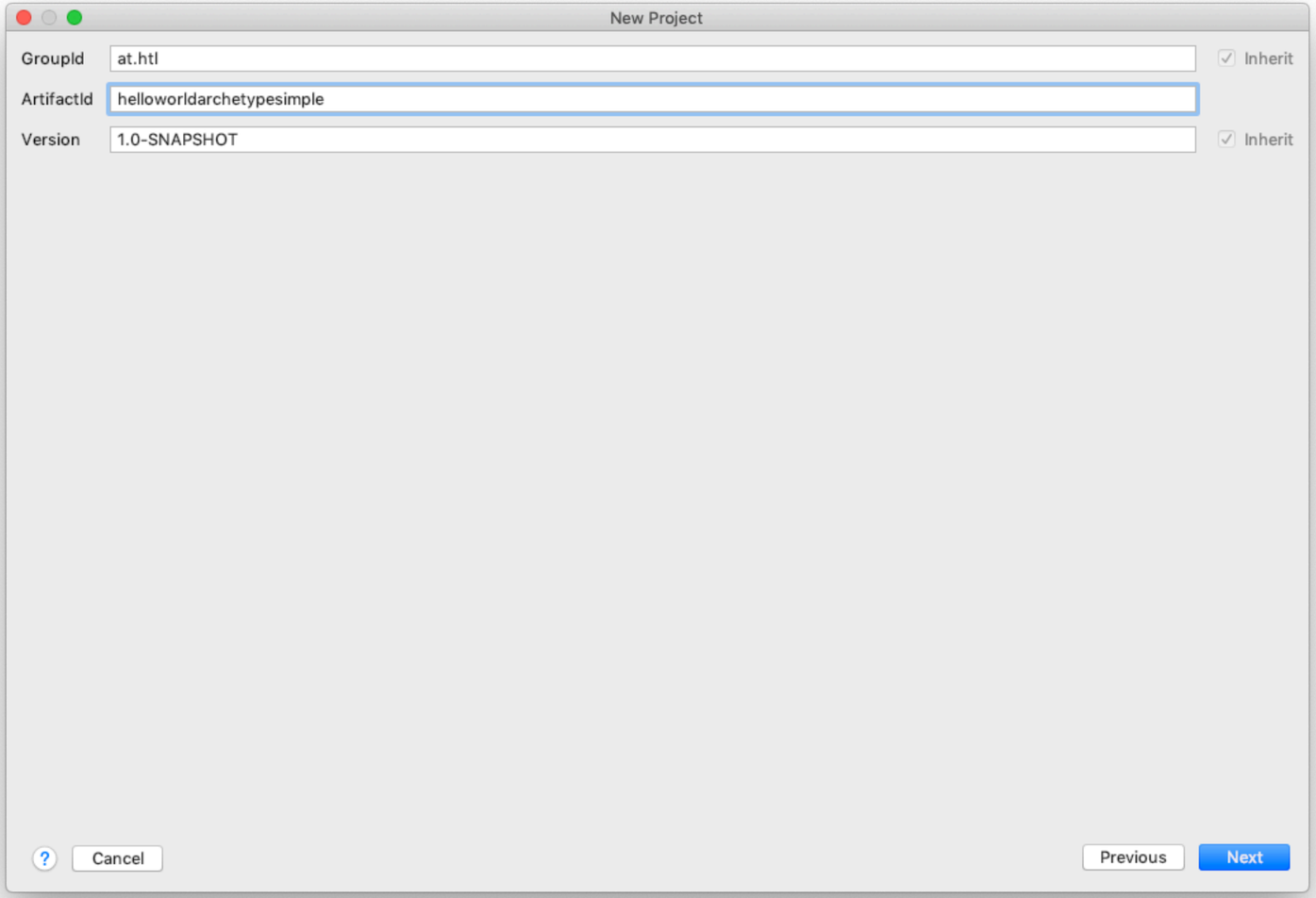
https://openjfx.io/openjfx-docs/#maven

Archetype auswählen

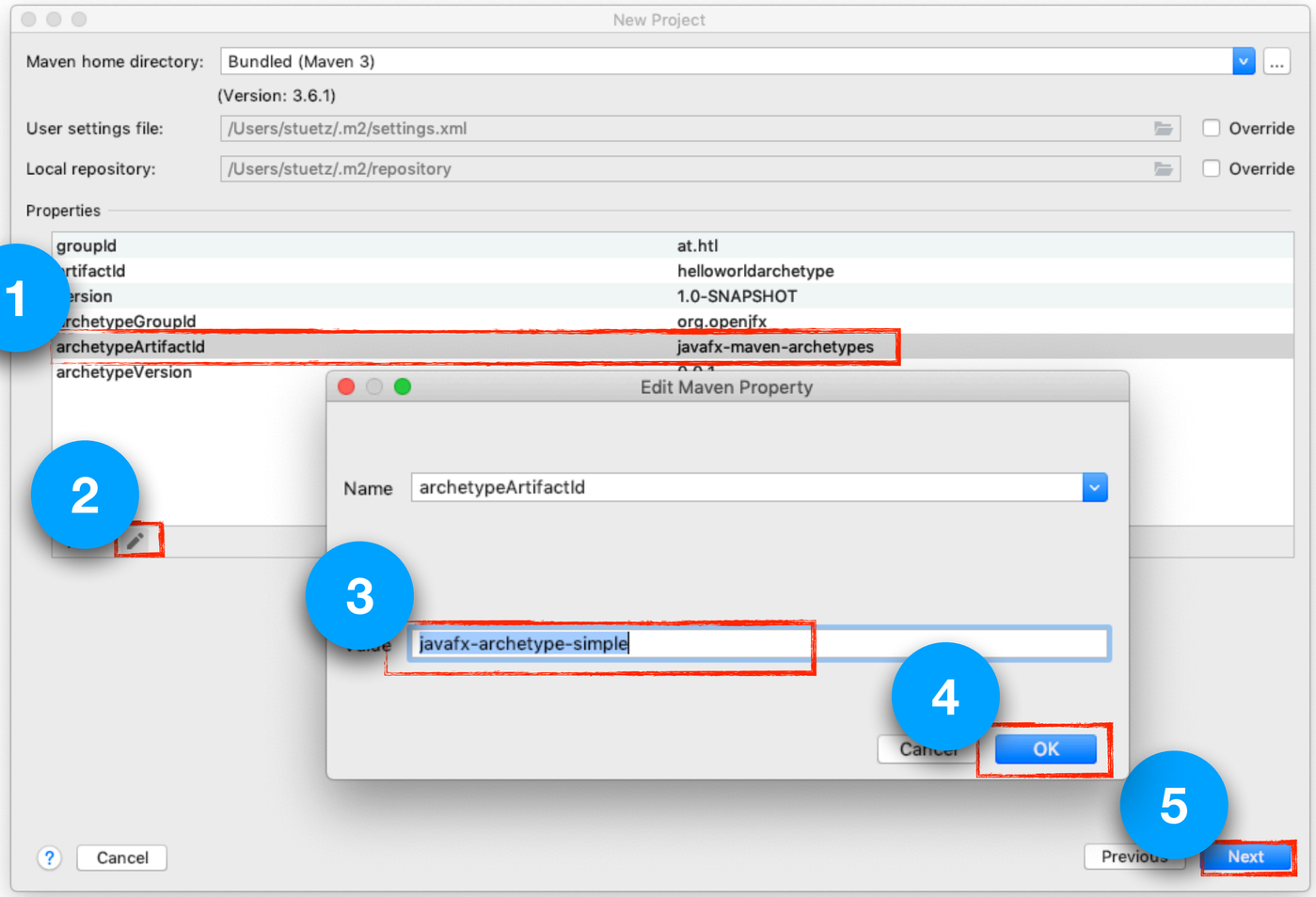


Erstellung eines JavaFX-Projekts ohne FXML (simple)

GroupId, ArtifactId

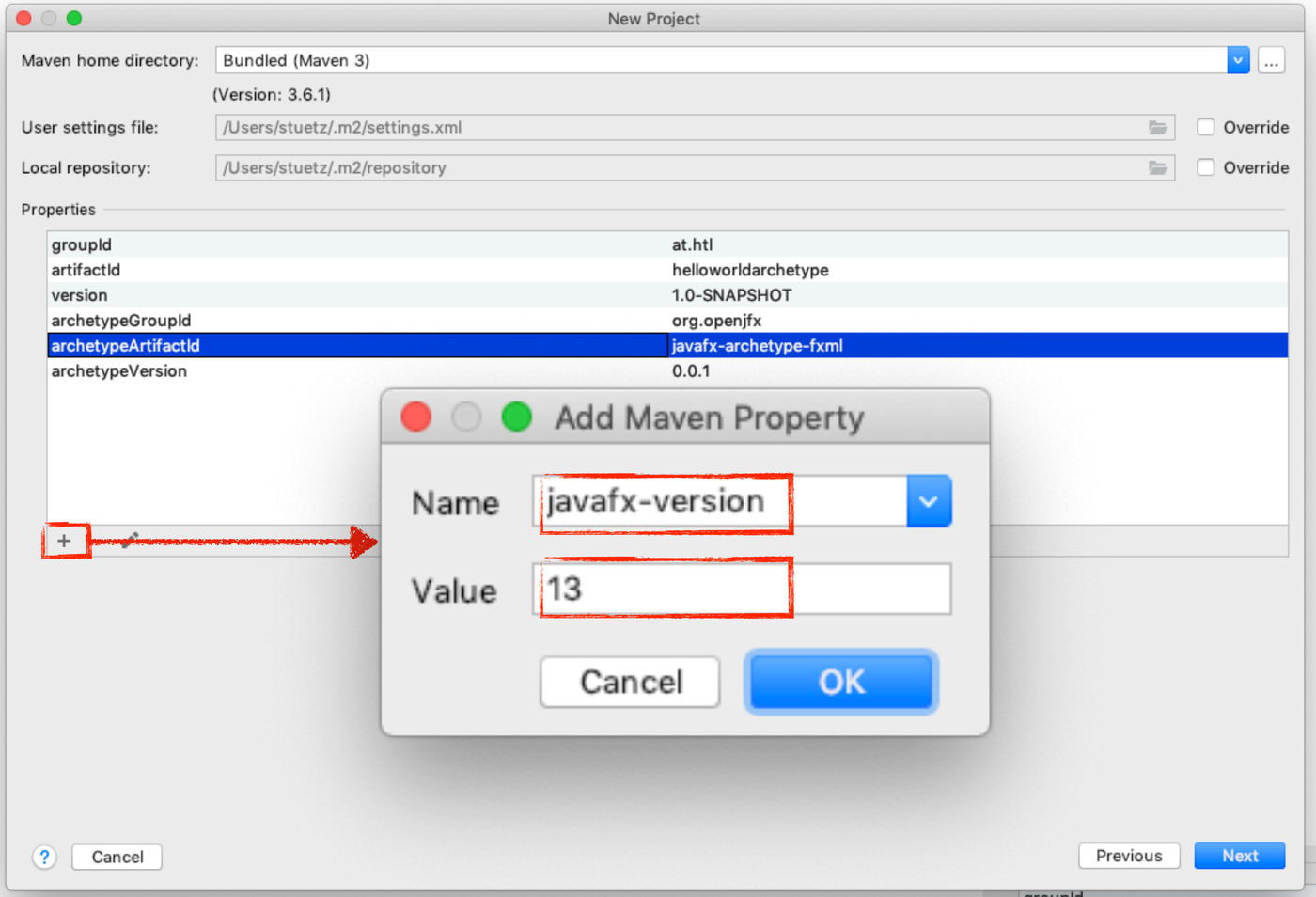


ArchetypeArtifactId eintragen

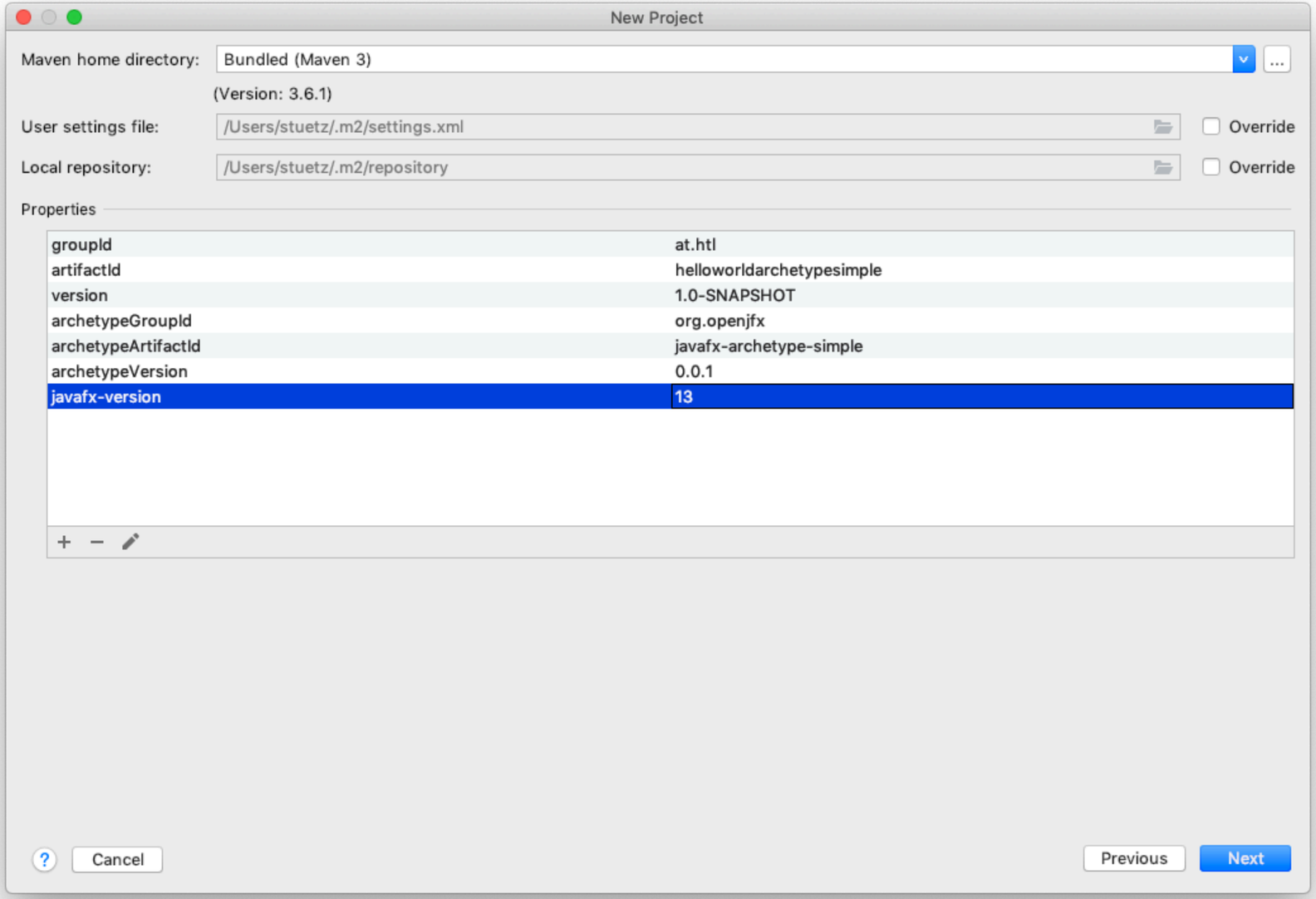


1. archetypeArtifactId auswählen
2. Edit anklicken
3. Entweder **„javafx-archetype-fxml“** eingeben oder falls fxml nicht gebraucht wird **„javafx-archetype-simple“**
4. OK
5. Next

