



# 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

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
- com.atlassian.maven.archetypes:jira-plugin-archetype
- com.rfc.maven.archetypes:...
- de.akquinet.jbosscc:jboss-archetype
- net.databinder:data-archetype
- net.liftweb.lift-archetype
- net.liftweb.lift-archetype
- net.sf.maven-har:maven-archetype
- net.sf.maven-sar:maven-archetype
- org.apache.camel.archetypes:...
- org.apache.cocoon:cocoon-archetype
- org.apache.cocoon:cocoon-archetype
- org.apache.cocoon:cocoon-archetype
- org.apache.cocoon:cocoon-archetype
- org.apache.maven.archetypes:...

? 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 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

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

Beachte: Die Packages der  
Klasse sind anzugeben

# HelloFX.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;

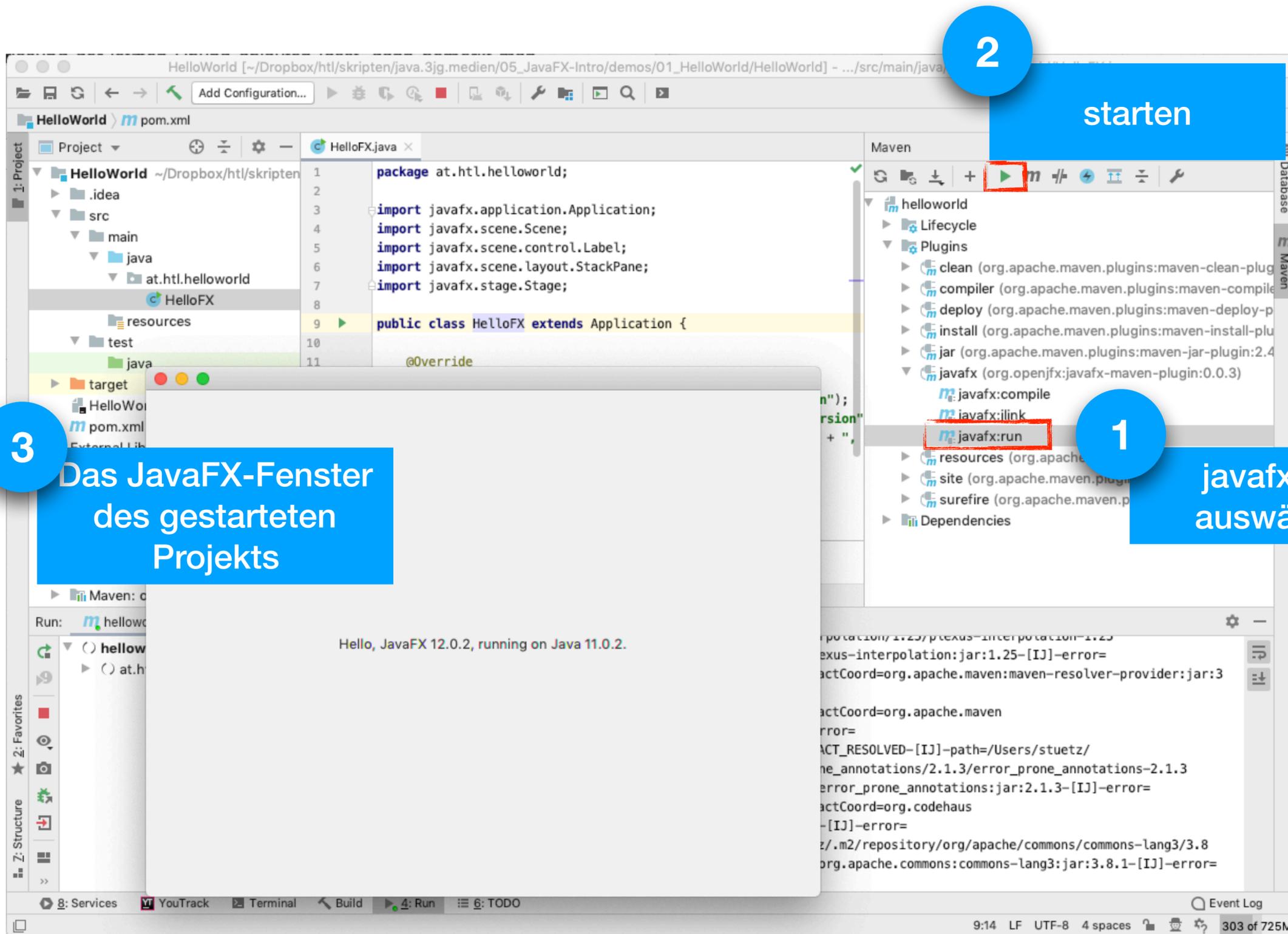
public class HelloFX extends Application {

    @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

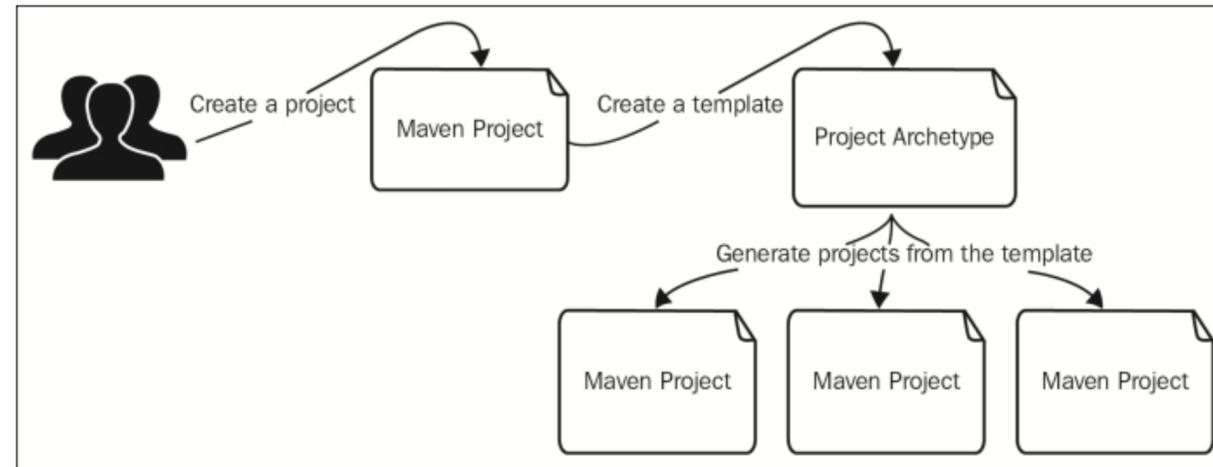
1

javafx:run  
auswählen

3

Das JavaFX-Fenster  
des gestarteten  
Projekts

Hello, JavaFX 12.0.2, running on Java 11.0.2.



*empfohlene  
Vorgehensweise*

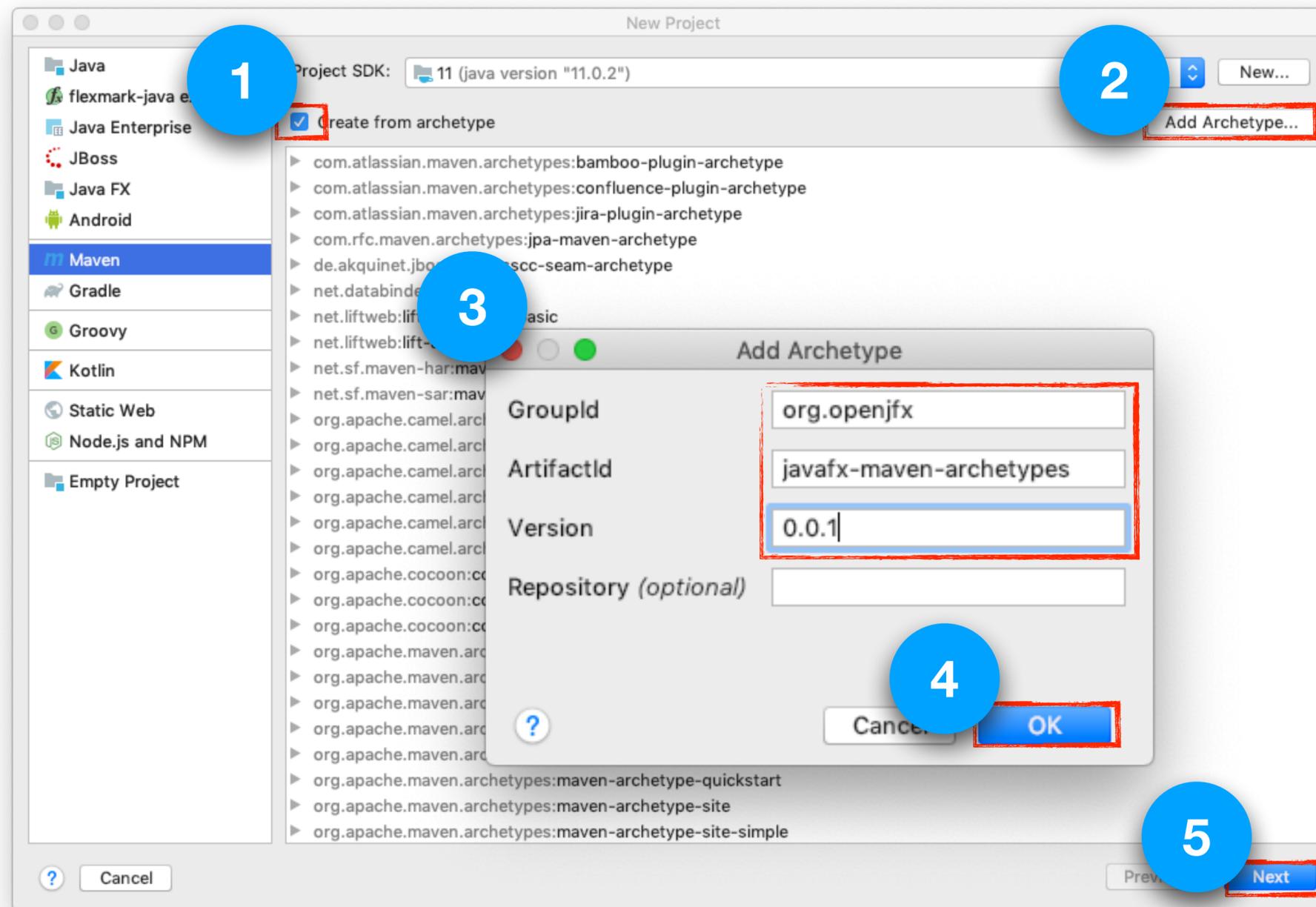
# Erstellung eines Projekts mit einem Archetype (Projektvorlage)

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

[https://youtu.be/rtdK8zWr\\_w](https://youtu.be/rtdK8zWr_w)

<https://youtu.be/Ri6No63fl-A>

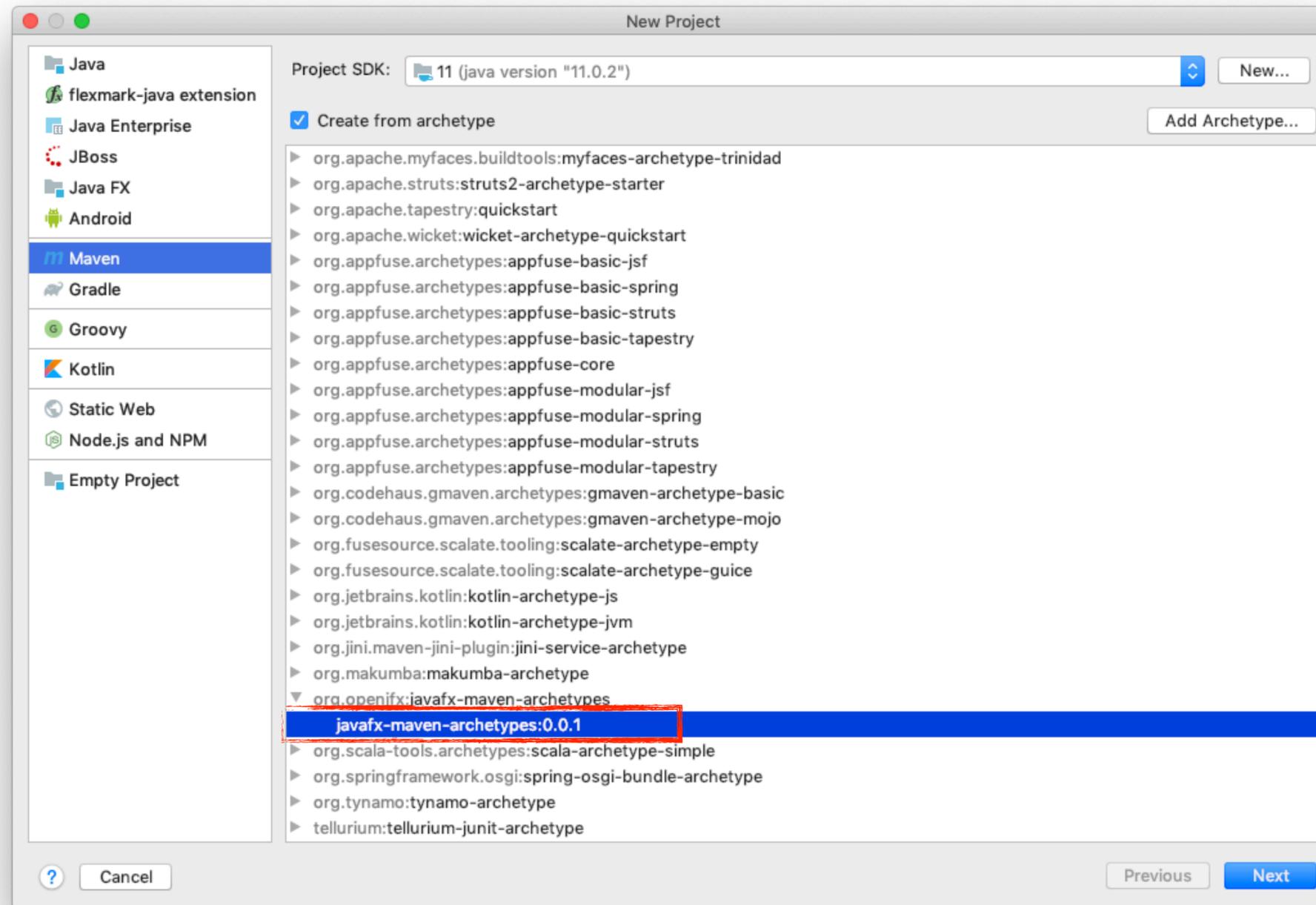
# Download des Archetypes in das lokale Repo



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

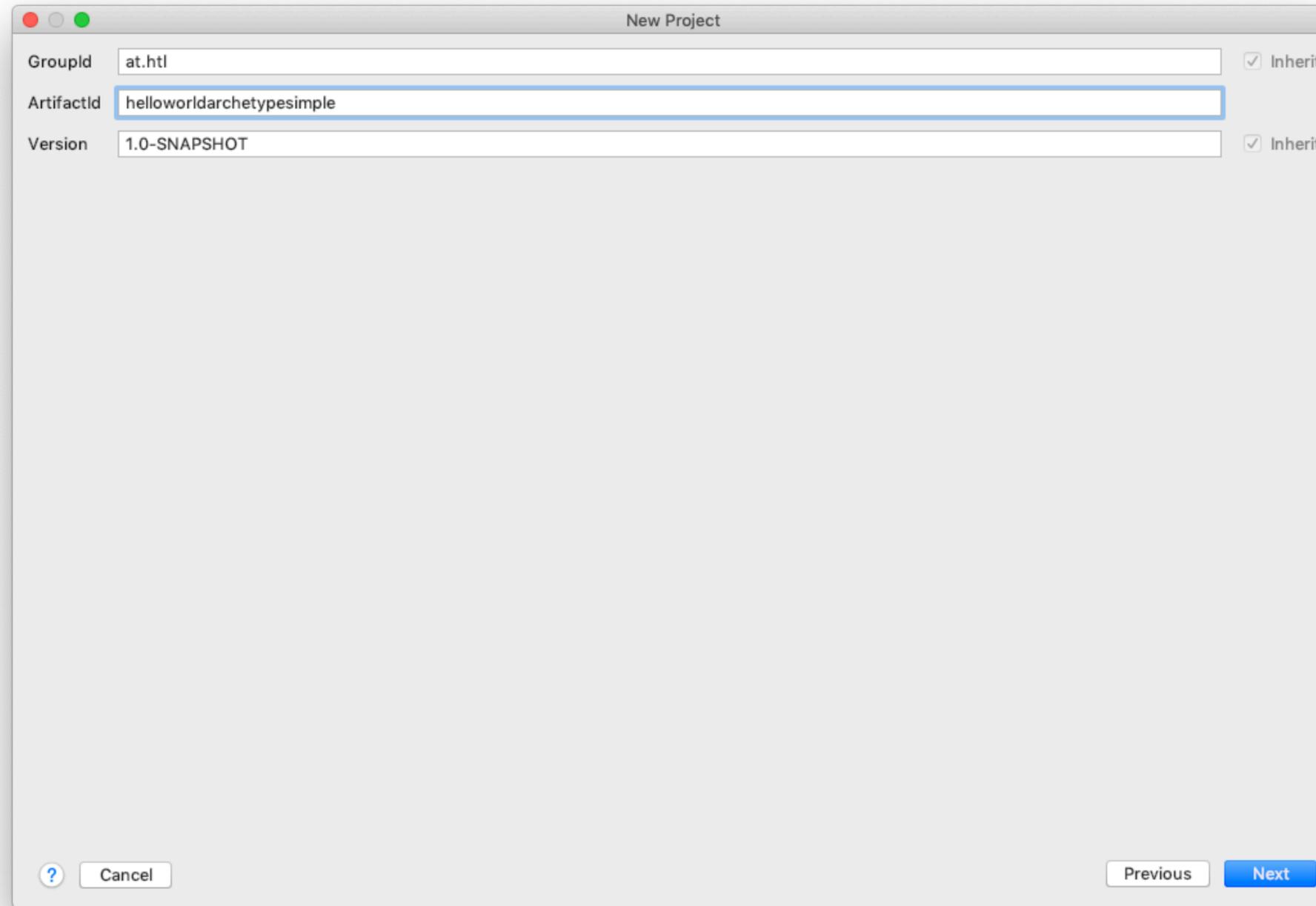
<https://openjfx.io/openjfx-docs/#maven>

# Archetype auswählen



# **Erstellung eines JavaFX- Projekts ohne FXML (simple)**

# GroupId, ArtifactId

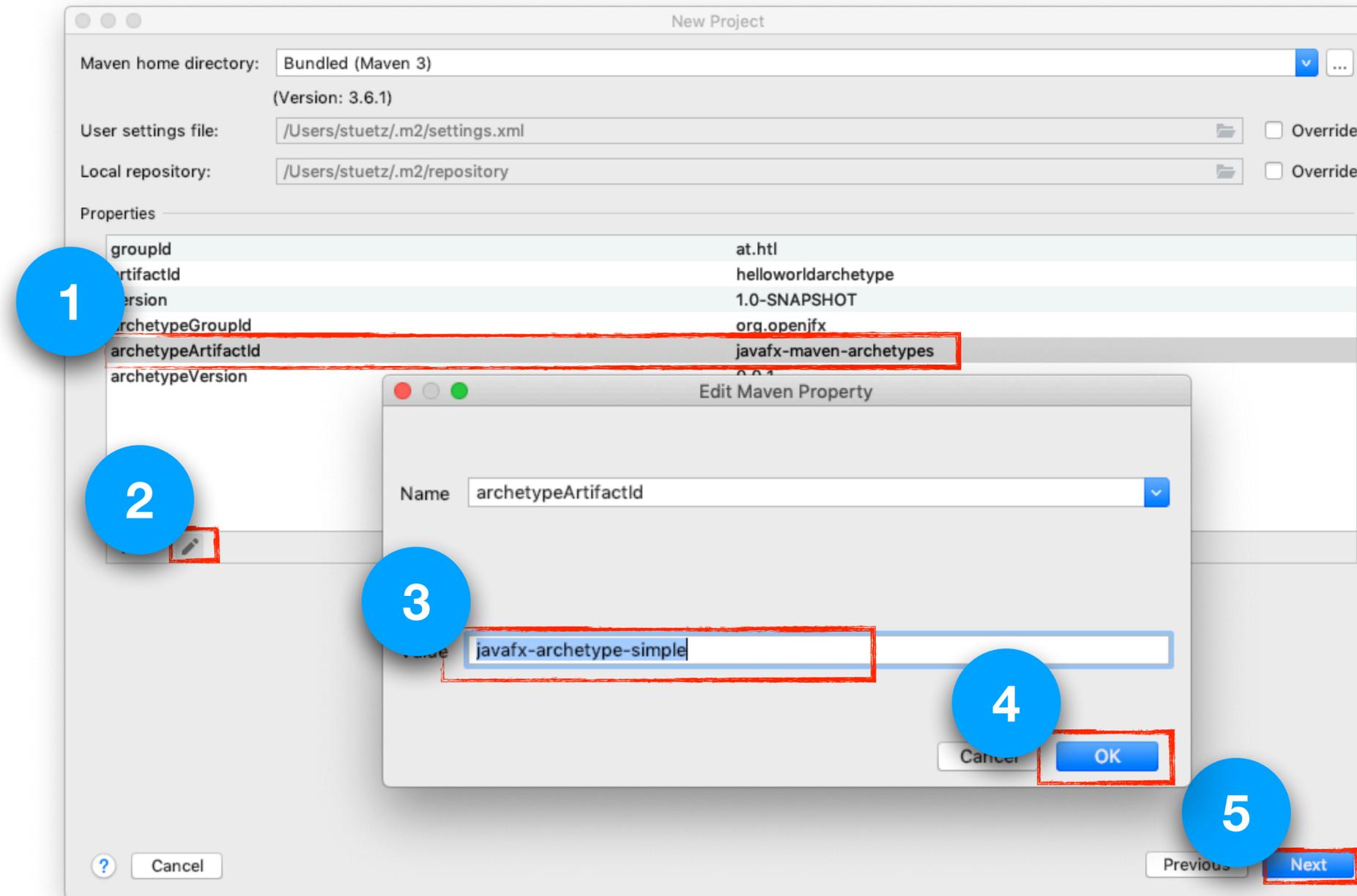


The image shows a 'New Project' dialog box with the following fields and options:

