

UE3 – Play Framework, JPA (25 Punkte)

In der zweiten Übung haben Sie eine serverseitige, MVC2-basierte Lösung auf Basis von Java-Technologien implementiert. Ziel dieses Übungsbeispiels ist es nun, diese Implementierung mit Hilfe des Play-Frameworks¹ umzusetzen und zu erweitern sowie eine Datenbankbindung zu integrieren.

Deadline der Abgabe via TUWEL²: **Sonntag, 10. Mai 2015 23:55 Uhr**

Nur ein Gruppenmitglied muss die Lösung auf TUWEL abgeben.

Jeopardy-Spiel

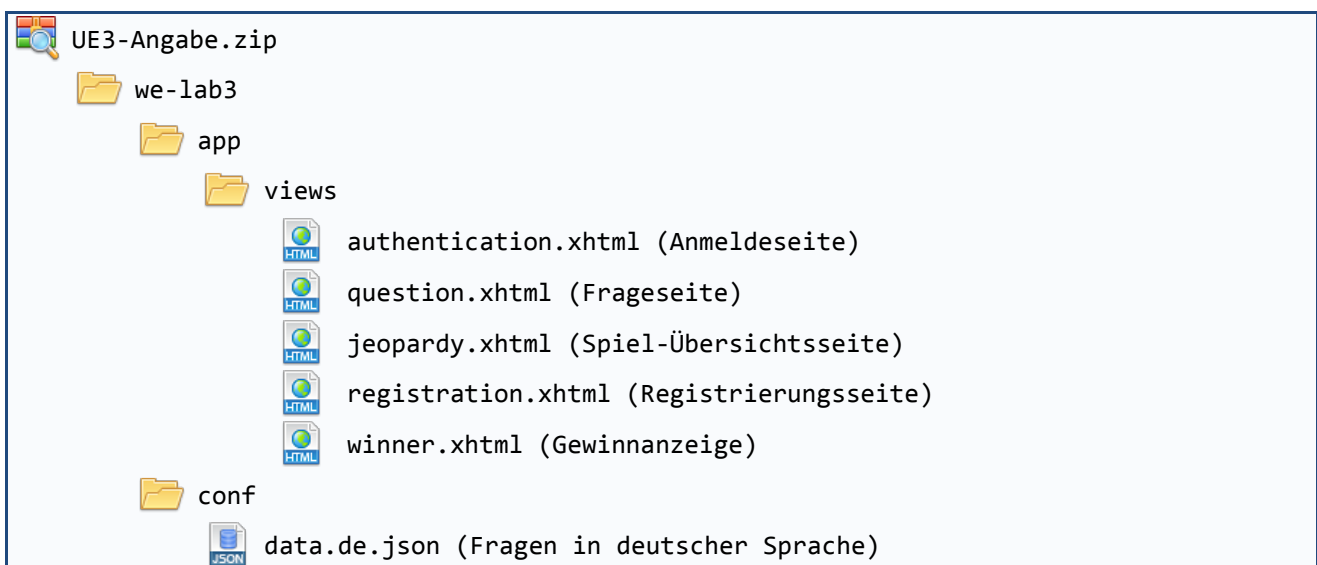
Bei „Jeopardy“ handelt es sich um ein Fragespiel, bei dem die KandidatInnen Fragen beantworten müssen, um Punkte zu sammeln. Jede Frage ist dabei einer Themenkategorie zugeordnet und hat einen bestimmten Wert, der den Schwierigkeitsgrad der Frage widerspiegelt. Eine Frage kann beliebig viele, aber mindestens zwei, Antwortmöglichkeiten haben, wobei mindestens eine dieser Antwortmöglichkeiten die korrekte Lösung auf die gestellte Frage ist („Multiple Choice“).