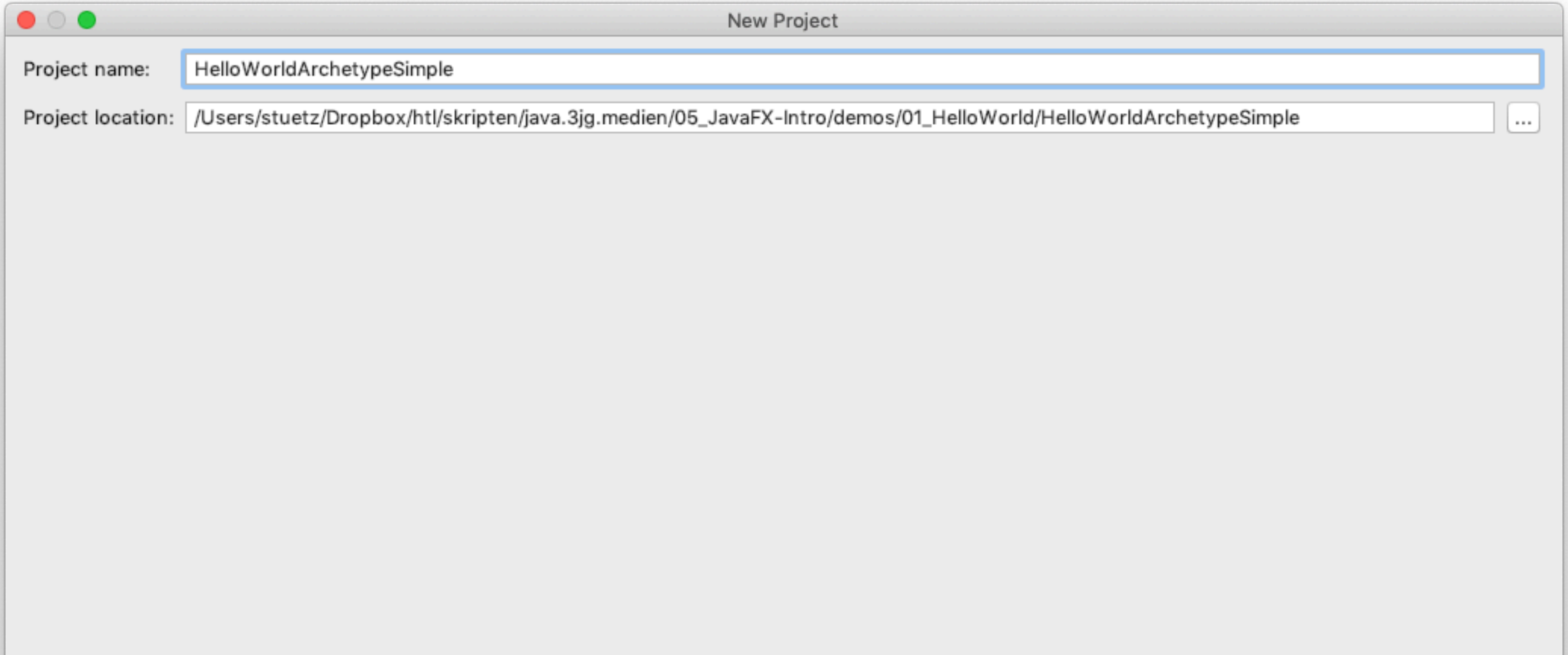
Einstellen der JavaFX-Version



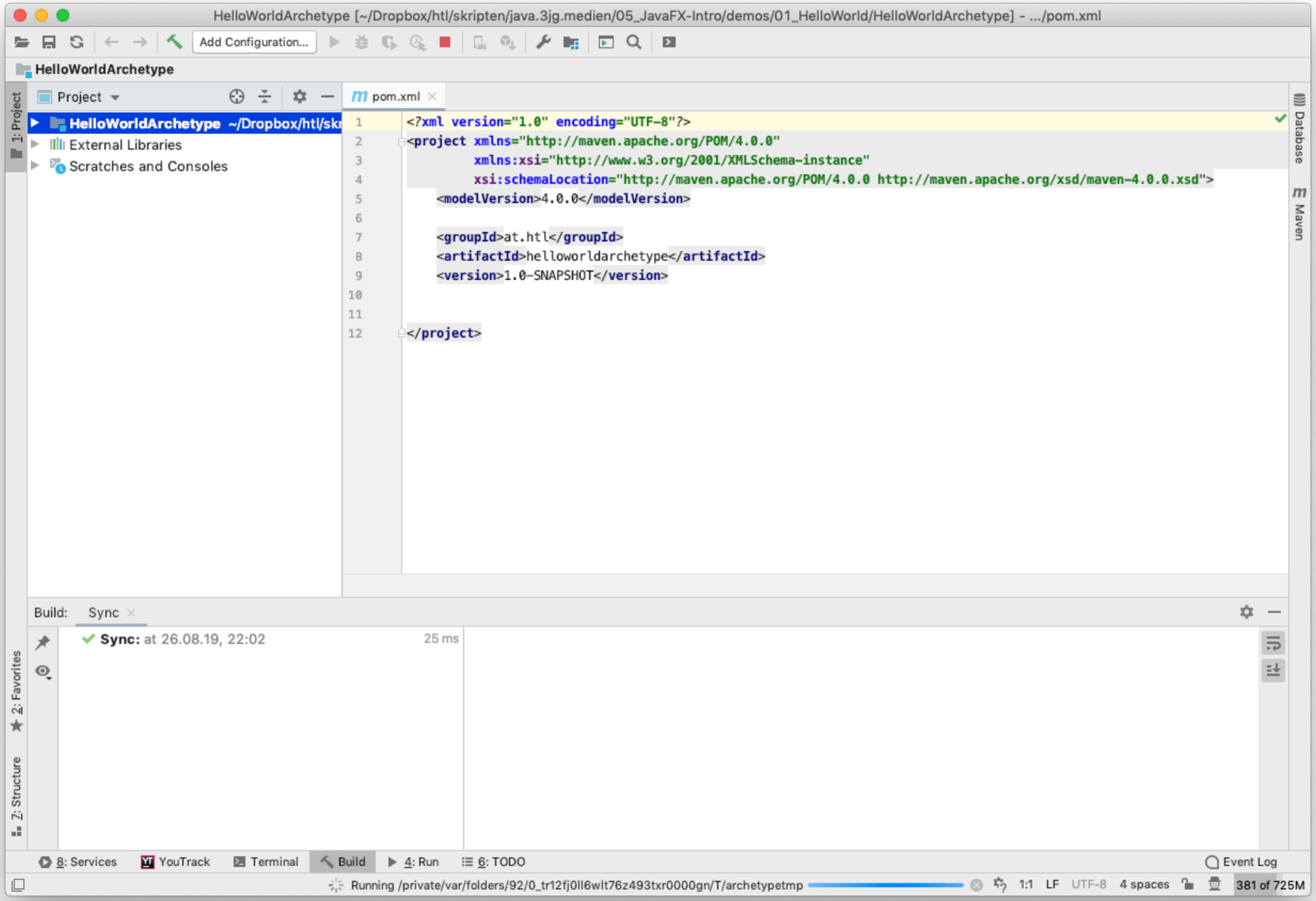
Archetype konfiguriert

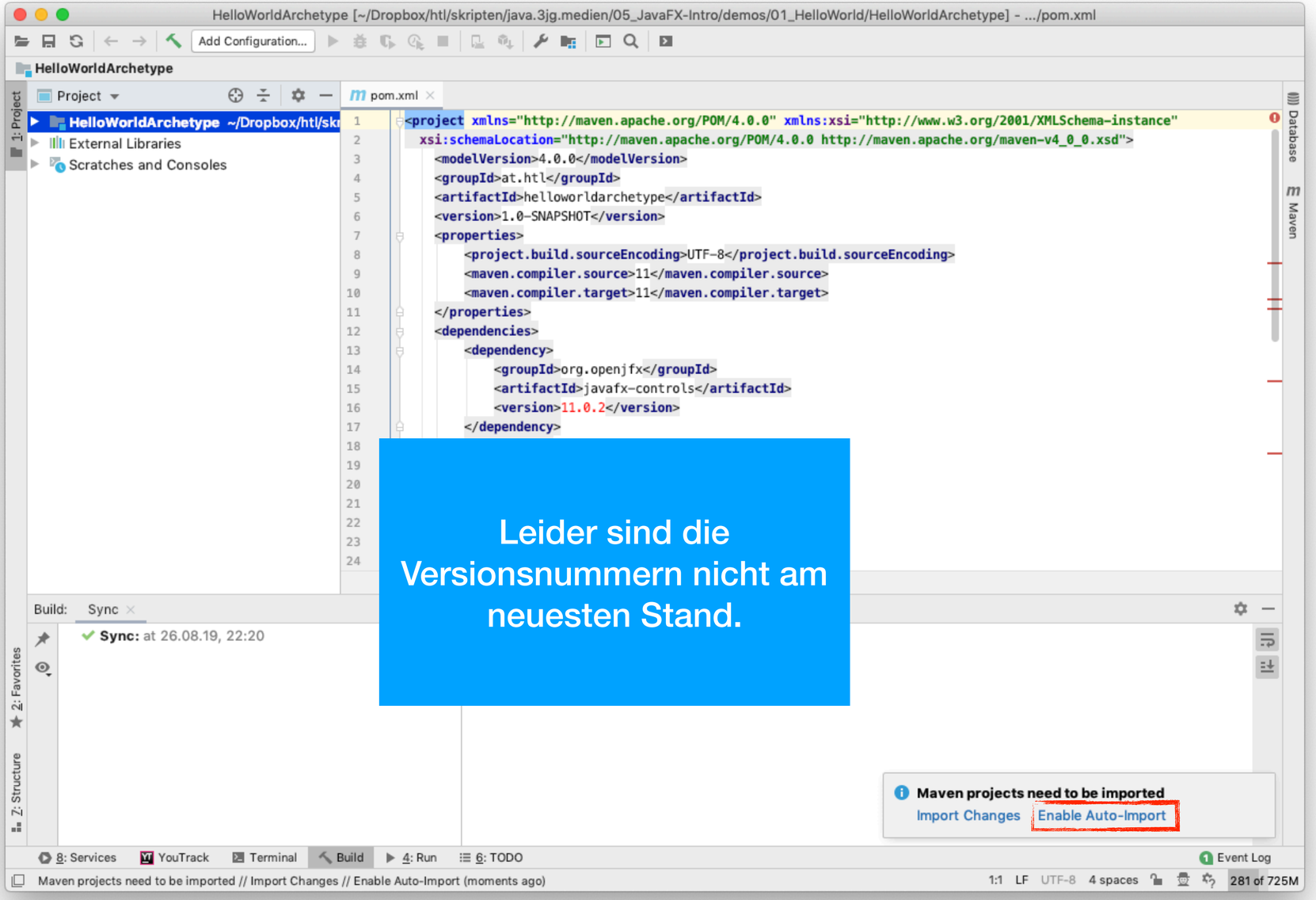


Projektname und Speicherort festlegen

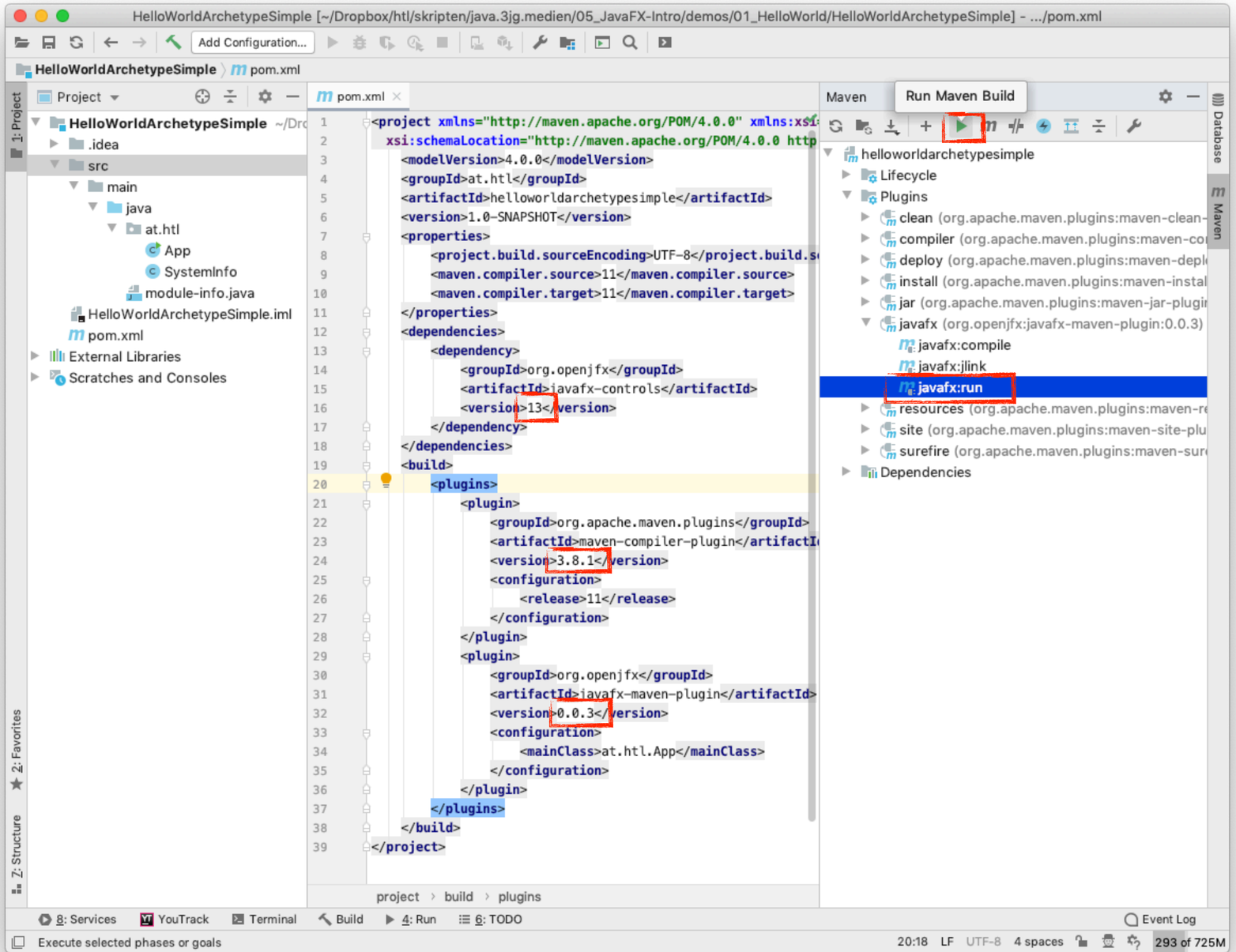


Der Download (Sync) wird durchgeführt

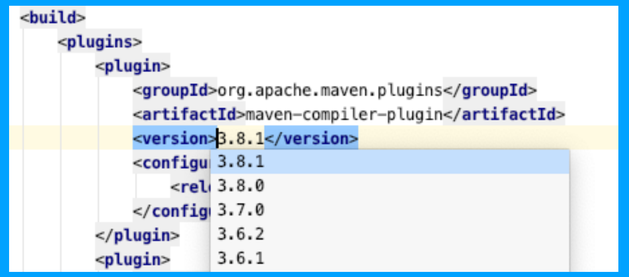



HelloWorldArchetype [~/Dropbox/htl/skripten/java.3jg.medien/05_JavaFX-Intro/demos/01_HelloWorld/HelloWorldArchetype] - .../pom.xml
Add Configuration...
HelloWorldArchetype
Project
HelloWorldArchetype ~/Dropbox/htl/sk
External Libraries
Scratches and Consoles
pom.xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>at.htl</groupId>
<artifactId>helloworldarchetype</artifactId>
<version>1.0-SNAPSHOT</version>
<properties>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
<maven.compiler.source>11</maven.compiler.source>
<maven.compiler.target>11</maven.compiler.target>
</properties>
<dependencies>
<dependency>
<groupId>org.openjfx</groupId>
<artifactId>javafx-controls</artifactId>
<version>11.0.2</version>
</dependency>
Leider sind die Versionsnummern nicht am neuesten Stand.
Build: Sync
Sync: at 26.08.19, 22:20
Maven projects need to be imported
Import Changes
Enable Auto-Import
8: Services
YouTrack
Terminal
Build
4: Run
6: TODO
Event Log
Maven projects need to be imported // Import Changes // Enable Auto-Import (moments ago)
1:1
LF
UTF-8
4 spaces
281 of 725M
1: Project
2: Favorites
7: Structure
Database
Maven

Versionsnummern aktualisieren



mit ^-<Space> kann man sich die Versionen ansehen …



… oder: um die neuesten Versionsnummern zu ersehen, einfach im mvnrepository nachsehen (siehe nächste Seite)

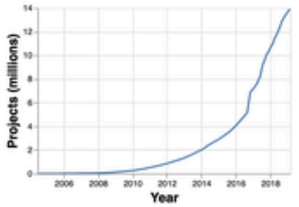
https://mvnrepository.com

MVNREPOSITORY

Search for groups, artifacts, categories

Search

Indexed Artifacts (15.1M)



Popular Categories

- Aspect Oriented
- Actor Frameworks
- Application Metrics
- Build Tools
- Bytecode Libraries
- Command Line Parsers
- Cache Implementations
- Cloud Computing
- Code Analyzers
- Collections
- Configuration Libraries
- Core Utilities
- Date and Time Utilities
- Dependency Injection
- Embedded SQL Databases

Home » org.apache.maven.plugins » maven-compiler-plugin

Apache Maven Compiler Plugin

The Compiler Plugin is used to compile the sources of your project.

License	Apache 2.0
Categories	Maven Plugins
Tags	plugin compiler build build-system maven apache
Used By	69 artifacts

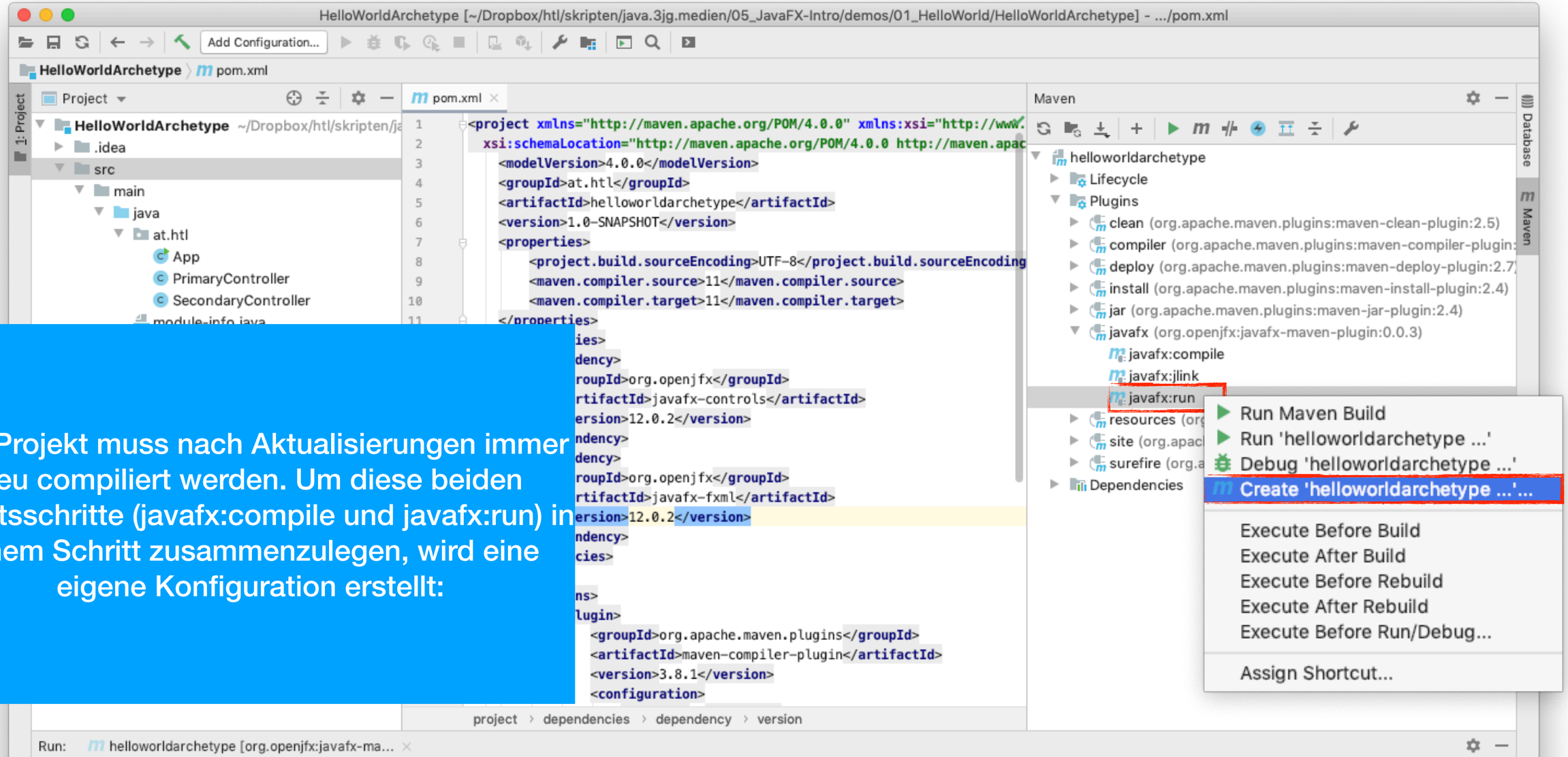
Im mvnrepository kann man sich alle verfügbaren Versionen ansehen

Central (24) | Spring Plugins (1) | Redhat GA (1) | Redhat EA (3) | JBoss 3rd-party (3) | JBoss Public (1) | ICM (7)

	Version	Repository	Usages	Date
3.8.x	3.8.1	Central	2	May, 2019
	3.8.0	Central	5	Jul, 2018
3.7.x	3.7.0	Central	11	Sep, 2017
3.6.x	3.6.2	Central	3	Jul, 2017
	3.6.1	Central	7	Jan, 2017
	3.6.0	Central	1	Oct, 2016
3.5.x	3.5.1	Central	9	Feb, 2016
	3.5	Central	0	Jan, 2016

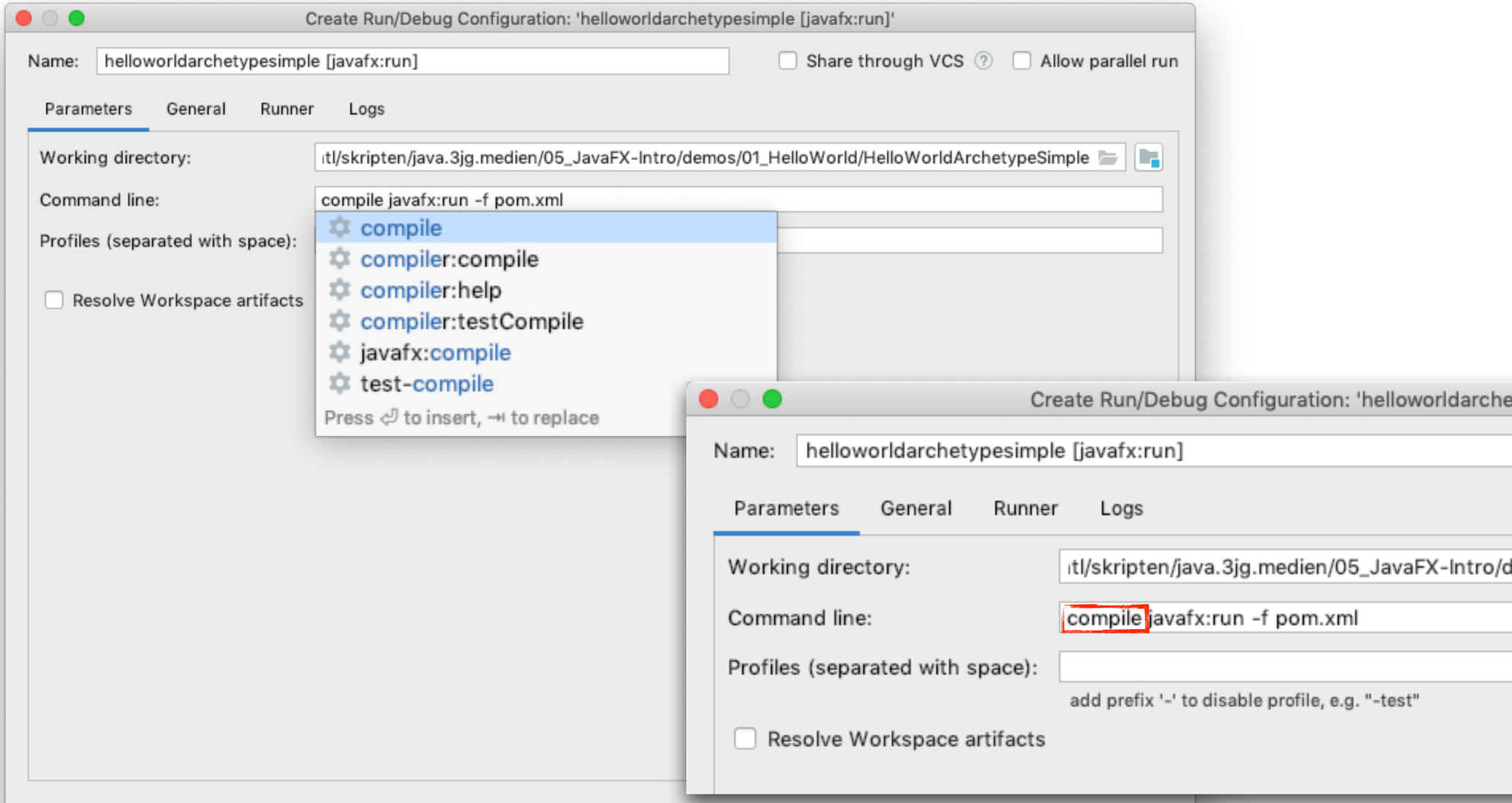
Ergebnis

Eigene Configuration erstellen

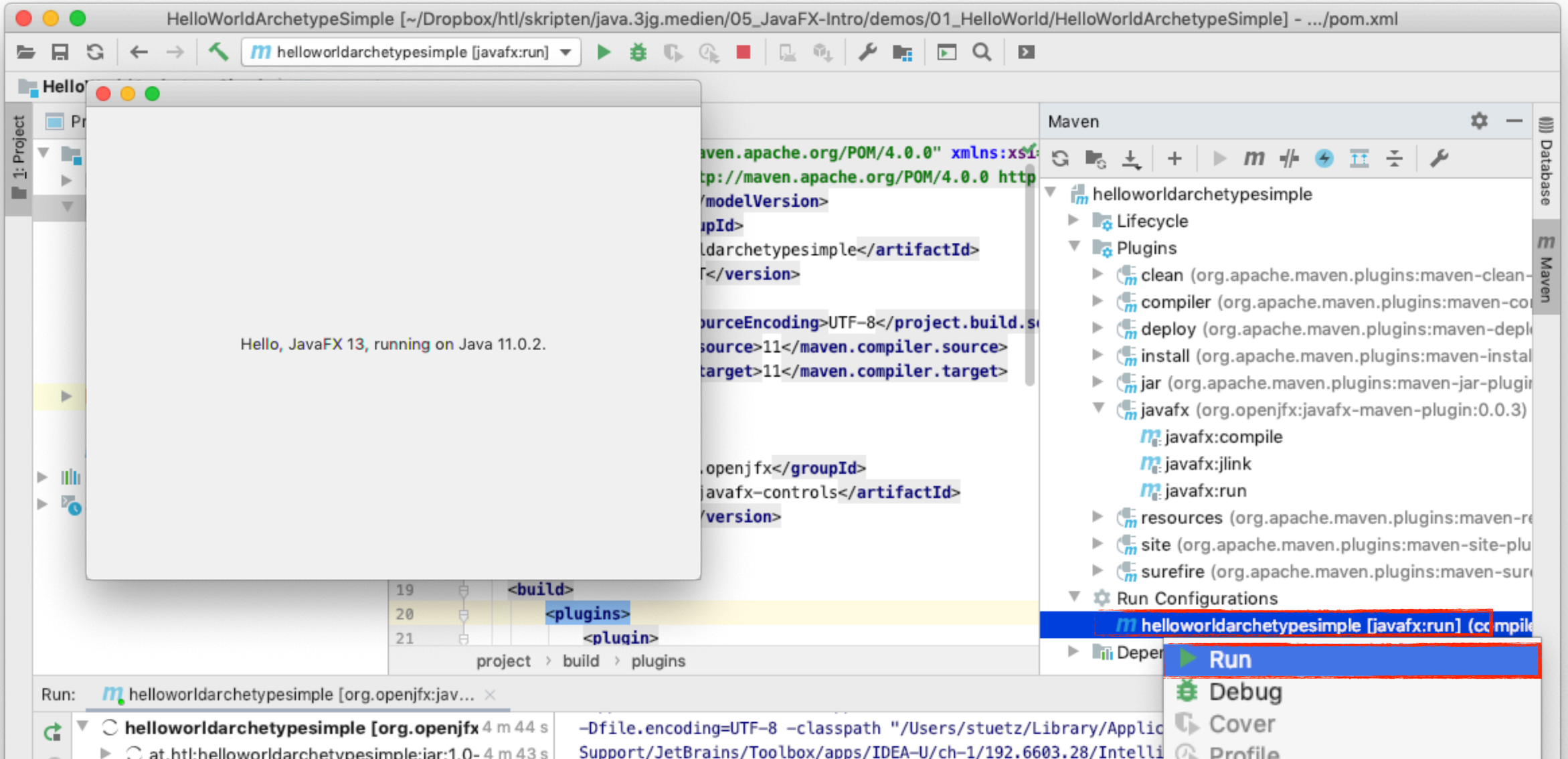


Das Projekt muss nach Aktualisierungen immer neu compiliert werden. Um diese beiden Arbeitsschritte (javafx:compile und javafx:run) in einem Schritt zusammenzulegen, wird eine eigene Konfiguration erstellt:

Compile eintragen



Ausführen: compile + run



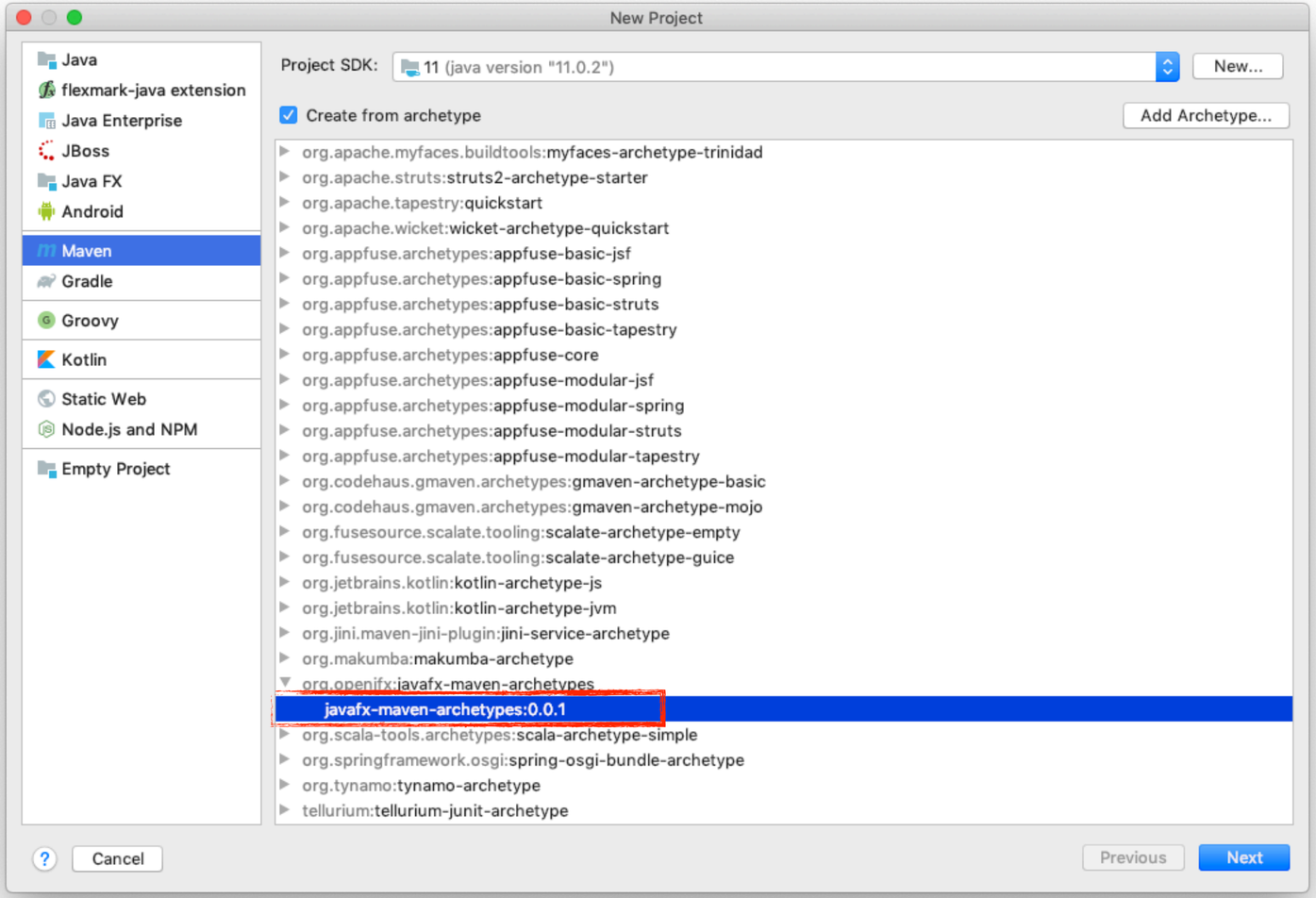
Code der FX-Anwendung

```java
package at.htl;

import ...

/**
 * JavaFX App
 */
public class App extends Application {

    @Override
    public void start(Stage stage) {
        var javaVersion = SystemInfo.javaVersion();
        var javafxVersion = SystemInfo.javafxVersion();

        var label = new Label("Hello, JavaFX "
                + javafxVersion
                + ", running on Java "
                + javaVersion + ".");
        var scene = new Scene(new StackPane(label), 640, 480);
        stage.setScene(scene);
        stage.show();
    }

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

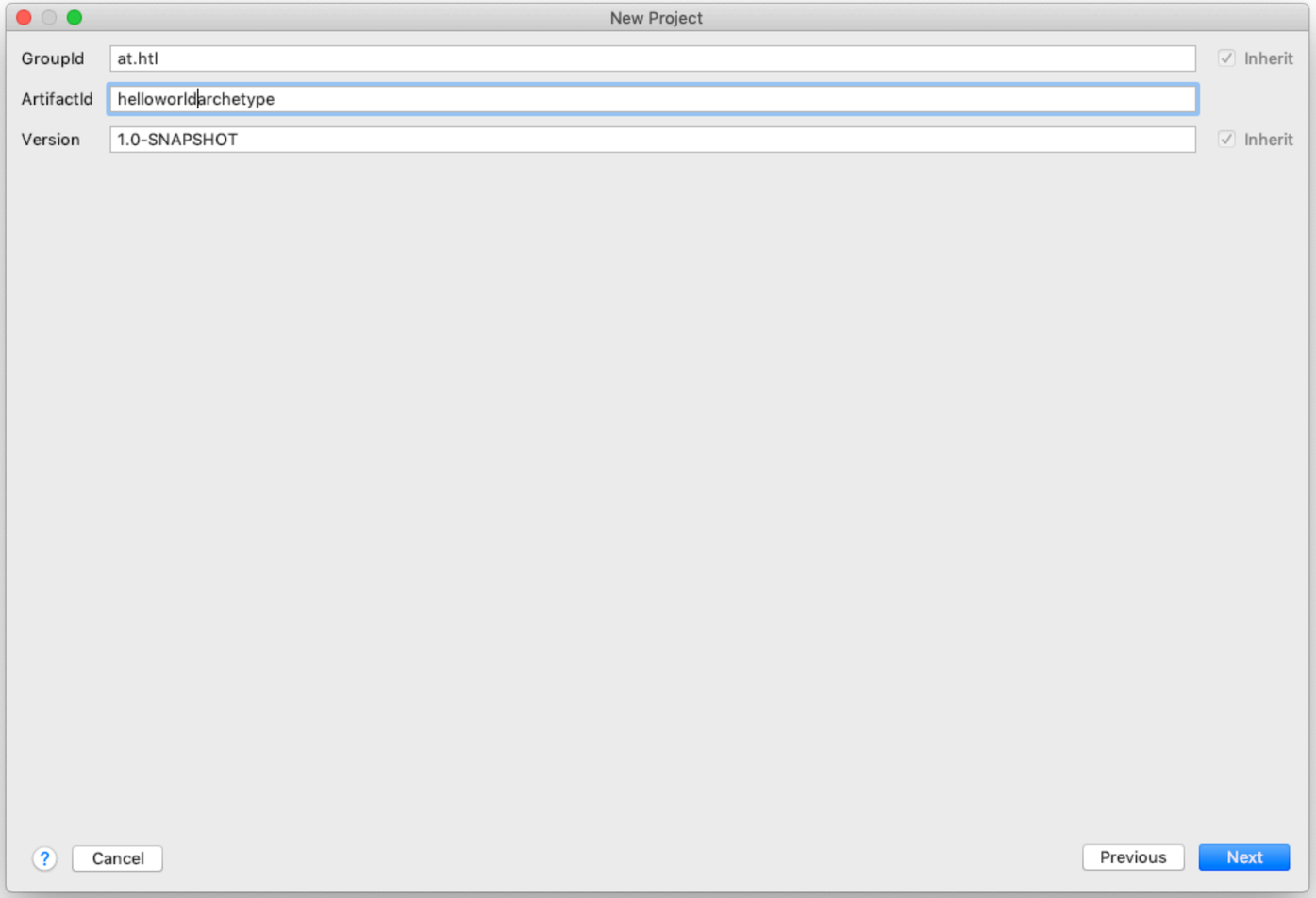
Wir besprechen später noch die Grundstruktur von FX-Applikationen

Erstellung eines JavaFX-Projekts mit FXML (fxml)

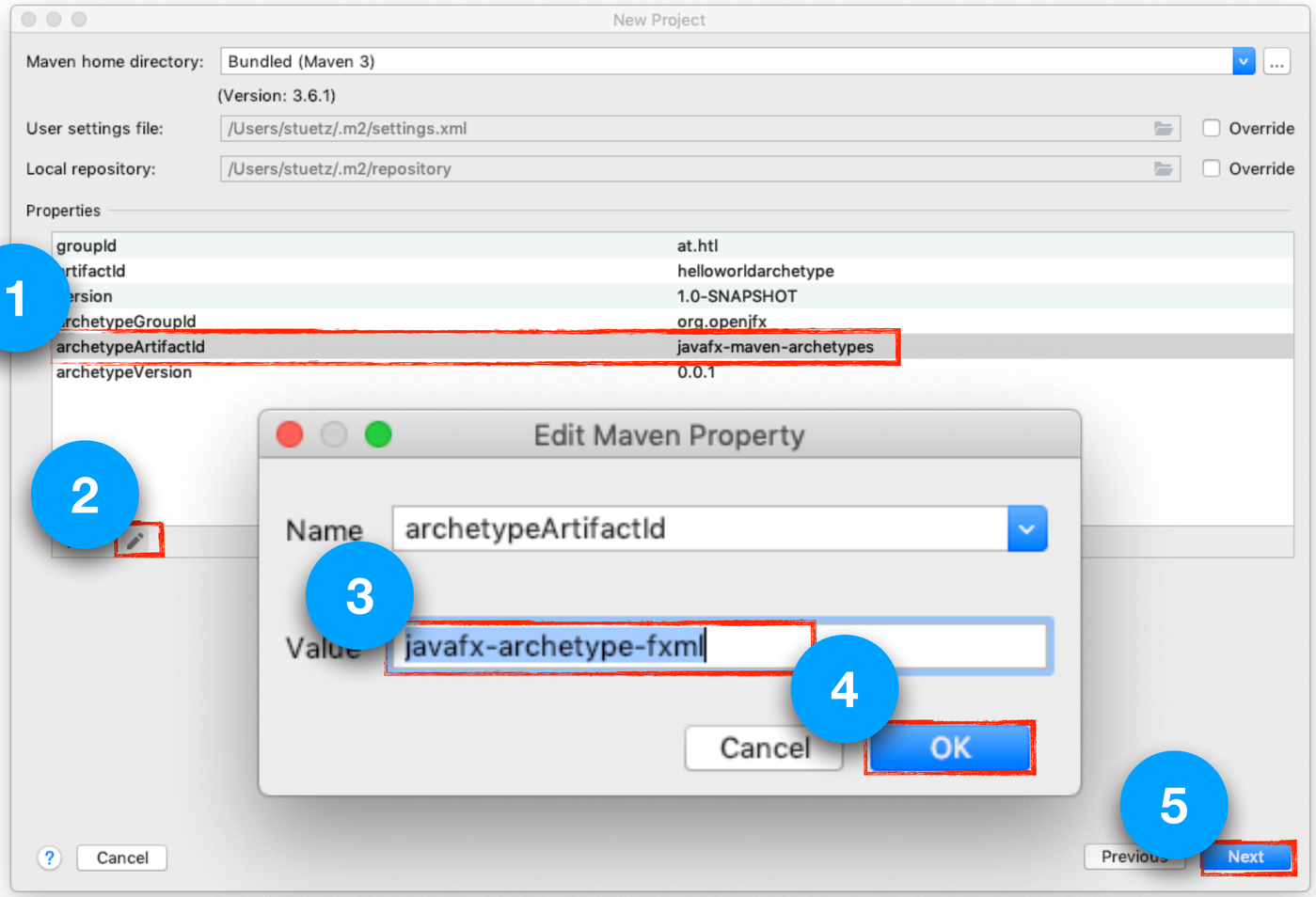
Archetype auswählen



GroupId, ArtifactId

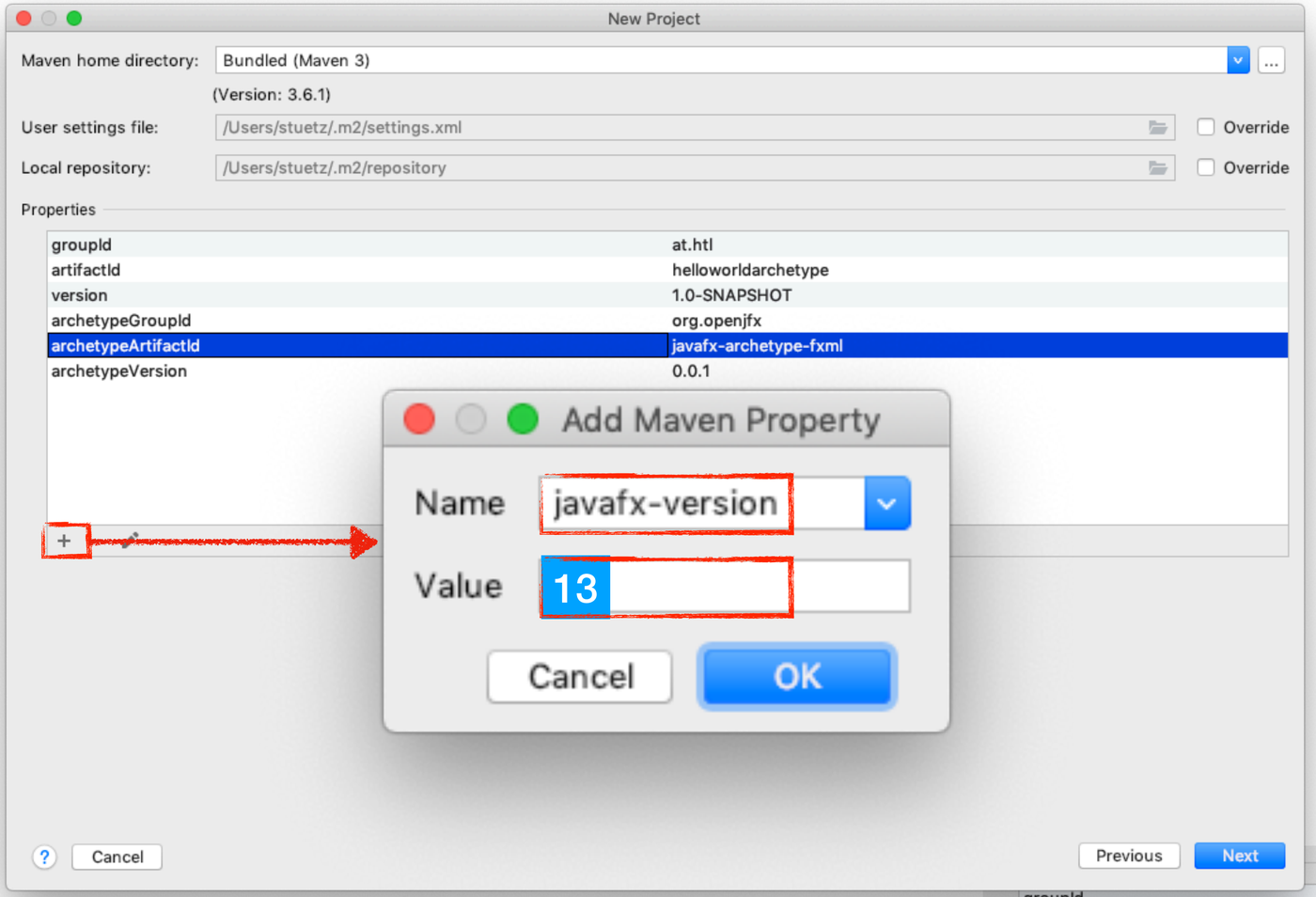


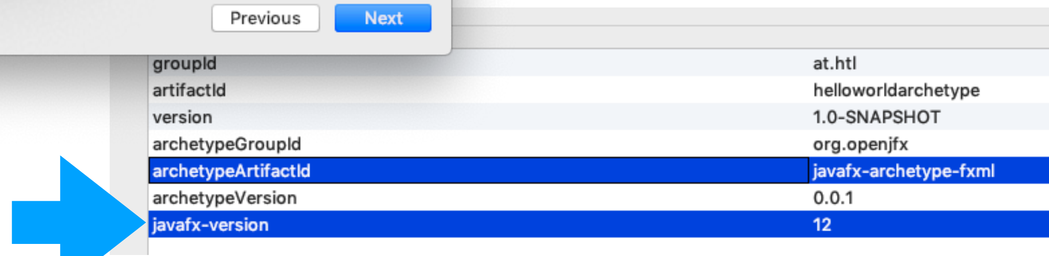
ArchetypeArtifactId eintragen



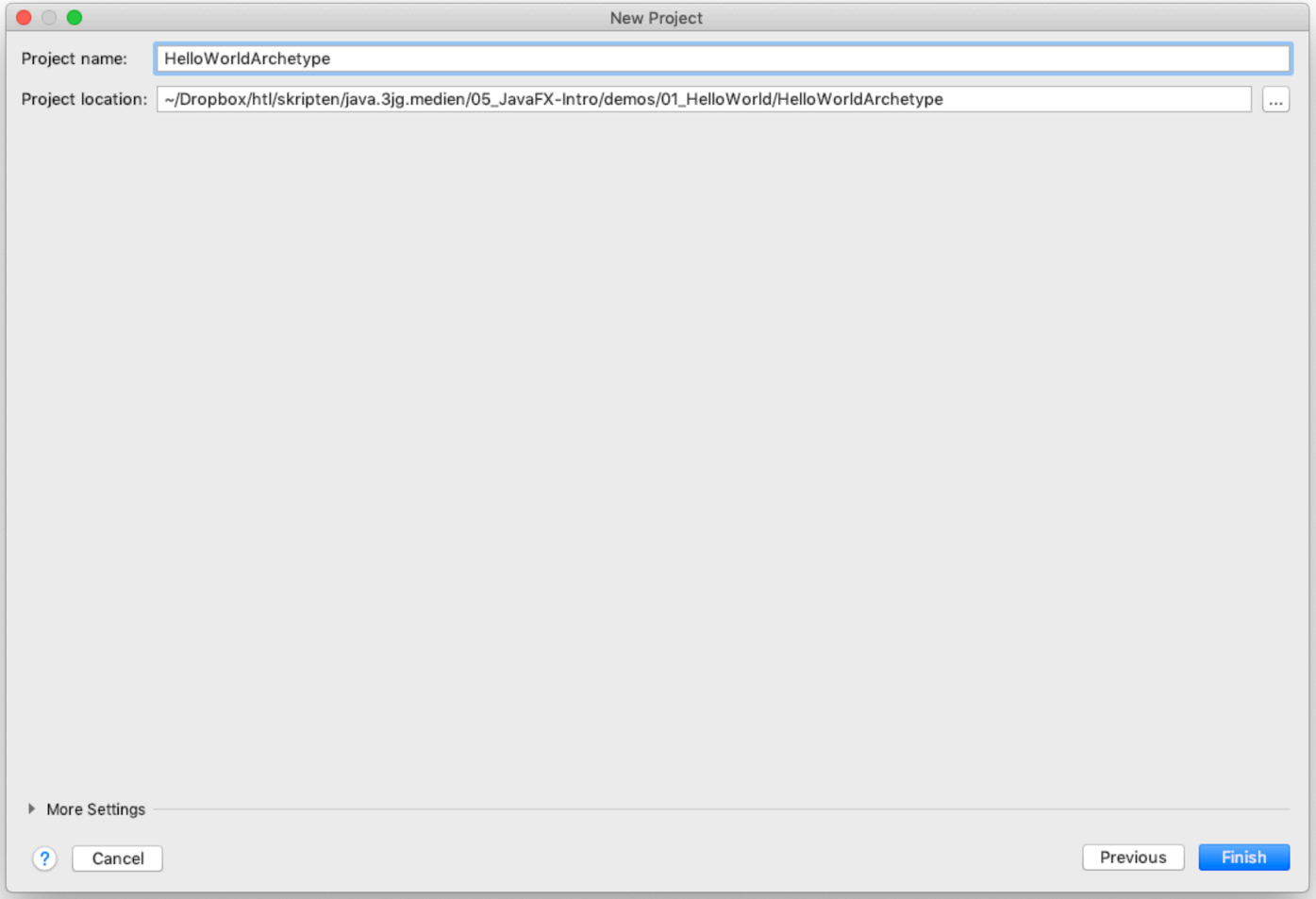
1. archetypeArtifactId auswählen
2. Edit anklicken
3. Entweder **„javafx-archetype-fxml“** eingeben oder falls fxml nicht gebraucht wird **„javafx-archetype-simple“**
4. OK
5. Next

Einstellen der JavaFX-Version

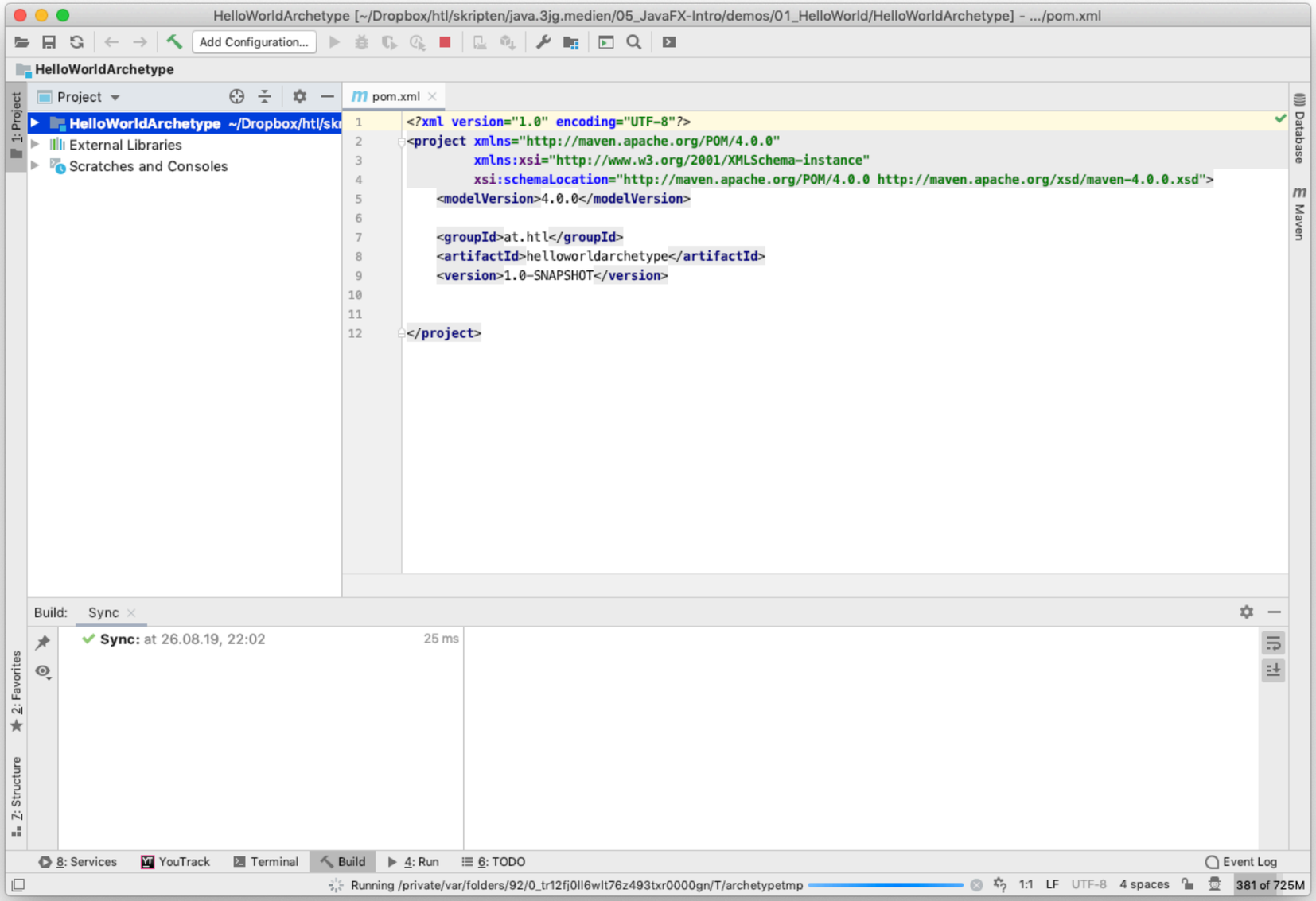


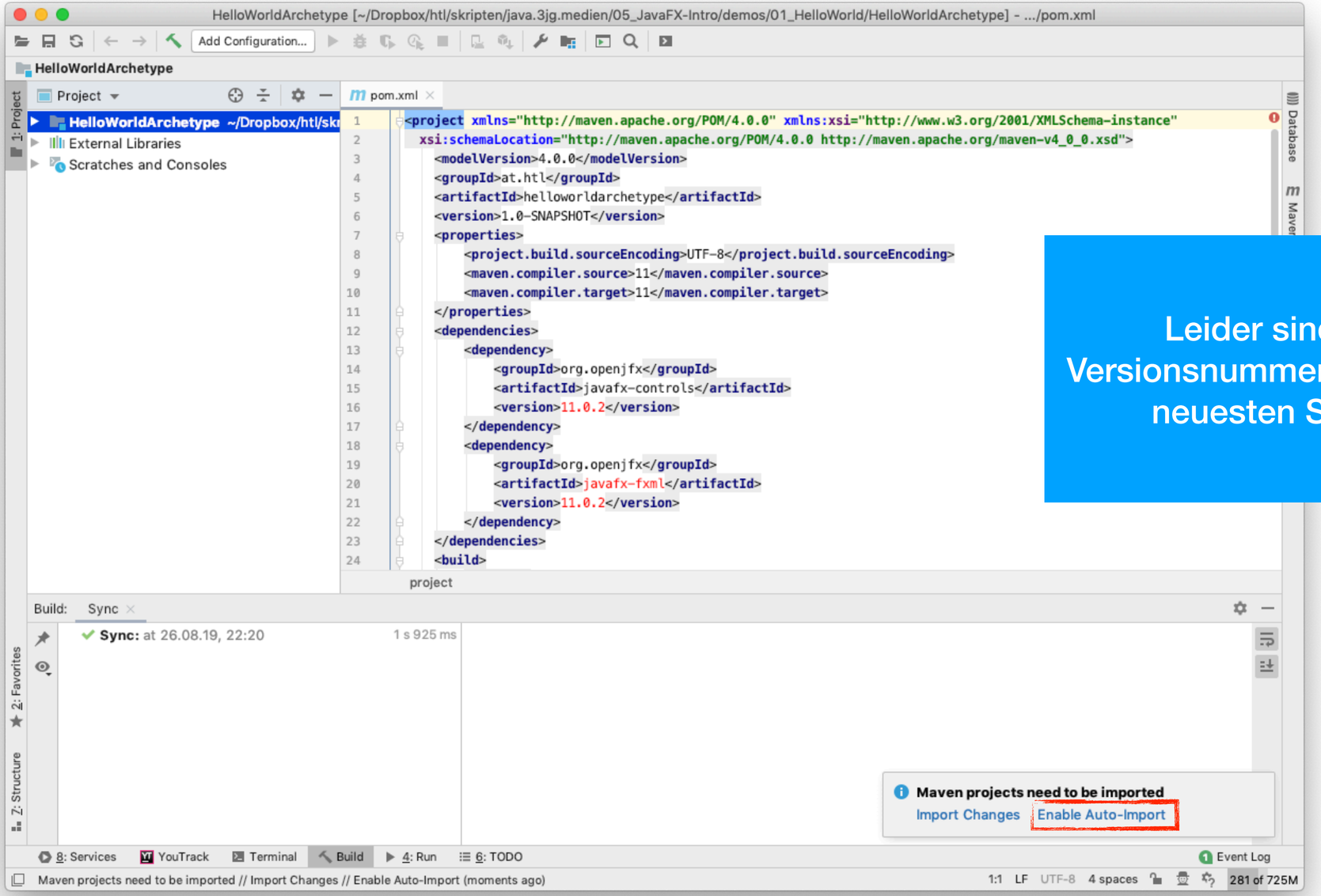


Projektname und Speicherort festlegen



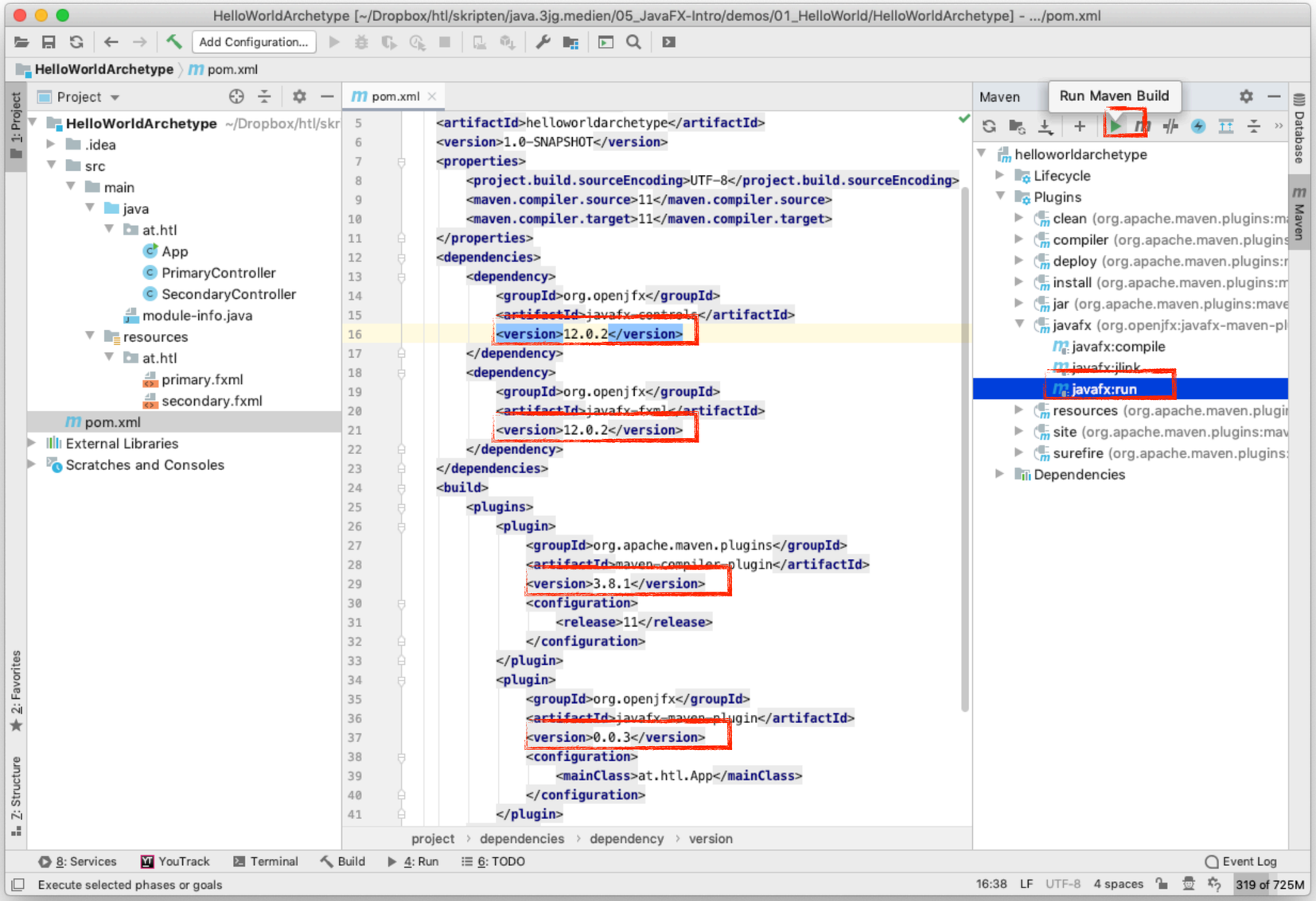
Der Download (Sync) wird durchgeführt



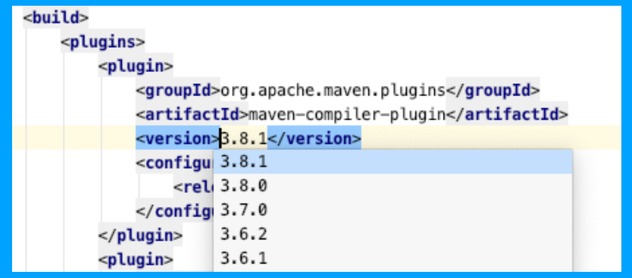


Leider sind die Versionsnummern nicht am neuesten Stand.

Versionsnummern aktualisieren



mit ^-<Space> kann man sich die Versionen ansehen …



… oder: um die neuesten Versionsnummern zu ersehen, einfach im mvnrepository nachsehen (siehe nächste Seite)

https://mvnrepository.com/search?q=org.openjfx

Most Visited | Your data in Search | LEO Deutsch-Englis... | Oracle Admin | HTL Leonding | Moodle-Kurse | Jenkins | News | Steuern und Mess

MVNREPOSITORY

org.openjfx

Search

Repository

- Central 10
- Gluon 4
- Gradle Plugins 1

Group

- org.openjfx 15

License

- GPL 8
- BSD 2

Found 15 results

Sort: **relevance** | popular | newest

1. **JavaFX Controls** — **125** usages
   org.openjfx » javafx-controls — GPL
   JavaFX Controls
   Last Release on Aug 20, 2019

2. **JavaFX Graphics** — **101** usages
   org.openjfx » javafx-graphics — GPL
