ICT Berufsbildung
Formation professionnelle
Formazione professionale

# Programmieren mit Java

## "Hello World" mit JavaFX


## Inhalt

# 1 Auftrag

Dein Auftrag ist es, den Anleitungen in diesem Dokument zu folgen. Du wirst den "Scene Builder" von Gluon auf auf Deinem Rechner installieren und danach Deine erste JavaFX-Applikation programmieren.

# 2 Ziel

Du lernst den "Scene Builder" zur Gestaltung graphischer Benutzeroberflächen (GUI) mit JavaFX kennen und erstellst dein erstes JavaFX-Programm (HelloWorld-Applikation).

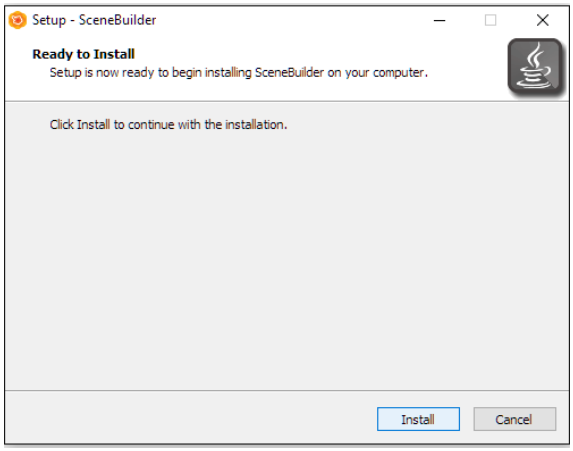
# 3 Scene Builder

Der JavaFX Scene Builder ist ein grafisches Tool, das die Erstellung von FXML-Dateien vereinfacht. Mit dem Tool können GUI-Elemente wie Buttons, Textfelder, etc. einfach entworfen und per Drag & Drop in eine JavaFX-Applikation integriert werden. Oracle hat die Entwicklung des Tools eingestellt und den Quelltext des Programms unter einer BSD-ähnlichen Lizenz freigegeben. Daraufhin hat die Firma Gluon die Wartung und den kommerziellen Support übernommen.

## 3.1 Installation

Installiere als erstes den Scene Builder von Gluon. Die Installationsdateien findest Du im Internet auf der folgenden Webseite: http://gluonhq.com/labs/scene-builder/

Lade die Installationsdateien (das Windows Installer x64 Paket falls Du einen Windows-Rechner hast) auf Deinen Rechner herunter und starte die Installation.




Falls der Scene Builder nach erfolgter Installation am Bildschirm angezeigt wird, kannst Du ihn getrost schliessen. Wir werden zuerst mit IntelliJ arbeiten und starten den Scene Builder später neu.

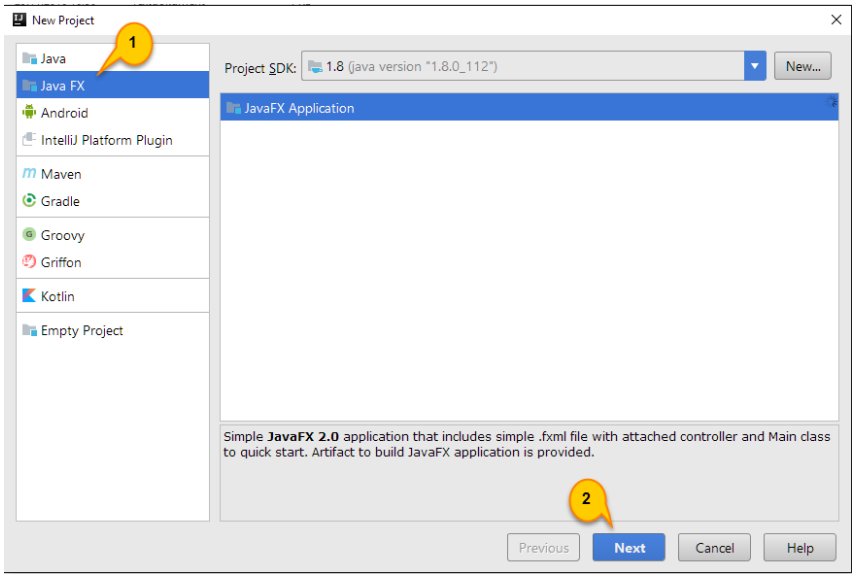
# 4 Deine erste JavaFX-Applikation

Und schon geht's los! Du wirst Deine erste JavaFX-Applikation mit Hilfe des Gluon Scene Builders erstellen.

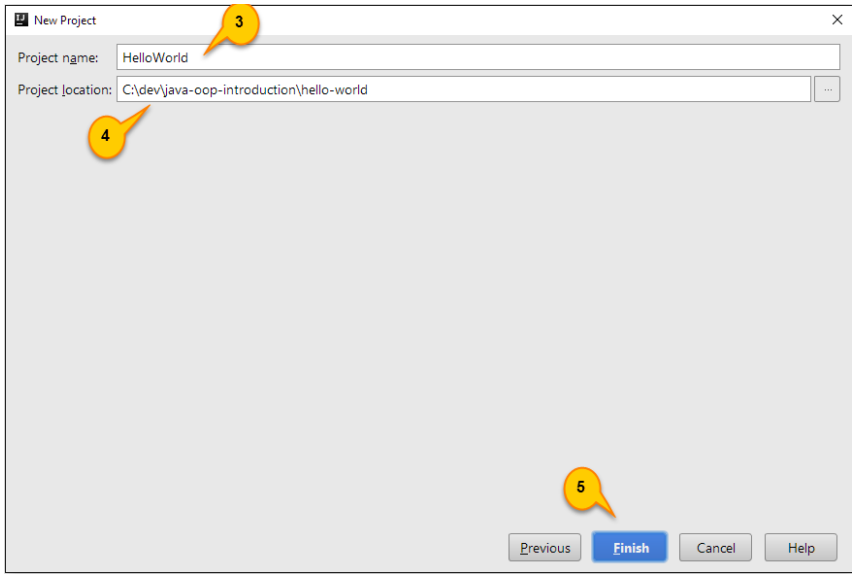
## 4.1 Java FX-Projekt einrichten

Starte die Entwicklungsumgebung IntelliJ und folge den unstenstehenden Schritten.

1. Erstelle ein neues Projekt und wähle aus der Liste der Projektvorlagen den Eintrag "Java FX".
2. Klicke "Next"



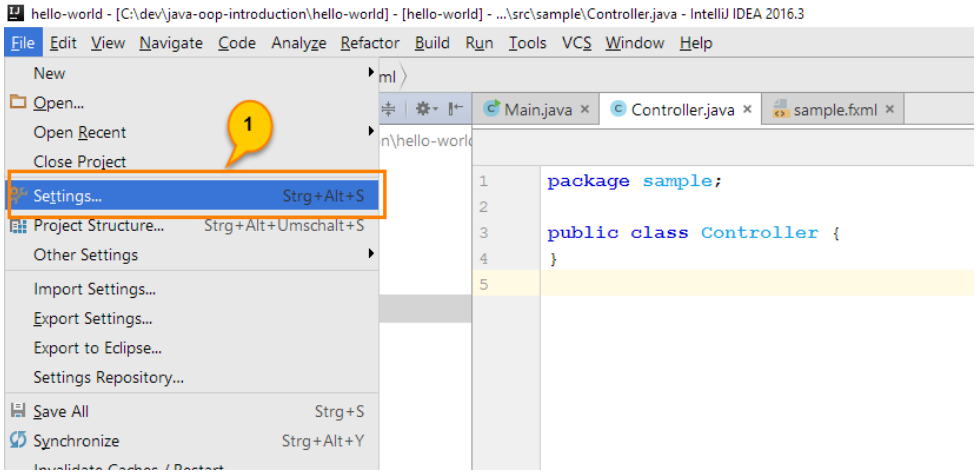

3. Gib dem Projekt einen sprechenden Namen
4. Bestimme den Ort, wo die Projektdateien gespeicher werden sollen (Project location)