Field	Value	Inherit
GroupId	at.htl	<input checked="" type="checkbox"/>
ArtifactId	helloworldarchetypesimple	<input type="checkbox"/>
Version	1.0-SNAPSHOT	<input checked="" type="checkbox"/>

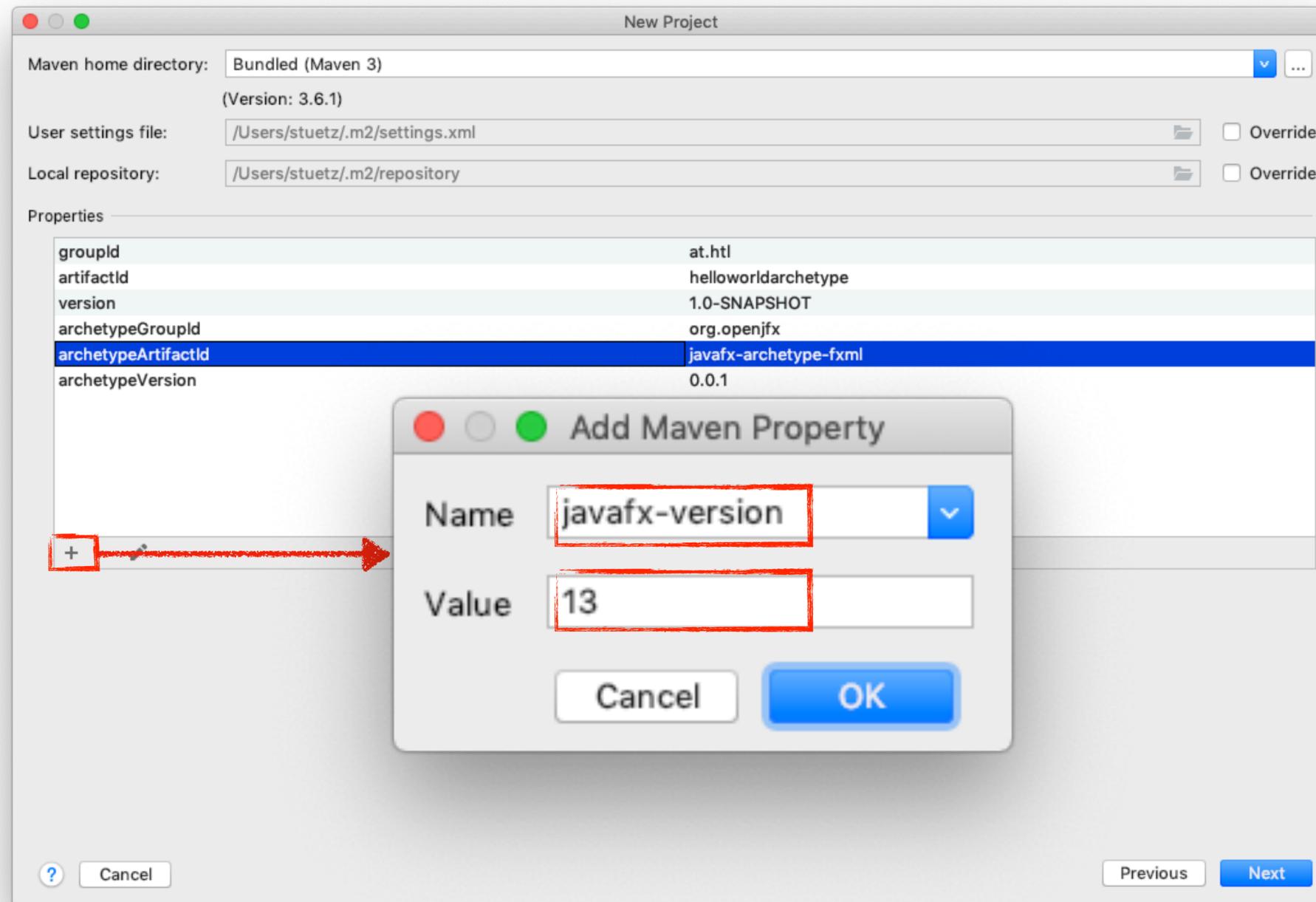
Buttons at the bottom: ? (help), Cancel, Previous, Next.

# 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



groupId	at.html
artifactId	helloworldarchetype
version	1.0-SNAPSHOT
archetypeGroupId	org.openjfx
archetypeArtifactId	javafx-archetype-fxml
archetypeVersion	0.0.1
javafx-version	12

# Archetype konfiguriert

New Project

Maven home directory:  ▼ ...  
(Version: 3.6.1)

User settings file:  📁  Override

Local repository:  📁  Override

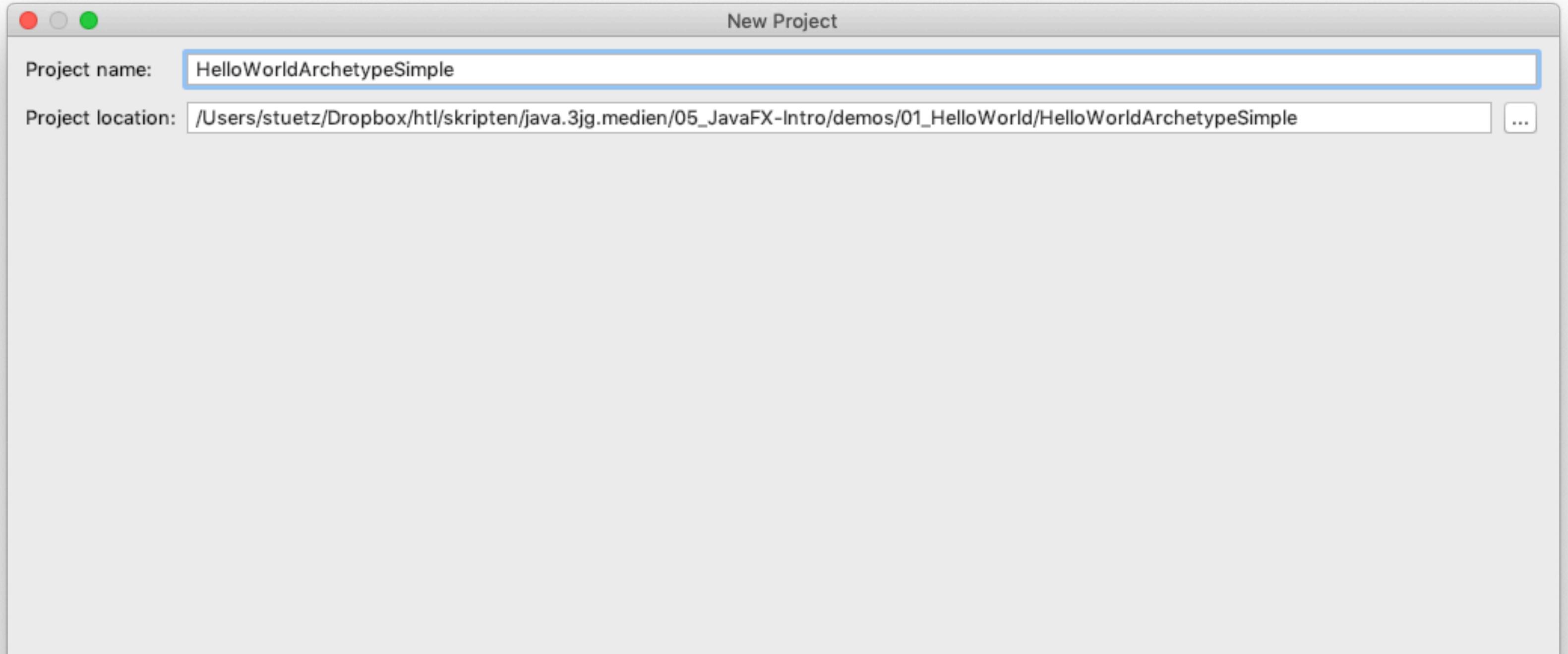
Properties

groupId	at.htl
artifactId	helloworldarchetypesimple
version	1.0-SNAPSHOT
archetypeGroupId	org.openjfx
archetypeArtifactId	javafx-archetype-simple
archetypeVersion	0.0.1
<b>javafx-version</b>	<b>13</b>

+ - ✎

?

# Projektname und Speicherort festlegen

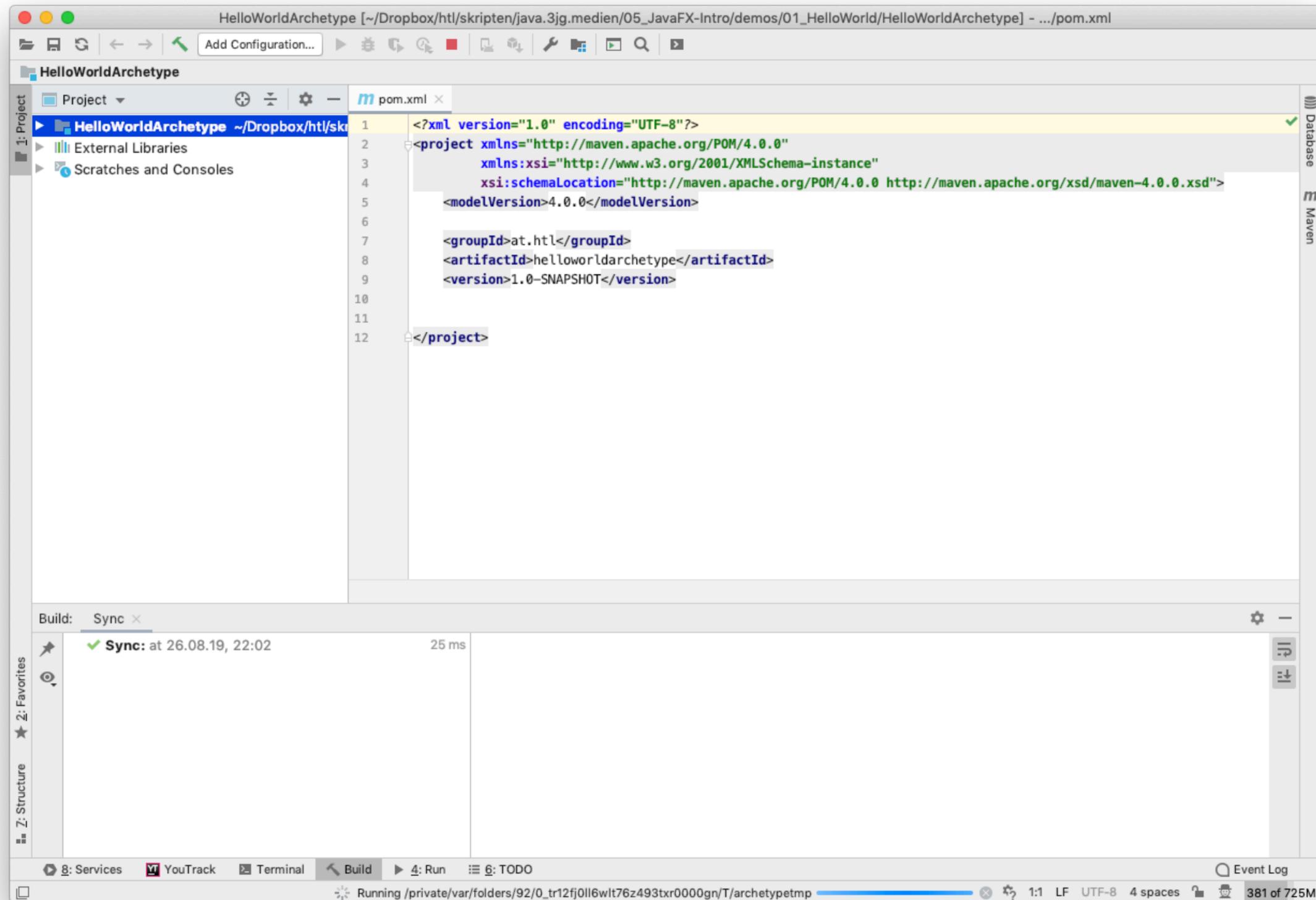


The image shows a 'New Project' dialog box from an IDE. It has a title bar with three window control buttons (red, yellow, green) on the left and the text 'New Project' in the center. Below the title bar, there are two input fields. The first is labeled 'Project name:' and contains the text 'HelloWorldArchetypeSimple'. The second is labeled 'Project location:' and contains the path '/Users/stuetz/Dropbox/htl/skripten/java.3jg.medien/05\_JavaFX-Intro/demos/01\_HelloWorld/HelloWorldArchetypeSimple'. To the right of the second input field is a small button with three dots '...'. The background of the dialog is a light gray color.

Project name: HelloWorldArchetypeSimple

Project location: /Users/stuetz/Dropbox/htl/skripten/java.3jg.medien/05\_JavaFX-Intro/demos/01\_HelloWorld/HelloWorldArchetypeSimple ...

# Der Download (Sync) wird durchgeführt



HelloWorldArchetype [~/Dropbox/htl/skripten/java.3jg.medien/05\_JavaFX-Intro/demos/01\_HelloWorld/HelloWorldArchetype] - .../pom.xml

1: Project  
Project  
HelloWorldArchetype ~/Dropbox/htl/sk...  
External Libraries  
Scratches and Consoles

```
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
3   <modelVersion>4.0.0</modelVersion>
4   <groupId>at.htl</groupId>
5   <artifactId>helloworldarchetype</artifactId>
6   <version>1.0-SNAPSHOT</version>
7   <properties>
8     <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
9     <maven.compiler.source>11</maven.compiler.source>
10    <maven.compiler.target>11</maven.compiler.target>
11  </properties>
12  <dependencies>
13    <dependency>
14      <groupId>org.openjfx</groupId>
15      <artifactId>javafx-controls</artifactId>
16      <version>11.0.2</version>
17    </dependency>
18  </dependencies>
19
20
21
22
23
24
```

Build: Sync ×  
✓ Sync: at 26.08.19, 22:20

2: Favorites  
Z: Structure

Database  
Maven

**Leider sind die Versionsnummern nicht am neuesten Stand.**

**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

# Versionsnummern aktualisieren

The screenshot shows an IDE window titled 'HelloWorldArchetypeSimple' with a 'pom.xml' file open. The XML content is as follows:

```
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://maven.apache.org/xsi:schemaLocation" http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
2   <modelVersion>4.0.0</modelVersion>
3   <groupId>at.htl</groupId>
4   <artifactId>helloworldarchetypesimple</artifactId>
5   <version>1.0-SNAPSHOT</version>
6   <properties>
7     <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
8     <maven.compiler.source>11</maven.compiler.source>
9     <maven.compiler.target>11</maven.compiler.target>
10  </properties>
11  <dependencies>
12    <dependency>
13      <groupId>org.openjfx</groupId>
14      <artifactId>javafx-controls</artifactId>
15      <version>13</version>
16    </dependency>
17  </dependencies>
18  <build>
19    <plugins>
20      <plugin>
21        <groupId>org.apache.maven.plugins</groupId>
22        <artifactId>maven-compiler-plugin</artifactId>
23        <version>3.8.1</version>
24        <configuration>
25          <release>11</release>
26        </configuration>
27      </plugin>
28      <plugin>
29        <groupId>org.openjfx</groupId>
30        <artifactId>javafx-maven-plugin</artifactId>
31        <version>0.0.3</version>
32        <configuration>
33          <mainClass>at.htl.App</mainClass>
34        </configuration>
35      </plugin>
36    </plugins>
37  </build>
38 </project>
```

The IDE also shows a Maven build output window with the following content:

```
helloworldarchetypesimple
├── Lifecycle
├── Plugins
│   ├── clean (org.apache.maven.plugins:maven-clean-plugin:3.1.0)
│   ├── compiler (org.apache.maven.plugins:maven-compiler-plugin:3.8.1)
│   ├── deploy (org.apache.maven.plugins:maven-deploy-plugin:3.0.0)
│   ├── install (org.apache.maven.plugins:maven-install-plugin:3.0.0)
│   ├── jar (org.apache.maven.plugins:maven-jar-plugin:3.2.0)
│   ├── javafx (org.openjfx:javafx-maven-plugin:0.0.3)
│   ├── javafx:compile
│   ├── javafx:jlink
│   └── javafx:run
│   ├── resources (org.apache.maven.plugins:maven-resources-plugin:3.2.0)
│   ├── site (org.apache.maven.plugins:maven-site-plugin:3.12.1)
│   └── surefire (org.apache.maven.plugins:maven-surefire-plugin:3.0.0)
└── Dependencies
```