   JavaFX Graphics
   Last Release on Aug 20, 2019

3. **JavaFX Base** — **98** usages
   org.openjfx » javafx-base — GPL
   JavaFX Base
   Last Release on Aug 20, 2019

4. **JavaFX FXML** — **79** usages
   org.openjfx » javafx-fxml — GPL
   JavaFX FXML
   Last Release on Aug 20, 2019

https://mvnrepository.com/artifact/org.openjfx/javafx-co

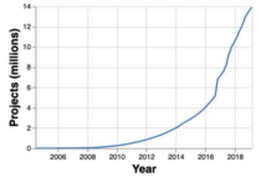
Most Visited | Your data in Search | LEO Deutsch-Englis... | Oracle Admin | HTL Leonding | Moodle-Kurse | Jenkins | News | Steuern und Mes

MVNREPOSITORY

Search for groups, artifacts, categories

Search

Indexed Artifacts (15.0M)



Popular Categories

- Aspect Oriented
- Actor Frameworks
- Application Metrics
- Build Tools
- Bytecode Libraries
- Command Line Parsers
- Cache Implementations
- Cloud Computing
- Code Analyzers
- Collections
- Configuration Libraries
- Core Utilities
- Date and Time Utilities
- Dependency Injection
- Embedded SQL Databases
- HTML Parsers
- HTTP Clients
- I/O Utilities
- JDBC Extensions
- JDBC Pools

Home » org.openjfx » javafx-controls

JavaFX Controls

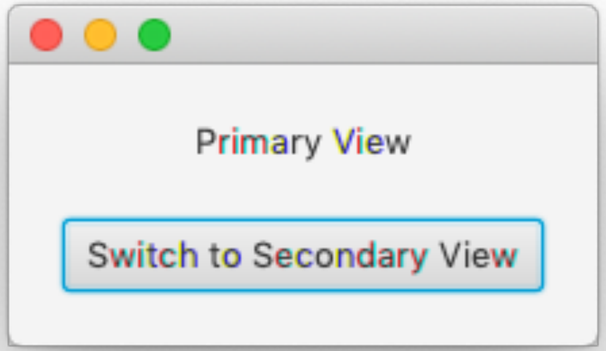
JavaFX Controls

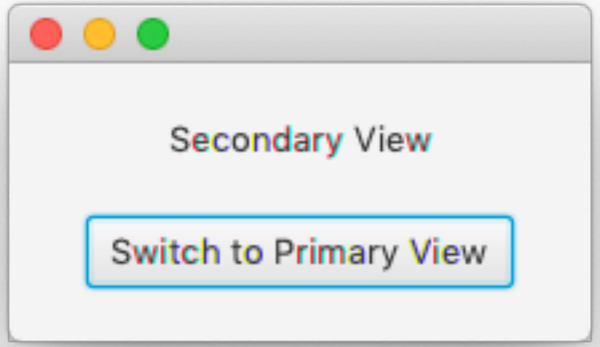
License	GPL 2.0
Used By	**125 artifacts**

Central (37) | ICM (2)

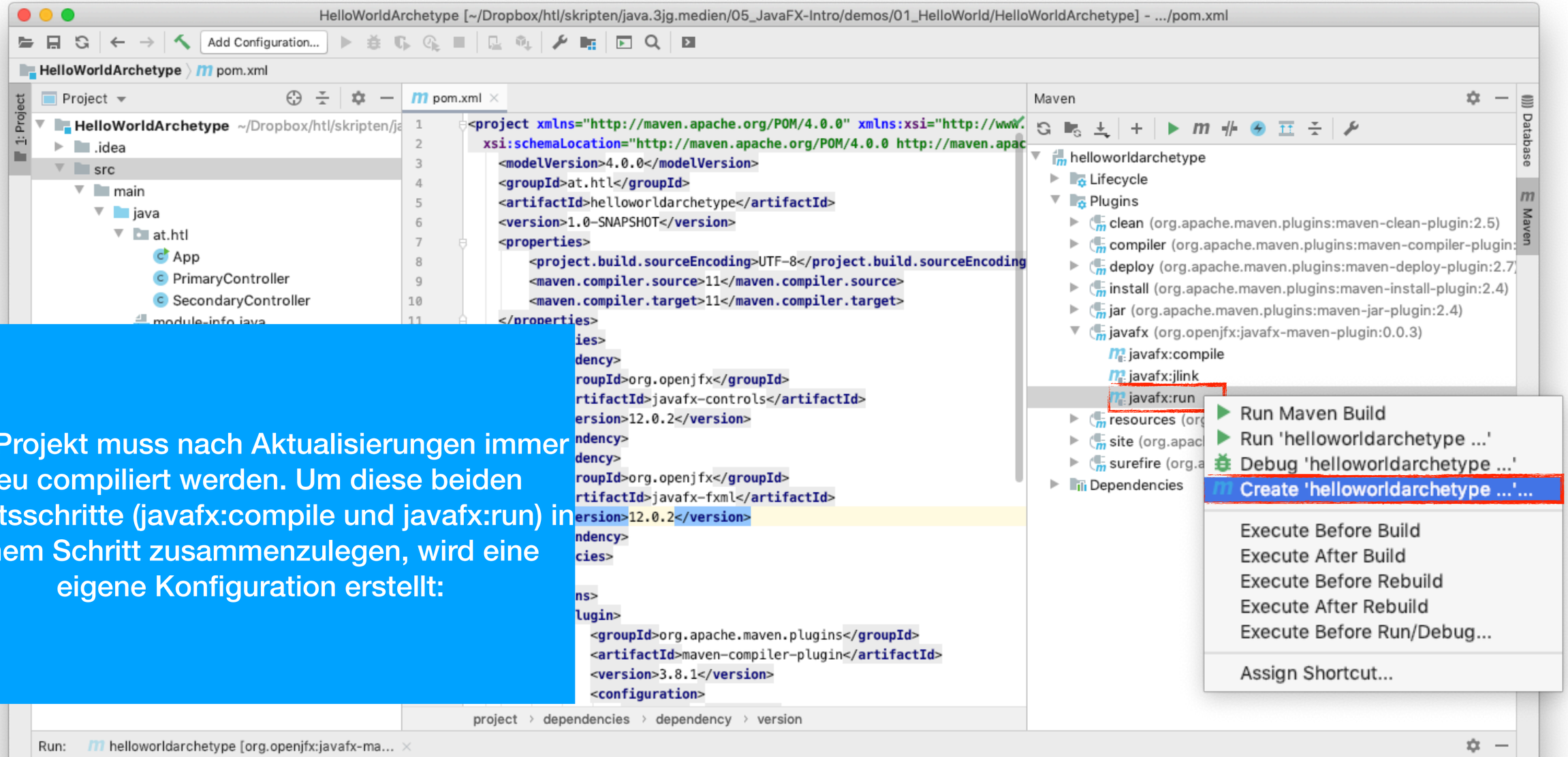
Version	Repository	Usages	Date
13-ea+13	Central	3	Aug, 2019
13-ea+12	Central	3	Aug, 2019
13-ea+11	Central	3	Jul, 2019
13-ea+10	Central	3	Jul, 2019
13-ea+9	Central	3	Jun, 2019
13-ea+8	Central	3	May, 2019
13-ea+7	Central	3	May, 2019
13-ea+6	Central	3	Apr, 2019
13-ea+5	Central	3	Apr, 2019
13-ea+4	Central	3	Apr, 2019
13-ea+3	Central	3	Mar, 2019
13-ea+2	Central	3	Mar, 2019
13-ea+1	Central	3	Mar, 2019
12.0.2	Central	4	Jul, 2019

Ergebnis


Primary View
Switch to Secondary View

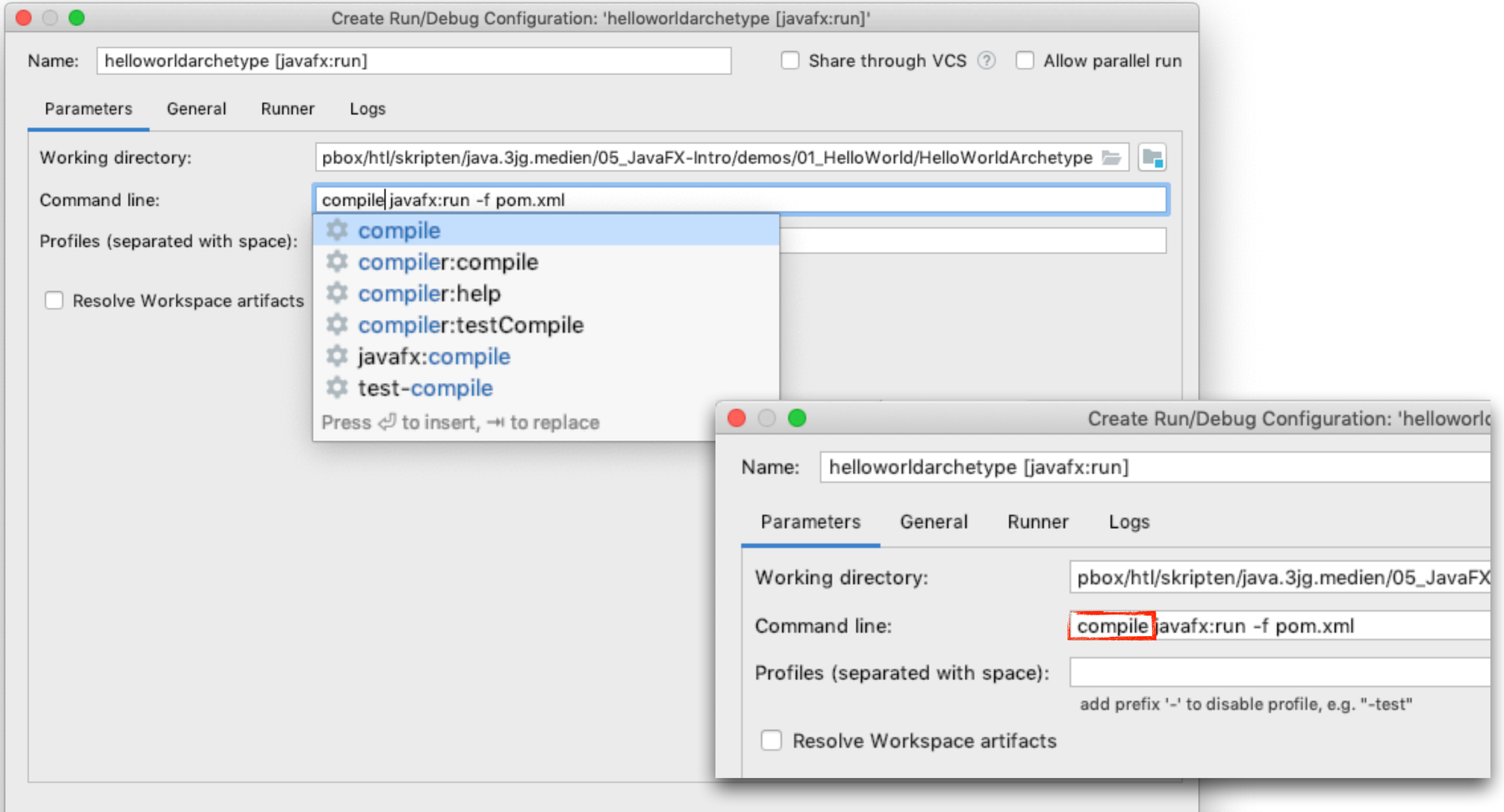

Secondary View
Switch to Primary View

Eigene Configuration erstellen

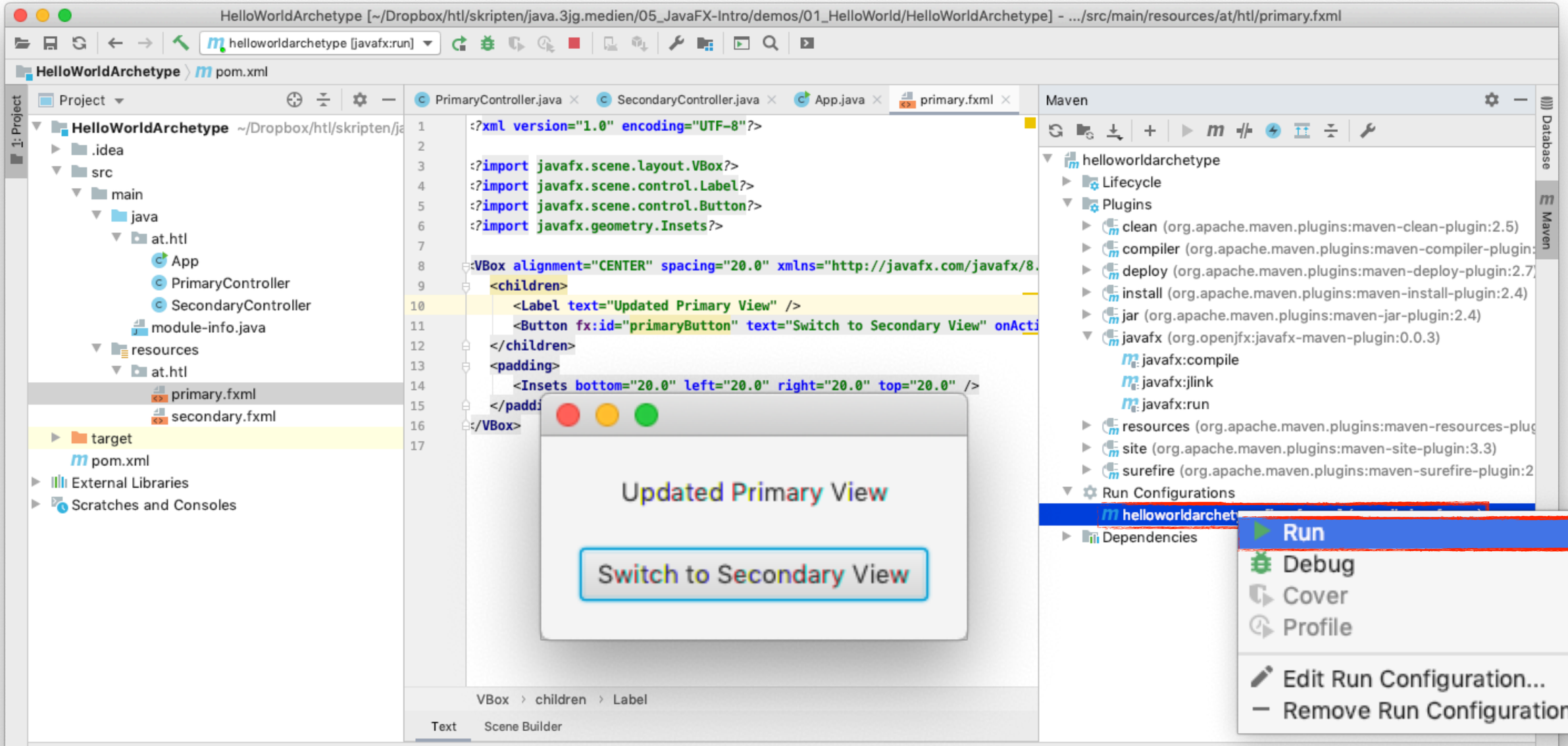


Das Projekt muss nach Aktualisierungen immer neu compiliert werden. Um diese beiden Arbeitsschritte (javafx:compile und javafx:run) in einem Schritt zusammenzulegen, wird eine eigene Konfiguration erstellt:

Compile eintragen



Ausführen: compile + run



Ende - Erstellen eines JavaFX-Projekts mit FXML (fxml)

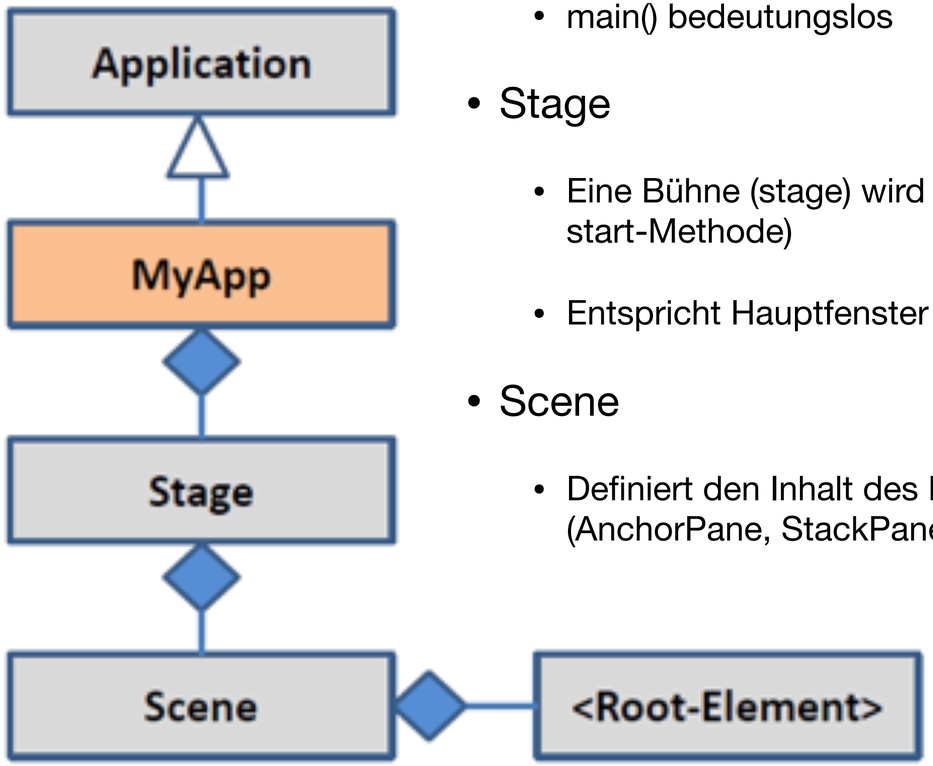
Grundstruktur der FX-Anwendung

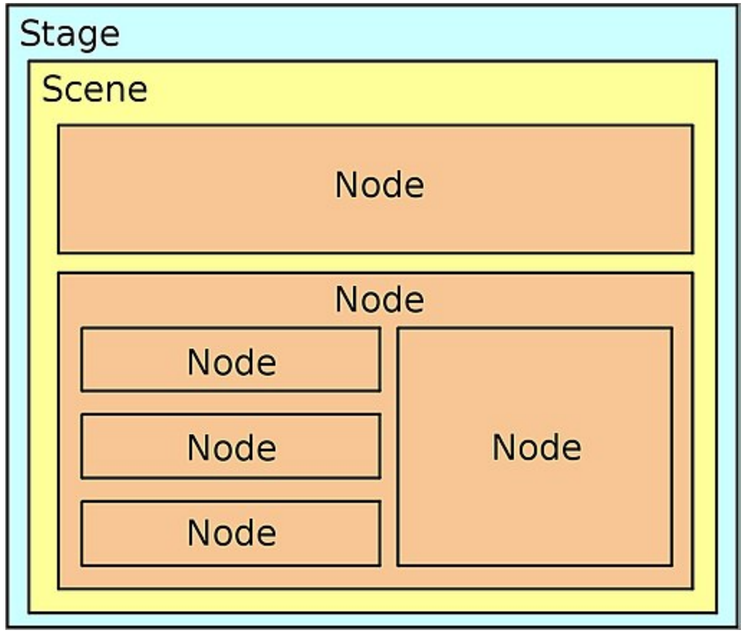
```java
package at.htl.javafxdemoapplication;