Ein Spiel besteht aus 10 Runden, wobei in jeder Runde eine Frage pro SpielerIn beantwortet wird. Der/Die SpielerIn (Mensch oder Computer) mit weniger Punkten darf zuerst eine Frage auswählen, bei Gleichstand wird der Mensch bevorzugt. Eine Frage gilt nur dann als richtig beantwortet, wenn alle korrekten Antwortmöglichkeiten ausgewählt und als Antwort an den Server übermittelt wurden. Zusätzlich gibt es bei jeder Frage ein Zeitlimit, bis zu dem diese Frage beantwortet werden muss. Ist dieses Zeitlimit abgelaufen, wird die aktuell ausgewählte Antwort gewertet.

Wurden von beiden SpielerInnen alle Fragen beantwortet, wird der Punktestand beider SpielerInnen ausgewertet und der/die GewinnerIn angezeigt. Herrscht Gleichstand, gewinnt der Computer.

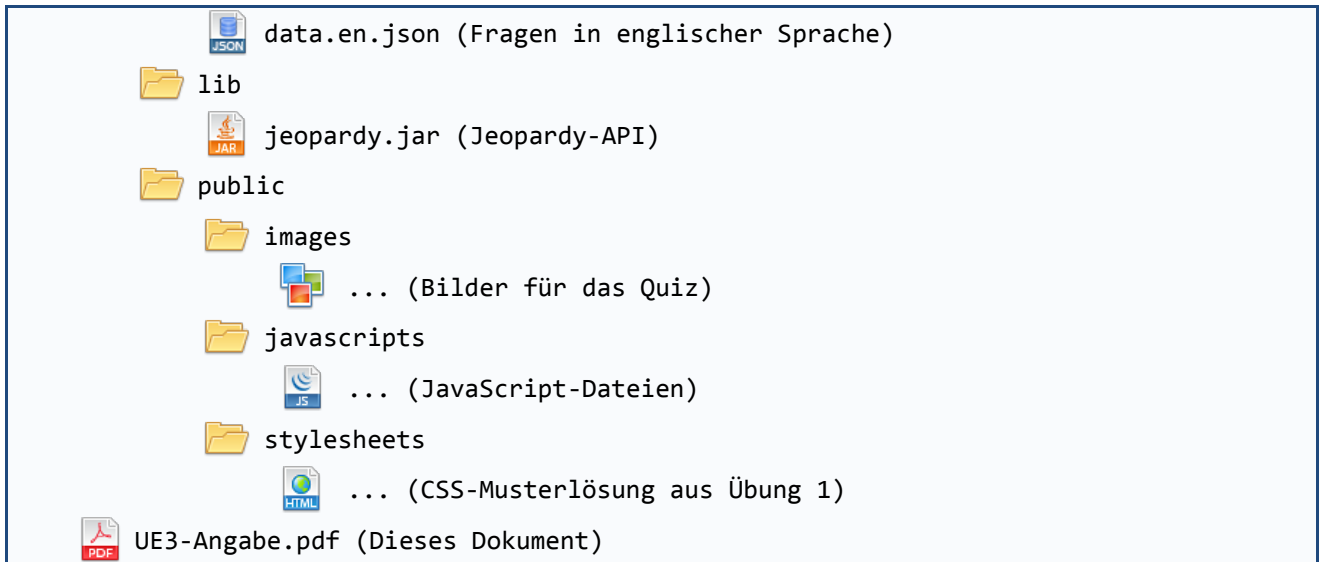
Angabe

Diese Angabe umfasst folgende Dateien:



¹ <http://www.playframework.com>

² <https://tuwel.tuwien.ac.at/course/view.php?id=5399>



Implementieren Sie mit Hilfe des Play Frameworks eine MVC2-Web-Applikation und achten Sie, wie auch mit bei Übung 2, auf eine Trennung von Model, View (nun Scala-HTML-Templates) und Controller. Legen Sie dazu erst eine Play-Anwendung an (siehe "Hinweise") und kopieren Sie die Dateien aus den Angaberessourcen in die gleichnamigen Verzeichnisse. Stellen Sie sicher, dass Ihre Anwendung mit der per Default erstellten Ressourcen läuft.