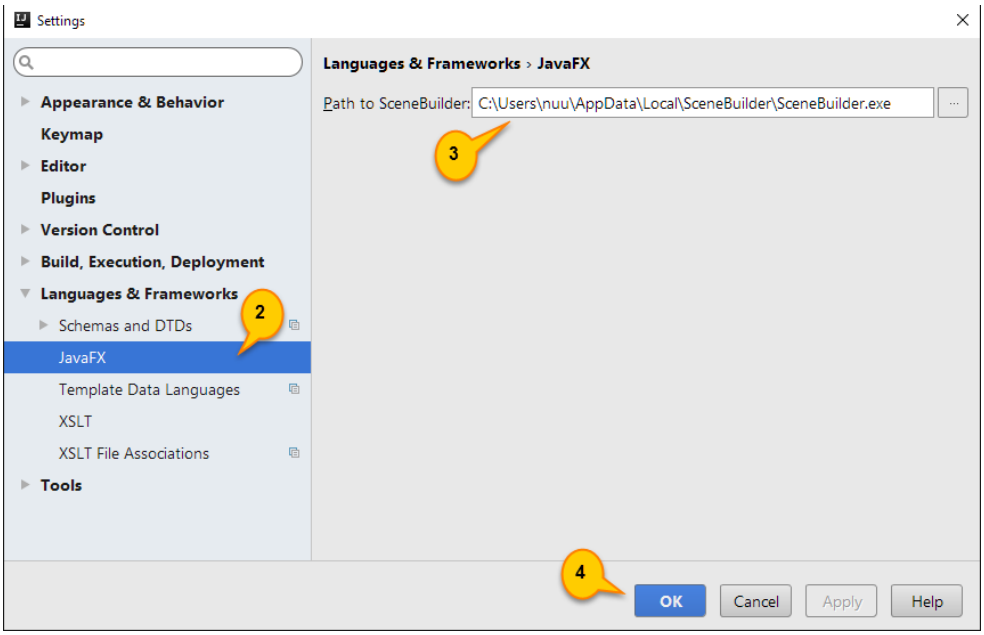
### 4.1.1 Pfad zum Scene Builder einrichten

Damit der Scene Builder zusammen mit IntelliJ verwendet werden kann, müssen die beiden einander bekannt gemacht werden. Stelle folgendes sicher:

1. Navigiere zum Menu-Eintrag *File* > *Settings…*



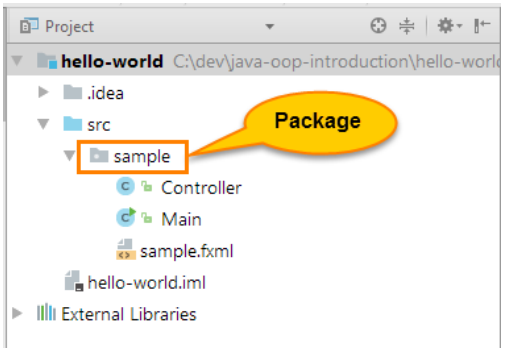

2. Im "Settings"-Dialog navigierst Du zum Eintrag *Languages & Frameworks* > *JavaFX*
3. Trage im Feld "Path to SceneBuilder" den Pfad zum Scene Builder Executable ein (typischerweise "c:\users\{benutzername}\AppData\Local\SceneBuilder\SceneBuilder.exe")



## 4.1.2 Projektstruktur

Wie Du in der Project View sehen kannst, hat IntelliJ hat automatisch ein Package namens "sample" erstellt und darin die Klassen "Main" und "Controller" sowie eine Datei mit Namen "sample.fxml". In diesem Kapitel wird kurz auf die Bedeutung dieser Projektelemente eingegangen und wir werden diesen Elementen einen passenderen Namen vergeben.

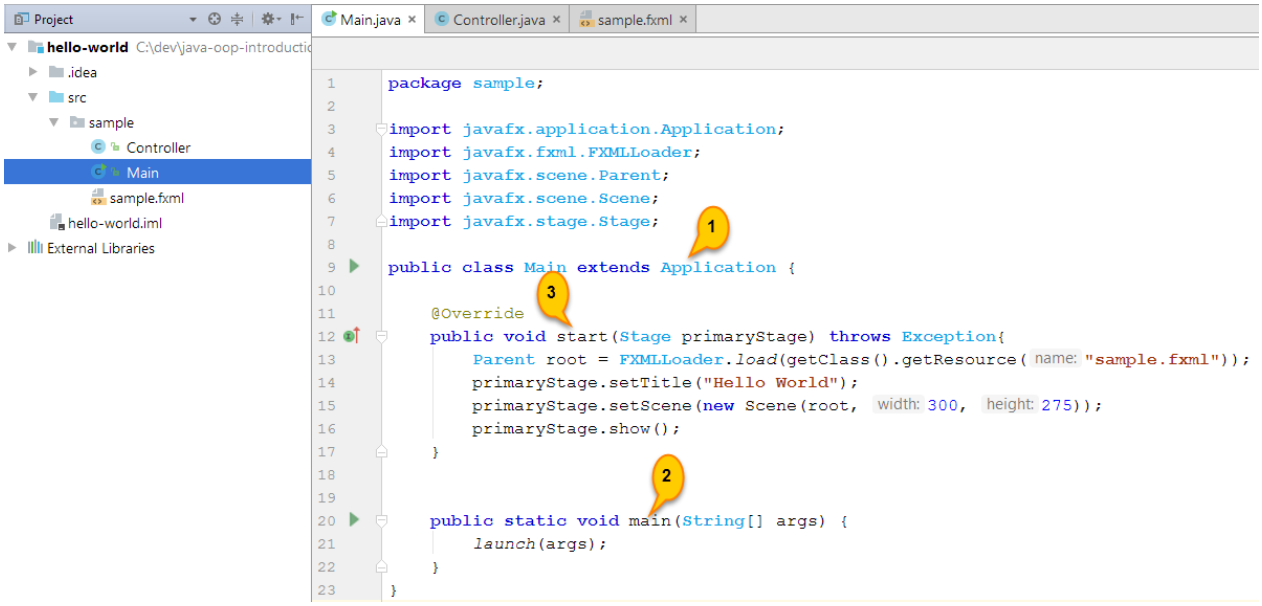
### 4.1.2.1 Packages




Packages sind quasi eine Schublade, in der Klassen und andere Dateien abgelegt werden können. Umfangreichere Java-Applikationen umfassen mehrere Klassen und um diese Ansammlung von Klassen zu organisieren und ihr eine Struktur zu geben werden Klassen in Packages zusammengefasst. Dies erhöht einerseits die Übersichtlichkeit des Projektes und andererseits können so einmal erstellte Packages direkt oder als jar-File in weiteren Applikationen wiederverwendet werden (Prinzip der Klassenbibliotheken). Eine wichtige Funktion von Packages ist ausserdem die sogenannte "Kapselung": die Klassen in einem Package werden gegen aussen abgeschirmt. Damit können unerwünschte Zugriffe und Namenskonflikte vermieden werden.

### 4.1.2.2 Die Klasse "Main"

Die von IntelliJ automatisch generierte "Main" Klasse ist zentraler Bestandteil jeder JavaFX-Applikation. Nachfolgend wird der von IntelliJ automatisch generierte Quellcode kurz erklärt.

