



UE4 – Datenbankbindung und Web Services (25 Punkte)

Ziel dieses Übungsbeispiels ist die Erweiterung der JSF-Web-Applikation aus Übung 3 um eine Datenbankbindung. Des Weiteren sollen über entsprechenden Web Services zusätzliche Anwendungen eingebunden werden.

Deadline der Abgabe via TUWEL¹: **Montag, 3. Juni 2013 23:55 Uhr**
Abgabegespräche: **Mittwoch, 5. Juni 2013**

Abgabemodalität

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

- src/main/java (Ordner)
 - ... (Java Quellcode)
- src/main/resources (Ordner)
 - META-INF (Ordner)
 - persistence.xml
 - ... (Properties Dateien für Internationalization)
- src/main/webapp (Ordner)
 - META-INF (Ordner)
 - context.xml
 - WEB-INF (Ordner)
 - web.xml (Angepasste web.xml)
 - index.xhtml (Login)
 - register.xhtml
 - table.xhtml
 - resources
 - CSS-Datei, jquery.js, Bilder, Custom Component
- pom.xml

Angabe

Erweitern Sie die Abgabe aus Übung 3 um nachstehende Funktionalität. Die in Übung 3 erstellte API kann weiterverwendet und erweitert werden. Sie können jedoch auch auf die zur Verfügung gestellte Musterlösung von UE3 aufbauen. Verwenden Sie als User Interface den XHTML- und CSS-Code der von uns zur Verfügung gestellten Musterlösung für Übung 1, die in den Angaberessourcen dieser Übung zu finden ist (verändern Sie keine ID Attributwerte - diese ist für die automatisierte Validierung Ihrer Lösung erforderlich!).

Nachfolgend werden für gewisse Elemente bestimmte Ids verlangt welche kein Vorschlag sondern Voraussetzung dafür sind, dass Ihre Lösung positiv bewertet werden kann! Bitte beachten Sie, dass beim Render-



Schritt durch die Verwendung von Naming Container (z.B. h:form) die Ids von Elementen, welche in einem Naming Container liegen, mit der Id des Naming Containers geprefixt werden. Wenn in den Anforderungen zu Ids der Text "respektive 'form:XXX'" steht, dann ist damit gemeint, dass die Elemente mit der jeweiligen Id "XXX" in dem NamingContainer mit der Id "form" liegen sollen.

Anforderungen an Ihre Implementierung:

1. Datenbankankbindung (5 Punkte)

Erweitern Sie die Registrierung aus UE3 so, dass die User Daten persistent in einer relationalen Datenbank gespeichert werden (Tabelle „Player“). Zusätzlich zu den bestehenden Attributen (Personal Data und Login Data) wird eine Referenz auf einen bei der Registrierung gewählten Avatar gespeichert (Siehe Punkt 4). Ein Avatar kann von mehreren Usern gewählt werden. Avatare werden in einer separaten Tabelle persistiert (Tabelle „Avatar“).

Die Login Funktion soll in weiterer Folge im Login Formular angegebene User Namen und Passwörter gegen in der Datenbank gespeicherte Userdaten validieren.

Verwenden Sie dafür die Java Persistence API, die O/R Mapper Implementierung Hibernate¹, sowie als Datenbank die In-Memory Datenbank H2². Die Konfiguration der Datenbankankbindung, sowie der Persistence Unit sind bereits in den Angaberessourcen (/src/main/webapp/META-INF/context.xml und /src/main/resources/META-INF/persistence.xml) vorgegeben.

Hinweis: H2 stellt ein Console Servlet bereit, mit dessen Hilfe Sie die von der Java Persistence API erstellten Tabellen und persistierten Daten einsehen können. Folgende Konfiguration ist im web.xml dafür notwendig:

```
<servlet>
  <servlet-name>H2Console</servlet-name>
  <servlet-class>org.h2.server.web.WebServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>H2Console</servlet-name>
  <url-pattern>/console/*</url-pattern>
</servlet-mapping>
```

Zusammenfassung der geforderten Funktionalität:

Die um einen Avatar erweiterten User Daten sollen in der Datenbank persistiert werden (2 Tabellen). Beim Login soll die Überprüfung von Benutzernamen und Passwort über die Datenbank erfolgen.