Wandeln Sie nun die von uns zur Verfügung gestellten XHTML5-Dateien (siehe Angaberessourcen) in entsprechende Scala-HTML-Templates³ um. Achten Sie darauf, dass dabei alle Element-IDs übernommen werden, da dies ist für die automatisierte Validierung Ihrer Lösung erforderlich ist. Das Aussehen der Seiten soll so gut es geht erhalten bleiben. Diese Templates (Endung muss .scala.html sein!) dienen als View. Erstellen Sie außerdem die Controller-Klassen.

Hauptanforderungen an Ihre Implementierung:

- **Seitenfluss:** Wird die Anwendung das erste Mal aufgerufen, soll der/die BenutzerIn auf die Anmeldeseite (authentication.xhtml) weiter geleitet werden. Von dort aus, kann er sich entweder anmelden oder auf die Registrierungsseite (registration.xhtml) navigieren. Nach erfolgreicher Registrierung wird der Benutzer wieder zurück auf die Anmeldeseite geleitet. Bei Fehlern im Registrierungsformular werden diese Fehler direkt beim Formular angezeigt. Ebenso werden Fehler beim Anmelden direkt bei der Anmeldeseite angezeigt. Hat sich der/die BenutzerIn erfolgreich angemeldet, so wird ein neues Spiel gestartet und der/die BenutzerIn wird auf die Übersichtseite (jeopardy.xhtml) weitergeleitet, von der eine Frage ausgewählt und anschließend beantwortet werden kann (question.xhtml). Am Ende des Spiels wird der/die GewinnerIn angezeigt (winner.xhtml). Auf der Gewinnanzeige-Seite hat man zusätzlich die Möglichkeit ein neues Spiel zu beginnen. In diesem Fall wird man direkt auf die Übersichtseite weitergeleitet. Meldet sich der/die SpielerIn während dem Spielen ab (Klick auf "Abmelden"), so wird das aktuell laufende Spiel abgebrochen und die Anmeldeseite angezeigt.

³ <http://www.playframework.com/documentation/2.3.8/JavaTemplates>

Verwenden Sie *Controller*-Klassen⁴ und *routes*⁵ um den Seitenfluss entsprechend dem Spielzustand zu steuern. Achten Sie auf eine korrekte Verwendung der HTTP-Methoden (GET, POST, etc.).

- *Spielablauf*: Das Spiel muss den am Anfang dieses Dokuments beschriebenen Regeln folgen (siehe Jeopardy-Spiel). Dabei können Sie davon ausgehen, dass die von uns zur Verfügung gestellten Fragen (data.*.json) bereits den Kriterien entsprechen (ausreichend Fragenkategorien, mindestens eine korrekte Antwortmöglichkeit etc.).

Implementieren Sie hierfür Controller-Klassen, welche die zur Verfügung gestellte API (jeopardy.jar, siehe auch Hinweise) verwenden.

- *Registrierung*: SpielerInnen sollen die Möglichkeit haben sich zu registrieren (registration.xhtml) und sich anzumelden (authentication.xhtml). Die Validierung der Registrierungsfelder soll im Gegensatz zu den bisherigen Übungen nun serverseitig erfolgen, d.h., eine Überprüfung mit Hilfe von JavaScript ist in dieser Übung nicht mehr notwendig. Die Kriterien der Felder können Sie der Angabe aus Übung 1 entnehmen. Zusätzlich gilt natürlich, dass es nicht zwei Benutzer mit demselben Benutzernamen geben darf.

Hierfür können Sie beispielsweise die Formular-Helper in Play⁶ verwenden.

- *Authentication*: SpielerInnen, die nicht eingeloggt sind, sollen auf die anderen Seiten der Anwendung keinen Zugriff haben.

Verwenden Sie hierfür den Security-Mechanismus von Play, d.h., implementieren Sie einen eigenen `Security.Authenticator`⁷ und verwenden Sie entsprechende Annotationen.