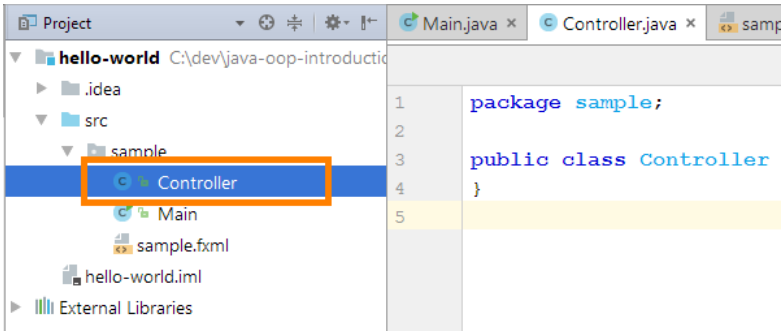



1. Main erbt von der Klasse javafx.application.Application, was am am Java-Schlüsselwort extends zu erkennen ist (Du brauchst zum jetzigen Zeitpunkt nicht zu verstehen, was genau "erben" bedeutet).

2. Die `main()` Methode ist der Einstiegspunkt unserer Applikation; sie wird beim Start als erstes aufgerufen. Darin wiederum wird die JavaFX Klasse `Application` angestossen und zwar mittels Aufruf der Methode `launch()`
3. Die `start()` Methode wird von JavaFX automatisch aufgerufen und dazu verwendet, das GUI zu initialisieren. In der `start()` Methode passiert insbesondere folgendes:
   - ...die fxml-Datei wird geladen → `load()`
   - ...der Titel der Applikation wird gesetzt → `setTitle()`
   - ...zur JavaFX-Stage wird eine Scene hinzugefügt → `setScene()`
   - ...die JavaFX-Stage wird am Bildschirm angezeigt → `show()`

#### 4.1.2.3 Die Klasse "Controller"

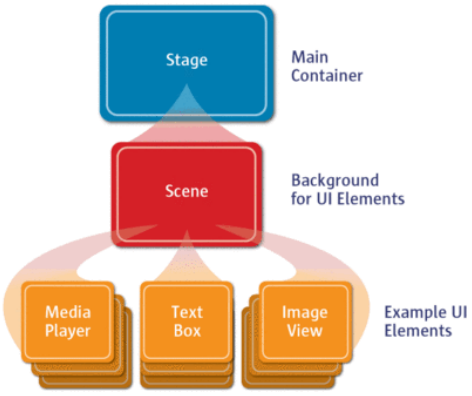
Die "Controller" Klasse ist im Moment noch "leer". Wir werden hier später die Programmlogik codieren, die u.a. für die Anzeige der Daten im GUI verantwortlich sein wird.




#### 4.1.2.4 Die Datei "sample.fxml"

Mit FXML-Dateien können in JavaFX grafische Benutzeroberflächen deklarativ (mit XML) beschrieben werden. Eine FXML-Datei beinhaltet ein komplettes GUI-Layout und kann für jeweils eine sogenannte *Scene* verwendet werden. Bei einer *Scene* handelt es sich um den Inhalt eines Fensters. Und ein Fenster wird in JavaFX als *Stage* bezeichnet. Die *Scene* ist also ein Container, welcher die UI-Elemente beinhaltet und repräsentiert.

Die folgende Grafik illustriert die grundlegende Struktur der grafischen Elemente einer JavaFX Applikation. Für weiterführende Informationen siehe: http://docs.oracle.com/javase/8/javafx/scene-graph-tutorial/scenegraph.htm




**`Stage`**: Dies ist eine von JavaFX bereitgestellte Klasse, die ein Window definiert, mit einem Rand und den typischen Minimize-, Maximize- und Close-Buttons rechts oben.

**`Scene`:** Zu einer Stage werden Scene's hinzugefügt. Auf einer Scene werden Buttons, Textfelder, Bilder etc. platziert.

## 4.1.3 Refactoring: Dateien umbenennen

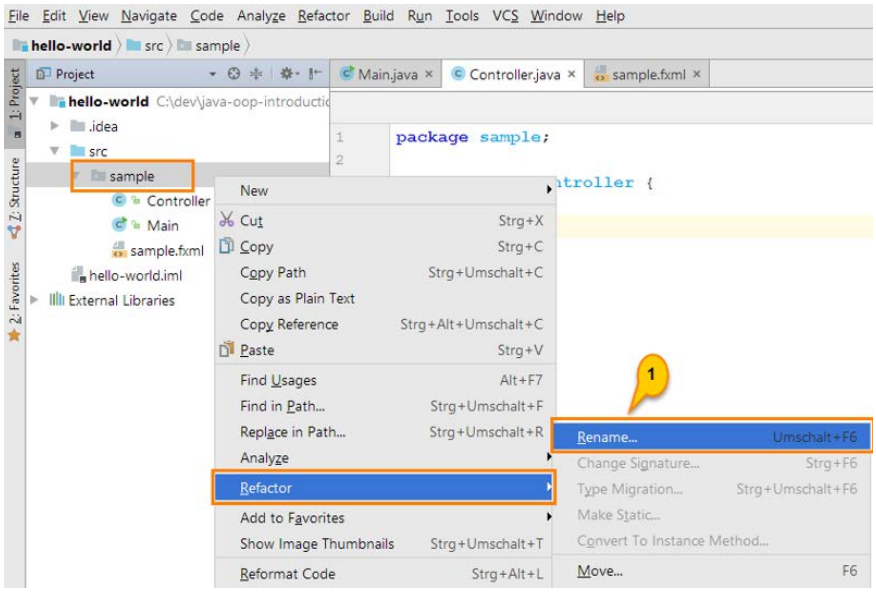
Die von IntelliJ vergebenen Namen für das Package und für die FXML-Datei sollen aussagekräftiger benannt werden. Um alle Abhängigkeiten zu bewahren, verwendest Du dazu am besten die Refactor-Funktionen von IntelliJ.

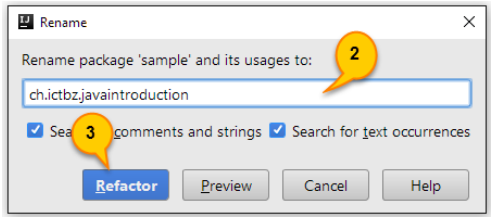
### 4.1.3.1 Package umbenennen

Als erstes vergibst Du unserem Package einen passenderen Namen. In der Java-Welt ist es üblich, dass zuerst das Länderkürzel gefolgt von einem Punkt und dann der Firmenname gefolgt von einem Punkt und dann der Projektname eingegeben wird, z.B. ch.ictbz.helloworld.

Führe folgende Schritte aus:

1. Navigiere in der Project View zum Package sample und dann über das Kontextmenu der rechten Maustaste zu Refactor > Rename…
2. Vergib einen passenden Namen für das Package
3. Klicke auf den Button Refactor um den Vorgang abzuschliessen

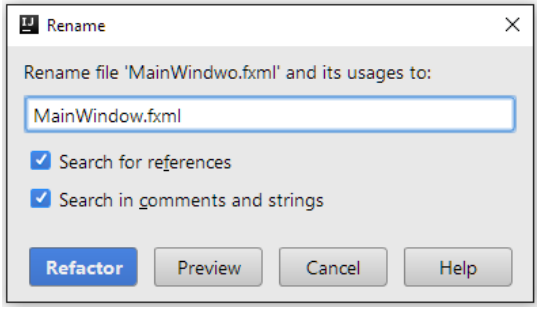



### 4.1.3.2 FXML-Datei umbenennen

Auf für die Umbenennung der Datei sample.fxml verwenden wir das Refactoring von InelliJ. Navigiere in der Project View zur Datei sample.fxml und aktiviere auch hier über das Kontextmenu der rechten Maustaste den Eintrag *Refactor > Rename..*