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

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.8.1</version>
      <configuration>
        <release>11</release>
      </configuration>
    </plugin>
    <plugin>
      <groupId>org.openjfx</groupId>
      <artifactId>javafx-maven-plugin</artifactId>
      <version>0.0.3</version>
      <configuration>
        <mainClass>at.htl.App</mainClass>
      </configuration>
    </plugin>
  </plugins>
</build>
```

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

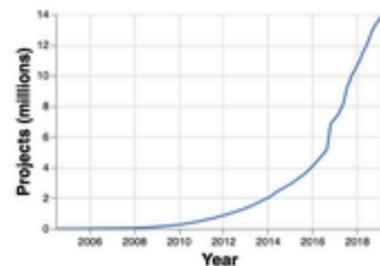
# https://mavenrepository.com

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.

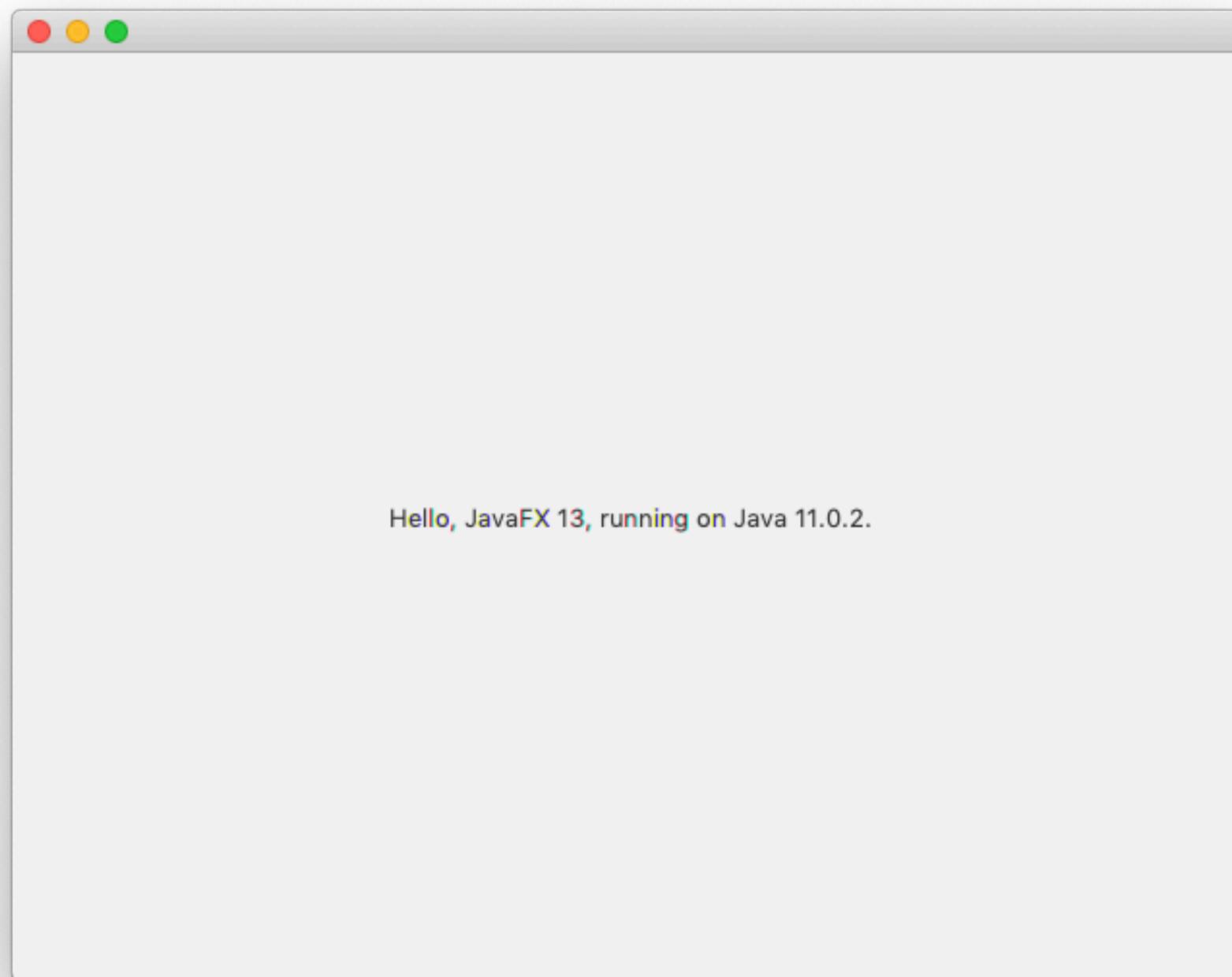
License	Apache 2.0
Categories	Maven Plugins
Tags	plugin compiler build build-system maven apache
Used By	69 artifacts

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

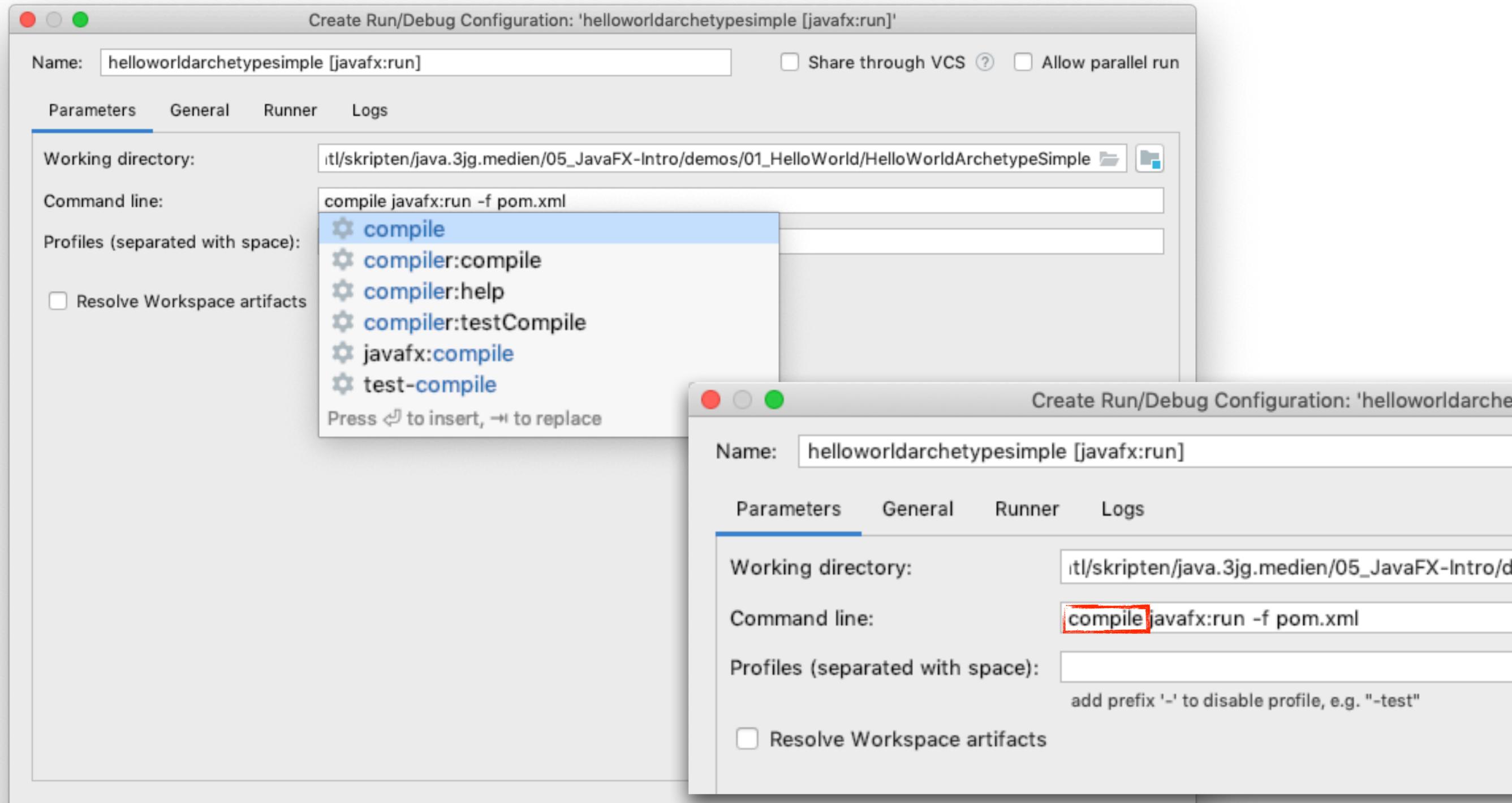
Das Projekt muss nach Aktualisierungen immer neu kompiliert werden. Um diese beiden Arbeitsschritte (javafx:compile und javafx:run) in einem Schritt zusammenzulegen, wird eine eigene Konfiguration erstellt:

The screenshot shows an IDE window titled "HelloWorldArchetype" with a pom.xml file open. The pom.xml content is as follows:

```
<?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/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>12.0.2</version>
    </dependency>
    <dependency>
      <groupId>org.openjfx</groupId>
      <artifactId>javafx-xml</artifactId>
      <version>12.0.2</version>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.1</version>
        <configuration>
```

The Maven tool window on the right shows the project structure with a custom configuration "javafx:run" highlighted in red. A context menu is open over this configuration, with the option "Create 'helloworldarchetype ...' ..." selected and highlighted in blue. Other options in the menu include "Run Maven Build", "Run 'helloworldarchetype ...'", "Debug 'helloworldarchetype ...'", "Execute Before Build", "Execute After Build", "Execute Before Rebuild", "Execute After Rebuild", "Execute Before Run/Debug...", and "Assign Shortcut...".

# Compile eintragen



# Ausführen: compile + run

The screenshot displays an IDE window for a project named "HelloWorldArchetypeSimple". The main editor shows the XML content of a `pom.xml` file, with the `<plugins>` section highlighted in yellow. The `javafx` plugin is configured with the following settings:

```
<plugin>  
  <groupId>org.openjfx</groupId>  
  <artifactId>javafx-controls</artifactId>  
  <version>11</version>  
</plugin>
```

The Maven tool window on the right shows the project structure, with the `javafx:run` goal selected under the `javafx` plugin. The `Run` button in the Maven toolbar is highlighted with a red box.

A small window in the foreground displays the output of the application:

```
Hello, JavaFX 13, running on Java 11.0.2.
```

The bottom status bar shows the command being executed: `Run: m helloworldarchetypesimple [org.openjfx:jav...]`. The bottom console shows the execution details for the `javafx:run` goal, including the classpath and the output of the `javafx:run` goal.

# Code der FX-Anwendung

```
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();

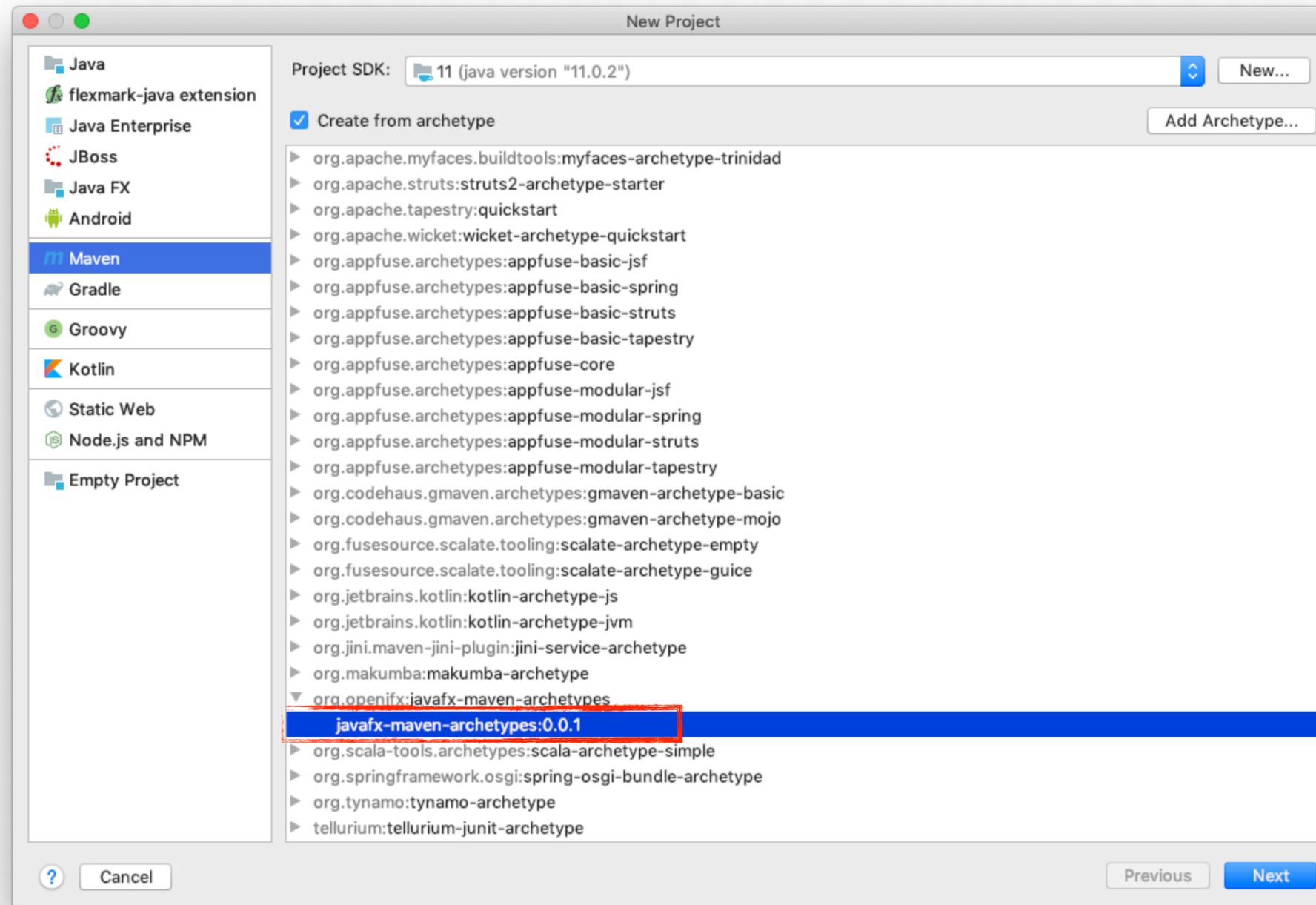
        var label = new Label("Hello, JavaFX "
            + javafxVersion
            + ", running on Java "
            + javaVersion + ".");
        var scene = new Scene(new StackPane(label), 640, 480);
        stage.setScene(scene);
        stage.show();
    }

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

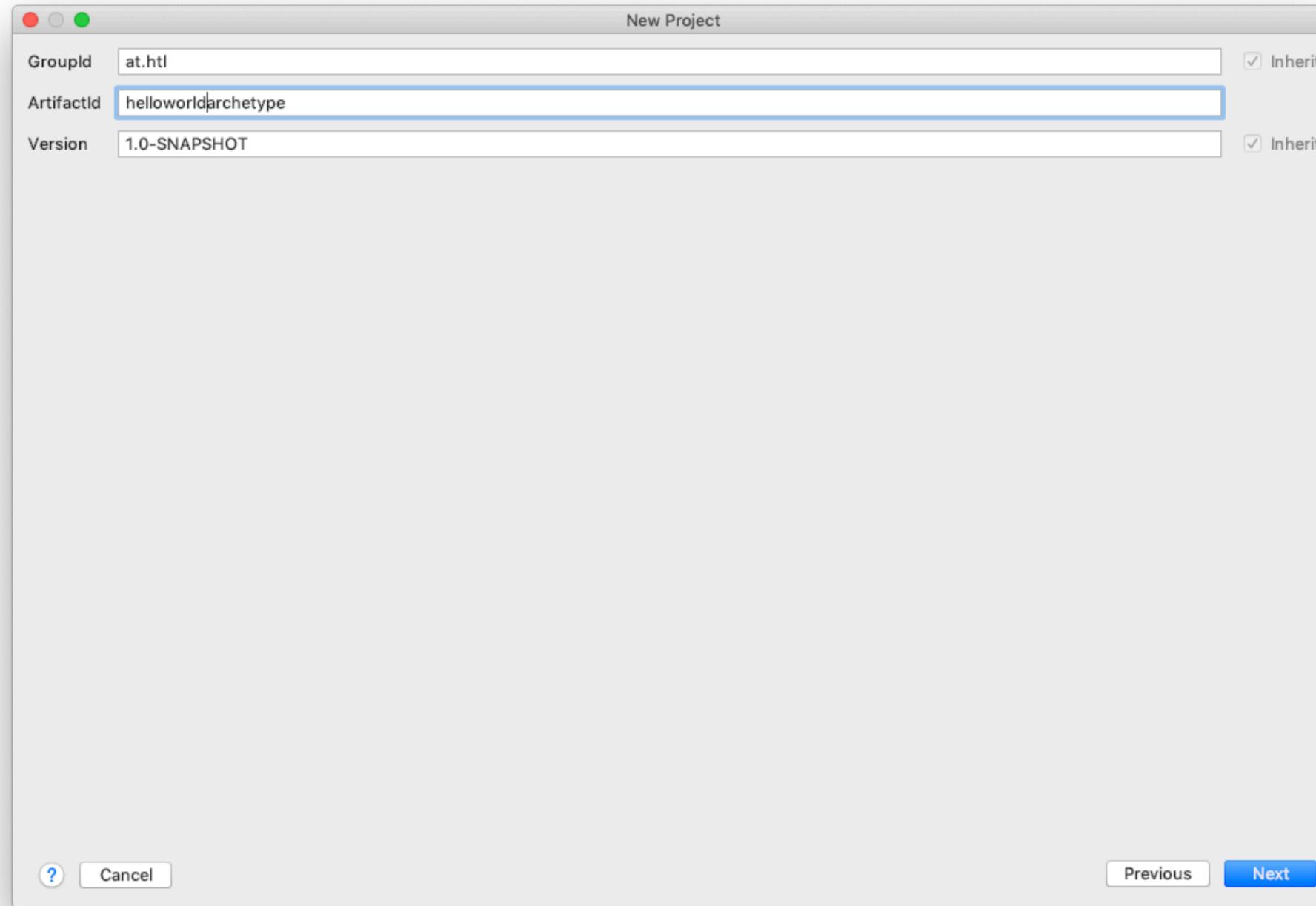
Wir besprechen später noch  
die Grundstruktur von FX-  
Applikationen

**Erstellung eines JavaFX-  
Projekts mit FXML (FXML)**

# Archetype auswählen



# GroupId, ArtifactId

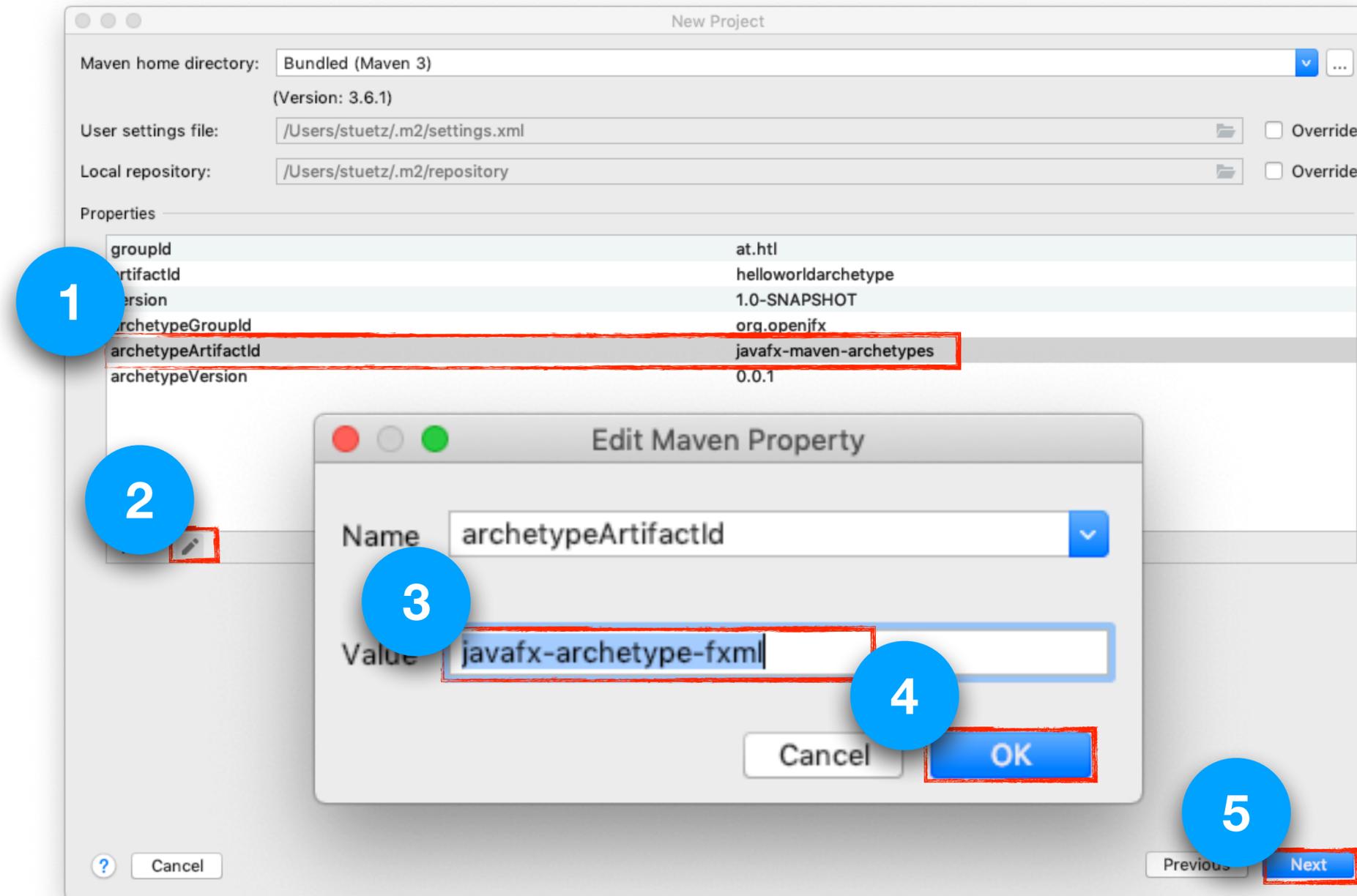


The image shows a 'New Project' dialog box with the following fields and options:

- GroupId:** at.html  Inherit
- ArtifactId:** helloworldarchetype
- Version:** 1.0-SNAPSHOT  Inherit