- *Datenbankanbindung*: Die Spielerdaten aus der Registrierung sollen in einer In-Memory-Datenbank (H2) gespeichert werden und im Falle eines Logins wieder aus dieser ausgelesen werden.

Implementieren Sie daher Ihre eigene User-Modell-Klasse (welche das Interface `User` implementiert) und verwenden Sie die Annotationen der Java Persistence API (siehe auch Hinweise).

- *Internationalisierung (i18n)*⁸: Ihre Web-Anwendung soll sowohl für deutschsprachige als auch für englischsprachige BenutzerInnen zugänglich sein. Integrieren Sie diese beiden Sprachen in Ihre Anwendung, in dem Sie sprachspezifische Meldungen extern verwalten und aufgrund der ausgewählten Sprache des Benutzers die korrekte Frage-Datei laden. Es ist für diese Übung ausreichend, die Fragen einmal am Anfang des Spiels korrekt zu laden, eine laufende Überprüfung der Sprache ist nicht notwendig.
- *Standards und Barrierefreiheit*: Das User Interface muss den Anforderungen von XHTML5 sowie WCAG-AA gerecht werden.

⁴ <http://www.playframework.com/documentation/2.3.8/JavaActions>

⁵ <http://www.playframework.com/documentation/2.3.8/JavaRouting>

⁶ <http://www.playframework.com/documentation/2.3.8/JavaForms>

⁷ <http://www.playframework.com/documentation/2.3.8/api/java/play/mvc/Security.Authenticator.html>

⁸ <http://www.playframework.com/documentation/2.3.8/JavaI18N>

Hinweise

Validierung

Verwenden Sie zur Validierung Ihrer XHTML Dateien den Validator <http://validator.nu/> und für Ihre CSS Dateien den vom W3C zur Verfügung gestellten Validation-Service <http://jigsaw.w3.org/css-validator/>. Beachten Sie, dass der Typ "date" für Eingabefelder derzeit nicht in allen Browsern unterstützt wird. Eine entsprechende Warnung bei der Validierung dürfen Sie in diesem Fall ignorieren. Zur Überprüfung der WAI-Tauglichkeit stehen Ihnen eine Vielzahl von Services im Internet zur Verfügung (z.B. AChecker, <http://achecker.ca/checker/index.php>). Nähere Infos dazu finden Sie in den Folien bzw. im TUWEL.

Anlegen einer Play-Anwendung








Die folgende Anleitung gibt nur einen kurzen Überblick über den typischen Ablauf der Anwendungserstellung in Play. Für detailliertere Informationen, lesen Sie bitte die Vorlesungsfolien (siehe TUWEL), die darin verlinkten Code-Beispiele, sowie die Dokumentation auf der Play-Webseite.

1. Laden Sie die aktuelle Version des Play-Frameworks herunter und befolgen Sie die Anleitung zur Installation des Play-Frameworks⁹.
2. Führen Sie zum Anlegen einer neuen Anwendung in der Konsole/Kommandozeile folgenden Befehl aus¹⁰:

```
activator new we-lab3-group<GruppenNr> play-java
```

Geben Sie beim Applikationsnamen denselben Namen (we-lab3-group<X>) noch einmal ein oder klicken Sie <enter>. Wählen Sie als Template die zweite Option (Java application).