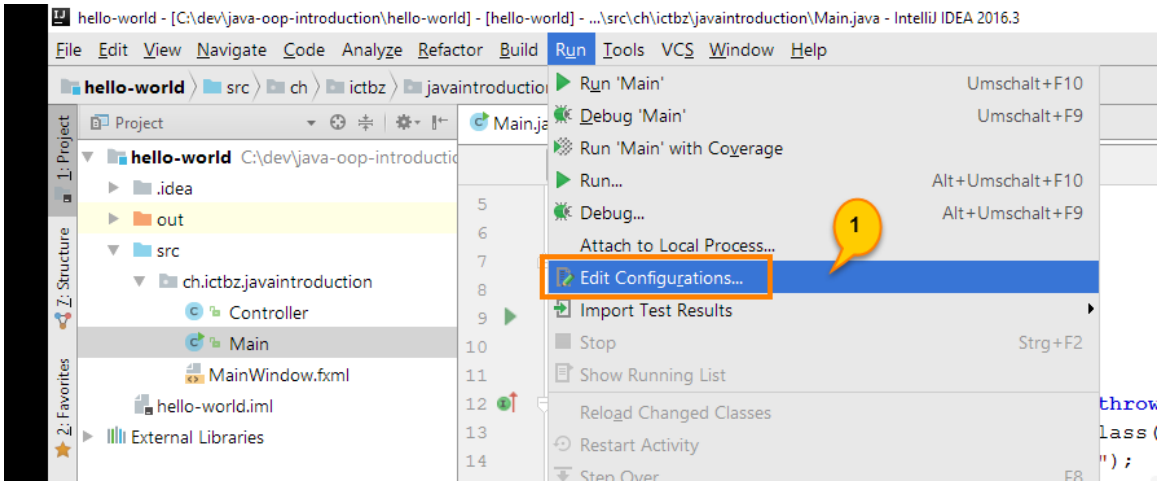
Gib der Datei den neuen Namen "MainWindow.fxml".



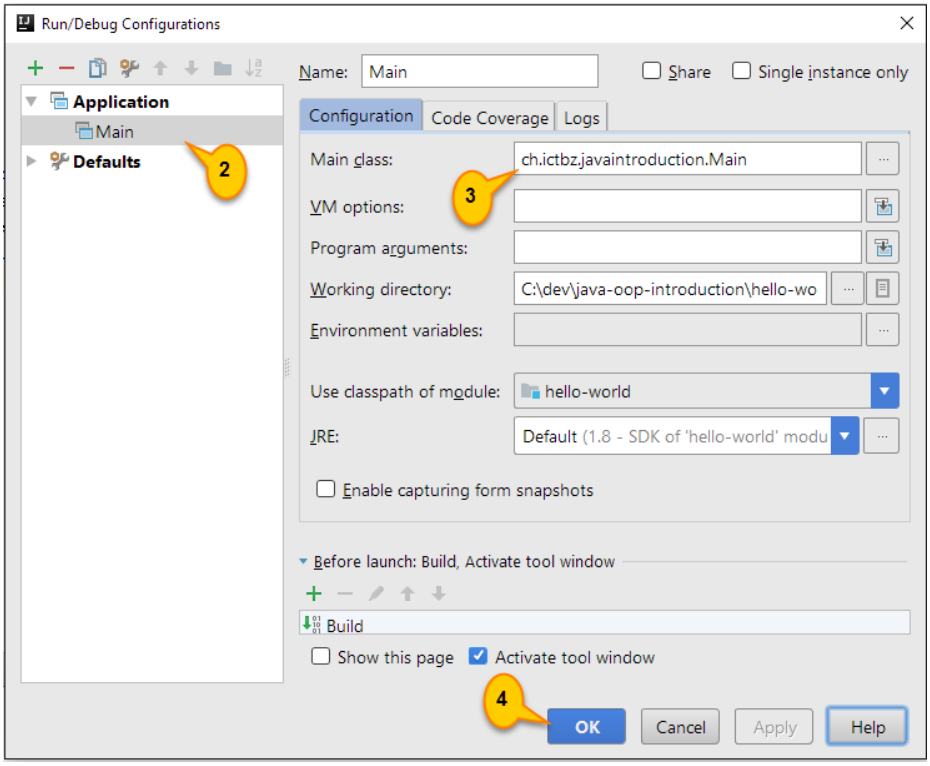
### 4.1.3.3 Die Run-Konfiguration erstellen

Damit Du nach nach dem erfolgten Refactoring Deine JavaFX Applikation starten kannst, musst Du in einem letzten Schritt noch die passende "Run-Konfiguration" erstellen. Führe dazu die folgenden Schritte aus.

1. Wähle in der Menu-Leiste den Eintrag *Run > Edit Configurations…*

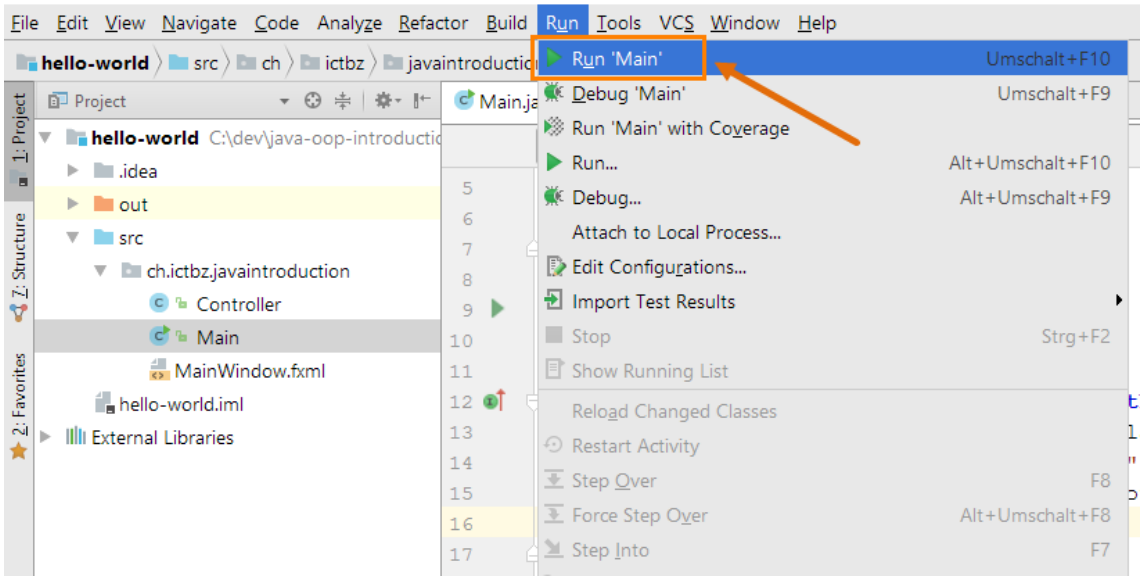



2. Aktiviere links in der Liste der vorhandenen Konfigurationen den Eintrag "Main".
3. Ändere im Reiter "Configuration" den Eintrag bei "Main class:" so, dass Deine `Main` Klasse referenziert wird (Name Deines Packages gefolgt von einem Punkt und dann der Name Deiner Main-Klasse)
4. Klicke "OK"

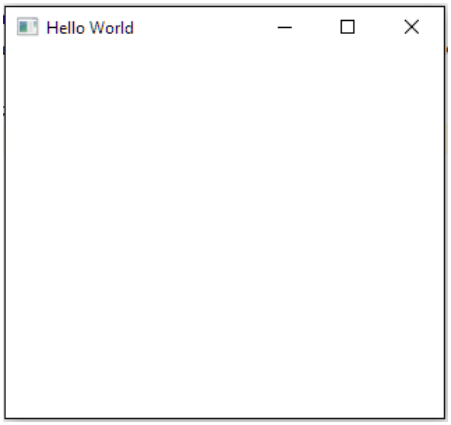


### 4.1.3.4 Die Run-Konfiguration testen

Teste Deine Run-Konfiguration indem Du Deine Applikation startest. Aktiviere hierzu über das Hauptmenu den Eintrag *Run > Run Main*.