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

public class JavaFXDemoApplication extends Application {

    @Override
    public void start(Stage stage) {...}

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

kann mit ^O generiert werden

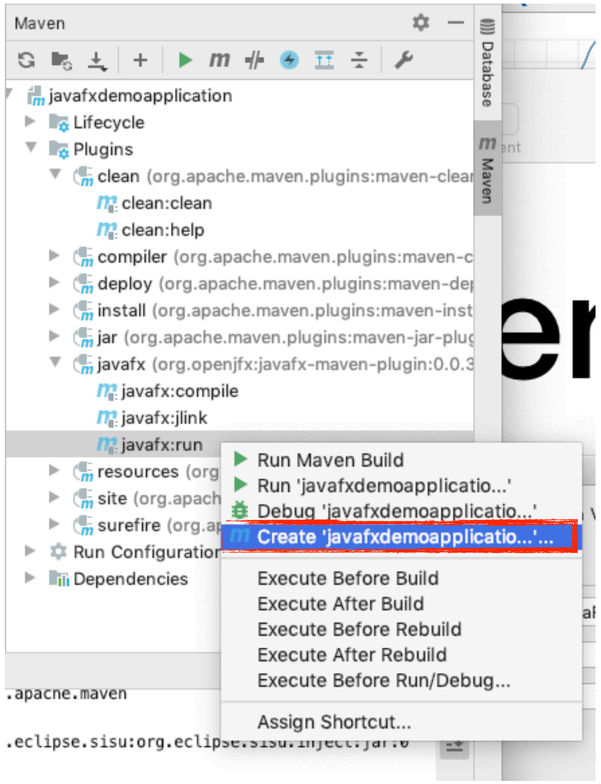
Grundelemente

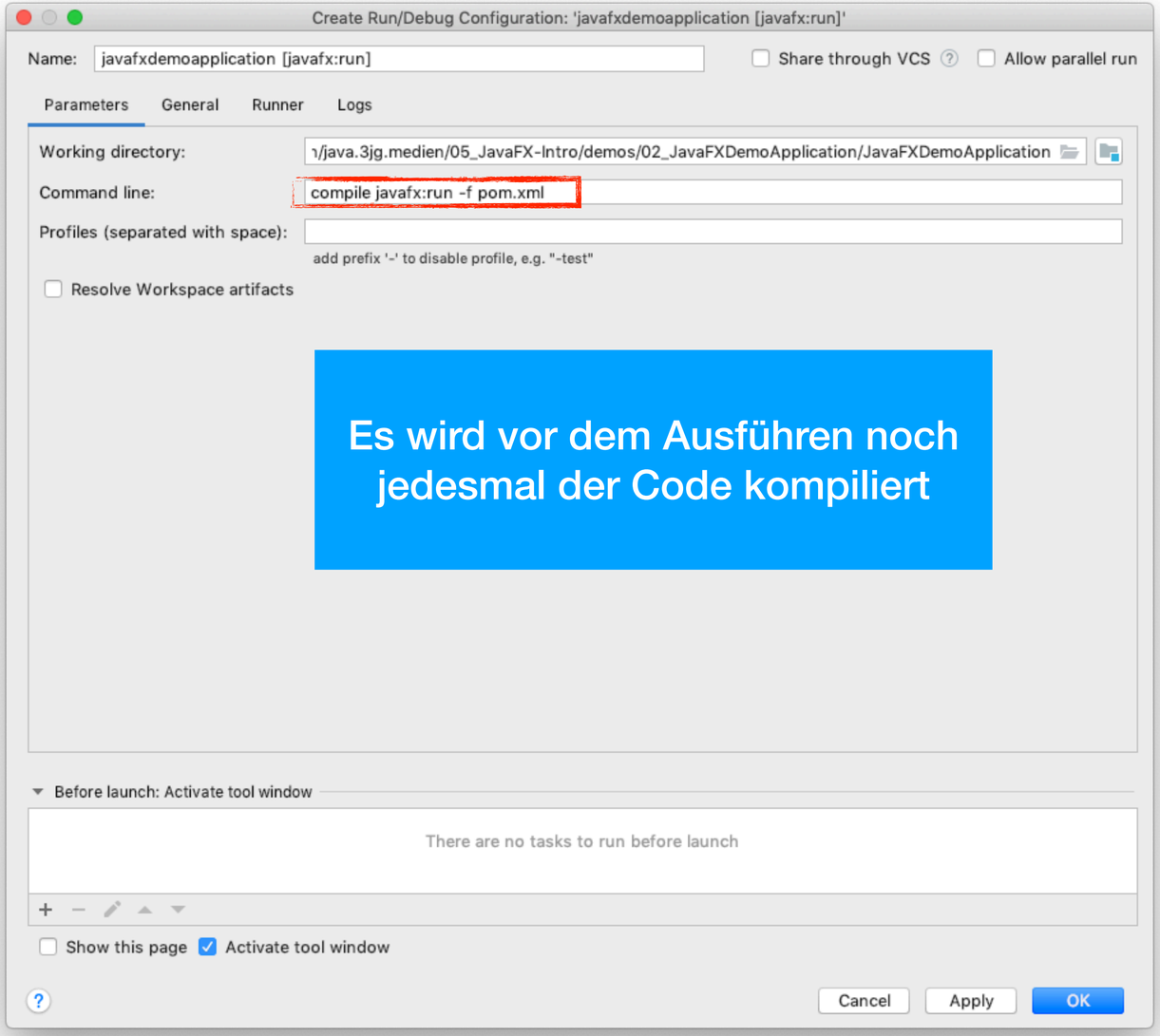
- MyFXApp
  - leitet von Application ab
  - start() wird überschrieben
  - main() bedeutungslos
- Stage
  - Eine Bühne (stage) wird beim Start an MyApp übergeben (Parameter der start-Methode)
  - Entspricht Hauptfenster der Anwendung und enthält genau eine Scene
- Scene
  - Definiert den Inhalt des Fensters ausgehend von einem Wurzelelement (AnchorPane, StackPane, GridPane, …)



