Programmierpraktikum/JavaFX

Programmierpraktikum/JavaFX

JavaFX-Projekt erzeugen, ausführen, einchecken

Inhaltsübersicht

zurück zur Übersicht

Leeres Projekt

[Bearbeiten]

Verwende dieses Programmgerüst zum Erstellen Deines JavaFX-Projektes (letzte Aufgaben der PS2-Übung und Programmierpraktikum).

```
└──src
   ├──main
   │   ├──java
   │   │   ├──gui
   │   │   │      ApplicationMain
   │   │   │      JarMain
   │   │   │      UserInterfaceController
   │   │   └──logic
   │   │          Logic
   │   └──resources
   │       └──gui
   │              UserInterface.fxml
   └──test
       └──java
           └──logic
javafx-pom.xml
pom.xml
```

Entpacke die zip-Datei in einen Ordner und benenne diesen wie den gewünschten Projektnamen. In der PS2-Übung verwende als Verzeichnisnamen "grpxx_uebyy" mit passender Gruppen- und Übungsnummer. Für das Programmierpraktikum nenne das Verzeichnis"pp_Spielname_MeinNachname". Ersetze auch in der `pom.xml` den Wert "grpxx_javafx" durch diesen Verzeichnisnamen.

Öffne erst danach dieses Projekt in *Intellij*. *Intellij* wird jetzt ein `.idea`-Verzeichnis mit Konfigurationsdateien erstellen.

Du kannst die `pom.xml` auch ändern, nachdem Du das Projekt in *Intellij* geöffnet hast, musst dann aber auch *Intellij*s Vorschlägen zur Übernahme der *MavenChanges* folgen.

JavaFX-Projekt ausführen [Bearbeiten]

Mit dieser Konfiguration lässt sich nur die **JarMain** ausführen, nicht aber die ApplicationMain.

Bei Ausführen der ApplicationMain erscheint der Fehler:
`Fehler: Zum Ausführen dieser Anwendung benötigte JavaFX-Runtime-Komponenten fehlen`

JavaFX-Projekt einchecken [Bearbeiten]

Die beiden `pom.xml` und `javafx-pom.xml` enthalten alle Konfigurationen für das Projekt (welche Bibliotheken verwendet und wie sie beim Kompilieren eingebunden werden sollen). Daher sollte das `.idea`-Verzeichnis auch **nicht** mit commitet werden, sondern ausschließlich das Projektverzeichnis mit `pom.xml` **und** `javafx-pom.xml` **und** `src` mit seinen Unterverzeichnissen.

Probleme bei der Projektanzeige/beim Kompilieren [Bearbeiten]

- Achte darauf, eine aktuelle Version von IntelliJ zu nutzen (mindestens 2021.2).
- Du musst zum Kompilieren JDK 17 oder neuer installiert haben.
- Lässt sich Dein Projekt nicht mehr richtig anzeigen oder kompilieren, versuche ein *Build/Rebuild* des Projektes.
- Manchmal erstellt *Intellij* eine `.iml`-Datei. Die gehört nicht in ein Maven-Projekt. Schließe dann IntelliJ, lösche die Datei und öffne das Projekt neu (aber nicht über Auswahl der zuletzt geöffneten Projekte).
- Hilft auch dies nicht, schließe das Projekt in *Intellij*, lösche das gesamte `.idea`-Verzeichnis und öffne das Projekt neu (aber nicht über Auswahl der zuletzt geöffneten Projekte).
- Leider behält *Intellij* in seinem Cache nicht immer die richtigen Konfigurationen. Nutze gerne ein *File/Invalidate Caches* oder *File/Repair IDE*.
- `WARNING: Unsupported JavaFX configuration: classes were loaded from 'unnamed module ...'` :
  Diese Warnung wird von uns toleriert. Es darf aber auch gerne eine `module-info.java` angelegt werden:
  Erstelle dafür im Verzeichnis `src/main/java` die Datei `module-info.java` (gibt es als eigenen Dateityp in *Intellij*). Der Name wird zum Projekt passen, aber z.B. Unterstriche durch Punkte ersetzen.
  Evtl. wird `junit` rot markiert und muss dem classpath zugefügt werden (dem Vorschlag von *Intellij* folgen).
  Hast Du in mehr Pakete strukturiert, kann es notwendig sein, ein Paket für den Zugriff zu öffnen (siehe auskommentierte opens-Beispiele):

```
module grpxx.javafx {
    requires javafx.controls;
    requires javafx.fxml;
    requires com.google.gson;
    requires junit;

    opens gui to javafx.fxml;              // to find UIController in gui and fxml-files in resources/gui
    opens logic to junit, com.google.gson; // to allow testing the logic and usage of gson
    // opens gui.img;                         // to find *.png in resources/gui/img
    // opens gui.fx to javafx.fxml;           // to find UIController in src/gui/fx

    exports gui;
}
```

  Die Einträge zu *gson* sind in der PS2-Übung nicht notwendig, sondern werden erst im Programmierpraktikum benötigt.

weitere Unterpakete anlegen [Bearbeiten]

Es ist sinnvoll, sein Projekt in zwei Pakete "gui" und "logic" aufzuteilen. Alle JavaFX-nutzenden Klassen gehören zum GUI und somit in das Paket (s.a. Programmaufteilung). Verschiebt man FXML-Datei und ihren Controller jeweils in ein andersnamiges Unterpaket, so muss die Position des Controllers in der FXML-Datei angepasst werden (funktioniert nicht über Refactoring). Hierfür kann man im SceneBuilder ganz links unten unter Controller die Controller-Klasse angeben oder direkt in der FXML-Datei das Attribut des äußersten Containers *fx:controller="gui.UserInterfaceController"* mit dem passenden Paketnamen vor dem Namen versehen.

Jar erzeugen und ausführen [Bearbeiten]

Am rechten Fensterrand findest Du die Möglichkeit, die Maven-Bedienung zu öffnen. 

Klappst Du diese auf, kannst Du mit Auswahl von *Lifecycle/package* eine `jar` erzeugen lassen. Diese findest Du dann im Ordner `target` mit der Bezeichnung `Projektname-1.0-SNAPSHOT.jar`.

Die `jar` ist auf Windows ausführbar und auf dem **Betriebsystem**, auf dem sie erstellt wurde.
Führe die `jar` in der **Powershell** (`java -jar pp_meins.jar`) aus. Dann siehst Du auch Ausgaben auf der Konsole wie z.B. Meldungen ungefangener Exceptions.

Sollte das Ausführen der `jar` sofort zu **Fehlermeldungen** führen, obwohl das Programm aus der IDE heraus funktioniert, liegt dies oftmals an Fehlern in der Dateiarbeit.
Sollten zum Ausführen der `jar` weitere Dateien (z.B. Libraries) notwendig sein, ist beim Erstellen der `jar` etwas schief gegangen. Möglicherweise hat *Intellij* das Projekt nicht als Maven-Projekt erkannt und eine .iml erstellt?

nach oben