Wenn das Refactoring und die Einstellung der Run-Konfiguration OK sind, startet nun Deine Applikation. Als Resultat solltest Du ein leeres Fenster mit einer Titelleiste (mit dem Minimize-, Maximizie- und Schliessen-Button) und dem Titel "Hello World" sehen.




Übrigens: Du kannst die Ausführung des Programms stoppen, indem Du in der Menu-Leiste den viereckigen roten Button klickst.


Im nächsten Kapitel wirst Du dieses leere Fenster erweitern und einen Button darauf platzieren.

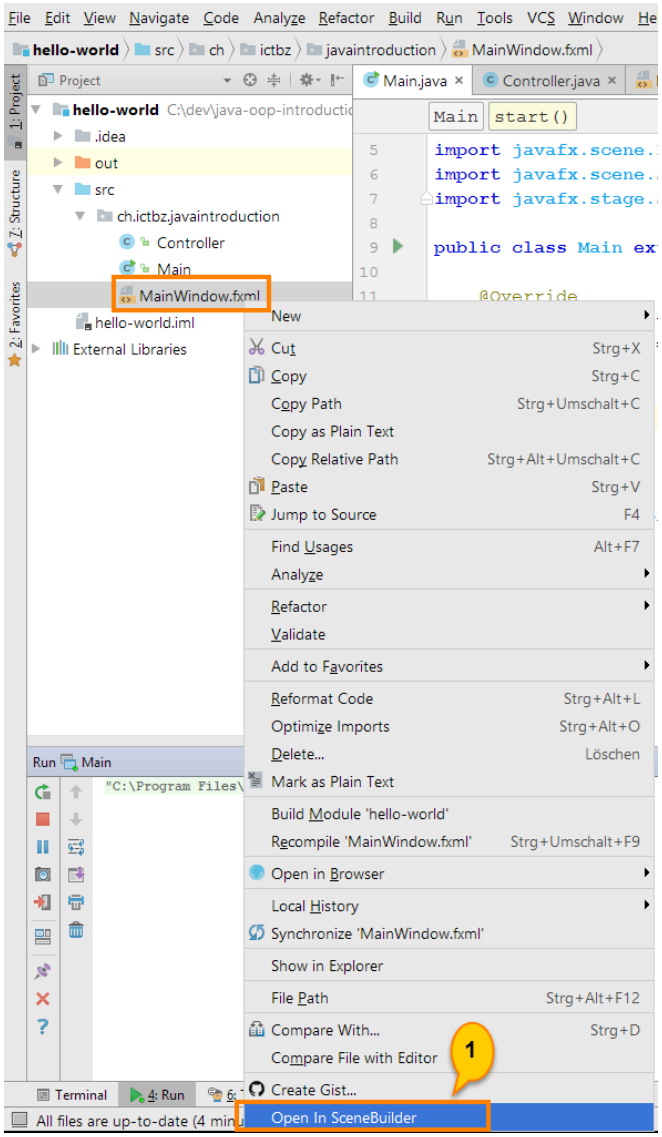
## 4.2 Das GUI-Design erstellen

Es gibt zwei Möglichkeiten, ein JavaFX Benutzerinterface zu erstellen: entweder alles mit Java zu programmieren oder aber deklaratives XML und FXML-Dateien zu verwenden. Wir werden die Variante XML verwenden, weil wir den "*Scene Builder"* nutzen wollen. Der Scene Builder ist ein grafischer Designer zum Erstellen von FXML-Dateien.

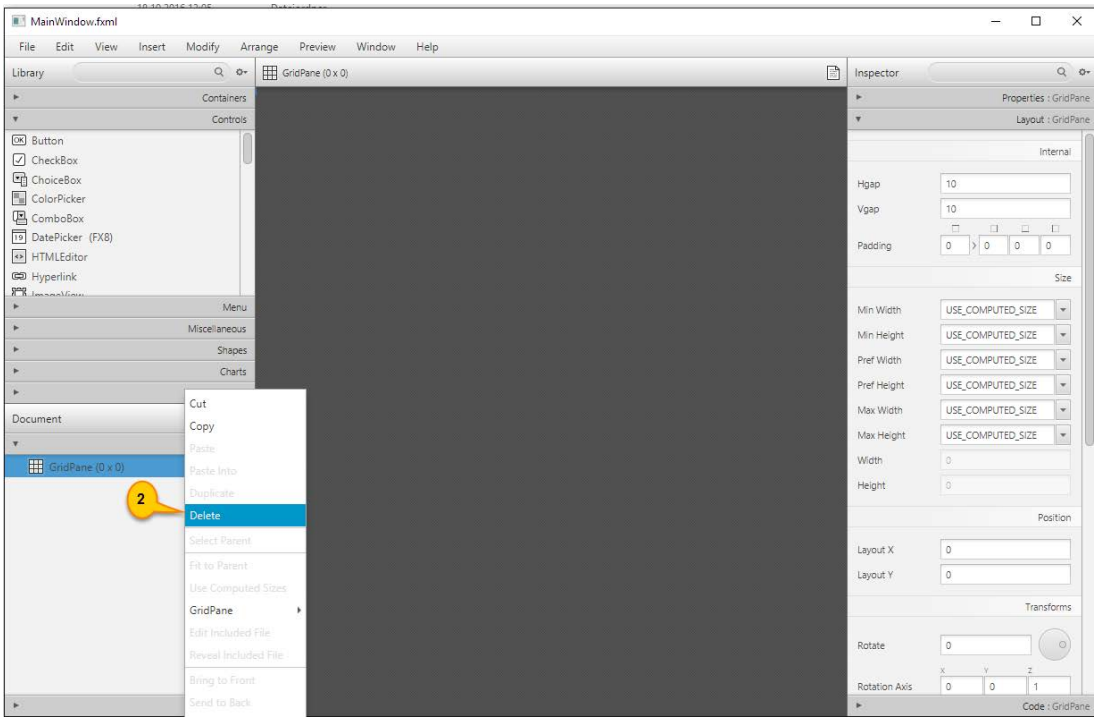
### 4.2.1 Ein AnchorPane verwenden

Verwende den Scene Builder, um ein AnchorPane (das ist eines von diversen Layouts, die uns JavaFX zur Verfügung stellt) zu initialisieren. Gehe dazu wie folgt vor.