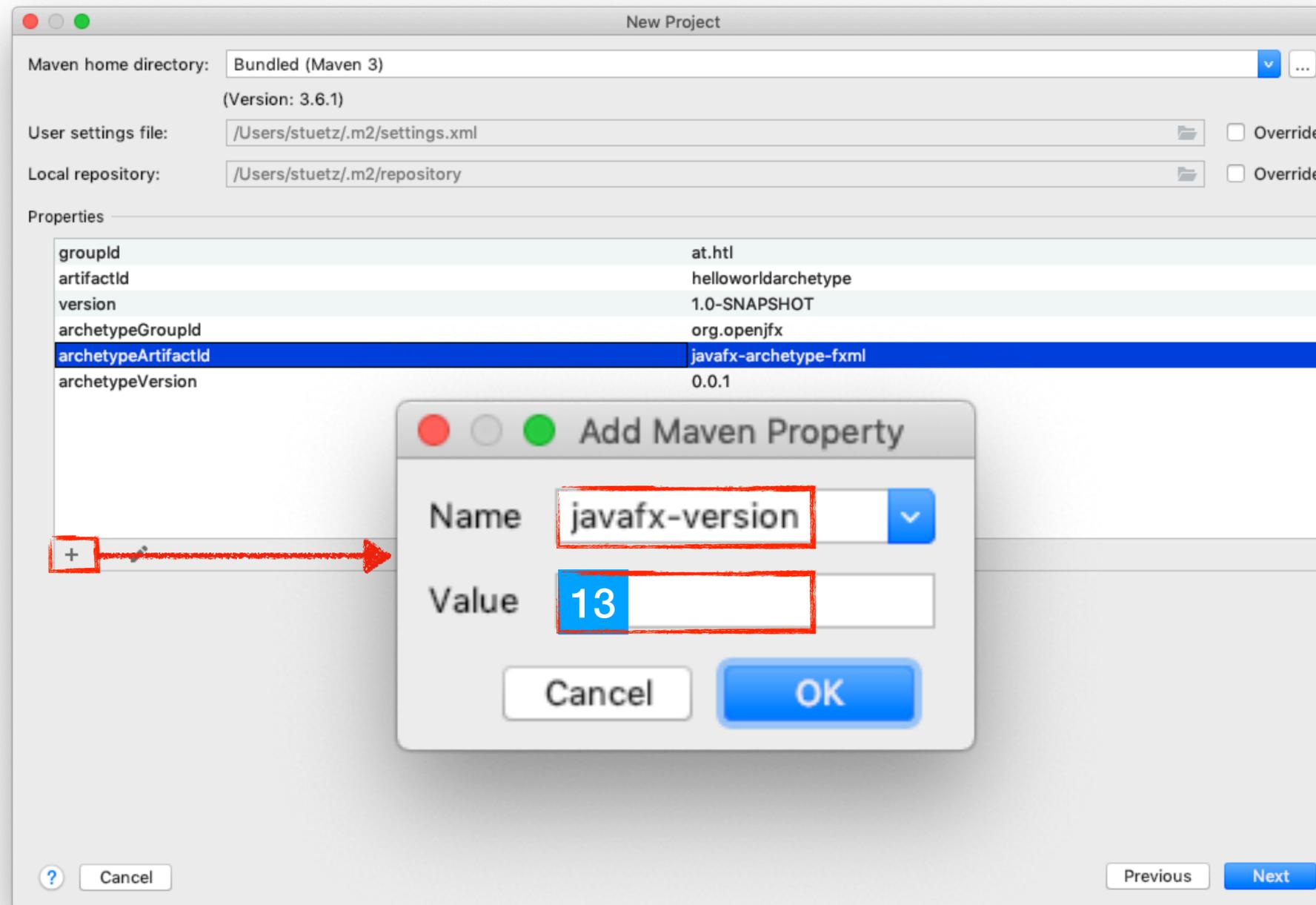
At the bottom of the dialog, there are four buttons: a help button (question mark icon), a 'Cancel' button, a 'Previous' button, and a 'Next' button.

# ArchetypeArtifactId eintragen



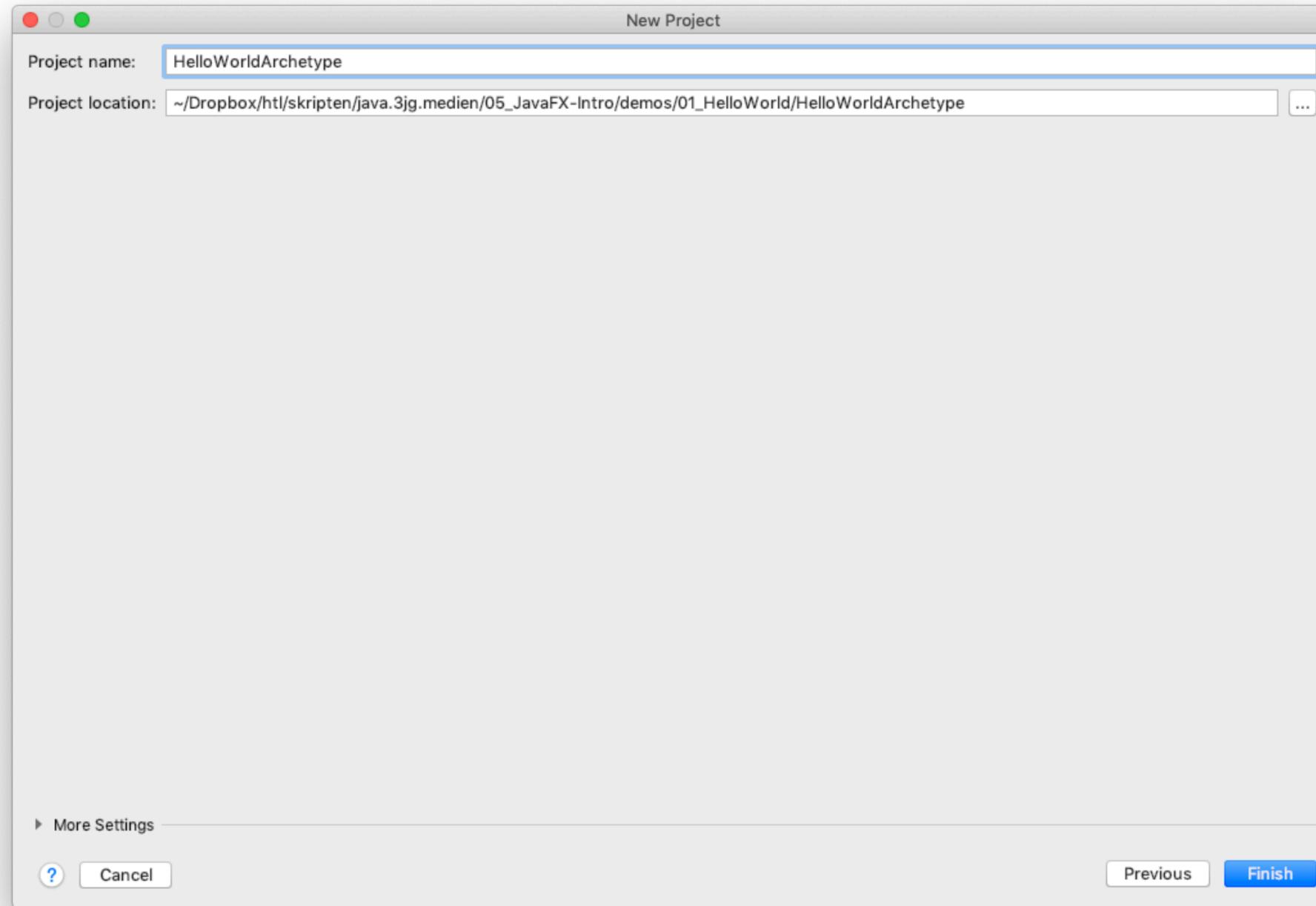
1. archetypeArtifactId auswählen
2. Edit  anklicken
3. Entweder „javafx-archetype-fxml“ eingeben oder falls fxml nicht gebraucht wird „javafx-archetype-simple“
4. OK
5. Next

# Einstellen der JavaFX-Version

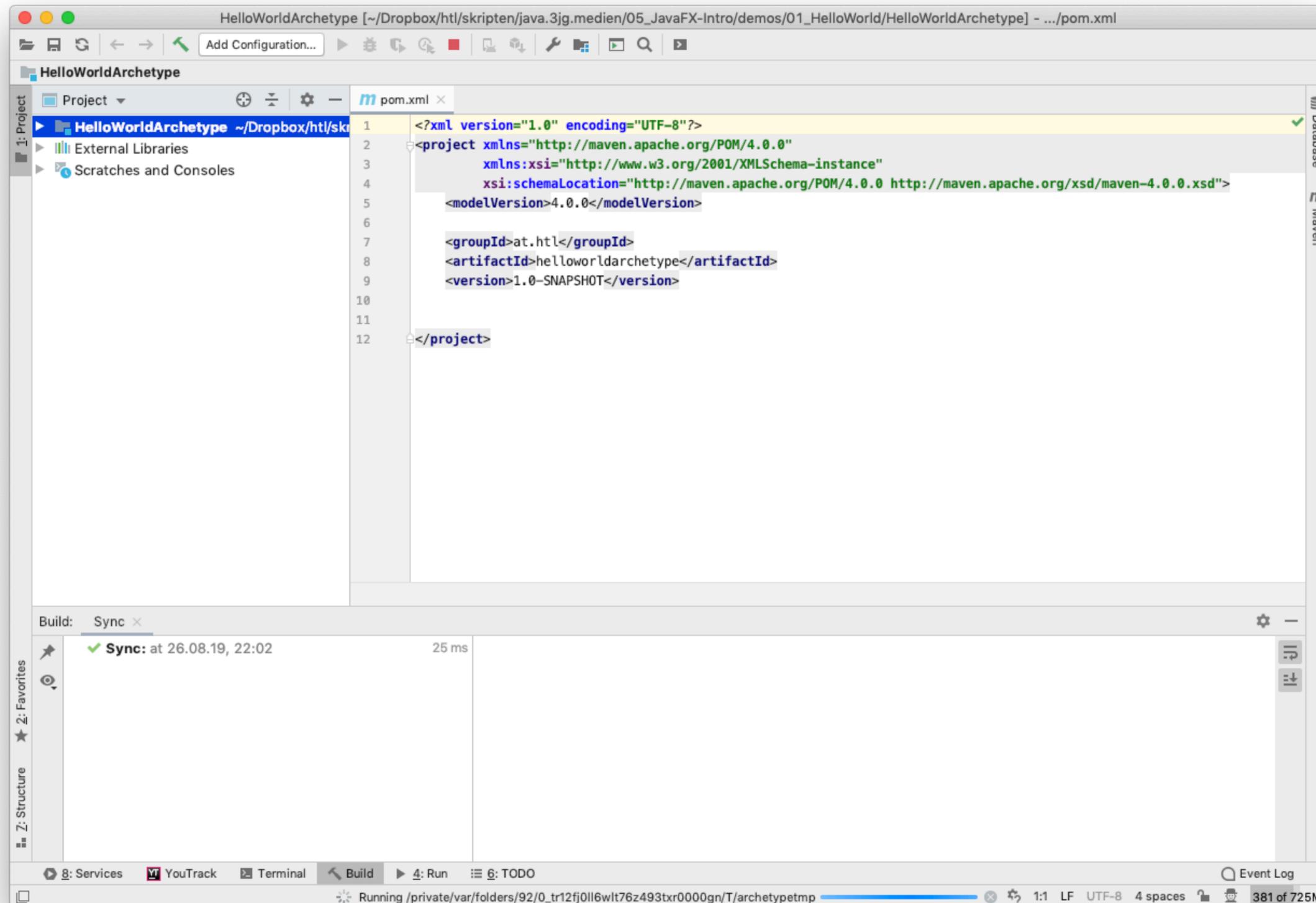


groupId	at.html
artifactId	helloworldarchetype
version	1.0-SNAPSHOT
archetypeGroupId	org.openjfx
archetypeArtifactId	javafx-archetype-fxml
archetypeVersion	0.0.1
javafx-version	12

# 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

1: Project

- Project
- 1: HelloWorldArchetype ~/Dropbox/htl/skripten/java.3jg.medien/05\_JavaFX-Intro/demos/01\_HelloWorld/HelloWorldArchetype
- External Libraries
- Scratches and Consoles

```
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
3   <modelVersion>4.0.0</modelVersion>
4   <groupId>at.htl</groupId>
5   <artifactId>helloworldarchetype</artifactId>
6   <version>1.0-SNAPSHOT</version>
7   <properties>
8     <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
9     <maven.compiler.source>11</maven.compiler.source>
10    <maven.compiler.target>11</maven.compiler.target>
11  </properties>
12  <dependencies>
13    <dependency>
14      <groupId>org.openjfx</groupId>
15      <artifactId>javafx-controls</artifactId>
16      <version>11.0.2</version>
17    </dependency>
18    <dependency>
19      <groupId>org.openjfx</groupId>
20      <artifactId>javafx-xml</artifactId>
21      <version>11.0.2</version>
22    </dependency>
23  </dependencies>
24  <build>
```

Database  
Maven

Build: Sync ×

✓ Sync: at 26.08.19, 22:20 1 s 925 ms

2: Favorites

7: Structure

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

**Maven projects need to be imported**  
Import Changes **Enable Auto-Import**

Leider sind die Versionsnummern nicht am neuesten Stand.

# Versionsnummern aktualisieren

The screenshot shows an IDE window titled "HelloWorldArchetype" with a pom.xml file open. The code is as follows:

```
5 <artifactId>helloworldarchetype</artifactId>
6 <version>1.0-SNAPSHOT</version>
7
8 <properties>
9   <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
10
11 </properties>
12 <dependencies>
13   <dependency>
14     <groupId>org.openjfx</groupId>
15     <artifactId>javafx-controls</artifactId>
16     <version>12.0.2</version>
17   </dependency>
18   <dependency>
19     <groupId>org.openjfx</groupId>
20     <artifactId>javafx-fxml</artifactId>
21     <version>12.0.2</version>
22   </dependency>
23 </dependencies>
24 <build>
25   <plugins>
26     <plugin>
27       <groupId>org.apache.maven.plugins</groupId>
28       <artifactId>maven-compiler-plugin</artifactId>
29       <version>3.8.1</version>
30       <configuration>
31         <release>11</release>
32       </configuration>
33     </plugin>
34     <plugin>
35       <groupId>org.openjfx</groupId>
36       <artifactId>javafx-maven-plugin</artifactId>
37       <version>0.0.3</version>
38       <configuration>
39         <mainClass>at.htl.App</mainClass>
40       </configuration>
41     </plugin>
42 </build>
```

The Maven tool window on the right shows the project structure with "javafx:run" selected.

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

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.8.1</version>
      <configuration>
        <release>11</release>
      </configuration>
    </plugin>
    <plugin>
      <groupId>org.openjfx</groupId>
      <artifactId>javafx-maven-plugin</artifactId>
      <version>0.0.3</version>
      <configuration>
        <mainClass>at.htl.App</mainClass>
      </configuration>
    </plugin>
  </plugins>
</build>
```

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

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

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

MVNREPOSITORY Search for groups, artifacts, categories Search

Home » org.openjfx » javafx-controls

**JavaFX Controls**  
JavaFX Controls

**License** GPL 2.0

**Used By** 125 artifacts

**Indexed Artifacts (15.0M)**

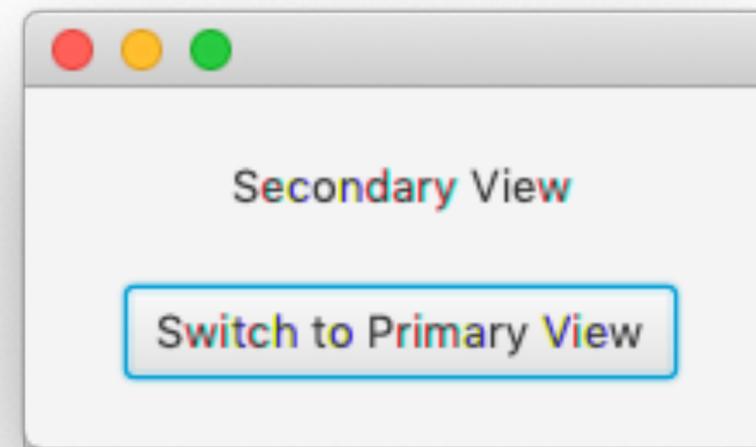
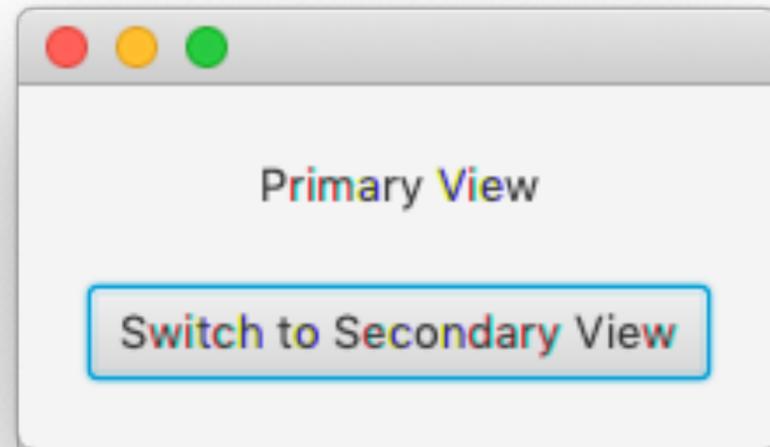
**Popular Categories**

- 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

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
<b>12.0.2</b>	Central	4	Jul, 2019

# Ergebnis



# Eigene Configuration erstellen

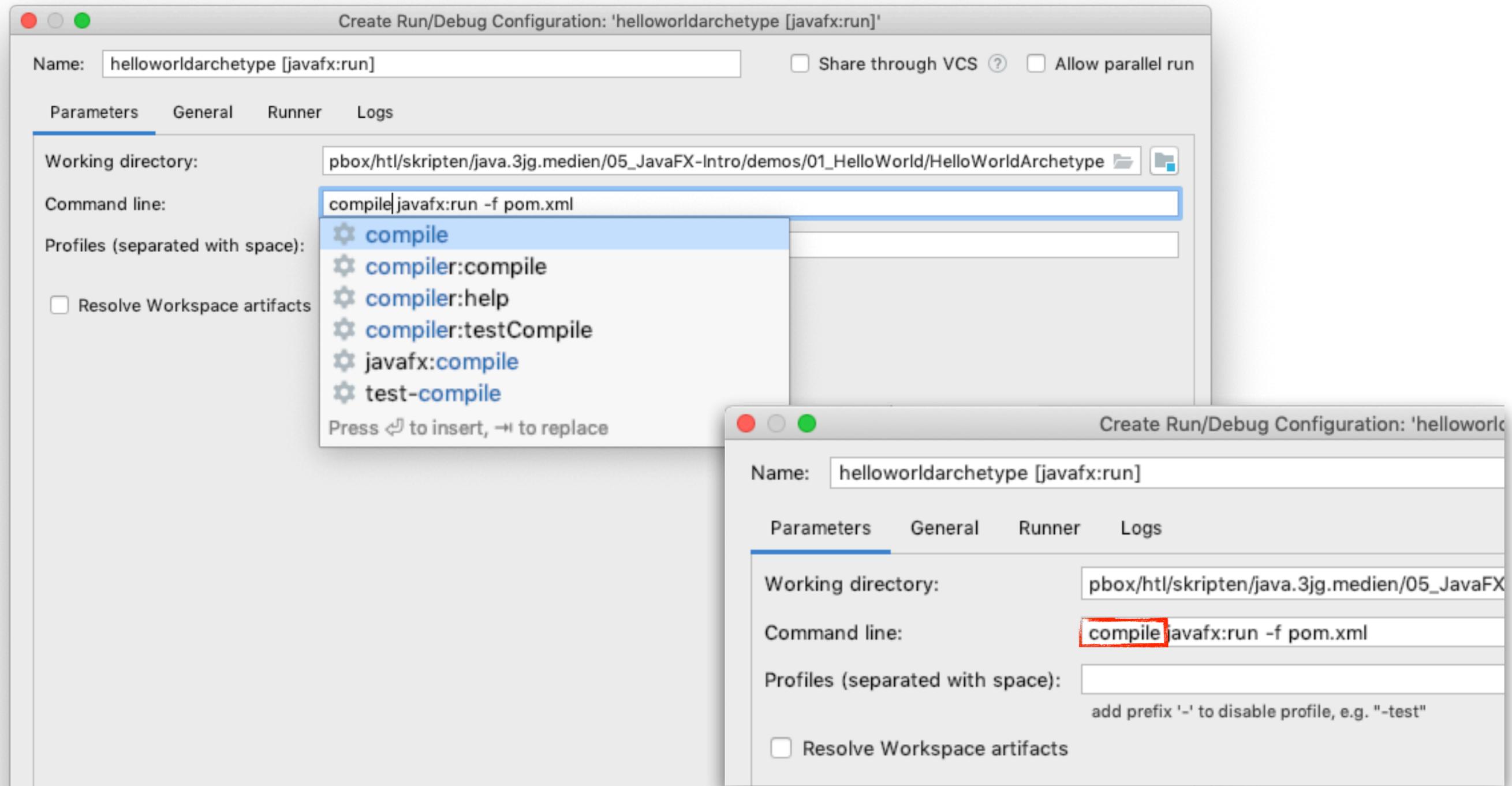
Das Projekt muss nach Aktualisierungen immer neu kompiliert werden. Um diese beiden Arbeitsschritte (javafx:compile und javafx:run) in einem Schritt zusammenzulegen, wird eine eigene Konfiguration erstellt:

The screenshot shows an IDE window titled "HelloWorldArchetype" with a pom.xml file open. The pom.xml content is as follows:

```
<?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/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>12.0.2</version>
    </dependency>
    <dependency>
      <groupId>org.openjfx</groupId>
      <artifactId>javafx-xml</artifactId>
      <version>12.0.2</version>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.1</version>
        <configuration>
```

The Maven tool window on the right shows the project structure with a custom configuration "javafx:run" highlighted in red. A context menu is open over this configuration, with the option "Create 'helloworldarchetype ...' ..." selected and highlighted in blue. Other options in the menu include "Run Maven Build", "Run 'helloworldarchetype ...'", "Debug 'helloworldarchetype ...'", "Execute Before Build", "Execute After Build", "Execute Before Rebuild", "Execute After Rebuild", "Execute Before Run/Debug...", and "Assign Shortcut...".

# Compile eintragen



# Ausführen: compile + run

The screenshot shows an IDE window titled "HelloWorldArchetype" with the following components:

- Project View:** Shows the project structure with folders like `.idea`, `src`, `main`, `resources`, and `target`. The `primary.fxml` file is selected.
- Code Editor:** Displays the content of `primary.fxml`, showing XML tags for `VBox`, `Label`, `Button`, and `Insets`. The text "Updated Primary View" and "Switch to Secondary View" are visible in the code.
- Maven View:** Shows the Maven configuration for the project, including plugins like `clean`, `compiler`, `deploy`, `install`, `jar`, `javafx`, `resources`, `site`, and `surefire`.
- Run Menu:** A context menu is open over the Maven view, showing options: `Run`, `Debug`, `Cover`, `Profile`, `Edit Run Configuration...`, and `Remove Run Configuration`.
- Application Window:** A small window titled "Updated Primary View" is shown in the foreground, containing a button labeled "Switch to Secondary View".

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

# Grundstruktur der FX-Anwendung

```
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 {

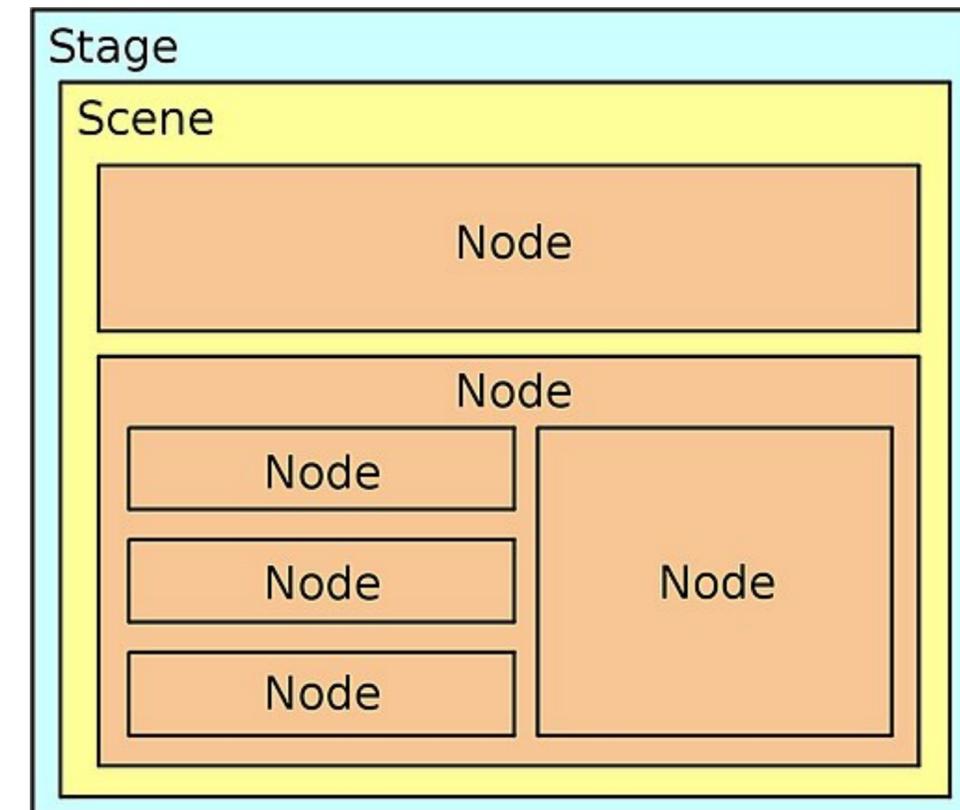
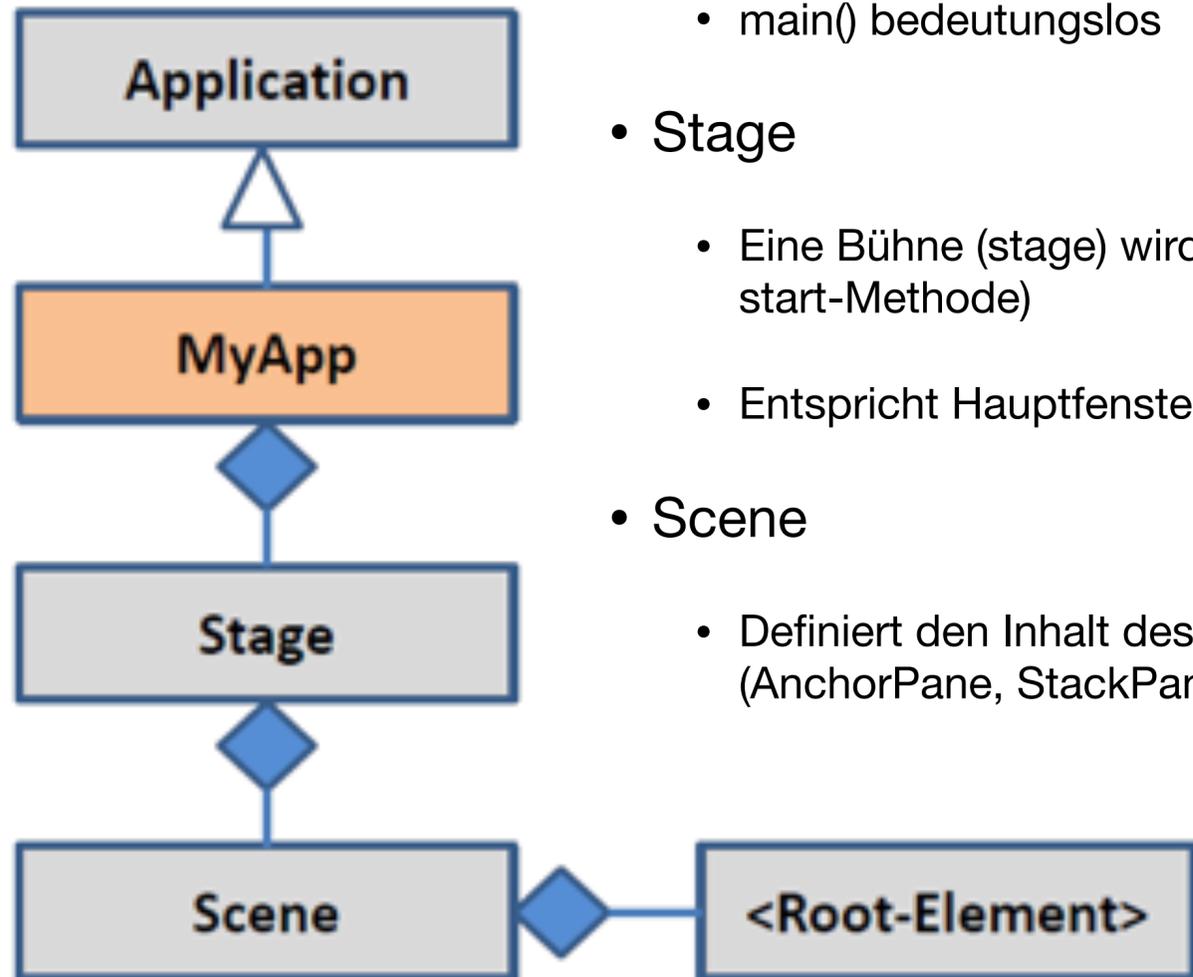
    @Override
    public void start(Stage stage) {...}

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

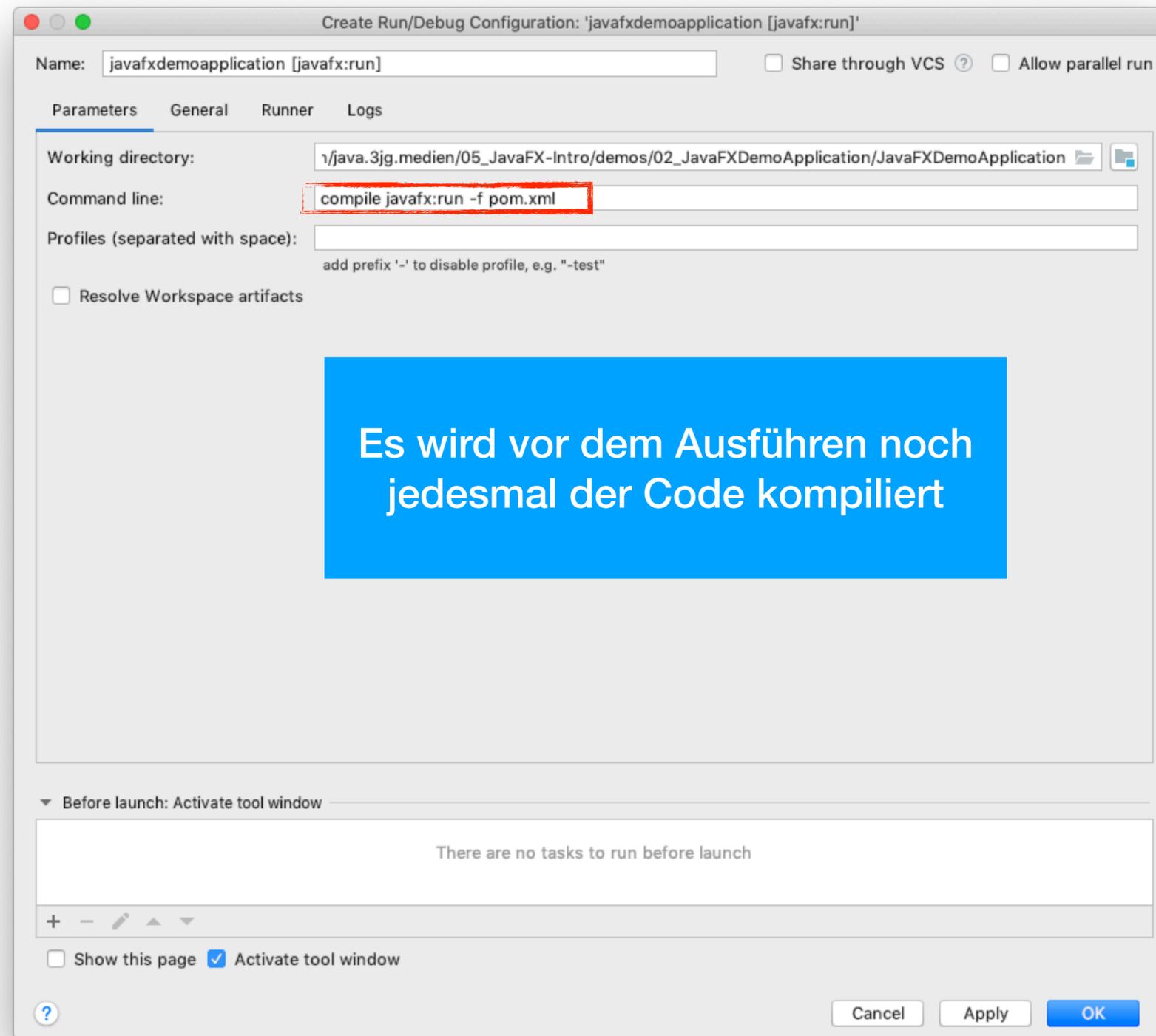
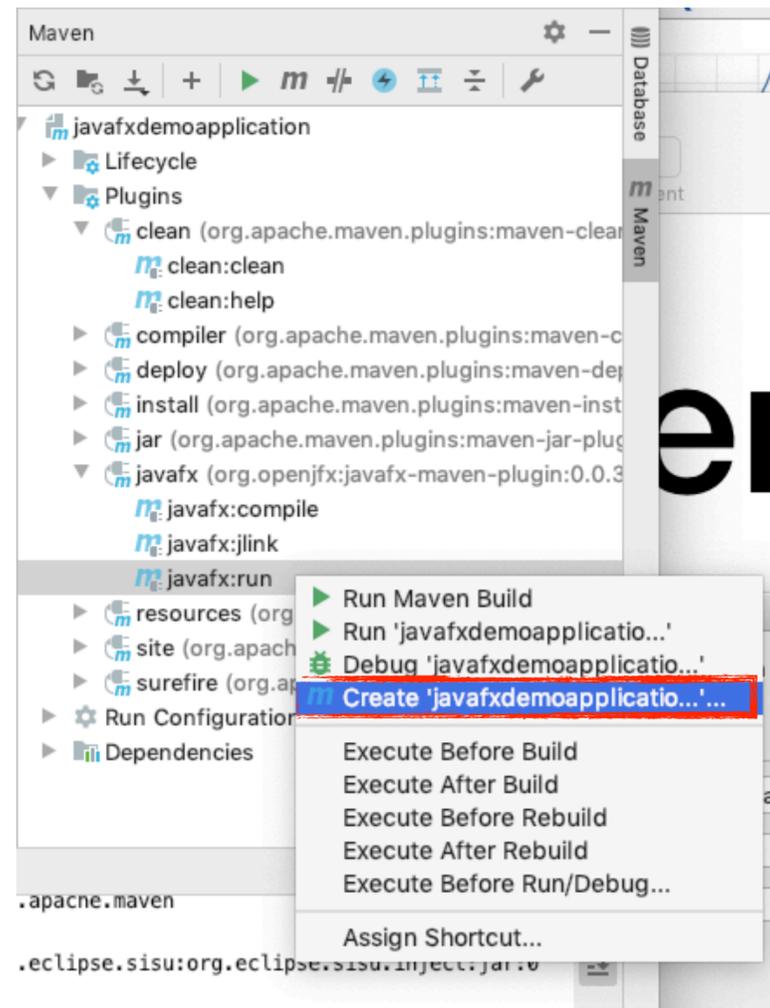
kann mit `^O`  
generiert werden

# 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 Wurzelement (AnchorPane, StackPane, GridPane, ...)



# Eigene Configuration erstellen



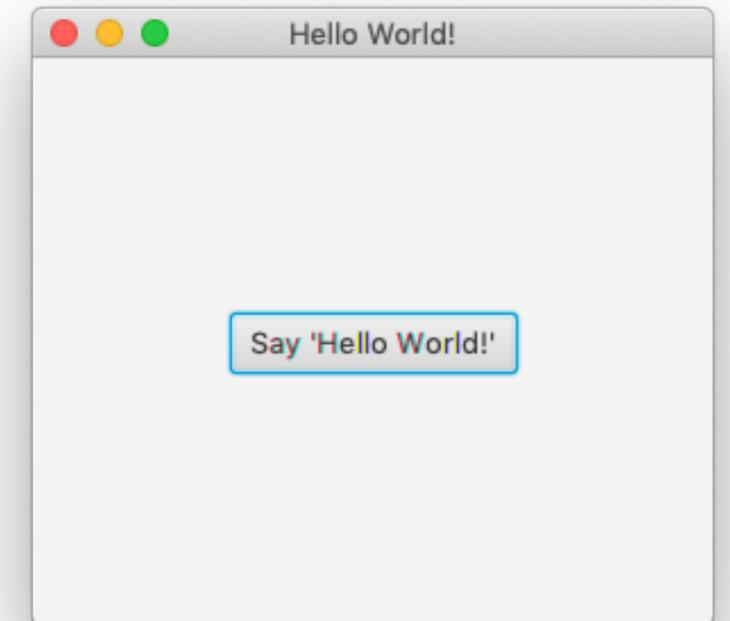
# Starten der Anwendung

The screenshot displays an IDE window for a JavaFX application. The main editor shows the following code in `JavaFXDemoApplication.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); }  
}
```

The Maven panel on the right shows the `javafx:run` goal selected under the `Run Configurations` section. The Run console at the bottom shows the output:

```
at.htl.javafxdemoapplication:jar:1.0-SNAPSHOT run  
[IJ]-1-ARTIFACT_RESOLVING-[IJ]-path=-[IJ]-artifactCoord=org.codehaus.plexus:plexus-1  
.25-[IJ]-error=  
[IJ]-1-ARTIFACT_RESOLVED-[IJ]-path=/Users/stuetz/.m2/repository/aopalliance/aopallia  
.0/aopalliance-1.0.jar-[IJ]-artifactCoord=aopalliance:aopalliance:jar:1.0-[IJ]-erro  
[IJ]-1-ARTIFACT_RESOLVED-[IJ]-path=/Users/stuetz/  
.m2/repository/org/codehaus/mojo/animal-sniffer-annotations/1.14/animal-sniffer-annotations-1.14  
.jar-[IJ]-artifactCoord=org.codehaus.mojo:animal-sniffer-annotations:jar:1.14-[IJ]-error=  
[IJ]-1-ARTIFACT_RESOLVING-[IJ]-path=-[IJ]-artifactCoord=javafx:javafx:jar:1-[IJ]-error=  
[IJ]-1-ARTIFACT_RESOLVING-[IJ]-path=-[IJ]-artifactCoord=org.codehaus  
.plexus:plexus-component-annotations:jar:1.7.1-[IJ]-error=  
Hello World!
```



# Startmethode und Ergebnis