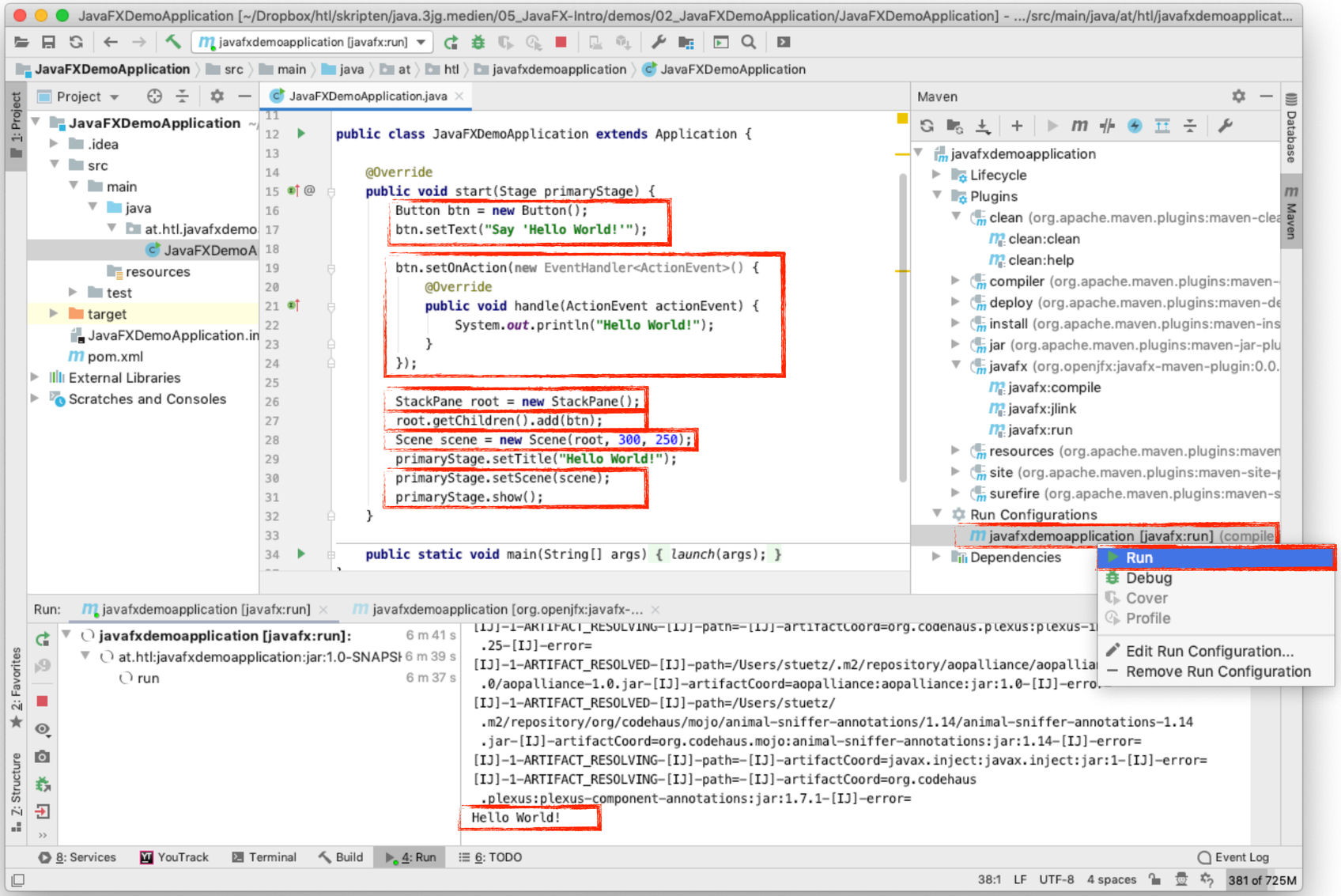


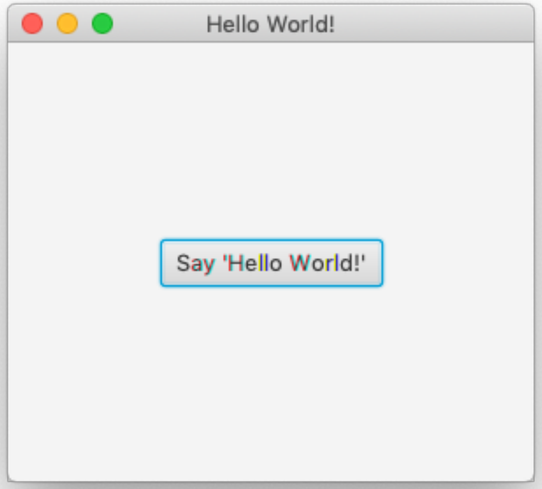
Eigene Configuration erstellen





Starten der Anwendung

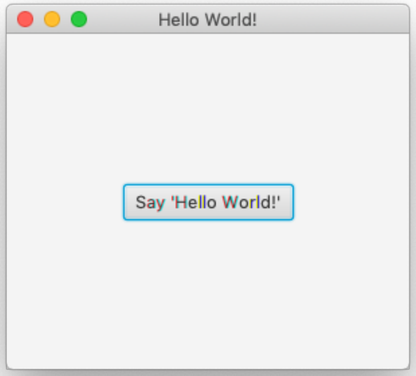




Startmethode und Ergebnis

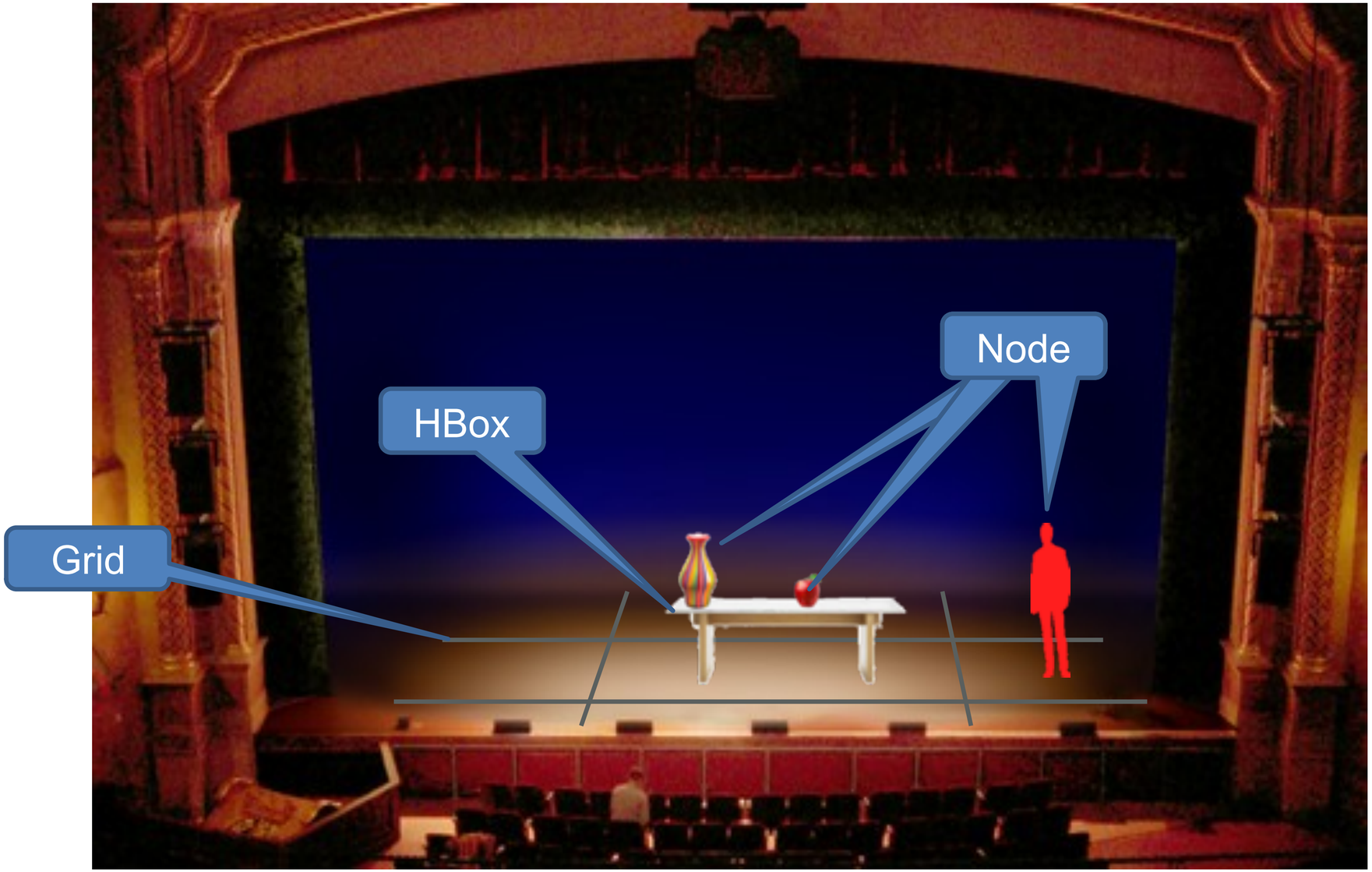
```java
public class JavaFXDemoApplication extends Application {

    @Override
    public void start(Stage primaryStage) {
        Button btn = new Button();
        btn.setText("Say 'Hello World!'");

        btn.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent actionEvent) {
                System.out.println("Hello World!");
            }
        });

        StackPane root = new StackPane();
        root.getChildren().add(btn);
        Scene scene = new Scene(root, 300, 250);
        primaryStage.setTitle("Hello World!");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

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

```java
btn.setOnAction(actionEvent -> System.out.println("Hello World!"));
```

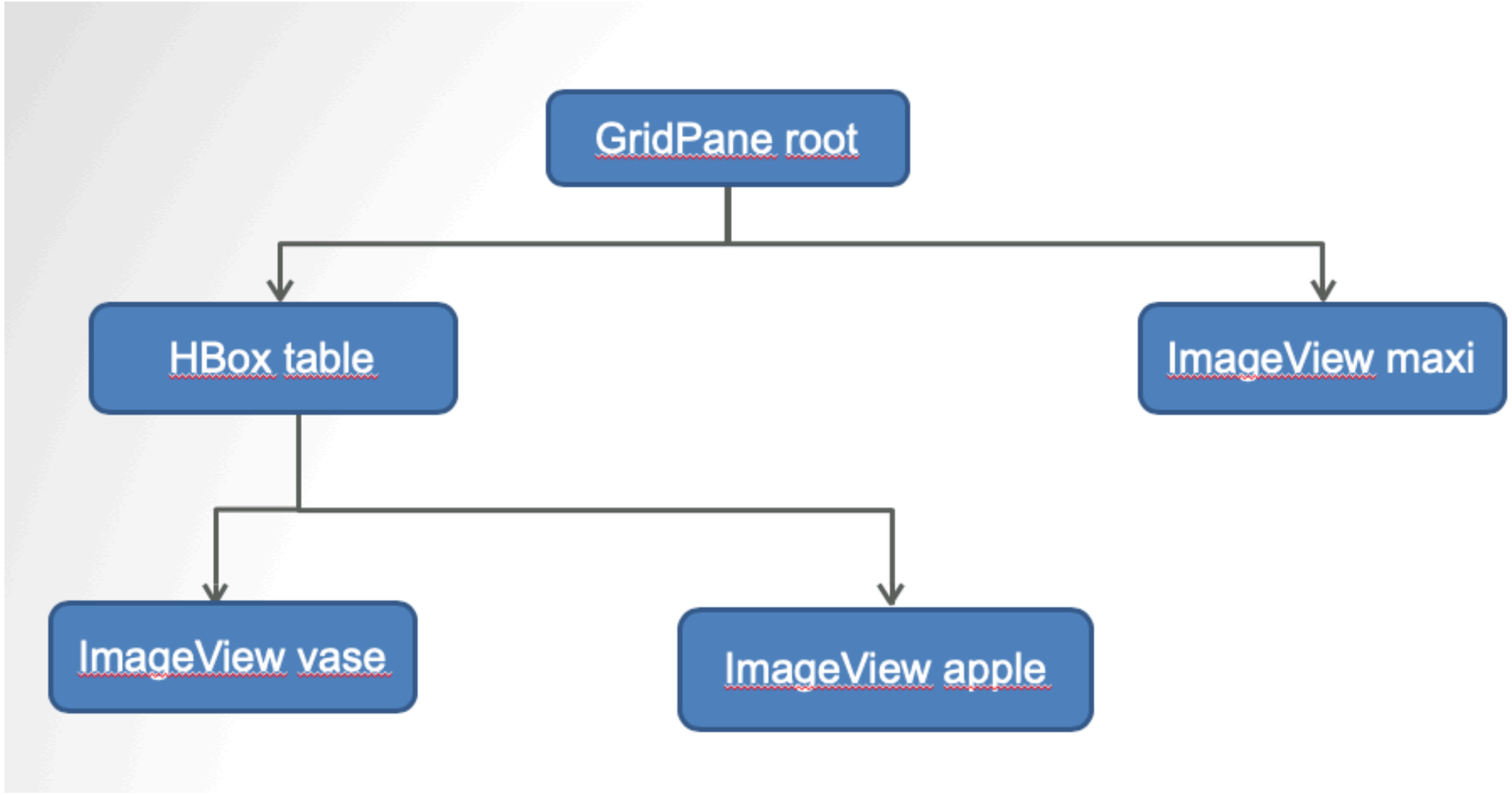



```
[IJ]-1-ARTIFACT_RESOLVING-[IJ
[IJ]-1-ARTIFACT_RESOLVING-[IJ
.plexus:plexus-component-ann
Hello World!
```

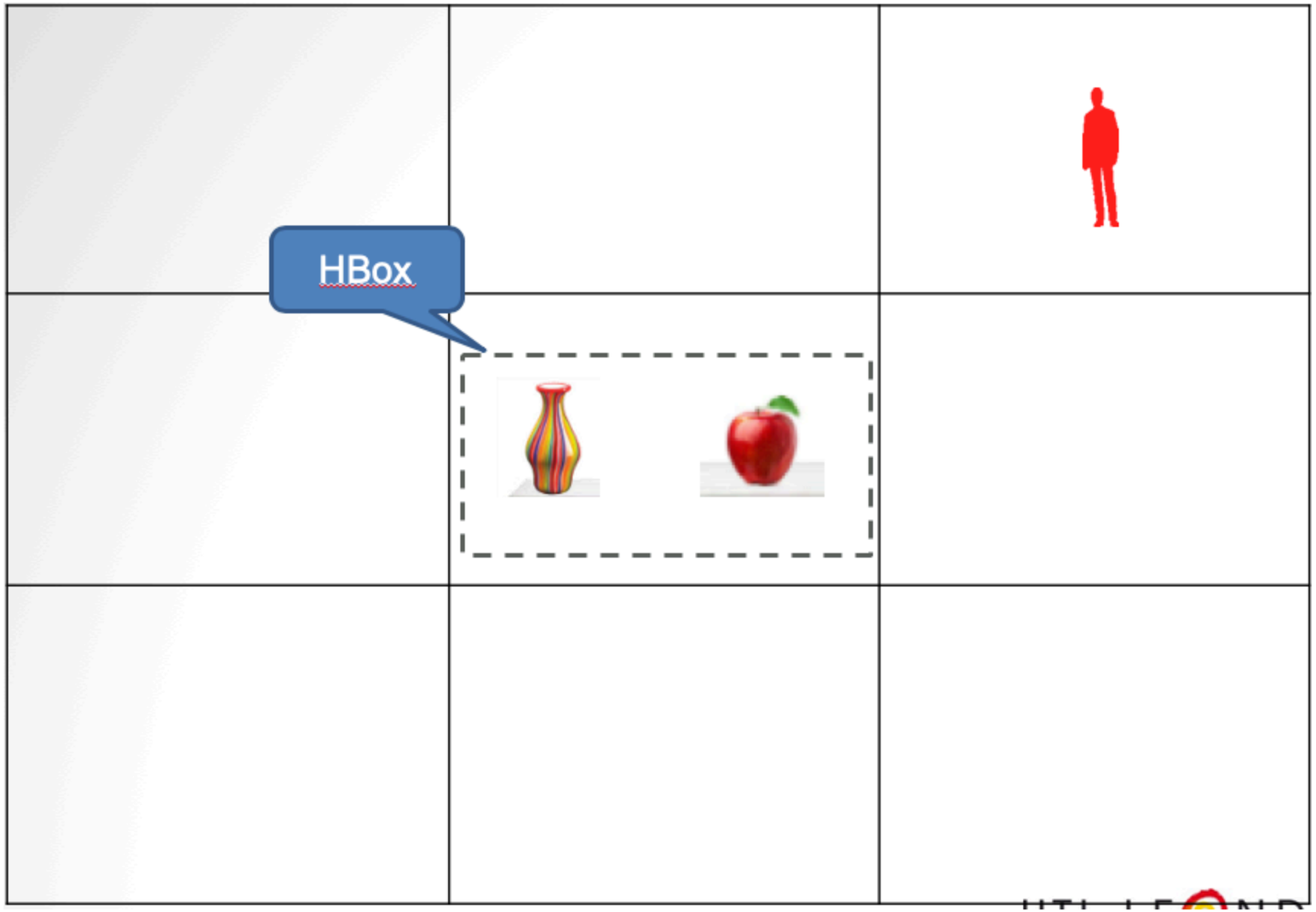
Stage -> Scene -> Grid



Scene Graph


GridPane root
HBox table
ImageView maxi
ImageView vase
ImageView apple

Scene Layout



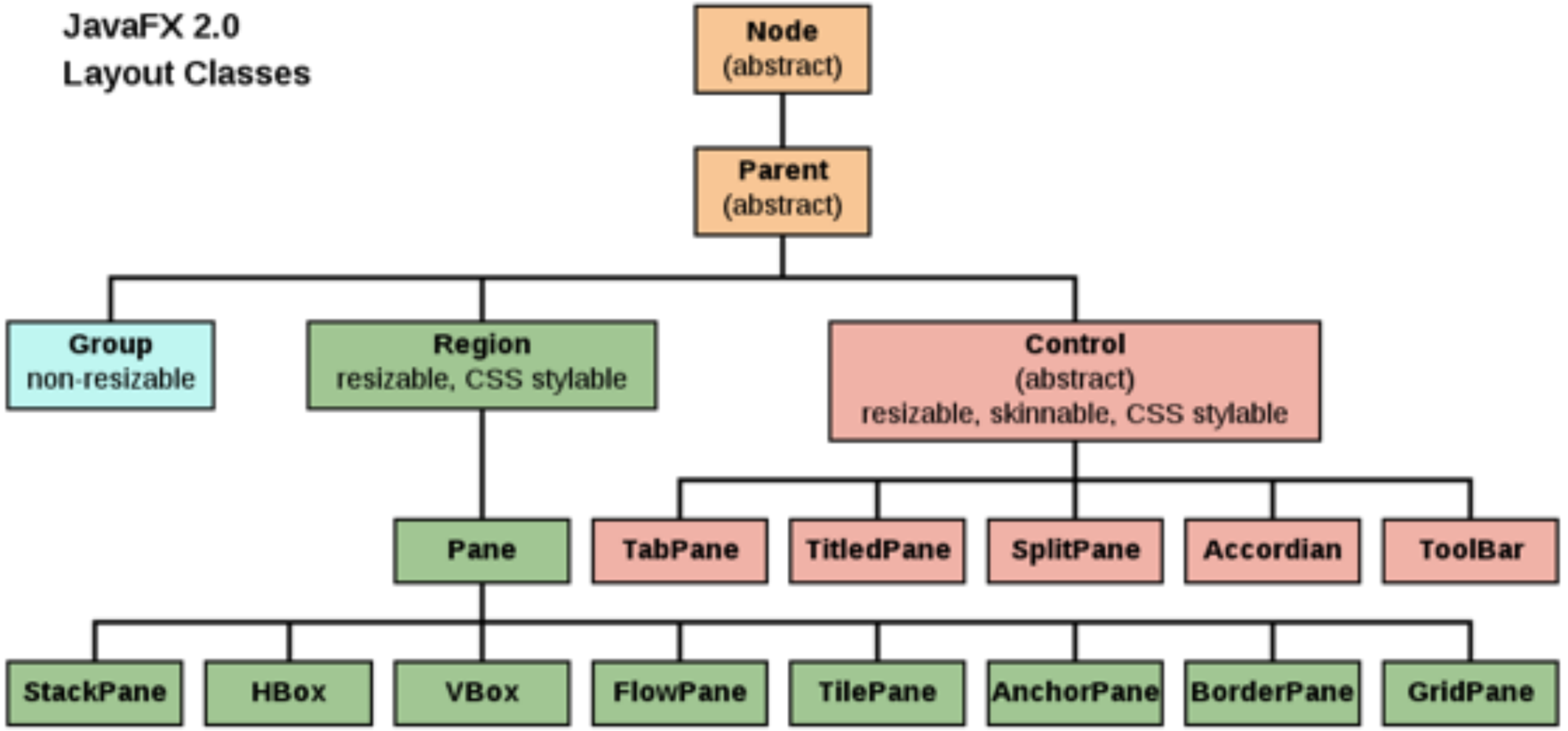
Source Code

```
public void start(Stage primaryStage) {
    ImageView maxi = new ImageView("maxi.png");
    ImageView vase = new ImageView("vase.png");
    ImageView apple = new ImageView("apple.png");

    HBox table = new HBox();
    table.getChildren().addAll(vase, apple);
    GridPane root = new GridPane();
    ColumnConstraints columnConstraints = new ColumnConstraints(150);
    RowConstraints rowConstraints = new RowConstraints(90);
    for (int i = 0; i < 2; i++){
        root.getColumnConstraints().add(columnConstraints);
        root.getRowConstraints().add(rowConstraints);
    }
    root.add(table, 1, 1);
    root.add(maxi, 2, 0);

    Scene scene = new Scene(root, 450, 270);
    primaryStage.setScene(scene);
    primaryStage.show();
}
```

HBox table = new HBox(vase, apple);

Zellengrößen definieren

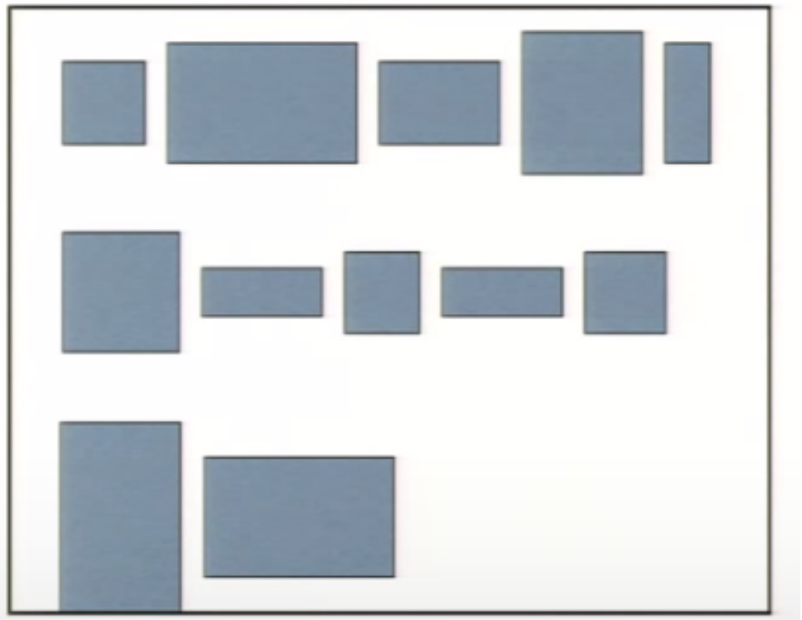
Hierarchie Container-Elemente



- **Group**: Nur logische Zusammenfassung von anderen Nodes
- **Region**: Standard-Layouts, CSS-fähig
- **Control**: Spezielle Layouts, die auch Benutzereingaben verarbeiten, css-fähig

FlowPane

- Anordnung der Controls nacheinander
- Horizontal oder vertikal
- Abstände zwischen den Elementen

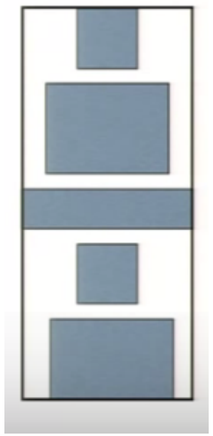