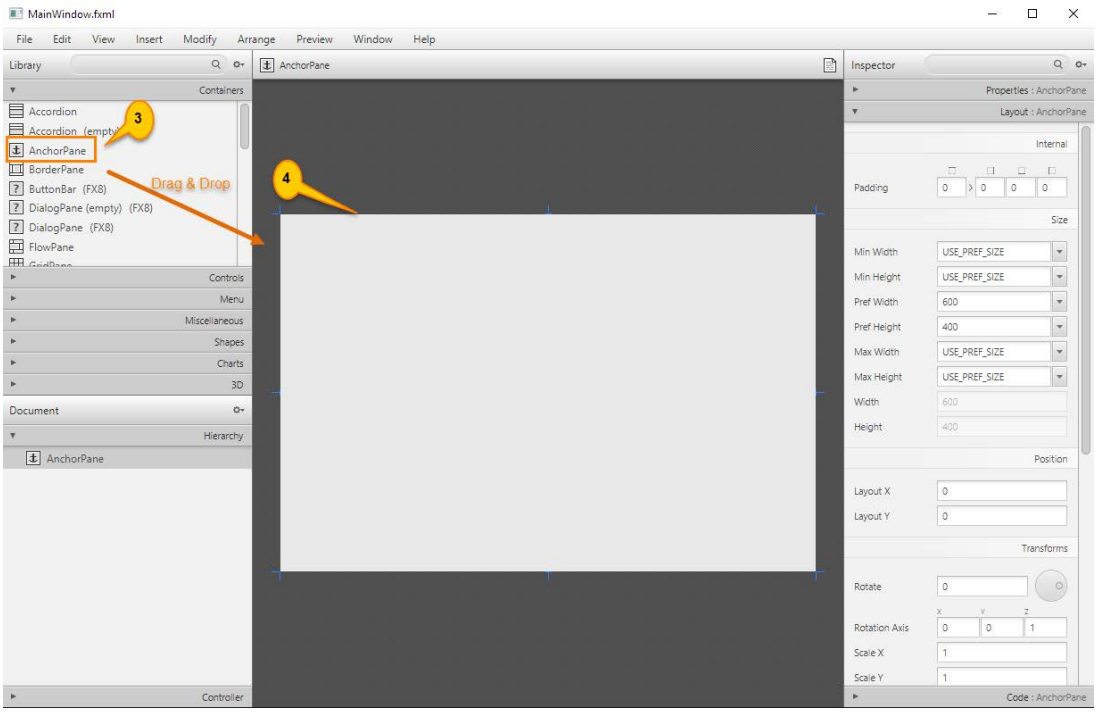
1. Navigiere in der Project View zur Datei MainWindow.fxml und öffne die Datei über Kontextmenu der rechten Maustaste im Scene Builder > *Open in SceneBuilder*



2. Im Scene Builder löschst Du bitte als erstes links unten den Eintrag *GridPane* aus der Hierarchie (falls er dort steht). Dies machst Du am besten über das Kontextmenü der rechten Maustaste.



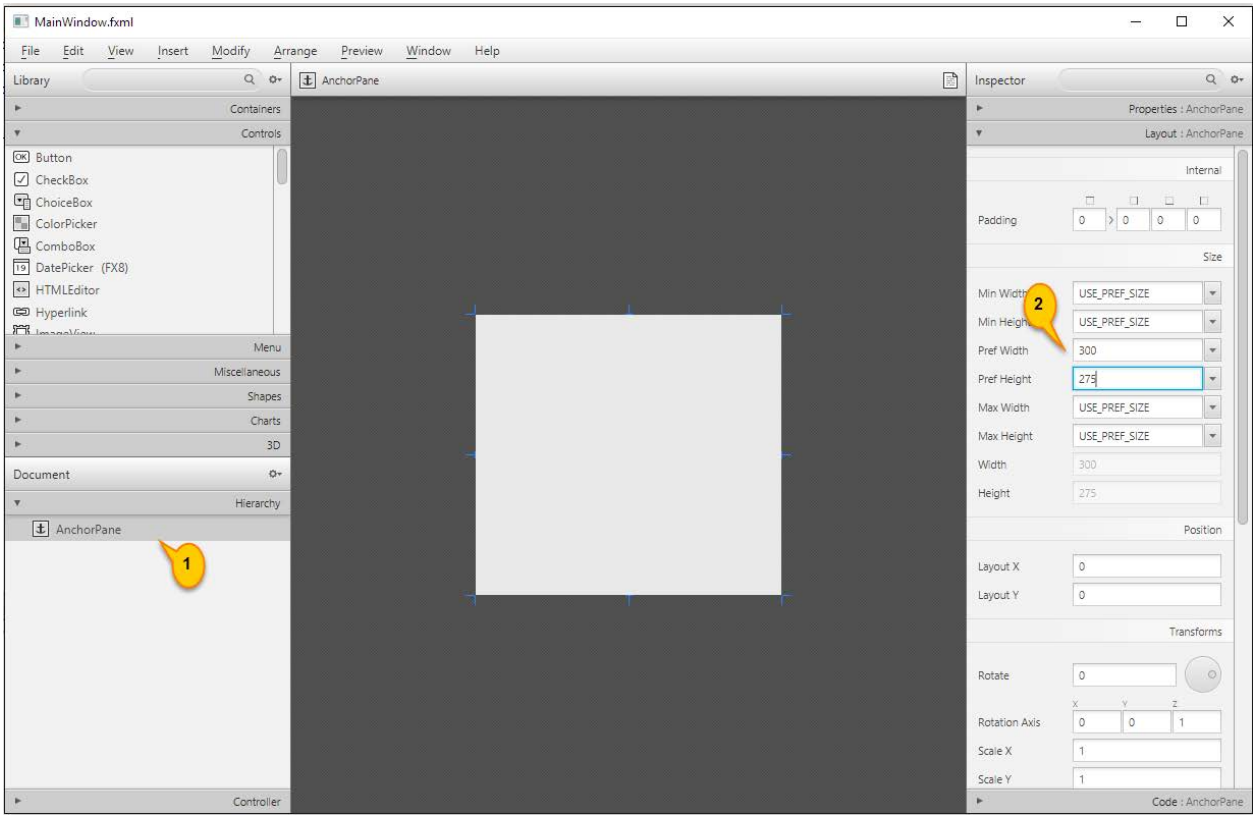

3. Nun wähle links aus den Containers das *AnchorPane* aus.
4. Kopiere das AnchorPane per Drag & Drop in den dunkelgrauen Designer-Bereich des Scene Builders.



#### 4.2.1.1 Das AnchorPane in der Grösse anpassen

Damit das AnchorPane innerhalb des JavaFX-Window (Scene) in der richtigen Grösse platziert wird, musst Du seine Breite und Höhe dem Window anpassen.

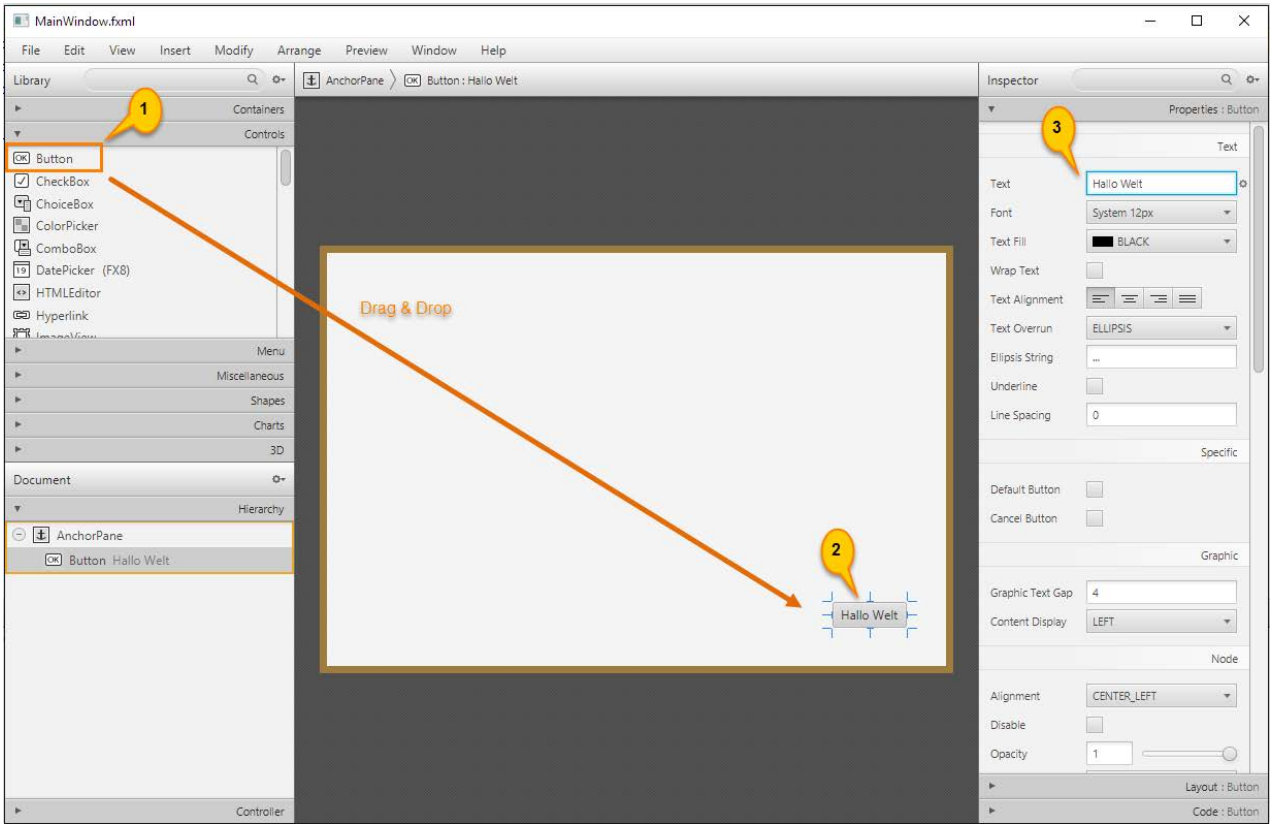
1. Wähle links im "Hierarchy"-Tab das AnchorPane aus
2. Rechts unter "Layout" kannst Du die Höhe sowie die Breite festlegen (*Pref Width* und *Pref Height*): setze die Breite auf 300 und die Höhe auf 275.