```
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();
```

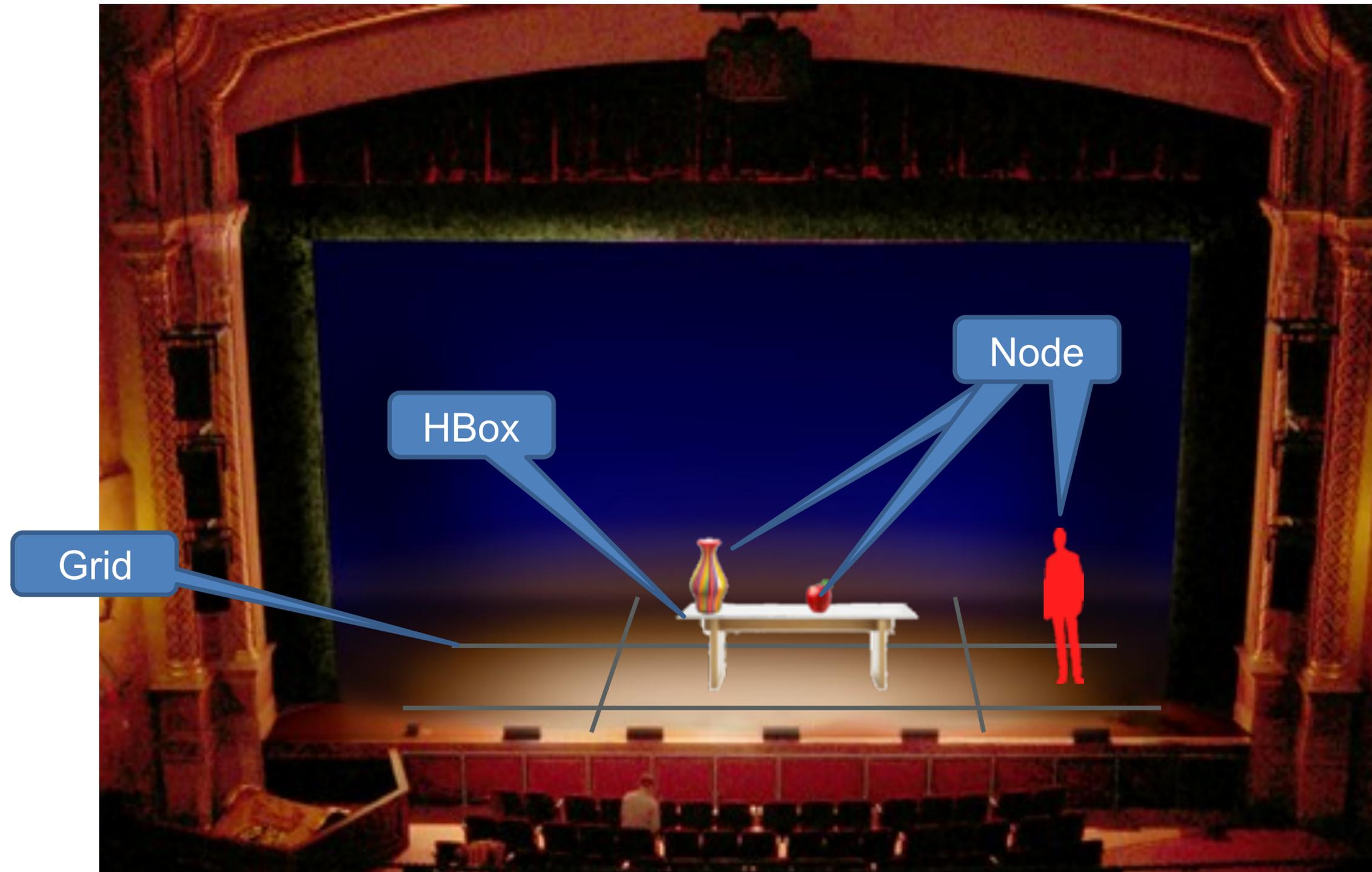
```
public static void main(String[] args) { launch(args); }
```

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

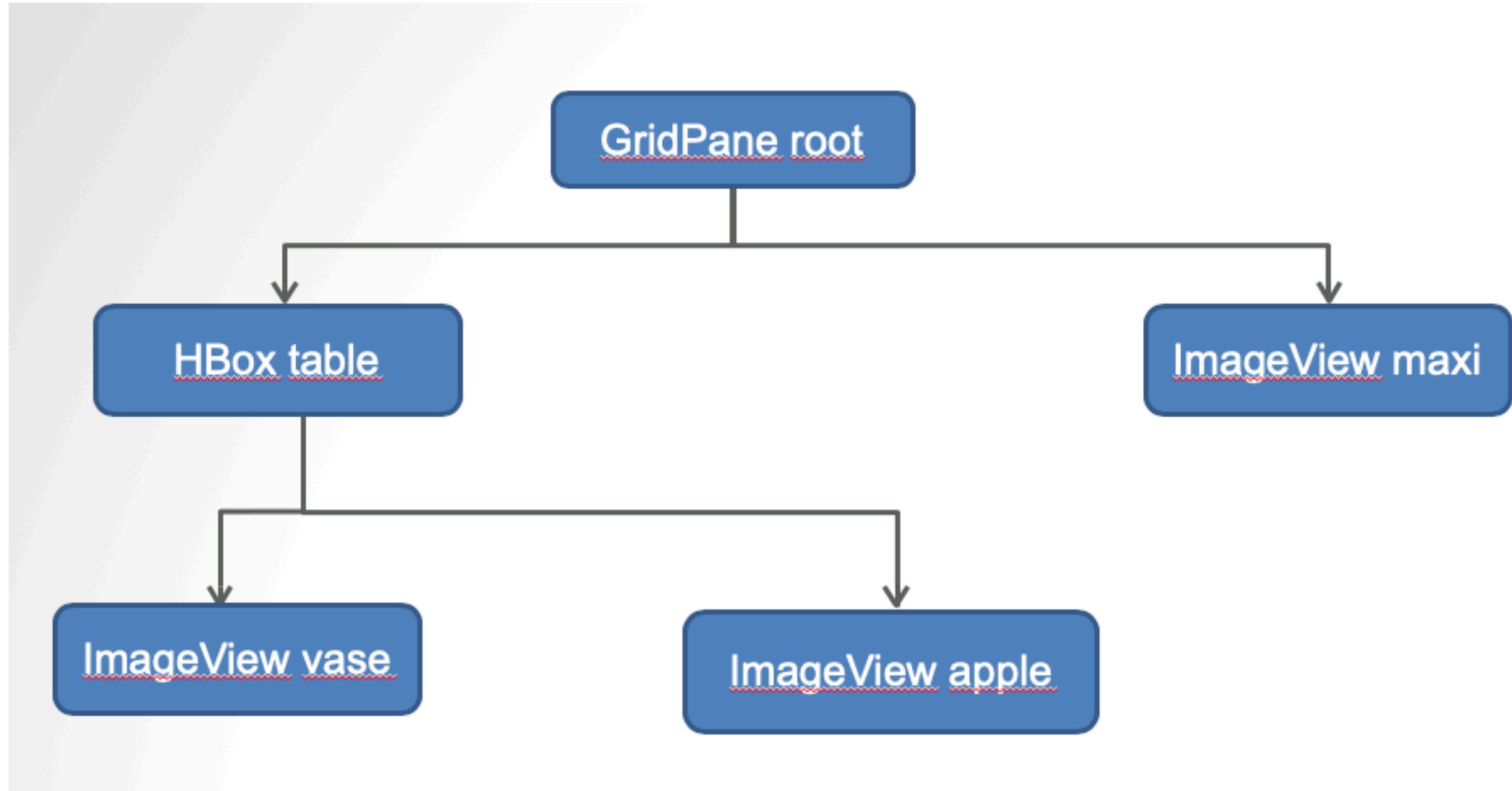


```
...jar-[IJ]-artifactCoord=org.  
[IJ]-1-ARTIFACT_RESOLVING-[IJ]  
[IJ]-1-ARTIFACT_RESOLVING-[IJ]  
...plexus:plexus-component-ann  
Hello World!
```

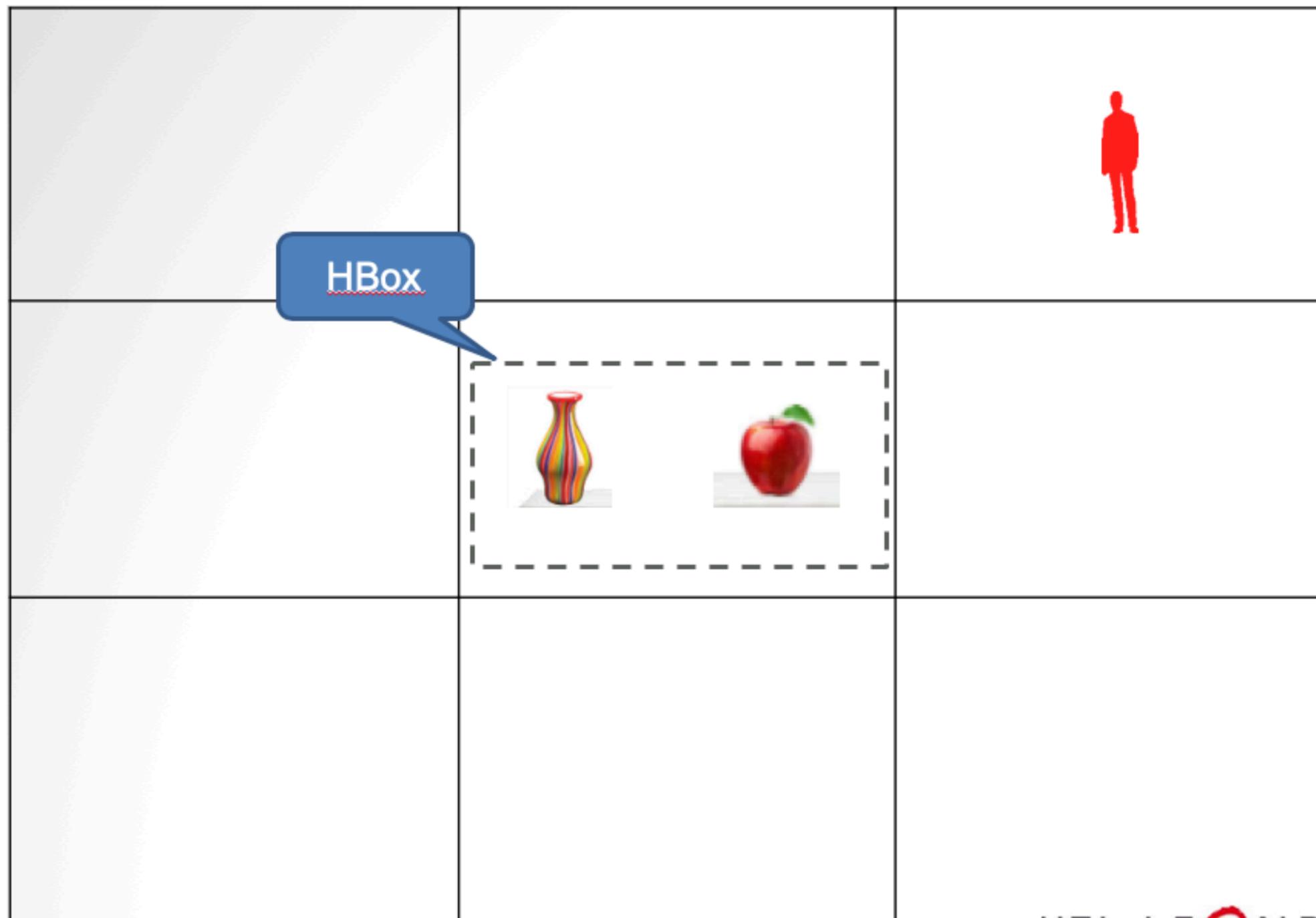
# Stage -> Scene -> Grid



# Scene Graph



# Scene Layout



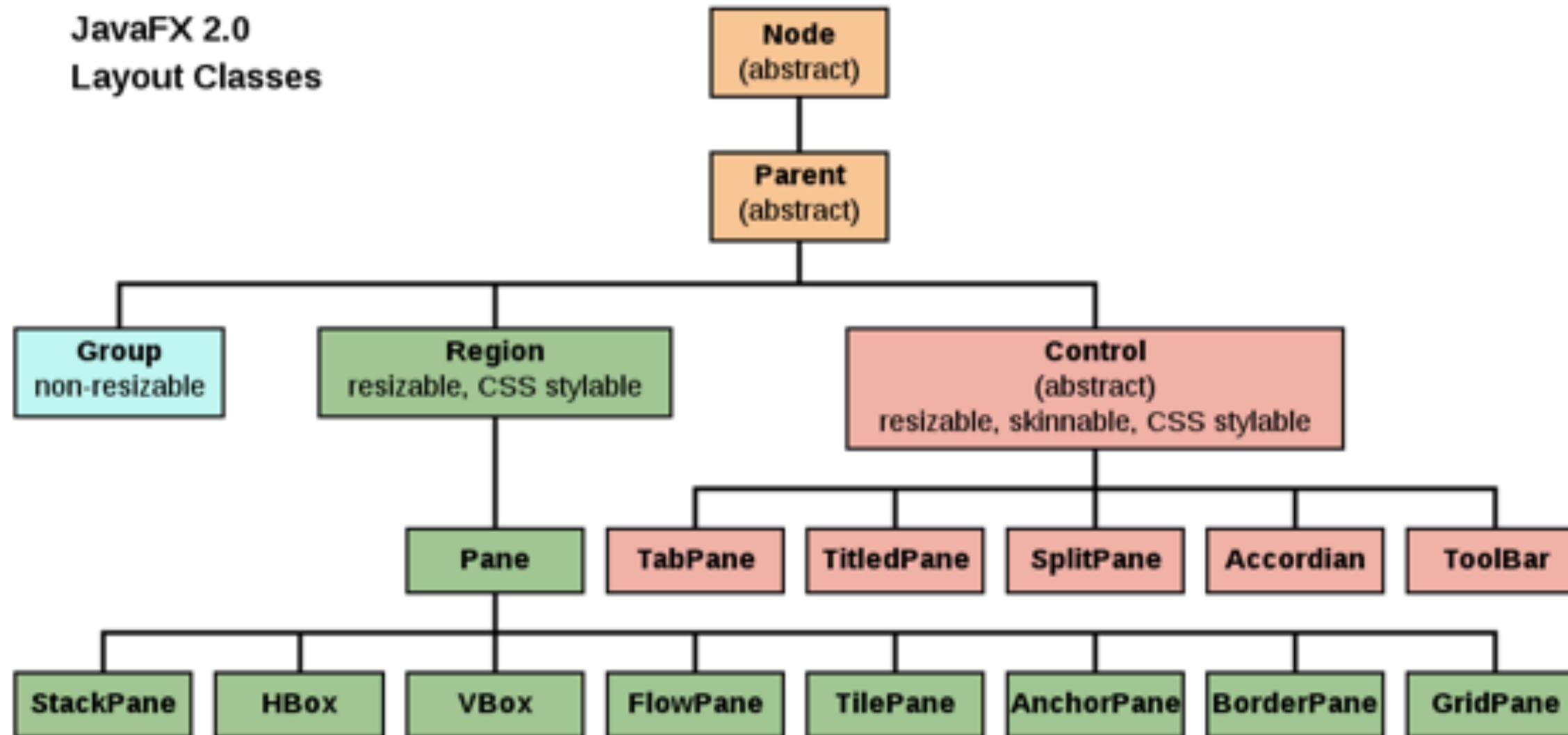
# Source Code