Folgende Dateien und Verzeichnisse sollten für Sie erstellt worden sein¹¹:

 we-lab3-group<X>	Anwendungsverzeichnis
 app	Ausführbare Artefakte
 controllers	Java-Controller-
 views	Scala-HTML-Templates
 conf	
 application.conf	Haupt-Konfigurations-Datei
 routes	Definierte Routen ("Redirects")
 project	
 build.properties	Marker für das sbt-Projekt
 plugins.sbt	sbt Plugins inkl. Play
 public	
 images	Bilder
 javascripts	JavaScript-Dateien

⁹ <http://www.playframework.com/documentation/2.3.8/Installing>

¹⁰ <http://www.playframework.com/documentation/2.3.8/NewApplication>

¹¹ <http://www.playframework.com/documentation/2.3.8/Anatomy>

 stylesheets	CSS-Dateien
 test	Auto-generierte JUnit-Tests
 build.sbt	Build-Script für die Anwendung

3. Wechseln Sie in das neu erstellte Verzeichnis und starten Sie die Applikation¹²:

```
cd we-lab3-group<GruppenNr>
activator start // "activator run" for developer mode
```

Während die Applikation kompiliert und gestartet wird, werden im Anwendungsverzeichnis neue Verzeichnisse angelegt: logs für Log-Dateien und target, project/project, und project/target für kompilierte und generierte Ressourcen.

4. Per Default bindet Play die neu erstellte Applikation an den Port 9000. Testen Sie die Applikation, indem Sie die folgende Adresse im Browser aufrufen.

<http://localhost:9000/>

Integration der H2-In-Memory-Datenbank der JPA in Play¹³

1. Fügen Sie die Abhängigkeiten in Ihrem Build-Script hinzu:

```
build.sbt

[...]
```

```
libraryDependencies += Seq(
  javaJdbc,
  javaCore,
  javaJpa,
  "org.hibernate" % "hibernate-entitymanager" % "4.3.1.Final"
)
```

2. Passen Sie die application.conf an, sodass H2 als Datenquelle verwendet wird und binden Sie die Datenquelle mit einem Namen an JNDI. Kommentieren Sie dafür folgende Absätze wieder ein (= Entfernung von #) oder fügen Sie sie hinzu:

```
conf/application.conf

db.default.driver=org.h2.Driver
db.default.url="jdbc:h2:mem:play"

db.default.jndiName=DefaultDS
jpa.name=defaultPersistenceUnit
jpa.default=defaultPersistenceUnit
```

¹² <http://www.playframework.com/documentation/2.3.8/PlayConsole>

¹³ <http://www.playframework.com/documentation/2.3.8/JavaJPA>

3. Konfigurieren Sie JPA mit Hilfe einer persistence.xml im conf/META-INF-Verzeichnis:

conf/META-INF/persistence.xml

```
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
    http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd"
  version="2.0">
```

```
<persistence-unit name="defaultPersistenceUnit" transaction-type="RESOURCE_LOCAL">
  <provider>org.hibernate.ejb.HibernatePersistence</provider>
  <non-jta-data-source>DefaultDS</non-jta-data-source>
  <properties>
    <property name="hibernate.dialect" value="org.hibernate.dialect.H2Dialect"/>
    <property name="hibernate.hbm2ddl.auto" value="create" />
  </properties>
</persistence-unit>
</persistence>
```

4. Verwenden Sie die JPA-Annotation über den Play-Actions, die auf JPA-Funktionen zugreifen.

```
@play.db.jpa.Transactional
public static Result index() {
  ...
}
```

Verwendung der beigefügten Spiel-API

Kopieren Sie die beigefügte API (big-quiz-api.jar) in ein (neu erstelltes) Verzeichnis "lib" und fügen Sie die notwendigen Abhängigkeiten in Ihrem Build-Script hinzu:

build.sbt

```
libraryDependencies += Seq(
  [...],
  "com.google.code.gson" % "gson" % "2.2"
)

templatesImport += "scala.collection._"
templatesImport += "at.ac.tuwien.big.we15.lab2.api._"
```

Die API selbst können Sie nun wie folgt in Java verwenden.