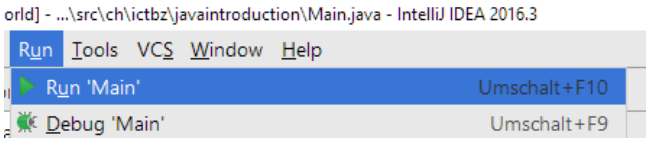

## 4.2.2 Einen Button zum AnchorPane hinzufügen

Als nächstes platzierst Du eine Schaltfläche (einen Button) auf dem AnchorPane.

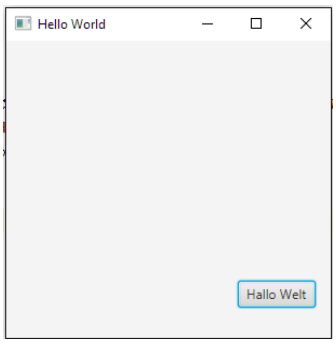
1. Aktivere den Bereich Controls und dort ganz zuoberst in der Liste das Steuerelement "Button"
2. Ziehe den Button mit der Maus auf das AnchorPane (Drag & Drop)
3. Beschrifte den Button mit "Hallo Welt" (indem Du im Scene Builder rechts bei den "Properties" die Eigenschaft "Text" setzt).




Speichere alles und wechsle zurück zu IntelliJ. Dann starte das Programm (*Run* > *Run Main*) um zu testen, ob der Button auf dem Window sichtbar ist.




Wenn alles korrekt funktioniert, ist nun der Button auf dem Window sichtbar.

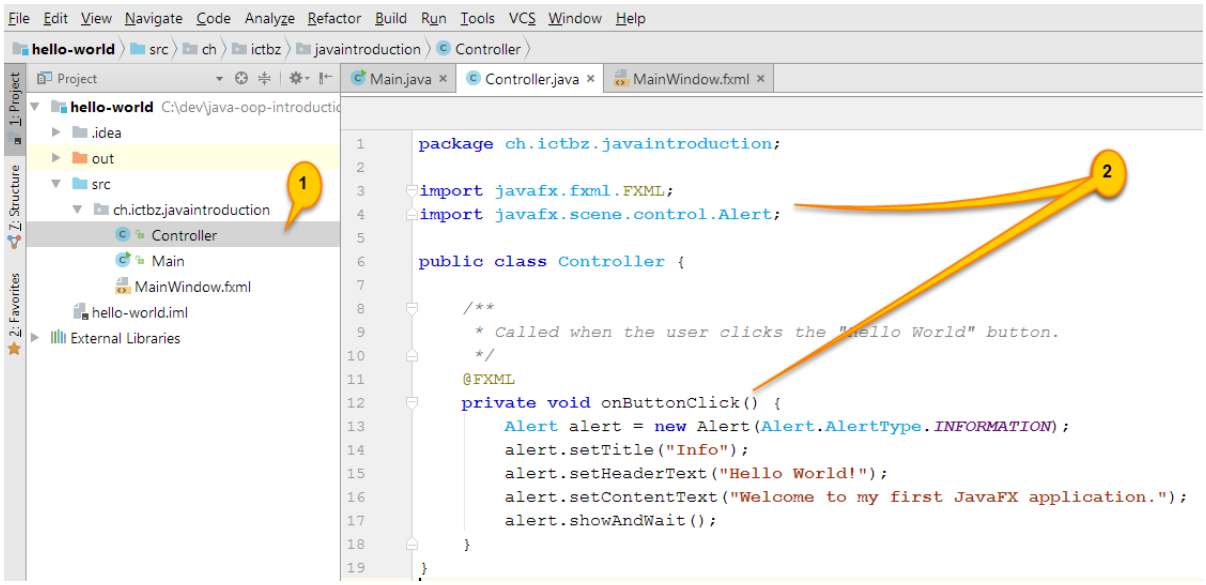


## 4.3 Auf Benutzerinteraktion reagieren

Soweit so gut: wir haben ein schönes Window mit einem schönen Button. Allerdings passiert nichts, wenn der Benutzer auf den Button klickt. Darum sollst Du nun dafür sorgen, dass dem Benutzer eine Meldung (ein "Alert") angezeigt wird, wenn er auf den Button klickt.

1. Navigiere in der Project View zur `Controller` Klasse und öffne das Codefenster für diese Klasse mit einem Doppelklick
2. Füge eine Methode *onButtonClick()* ein und darin der Java-Code, der einen Alert am Bildschirm ausgibt. Oberhalb dieser Methode musst Du die Annotation @FXML angeben, die bewirkt, dass die Methode mit der FXML-Datei verknüpft wird und dort dann zur Verfügung steht.