```
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

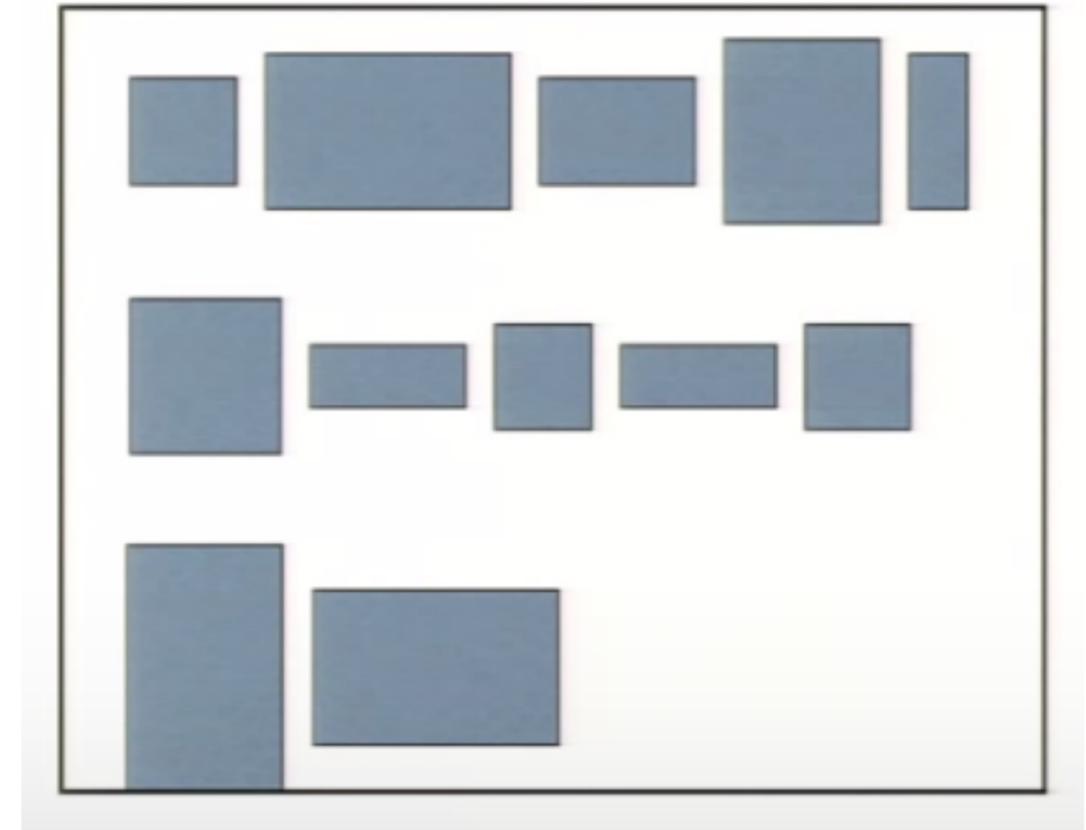
# Hierarchie Container-Elemente



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

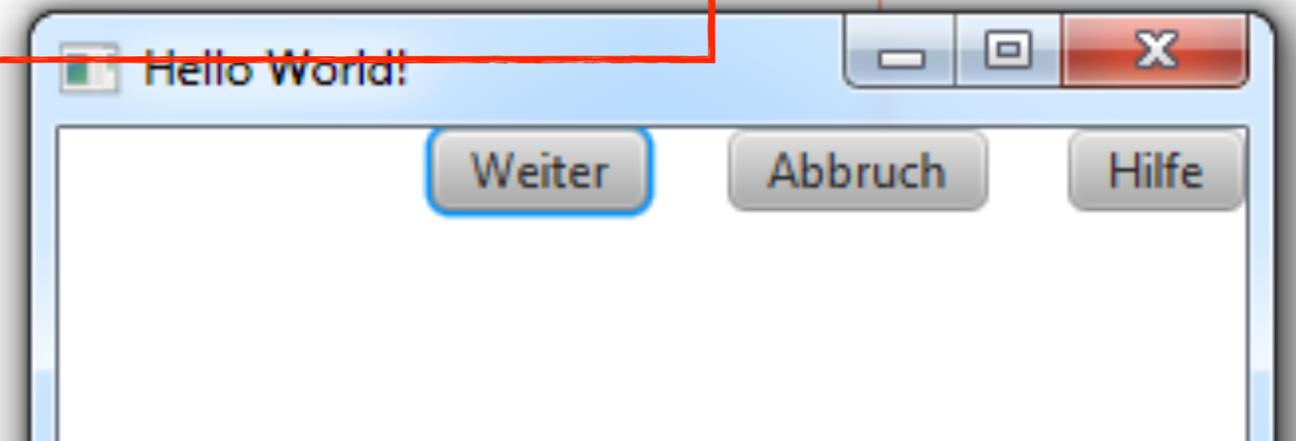
# FlowPane

- 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();
```

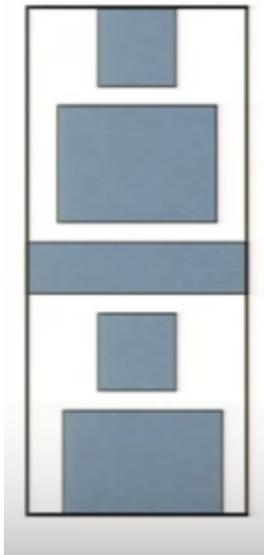


# Java FX – HBox, VBox

- HBOX



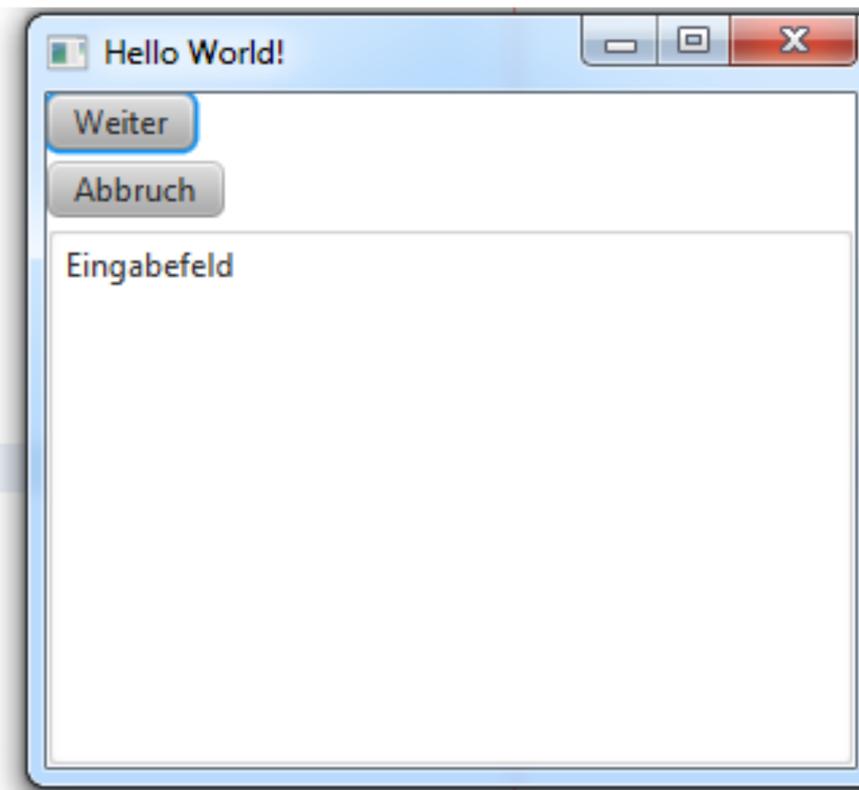
- VBox



# VBox/HBox

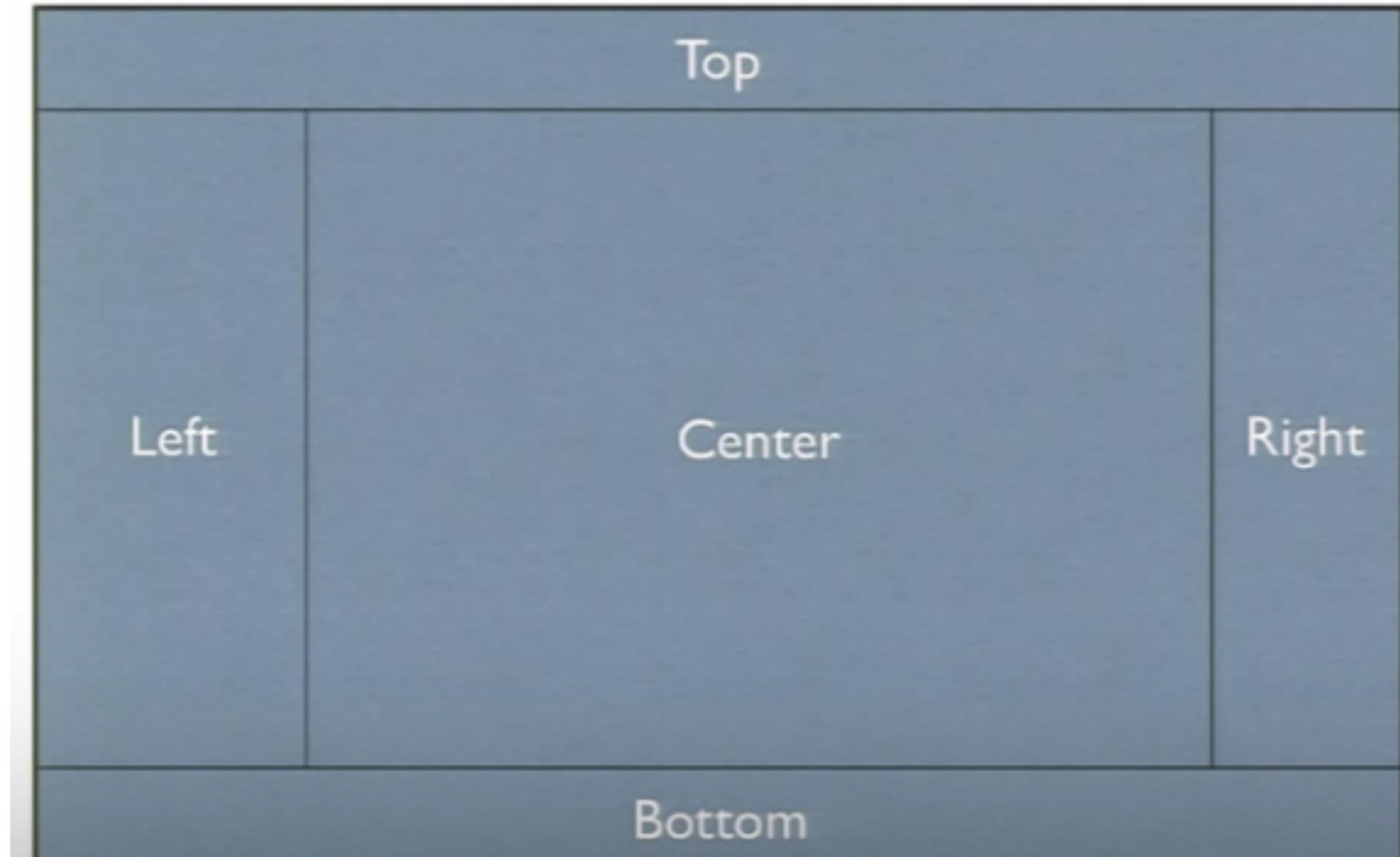
- Anordnung nacheinander/untereinander ohne Umbruch (FlowPane)
- Abstände können angegeben werden
- `setVgrow()/setHGrow()` bestimmt, was mit dem restlichen Platz geschieht

```
@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();
}
```



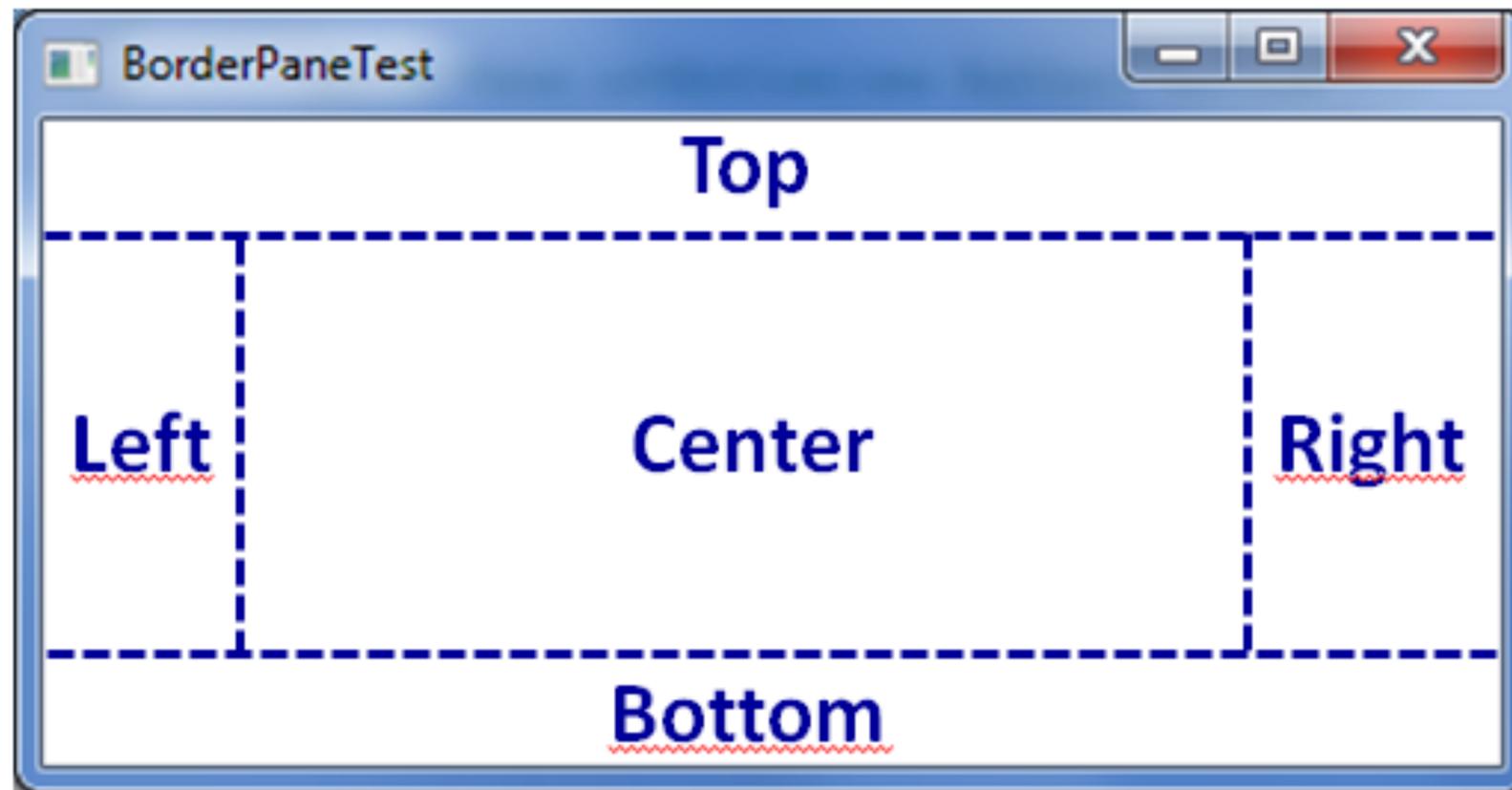
# Java FX – BorderPane

- BorderPane



# BorderPane

- Bis zu 5 Platzierungsmöglichkeiten



bevorzugte Höhe, fest

restlicher Platz,  
wächst/schrumpft

bevorzugte Höhe, fest

bevorzugte  
Breite, fest

restlicher Platz,  
wächst/schrumpft

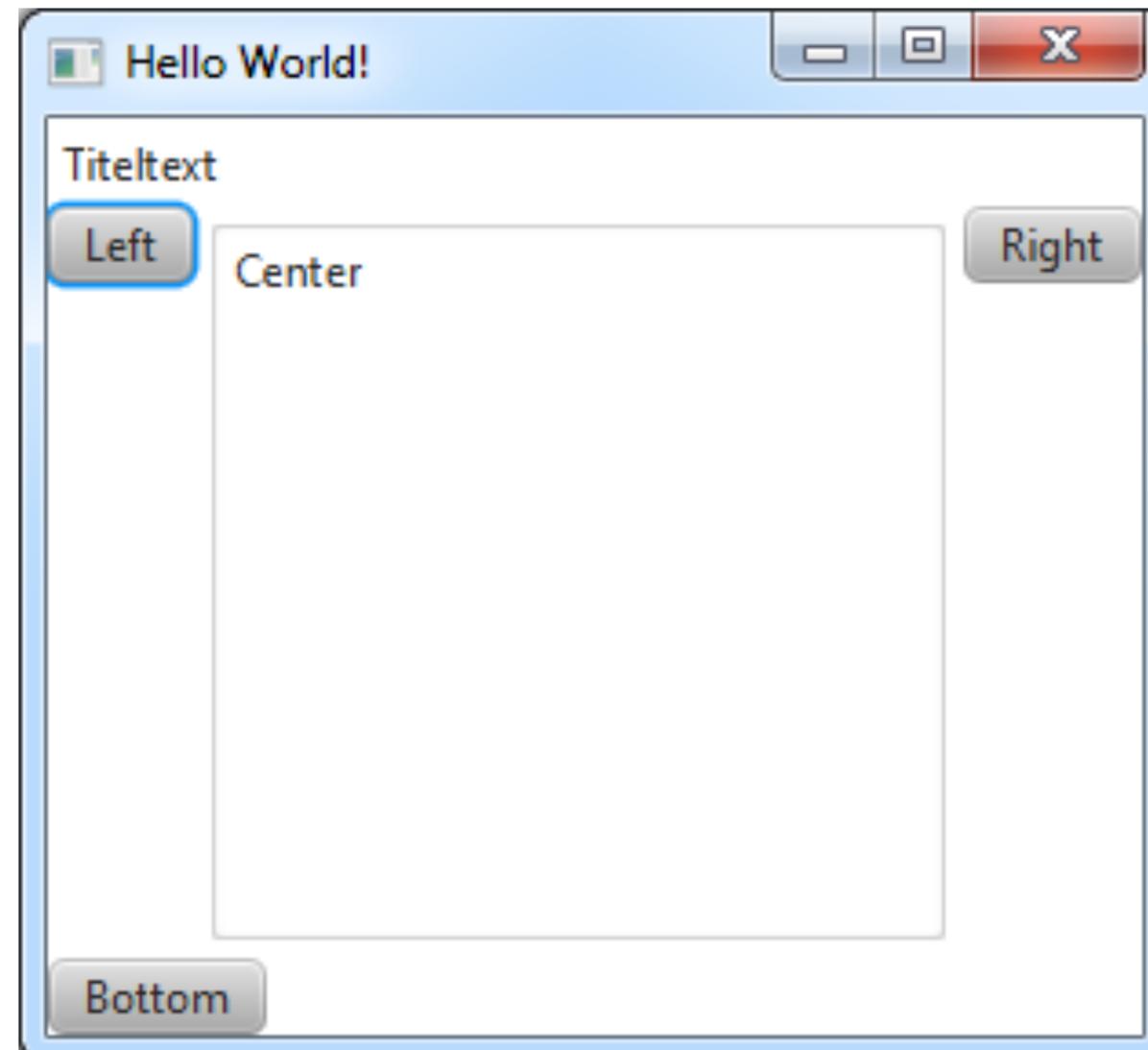
bevorzugte  
Breite, fest

# 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);
```