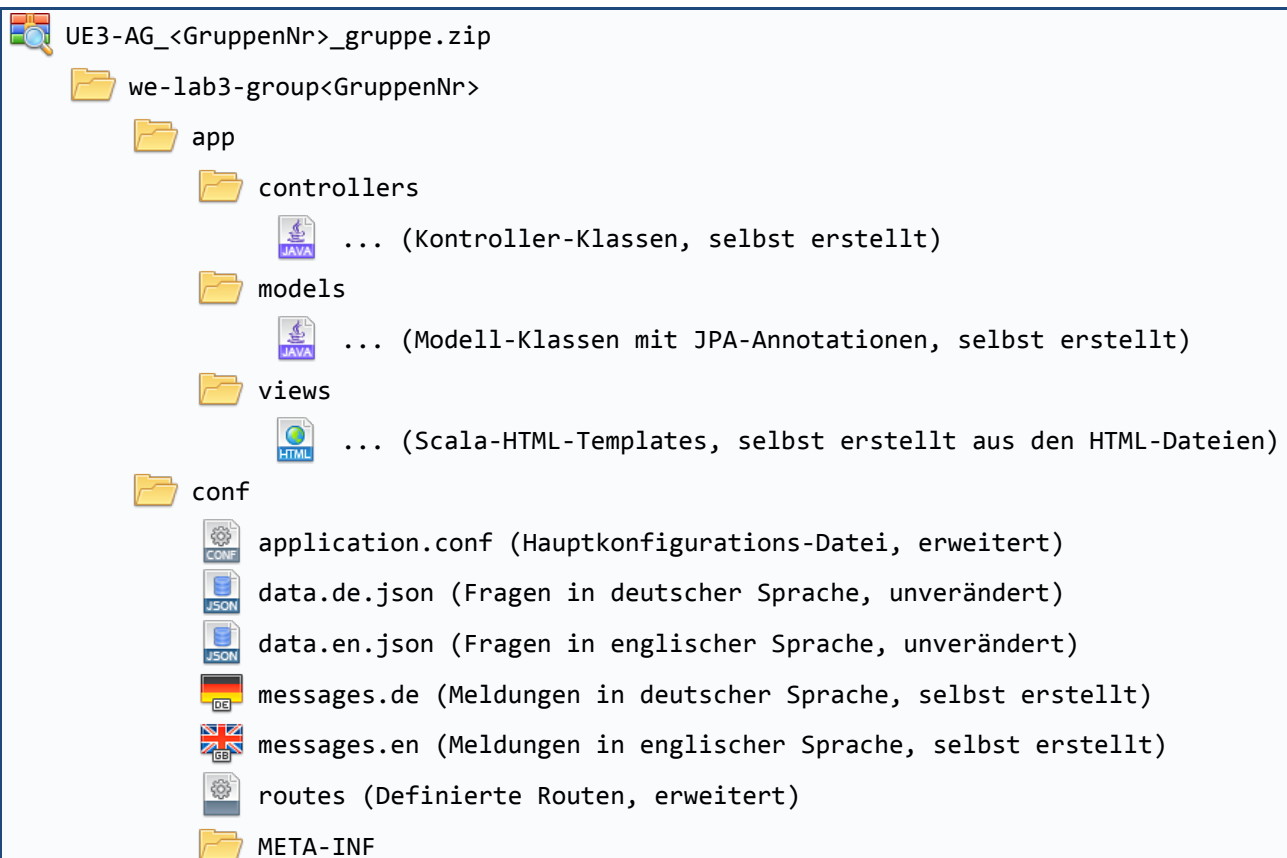
```
import at.ac.tuwien.big.we15.lab2.api.*;
...
JeopardyFactory factory = new PlayQuizFactory(jsonFilePat);
JeopardyGame game = factory.createGame(user);
Player human = game.getHumanPlayer(); // get human player
human.getProfit(); // get current profit of player
human.getLatestProfitChange(); // get last profit change (for display)
game.chooseHumanQuestion(questionId); // choose next human question
game.answerHumanQuestion(answerIds); // answer current human questions
game.isRoundStart(); // check if we are at the beginning of a new round
game.isAnswerPending(); // check if the current question needs to be answered
game.isGameOver(); // check if game is over
game.getWinner(); // winner of round or null if no winner exists yet
```