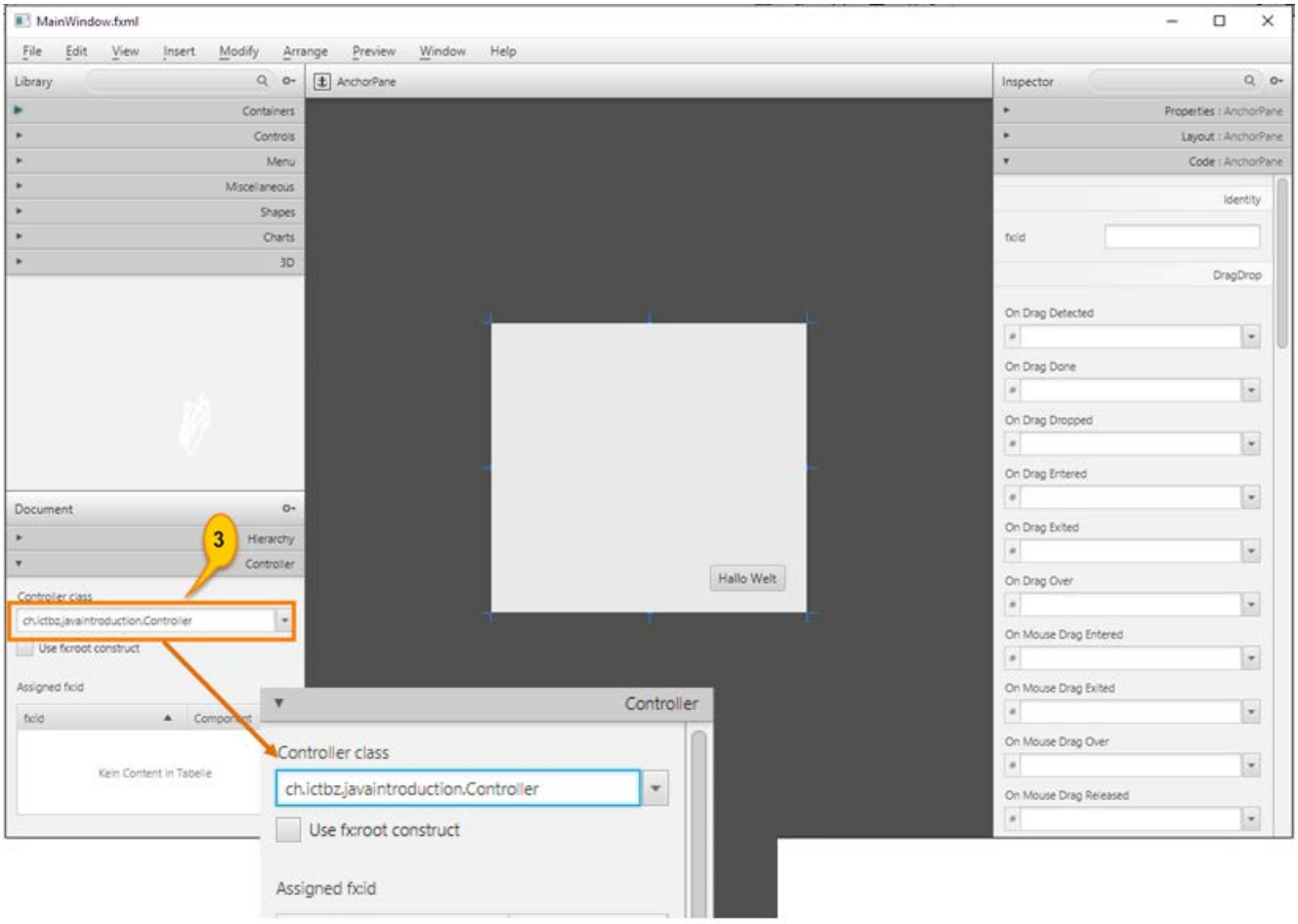
   Du kannst den Code gerne von unten kopieren, damit Du nicht alles abtippen musst (vergiss die `import` Anweisungen nicht!).



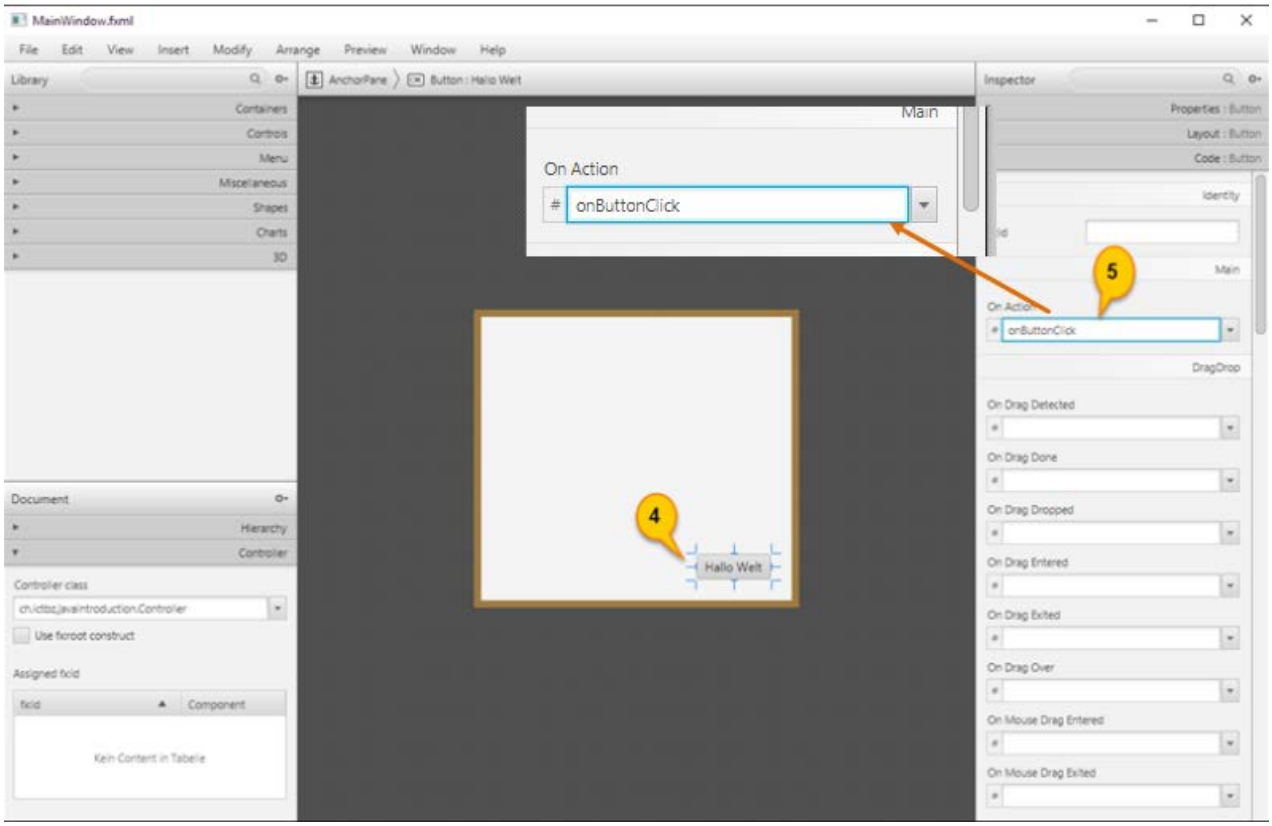

Der Code zum Kopieren:

```
package ch.ictbz.javaintroduction;

import javafx.fxml.FXML;
import javafx.scene.control.Alert;

public class Controller {
    /**
     * Called when the user clicks the "Hello World" button.
     */
    @FXML
    private void onButtonClick() {
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("Info");
        alert.setHeaderText("Hello World!");
        alert.setContentText("Welcome to my first JavaFX application.");
        alert.showAndWait();
    }
}
```

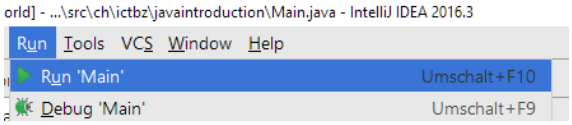
3. Nachdem Du die Ereignisbehandlung im Controller erstellt hast (speichern nicht vergessen!) wechselst Du zurück zum Scene Builder, wo Du links die Controller-Klasse auswählen kannst. Dies ist wichtig, damit der Scene Builder weiss, wo sich der Controller für das GUI befindet.



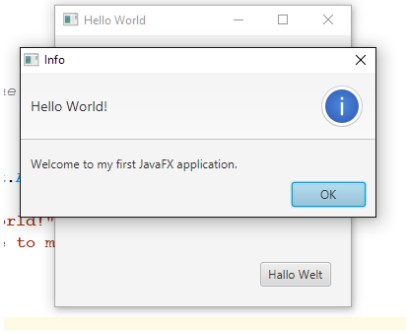
4. Markiere mit der Maus den "Hallo Welt" Button.
5. Wähle rechts unter *Code* > *On Action* die vorhin im Controller programmierte Methode "onButtonClick" aus. Damit sagst Du JavaFX, dass beim Klick auf den Button ebendiese Methode aufgerufen werden soll.




Speichere alles, wechsle zurück zu IntelliJ und starte die Applikation.




Klicke auf den Button: wenn Du alles richtig programmiert hast, wird eine Meldung am Bildschirm angezeigt.




**Herzliche Gratulation! Du hast soeben Deine erste JavaFX Anwendung programmiert!**