¹ <http://www.hibernate.org/>

² <http://www.h2database.com>



2. Integration eines Highscore Services (5 Punkte)

Nach erfolgreichem Abschluss eines Spiels sollen die Spielergebnisse automatisch auf einem Highscoreboard gepostet werden. Für jedes Spiel soll dabei jeweils genau einmal das Highscoreergebnis gespeichert werden. Highscoreergebnisse können mittels eines SOAP/WSDL Web Services gepostet werden.

URL des Highscoreboard:

<http://playground.big.tuwien.ac.at:8080/highscore/>

Service URL:

<http://playground.big.tuwien.ac.at:8080/highscore/PublishHighScoreService>

User-Key:

34EphAp2C4ebaswu

Die Highscoreergebnisse werden mit Hilfe der Datenstruktur übermittelt, welche Sie im Rahmen der VU Semistrukturierte Daten entworfen haben. Dabei müssen vom Schema jedoch nicht alle Attribute und Elemente übermittelt werden, sondern nur jene, die in nachfolgendem Ausschnitt beschrieben sind.

Für `start-date`, `end-date` und `registration-date` nehmen Sie das aktuelle Datum. Es wird genau ein `player` mit `date-of-birth` und `gender` übergeben.

Es wird genau eine `round` übergeben, welche die `number` '0' haben muss. In der `round` ist genau ein `game` enthalten. `date` entspricht dabei wieder dem aktuellen Datum. Der `status` muss 'finished' sein. Unter `duration` übergeben Sie die Anzahl der Sekunden, die während des Spiels verstrichen sind (= Spieldauer). Unter `winner` übergeben Sie den Gewinner des Spiels (= entweder 'Computer' oder der Name des Spielers).

Wenn Sie alle Elemente und Attribute korrekt übergeben haben, bekommen Sie im Response eine UUID retourniert, welche Sie dann für die nächste Teilaufgabe benötigen. Wenn Sie fehlerhafte Daten übergeben, werden entsprechenden Fehlermeldungen vom Service retourniert.



```
<data:HighScoreRequest xsi:schemaLocation="http://big.tuwien.ac.at/we/highscore/data wehighscore-
datatypes.xsd" xmlns:data="http://big.tuwien.ac.at/we/highscore/data"
xmlns:ssd="http://www.dbai.tuwien.ac.at/education/ssd/SS13/uebung/Tournament"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <data:UserKey>34EphAp2C4ebaswu</data:UserKey>
  <ssd:tournament start-date="2012-04-23" end-date="2012-04-23" registration-
deadline="2013-04-23T00:00:00Z">
    <ssd:players>
      <ssd:player username="Max Mustermann">
        <ssd:date-of-birth>1965-08-13</ssd:date-of-birth>
        <ssd:gender>MALE</ssd:gender>
      </ssd:player>
    </ssd:players>
    <ssd:rounds>
      <ssd:round number="0">
        <ssd:game date="2012-04-23" status="finished" duration="23"
winner="The computer">
          <ssd:players/>
        </ssd:game>
      </ssd:round>
    </ssd:rounds>
  </ssd:tournament>
</data:HighScoreRequest>
```

Zusammenfassung der geforderten Funktionalität:

- Highscoreergebnisse müssen auf dem Highscoreboard veröffentlicht werden und die UUID muss korrekt retourniert werden
- Damit man das korrekte Ansprechen des Highscore Services nachvollziehen kann, muss die retournierte UUID als Logging-Ausgabe ausgegeben werden (und scheint somit in den Tomcat Logs in catalina.out auf).
- Für den Fall, dass das Highscore Service nicht erreichbar ist, muss ein entsprechendes Exception Handling vorhanden sein (der User darf zB keinen HTTP 500 Fehler sehen)

3. Publizieren der UUID auf Twitter (5 Punkte)

Die in Aufgabe 2 gewonnene UUID soll in diesem Schritt per Aufruf der Twitter API als Tweet zu Twitter geschickt werden. Dabei sind folgende zwei Java Klassen verpflichtend zu verwenden:

- TwitterStatusMessage.java
- ITwitterClient.java

Der Text, welcher als Tweet auf Twitter gepostet werden soll, wird durch Aufruf der Methode

```
public String getTwitterPublicationString()
```



in der Klasse `TwitterStatusMessage` retourniert. Ihr `TwitterClient` muss das Interface `ITwitterClient` implementieren.