```
Button okButton = new Button("Weiter");
Button cancelButton = new Button("Abbruch");
Button helpButton = new Button("Hilfe");
FlowPane flowPane = new FlowPane(Orientation.HORIZONTAL, 20, 20);
flowPane.getChildren().addAll(okButton, cancelButton, helpButton);
flowPane.setAlignment(Pos.TOP_RIGHT);

Scene scene = new Scene(flowPane, 300, 250);
primaryStage.setTitle("Hello World!");
primaryStage.setScene(scene);
primaryStage.show();
```

Java FX – HBox, VBox

- HBOX

- VBOX

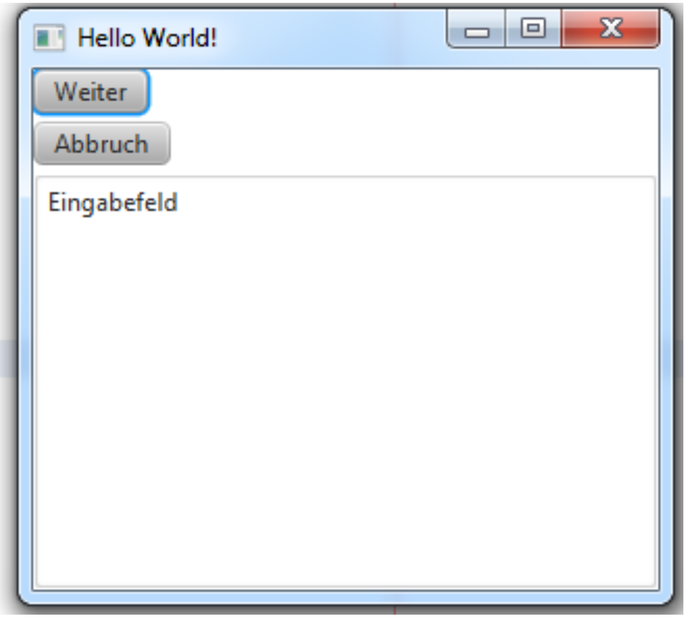


siehe auch: https://www.schmiedecke.info/Prg1/Konzepte/VL19-%20JAVAFX%20Panes%20and%20Controls.pdf

VBox/HBox

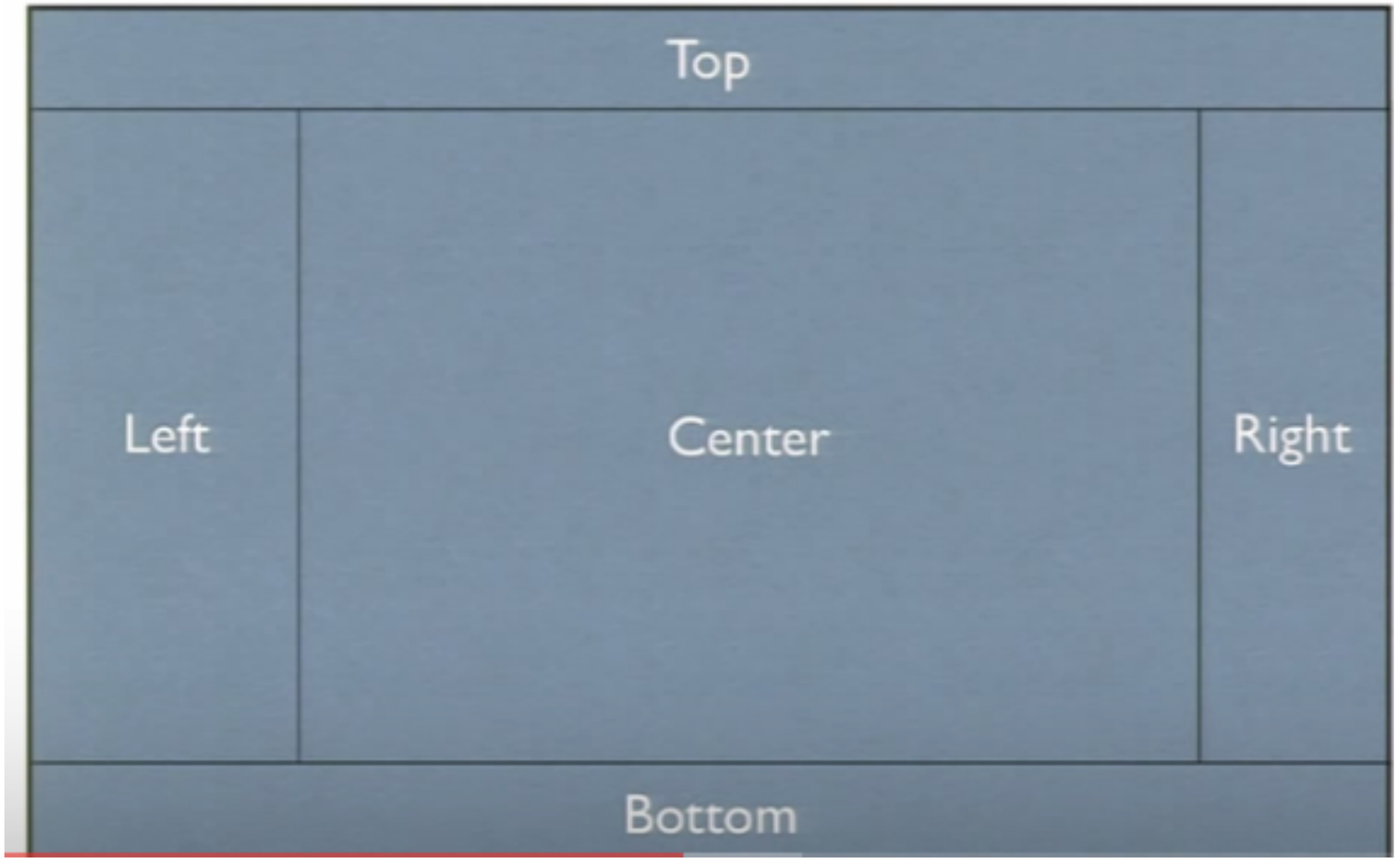
- Anordnung nacheinander/untereinander ohne Umbruch (FlowPane)
- Abstände können angegeben werden
- setVgrow()/setHGrow() bestimmt, was mit dem restlichen Platz geschieht

```
@Override
public void start(Stage primaryStage) {
 VBox vBox = new VBox(4.0);
    Button okButton = new Button("Weiter");
    Button cancelButton = new Button("Abbruch");
    TextArea text = new TextArea("Eingabefeld");
    VBox.setVgrow(text, Priority.ALWAYS);
    vBox.getChildren().
            addAll(okButton, cancelButton, text);
    Scene scene = new Scene(vBox, 300, 250);
    primaryStage.setTitle("Hello World!");
    primaryStage.setScene(scene);
    primaryStage.show();
}
```



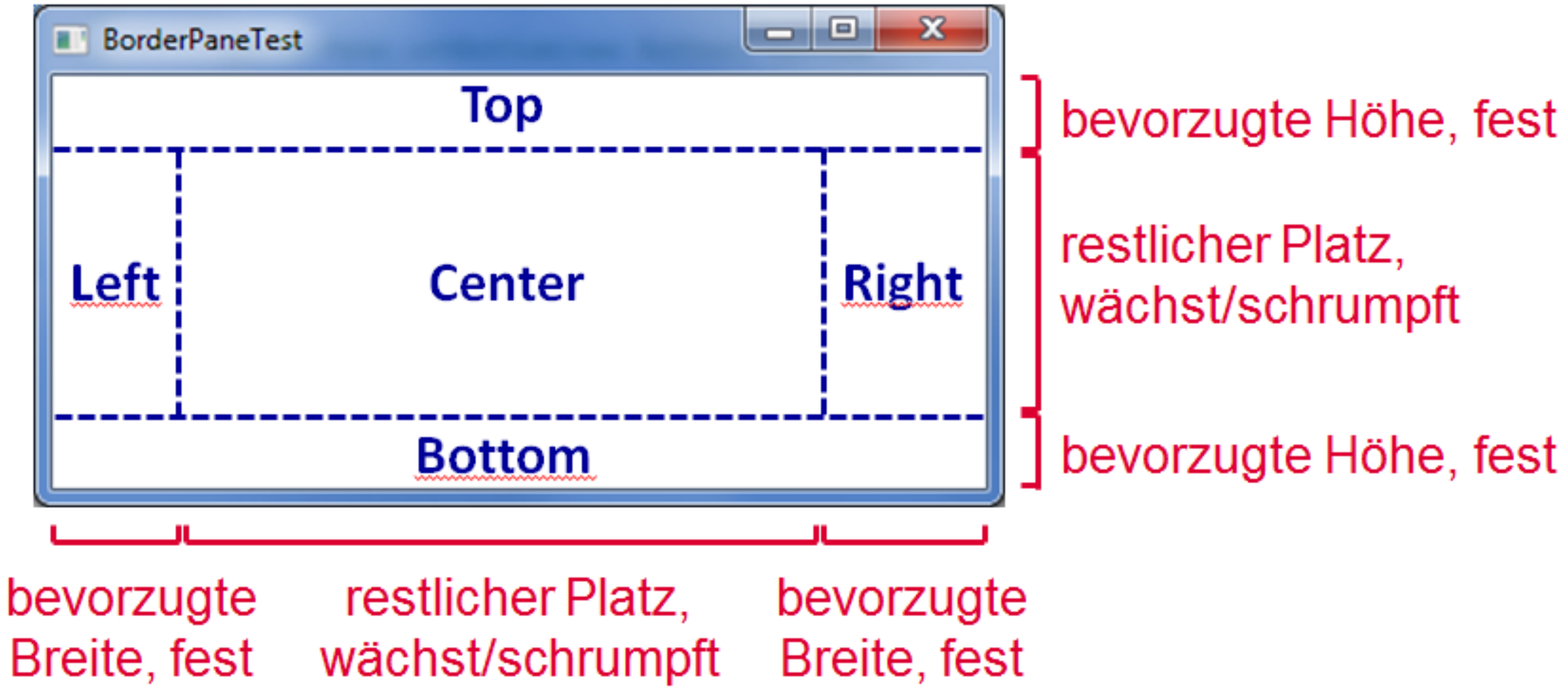
Java FX – BorderPane

- BorderPane



BorderPane

- Bis zu 5 Platzierungsmöglichkeiten

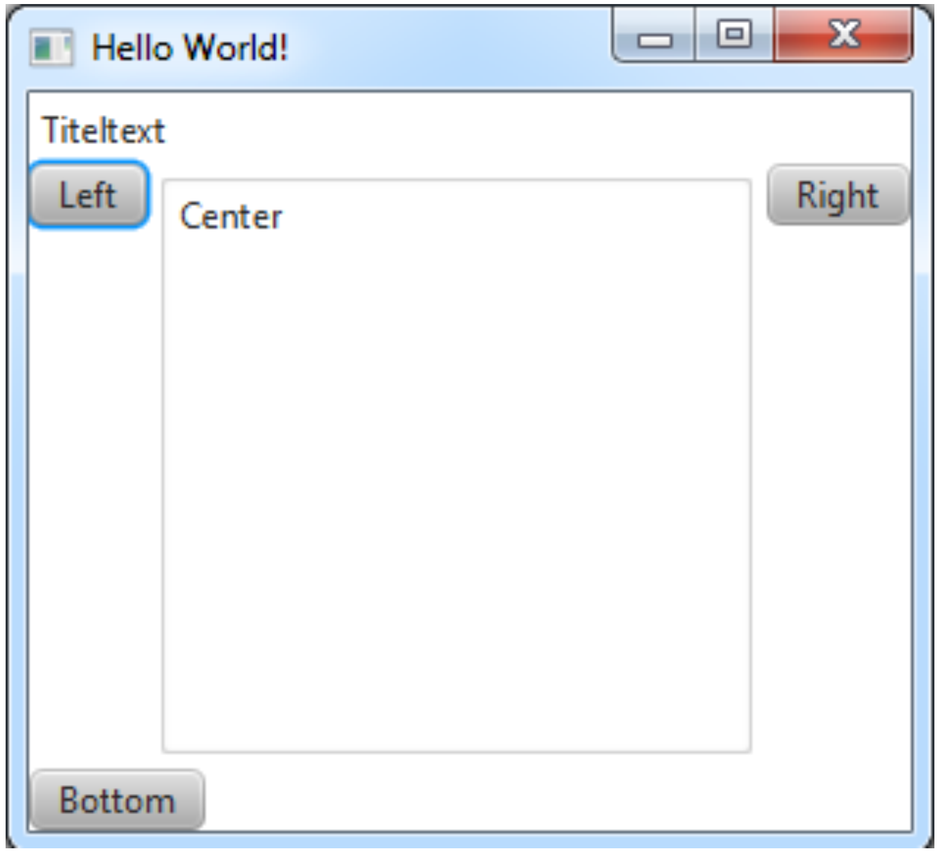


BorderPane - Beispiel