Testen während der Entwicklung

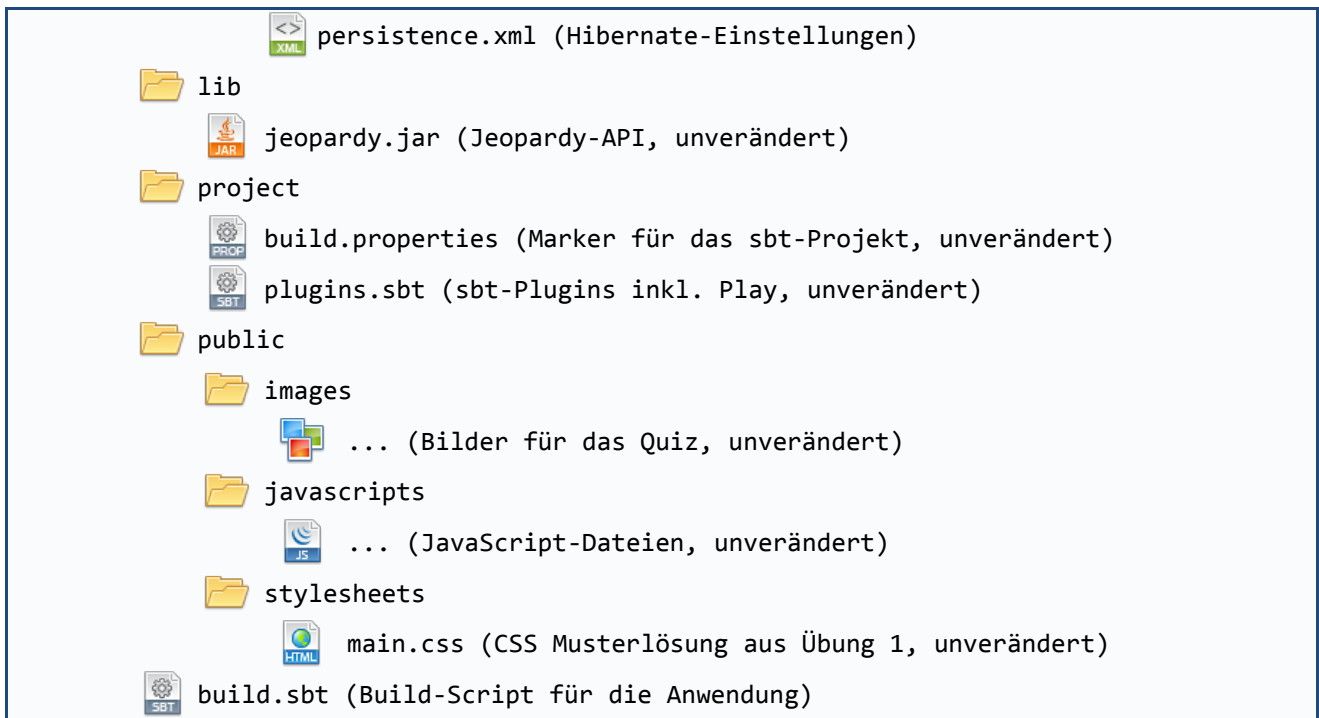
Das Neukompilieren während die Play-Anwendung in einem Browser läuft kann zu Null Pointer Exceptions führen. Um dieses Problem zu vermeiden können Sie ein Inkognito-Fenster in Ihrem Browser verwenden.

Abgabemodalität

Beachten Sie die allgemeinen Abgabemodalitäten des TUWEL-Kurses¹⁴. Zippen Sie Ihre Abgabe, sodass sie die folgende Struktur aufweist:



¹⁴ <https://tuwel.tuwien.ac.at/course/view.php?id=6324>



Alle Dateien müssen UTF-8 codiert sein!

ACHTUNG: Wird das Abgabeschema nicht eingehalten, so kann es zu Punkteabzügen kommen!