Twitter URL: <https://twitter.com/BIGEWA2013>

Consumer Key:
GZ6tiy1XyB9W0P4xEJudQ

Consumer Secret:
gaJDIW0vf7en46JwHAOkZsTHvtAiZ3QUd2mD1x26J9w

Access Token:
1366513208-MutXEbBMAVOwrbFmZtj1r4Ih2vcoHGHE2207002

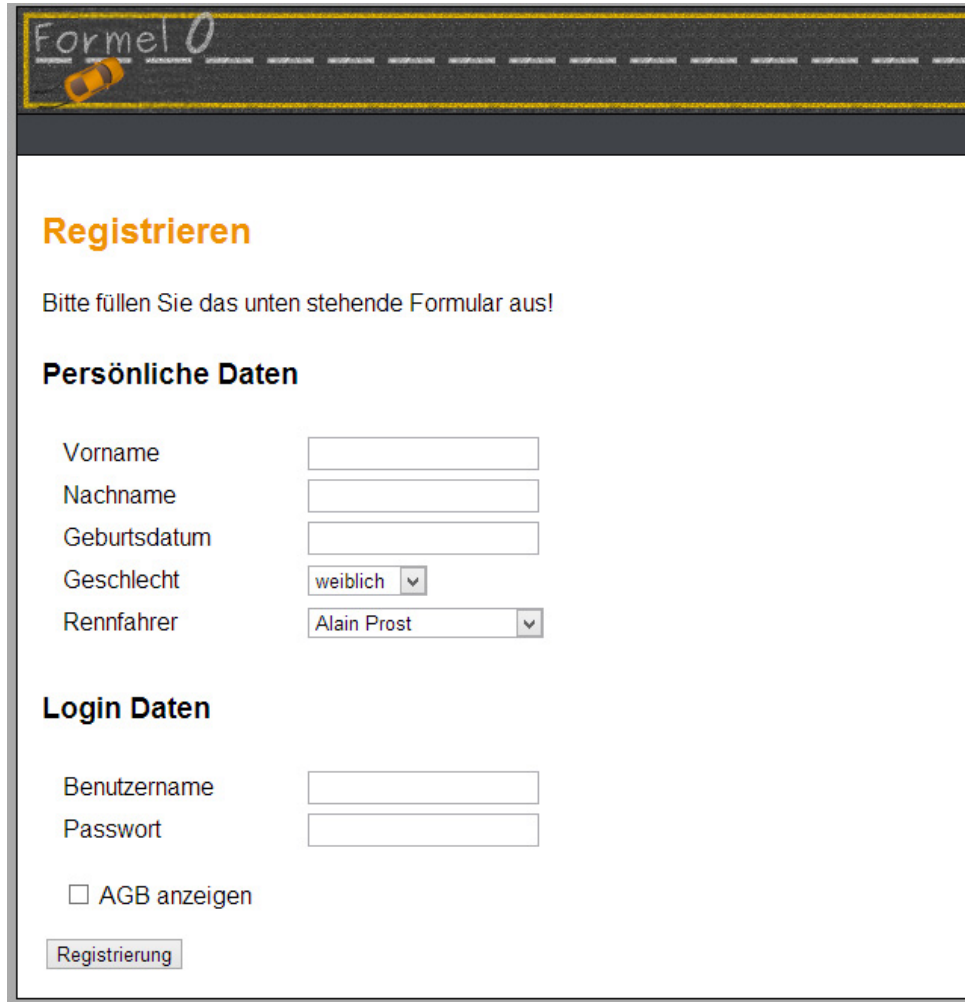
Access Token Secret:
RMPWOePlus3xtURWRVnv1TgrjTyK7Zk33evp4KKyA

Zusammenfassung der geforderten Funktionalität:

- Die UUID, welche durch den Aufruf des Highscore Service gewonnen wurde, soll als Tweet auf Twitter veröffentlicht werden
- Nach erfolgreichem Aufruf des Highscore Services und dem erfolgreichem Posten der UUID auf Twitter, soll dem Benutzer eine Statusmeldung „UUID xyz wurde auf Twitter veröffentlicht“ angezeigt werden.
- Für den Fall, dass das Publizieren auf Twitter fehlgeschlagen ist, muss ein entsprechendes Exception Handling vorhanden sein (der User darf zB keinen HTTP 500 Fehler sehen)

4. Auswahl eines Rennfahrers über Picasa (5 Punkte)

Bei der Neuregistrierung eines Benutzers fügen Sie ein neues Auswahlfeld mit der Id „avatar“ hinzu, über welches der Benutzer einen Rennfahrer wählen kann.



Formel 0

Registrieren

Bitte füllen Sie das unten stehende Formular aus!

Persönliche Daten

Vorname

Nachname

Geburtsdatum

Geschlecht

Rennfahrer

Login Daten

Benutzername

Passwort

☐ AGB anzeigen

Die Menge der verfügbaren Rennfahrer ergibt sich aus den Fotos in folgendem Picasa Album:

<https://picasaweb.google.com/data/feed/api/user/107302466601293793664>

Lesen Sie die Fotos aus dem Album mit Hilfe der Picasa API³ aus. Dabei sind folgende zwei Java Klassen verpflichtend zu verwenden:

- RaceDriver.java
- IRaceDriverService.java

³

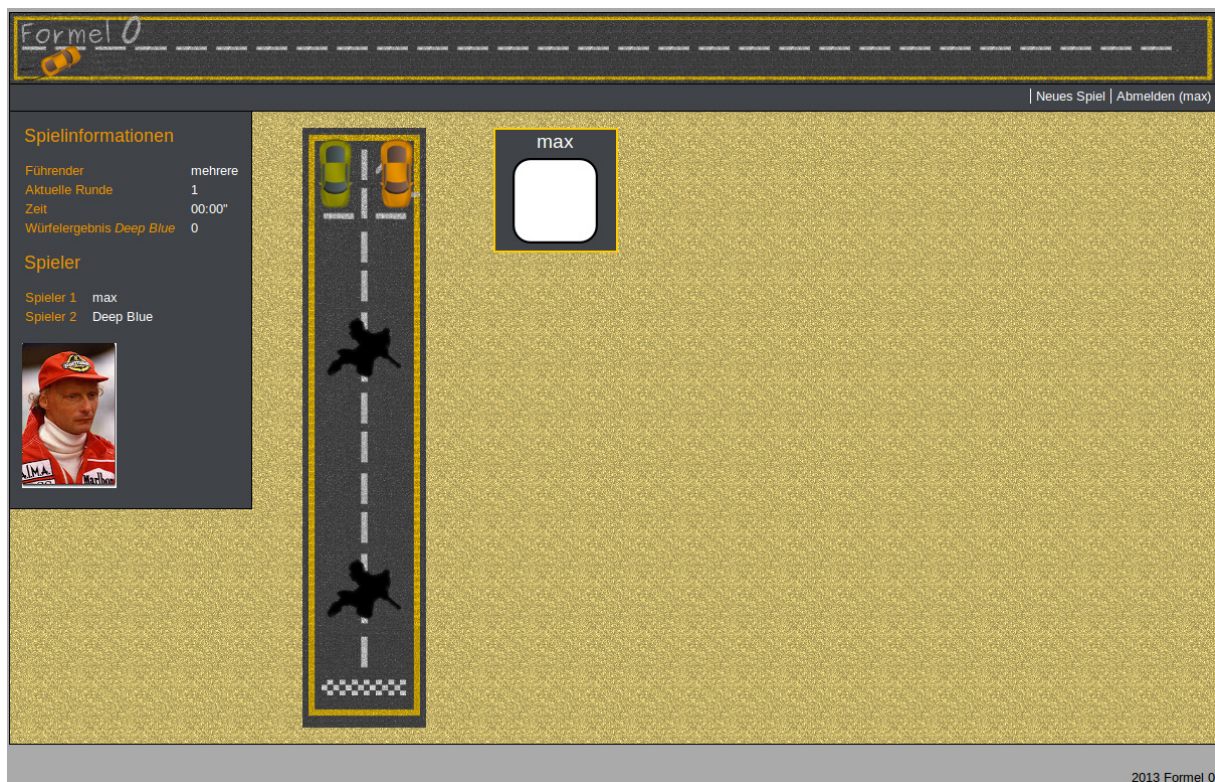
https://developers.google.com/picasa-web/docs/2.0/developers_guide_java

Ein Foto ist dann ein Rennfahrer, wenn es den Tag "Driver" hat. Fotos welche keinen "Driver" Tag haben, dürfen nicht verwendet werden.