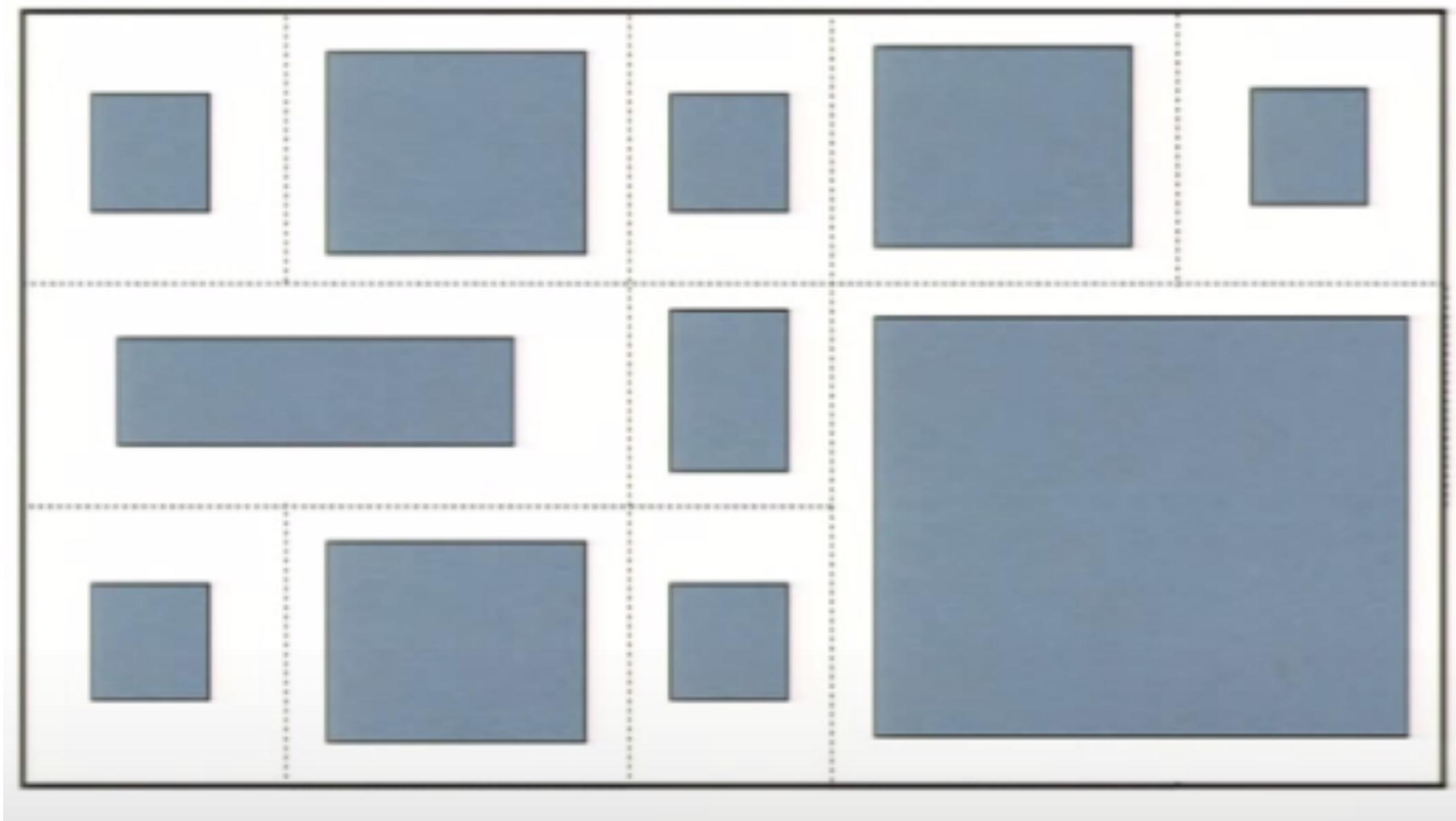
# BorderPane - Ergebnis

- Margin für Titeltext und Text im Zentrum



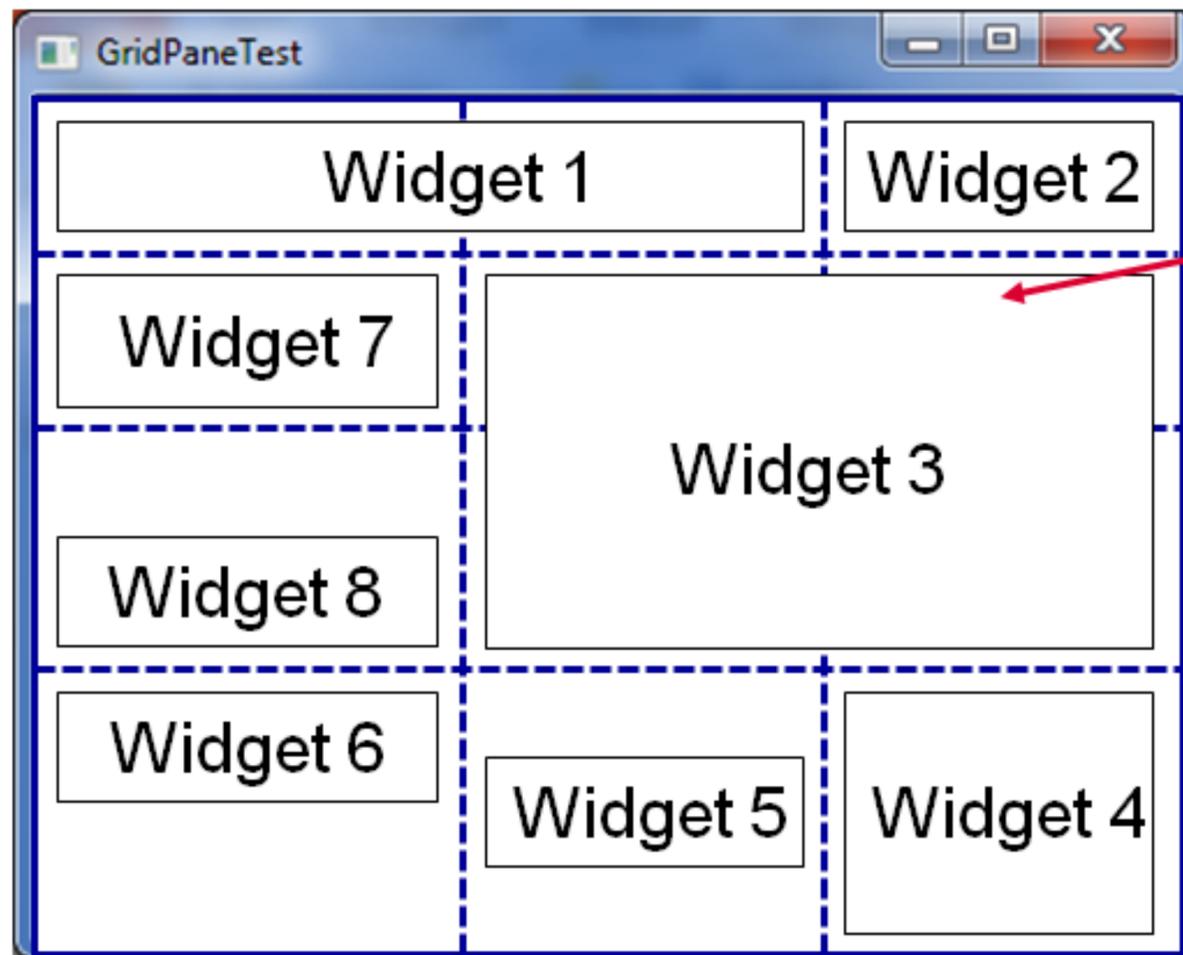
# Java FX – GridPane

- GridPane



# GridPane

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



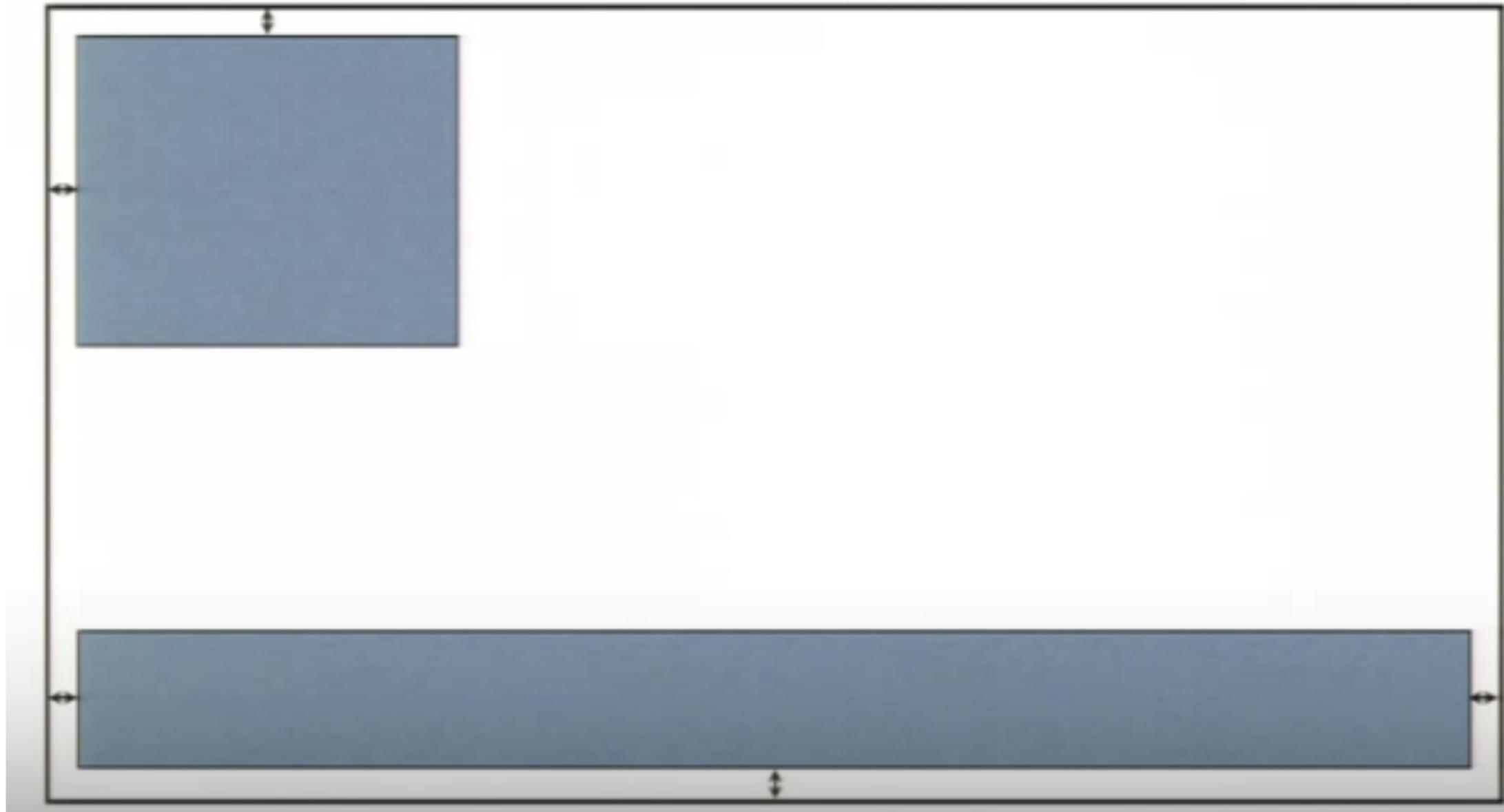
Widgets dürfen sich über mehrere Zeilen und/oder Spalten erstrecken.

Zeilen dürfen unterschiedliche Höhen, Spalte unterschiedliche Breiten besitzen.

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

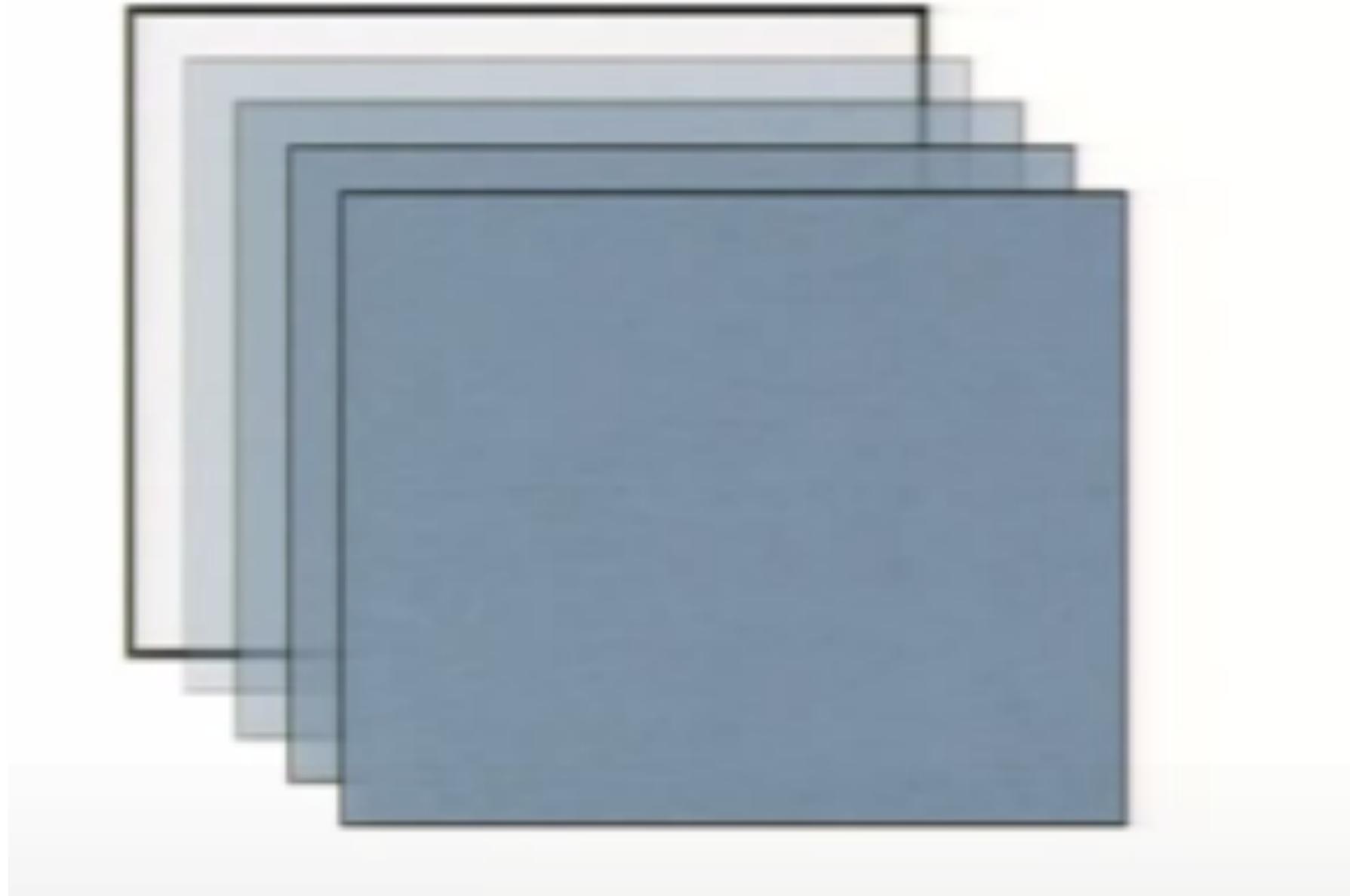
# Java FX – AnchorPane

- AnchorPane



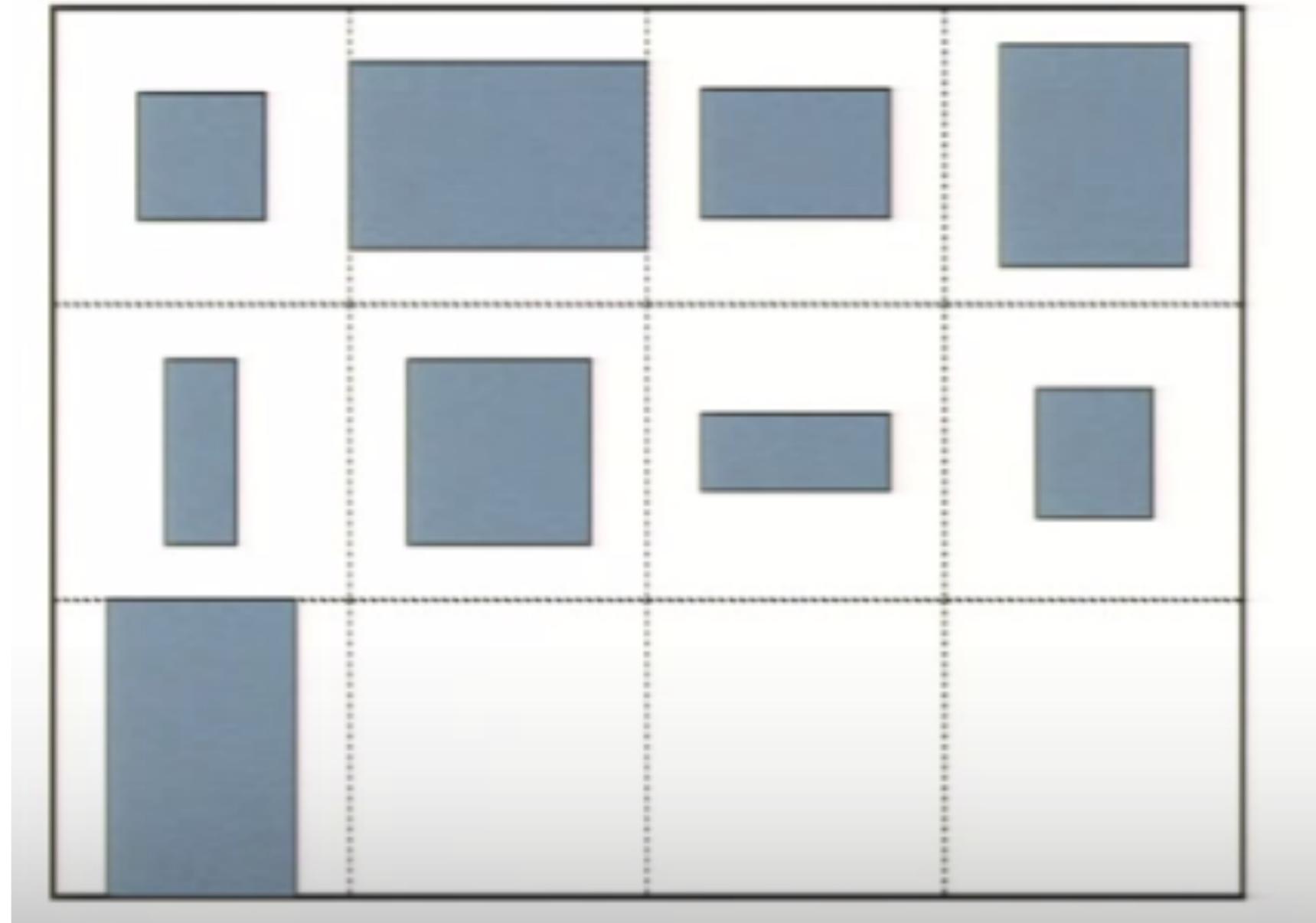
# Java FX – StackPane

- StackPane



# Java FX – TilePane

- TilePane

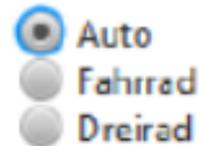


# Beispiele für UI-Controls in JavaFX

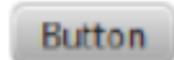
## CheckBox



## RadioButton



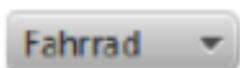
## Button



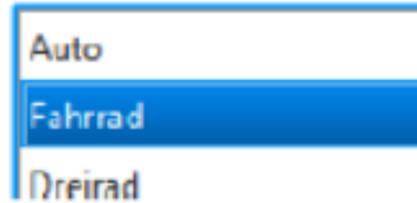
## ToggleButton



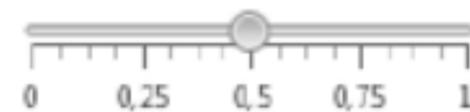
## ComboBox



## ListView



## Slider



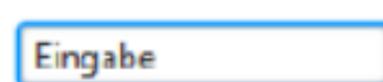
## Hyperlink



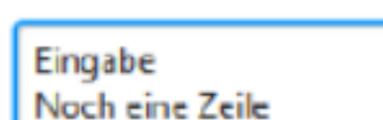
## Label

Vorname:

## TextField



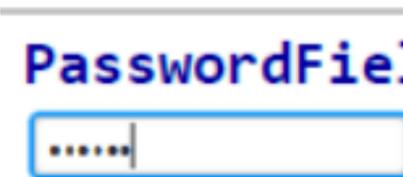
## TextArea



## HTMLEditor



## PasswordField



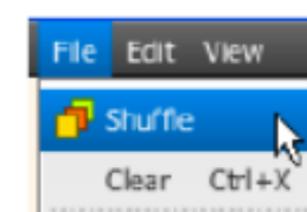
## TableView



## TreeView



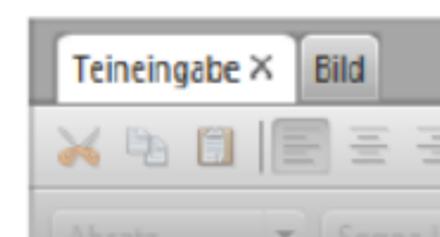
## Menu



## ToolBar



## TabPane



## SplitPane



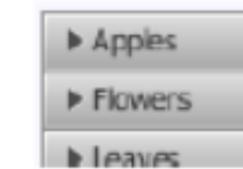
## ScrollPane



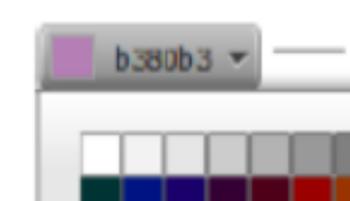
## TitledPane



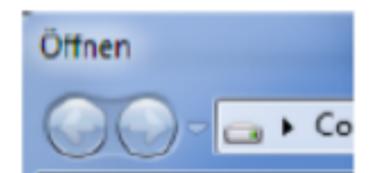
## Accordion



## ColorPicker



## FileChooser



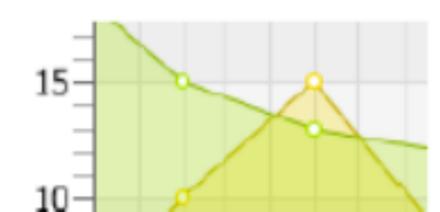
## DirectoryChooser



## LineChart



## AreaChart



# Button mit Eventhandling

- `javafx.scene.control.Button`;
- Observer anmelden (registrieren) mit `setOn...()`
  - `setOnAction` —> `ActionEventHandler`
  - Generische Implementierung
- Implementierung der Behandlungsroutine
  - Erzeugung und Instanziierung 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

```
btn.setOn|
b) ○ setOnDragEntered(EventHandler<? super
○ setOnDragExited(EventHandler<? super D
○ setOnDragOver(EventHandler<? super Dra
○ setOnInputMethodTextChanged(EventHandl
○ setOnKeyPressed(EventHandler<? super K
} ○ setOnKeyReleased(EventHandler<? super
S ○ setOnKeyTyped(EventHandler<? super Key
r ○ setOnMouseClicked(EventHandler<? super MouseEvent> eh) void
S ○ setOnMouseDragEntered(EventHandler<? super MouseDragEvent> eh) void
P ○ setOnMouseDragExited(EventHandler<? super MouseDragEvent> eh) void
P ○ setOnMouseDragOver(EventHandler<? super MouseDragEvent> eh) void
P ○ setOnMouseDragReleased(EventHandler<? super MouseDragEvent> eh) void
○ setOnMouseDragged(EventHandler<? super MouseEvent> eh) void
○ setOnMouseEntered(EventHandler<? super MouseEvent> eh) void
○ setOnMouseExited(EventHandler<? super MouseEvent> eh) void
Ph ○ setOnMouseMoved(EventHandler<? super MouseEvent> eh) void
na ○ setOnMousePressed(EventHandler<? super MouseEvent> eh) void
```

← → 📄 🗑

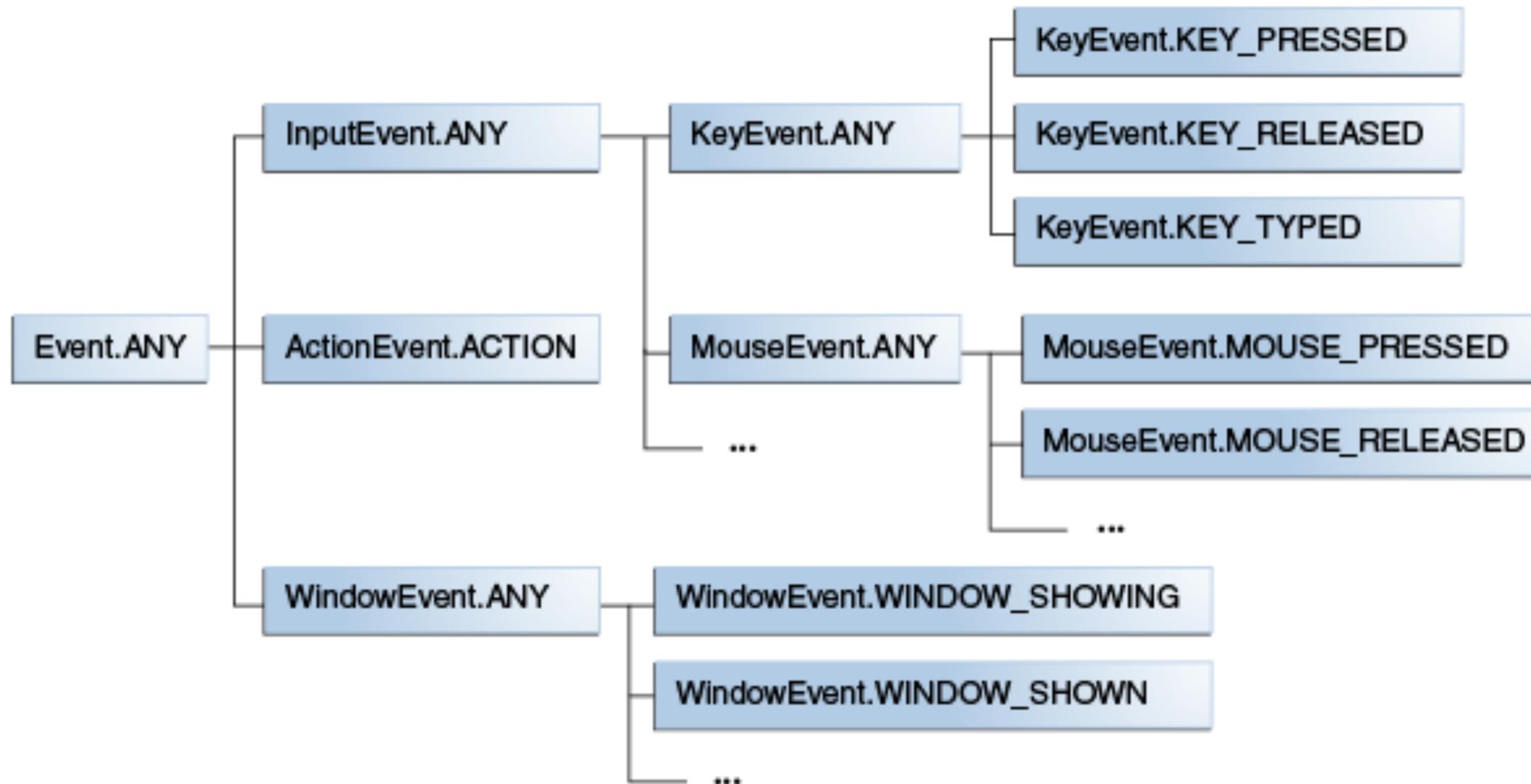
[javafx.scene.Node](#)

```
public final void setOnMousePressed(EventHandler<? super MouseEvent> eh)
```

Sets the value of the property onMousePressed.

Property description:  
Defines a function to be called when a mouse button has been pressed on this Node.

# Hierarchie der Events



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

[https://docs.oracle.com/javafx/2/ui\\_controls/jfxpub-ui\\_controls.htm](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](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%20Panee%20and%20Controls.pdf>
- <https://www.schmiedecke.info/Prg1/Konzepte/VL20-%20JAVAFX-Interaktion.pdf>
- <https://www.schmiedecke.info/Prg1/Konzepte/VL21-%20JAVAFX-Navigation.pdf>

# HTL LE NDING



Noch  
Fragen?

HTL LE  NDING

**Schön, hier zu lernen**