```
BorderPane borderPane = new BorderPane();
Label topLabel = new Label("Titeltext");
BorderPane.setAlignment(topLabel, Pos.CENTER_LEFT);
BorderPane.setMargin(topLabel, new Insets(4.0, 4.0, 4.0, 4.0));
borderPane.setTop(topLabel);
borderPane.setLeft(new Button("Left"));
borderPane.setRight(new Button("Right"));
TextArea centerText = new TextArea("Center");
BorderPane.setAlignment(centerText, Pos.CENTER);
BorderPane.setMargin(centerText, new Insets(4.0, 4.0, 4.0, 4.0));
borderPane.setCenter(centerText);
borderPane.setBottom(new Button("Bottom"));
Scene scene = new Scene(borderPane, 300, 250);
```

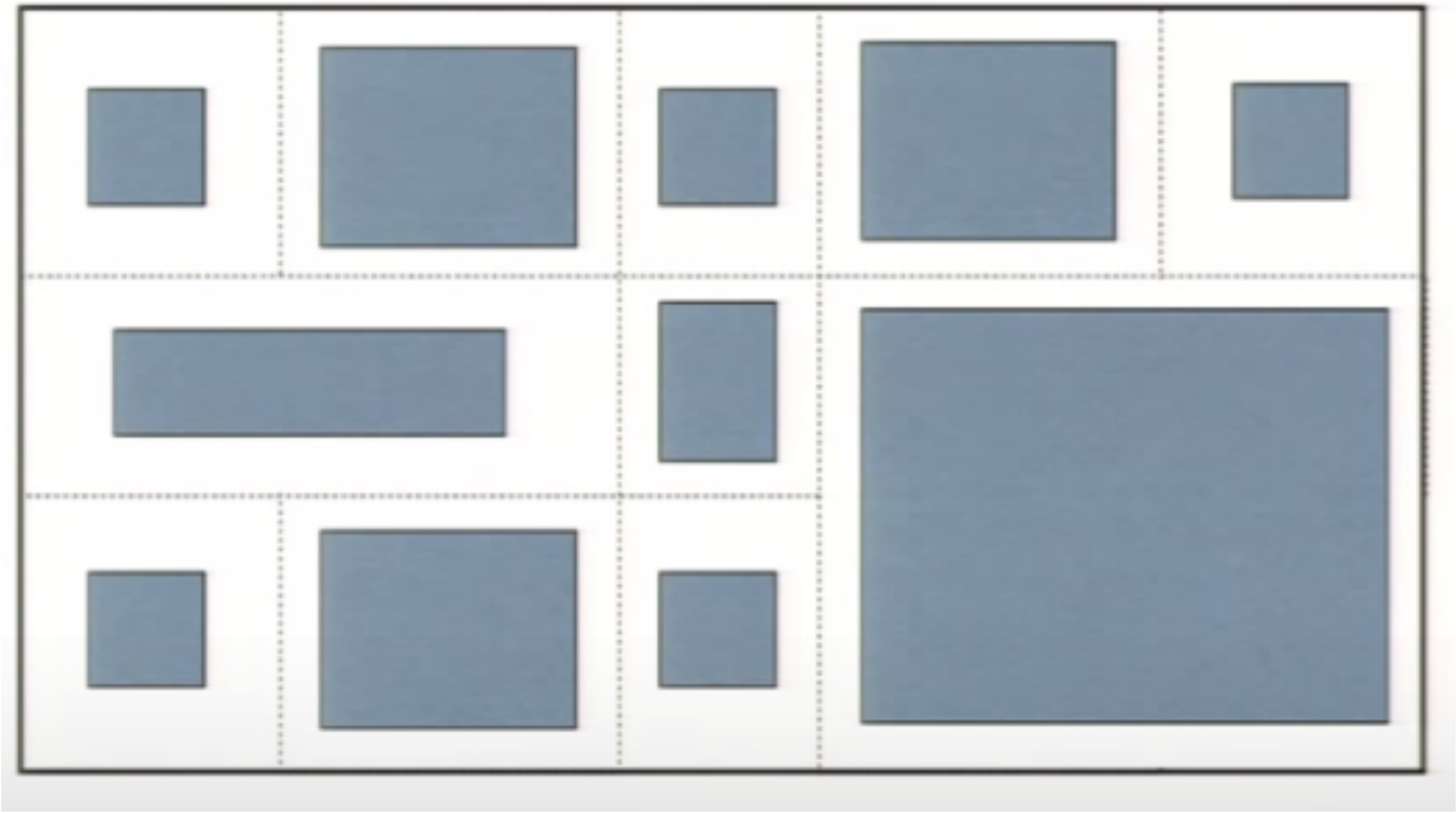
BorderPane - Ergebnis

- Margin für Titeltext und Text im Zentrum



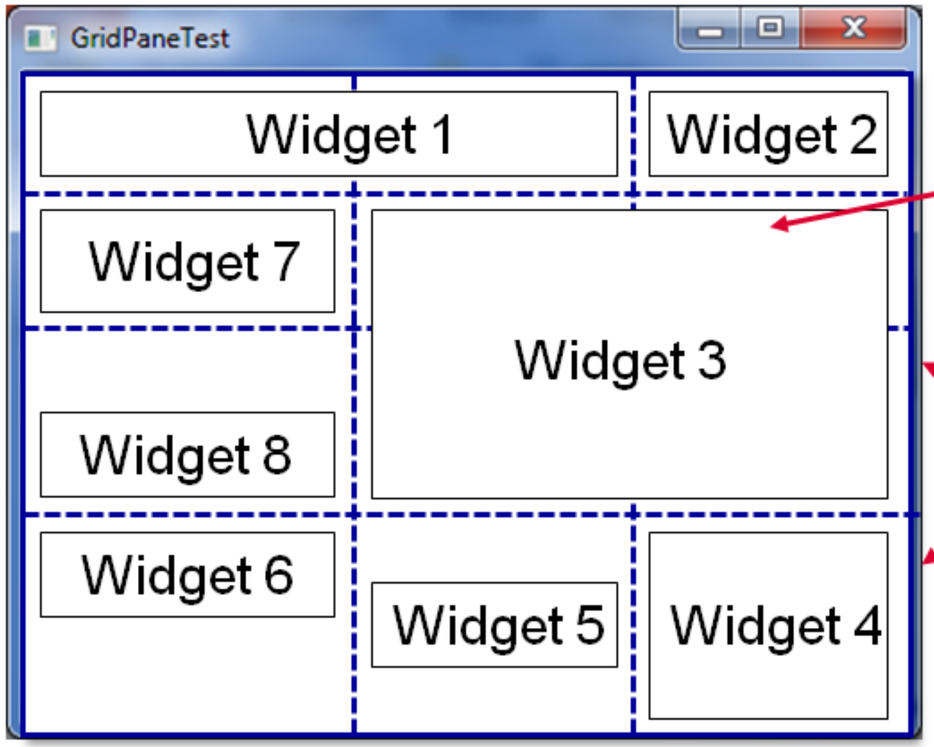
Java FX – GridPane

- GridPane



GridPane

- Sehr flexibel aber auch komplex
- Verwendung bei SceneBuilder (später)



Widgets dürfen sich über mehrere Zeilen und/oder Spalten erstrecken.

Zeilen dürfen unterschiedliche Höhen, Spalte unterschiedliche Breiten besitzen.

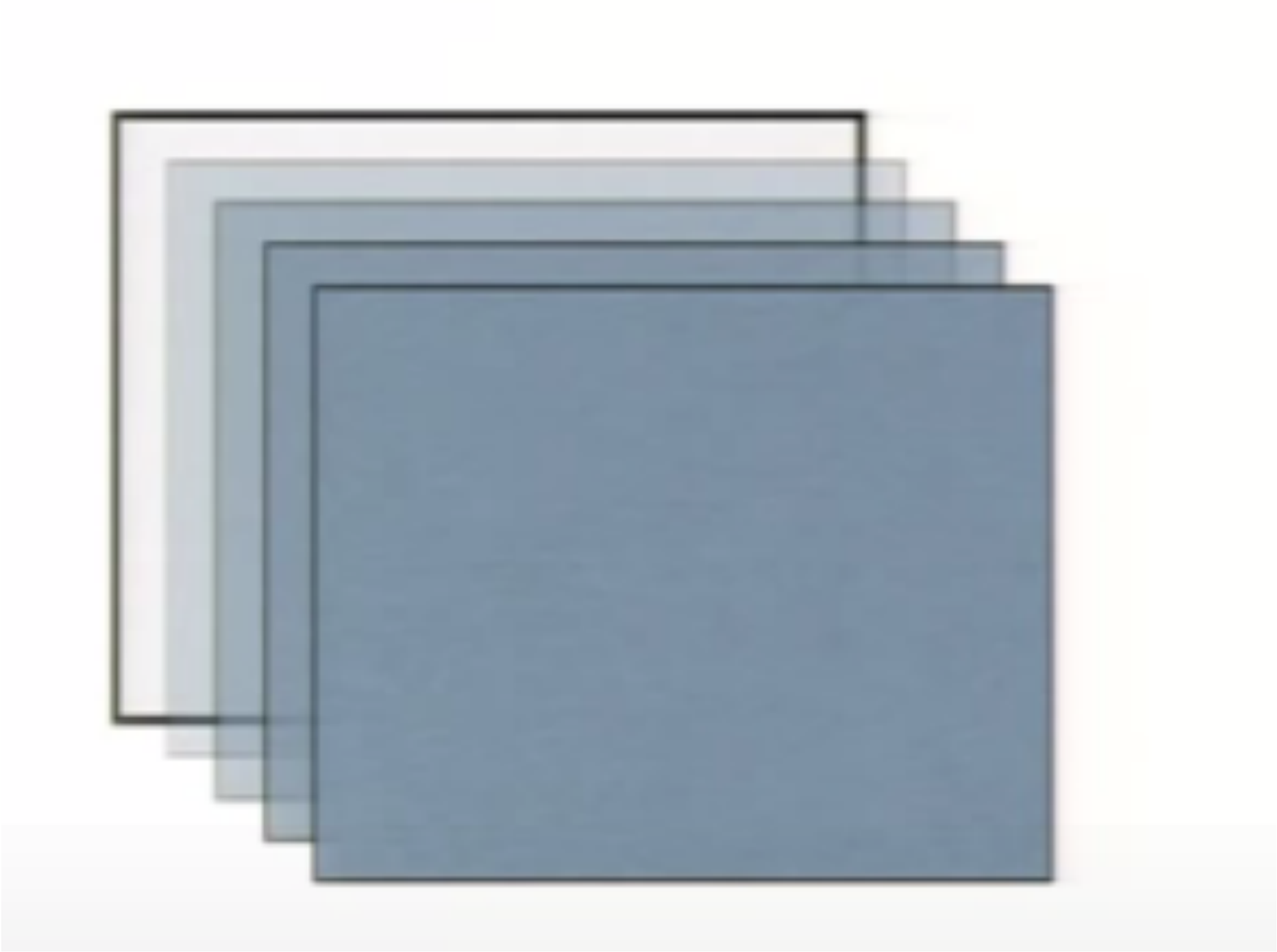
Die Größen der Zellen werden aus den bevorzugten Größen der Widgets berechnet.

Java FX – AnchorPane

- AnchorPane

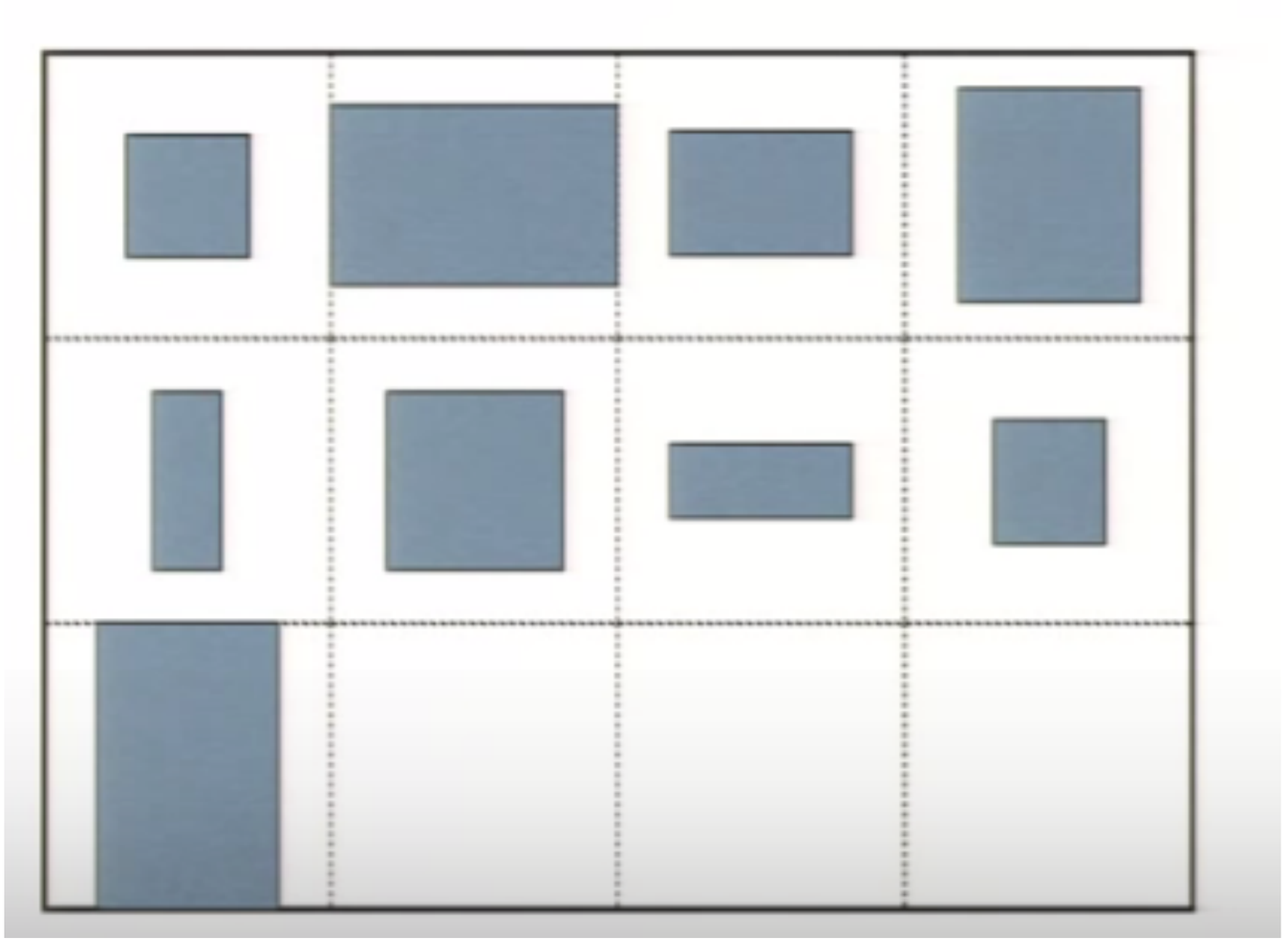
Java FX – StackPane

- StackPane

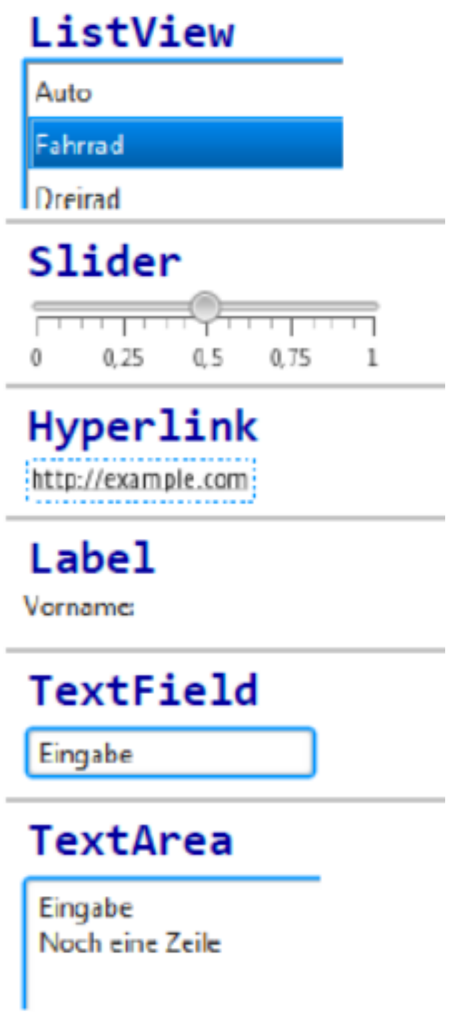


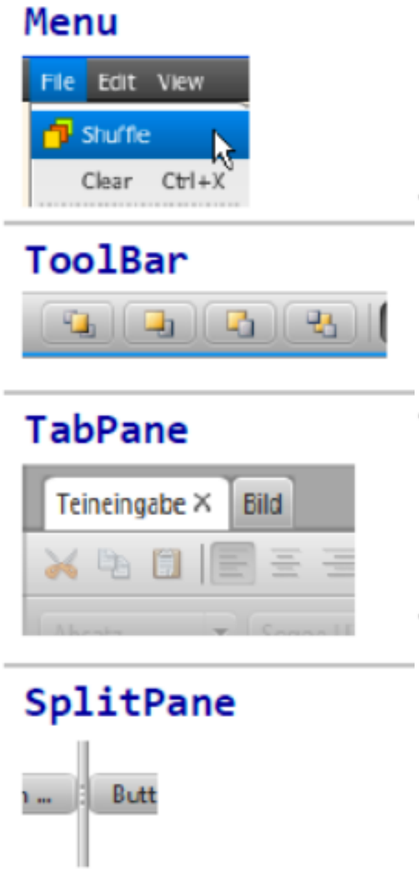
Java FX – TilePane

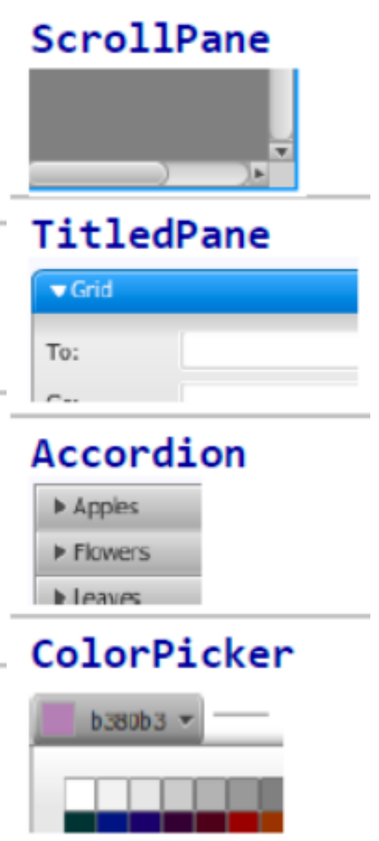
- TilePane

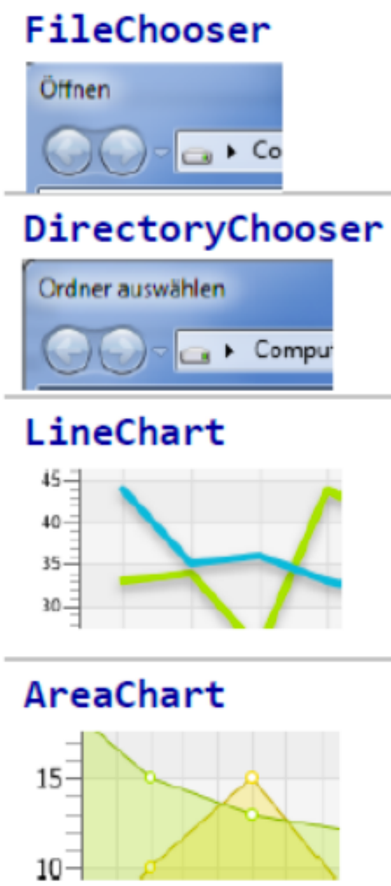


Beispiele für UI-Controls in JavaFX









Quelle: https://www.schmiedecke.info/Prg1/Konzepte/VL19-%20JAVAFX%20Panes%20and%20Controls.pdf

Button mit Eventhandling

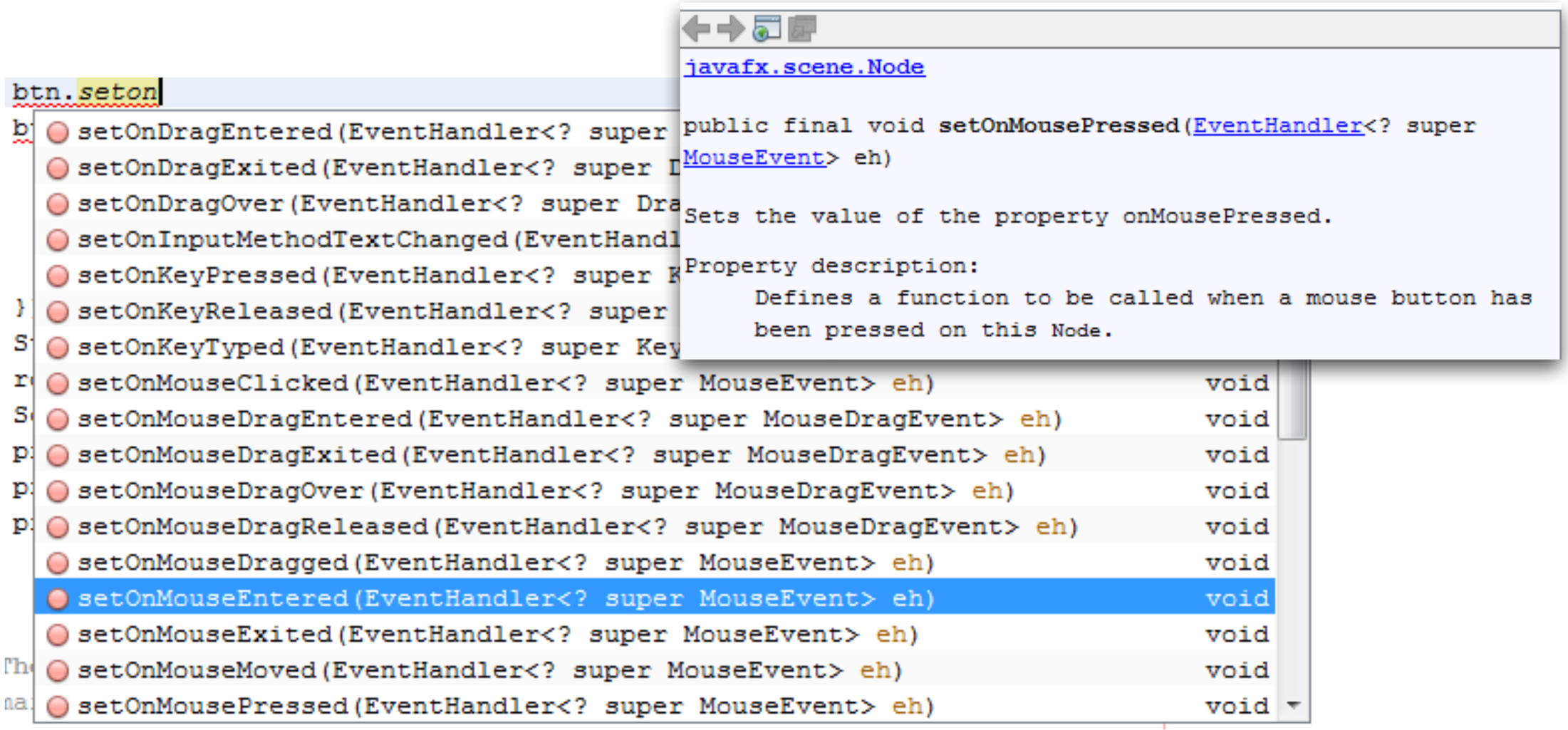
- javafx.scene.control.Button;
- Observer anmelden (registrieren) mit setOn…()
  - setOnAction —> ActionEventHandler
  - Generische Implementierung
- Implementierung der Behandlungsroutine
  - Erzeugung und Instanzierung einer anonymen inneren Klasse

```
Button btn = new Button();
btn.setText("Say 'Hello World'");
btn.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        System.out.println("Hello World!");
    }
});
```

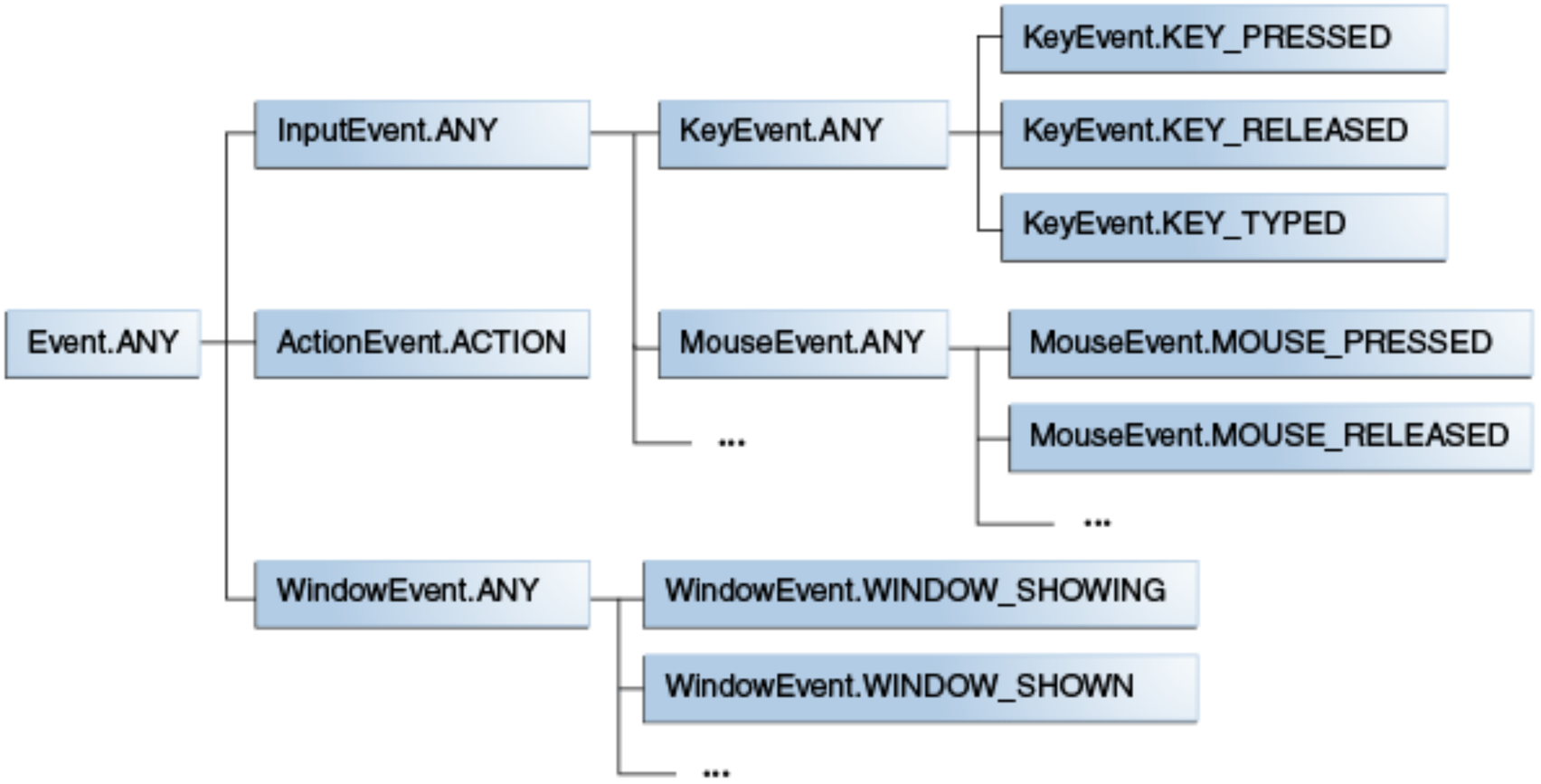
- -Alternativ: Lambda-Ausdruck

```
btn.setOnAction((event) -> {
    System.out.println("Hello World!");
});
```

Vielfältige Ereignisse sind registrierbar



Hierarchie der Events



Oracle-Unterlagen zum Einsatz der Controls (outdated, but ok)

https://docs.oracle.com/javafx/2/ui_controls/jfxpub-ui_controls.htm

- JavaFX UI Controls
- Label
- Button
- Radio Button
- Toggle Button
- Checkbox
- Choice Box
- Text Field
- Password Field
- Scroll Bar
- Scroll Pane
- List View
- Table View
- Tree View
- Combo Box
- Separator
- Slider
- Progress Bar and Progress Indicator
- Hyperlink
- Tooltip
- HTML Editor
- Titled Pane and Accordion
- Menu
- Color Picker
- Pagination Control
- File Chooser
- Customization of UI Controls

Neuere Ressourcen (openJFX)

- **Introduction to FXML**
  https://openjfx.io/javadoc/11/javafx.fxml/javafx/fxml/doc-files/introduction_to_fxml.html
- **JavaFX CSS Reference Guide**
  https://openjfx.io/javadoc/11/javafx.graphics/javafx/scene/doc-files/cssref.html
- **openJFX am OpenJDKwiki**
  https://wiki.openjdk.java.net/display/OpenJFX
- https://www.schmiedecke.info/Prg1/Konzepte/VL18-GUIs-mit-JavaFX-Intro.pdf
- https://www.schmiedecke.info/Prg1/Konzepte/VL19-%20JAVAFX%20Panes%20and%20Controls.pdf
- https://www.schmiedecke.info/Prg1/Konzepte/VL20-%20JAVAFX-Interaktion.pdf
- https://www.schmiedecke.info/Prg1/Konzepte/VL21-%20JAVAFX-Navigation.pdf

HTL LEONDING

Noch
Fragen?

HTL LEONDING

Schön, hier zu lernen