Lesen Sie folgende Informationen aus:

- URL des Bildes
- Name des Rennfahrers
- Wikipedia-URL des Rennfahrers. Jeder Rennfahrer hat einen Tag, welcher mit "wiki:" beginnt und die Wikipedia-Adresse beinhaltet.

Stellen Sie den Rennfahrer unter dem Spielinformationfeld dar. Bei einem Klick auf das Bild des Rennfahrers soll sich ein neues Fenster mit dem Wikipedialink des Rennfahrers öffnen.



Zusammenfassung der geforderten Funktionalität:

- Die Liste der Rennfahrer wird mit Hilfe der Picasa API von Picasa bezogen. Holen Sie dabei die Liste der Rennfahrer nur einmal und halten Sie die Liste anschließend persistent in Ihrer Anwendung. Pro gestarteter Anwendung darf nur einmal auf die Picasa API zugegriffen werden!
- Bei der Neuregistrierung muss der Benutzer verpflichtend einen Rennfahrer auswählen
- Der aktuelle Rennfahrer wird unter dem Spielinformationfeld angezeigt
- Bei einem Klick auf das Foto des Rennfahrers öffnet sich der entsprechenden Link auf den Rennfahrereintrag in der Wikipedia



Angabe

Die Angaberessourcen können in TUWEL heruntergeladen werden. Diese beinhalten die folgenden Dateien:

- UE4-Angabe.zip
 - UE4.pdf (Dieses Dokument)
 - src/main/java/formel0api (Ordner)
 - Dice.java
 - Game.java
 - Player.java
 - src/main/java/tuwien/big/formel0/picasa
 - Interfaces und Javaklassen für Picasazugriff
 - src/main/java/tuwien/big/formel0/twitter
 - Interfaces und Javaklassen für Twitterzugriff
 - src/main/webapp
 - META-INF (Ordner)
 - context.xml (Konfiguration der Datenbankansbindung)
 - WEB-INF (Ordner)
 - web.xml
 - register.html
 - table.html
 - resources (Ordner)
 - styles (Ordner)
 - screen.css (CSS-Datei der Musterlösung, aktualisiert)
 - js (Ordner)
 - jquery.js (jQuery JavaScript Library)
 - img (Ordner)
 - ... (Benötigte Bilder)
 - src/resources
 - META-INF (Ordner)
 - persistence.xml
 - pom.xml (Maven Projekt Deskriptor)

Hinweis

Testen Sie ihre Abgabe in den Browsern Firefox und Chrome.

Das JSF2.0 Sample Store Beispiel, welches Sie als ZIP auch über TUWEL runterladen können, zeigt Ihnen die Verwendung von JSF 2.1. Es wird daher empfohlen, sich dieses Beispiel und den Source Code genau anzusehen. Zusätzlich zu den Folien des JSF Gastvortrages empfehlen wir Ihnen, die BIG-internen Folien (vor allem zusätzlich Templates [S 27], Compositions [S 28], Composite Components [S 29] und Internationalisierung I18N [S 46]) und das Beispiel JSF2.0 Sample Store durchzusehen. Wir sind uns im Klaren, dass einige dieser Features in der Vorlesung aus Zeitgründen nur angeschnitten wurden, sie aber dennoch einen wichtigen Teil von JSF 2.1 darstellen und mit den angegebenen



Ressourcen leicht zu verwenden sind. Es wird Ihnen viel zur Lösung der Übung beitragen. Des Weiteren ist <http://jsfatwork.irian.at/> sehr zu empfehlen.

Das Maven Projekt können Sie mithilfe von Netbeans öffnen (File → Open Project) und ausführen. Die .war Datei finden Sie im Ordner „target“ des Projekts.

Sie können Ihre Entwicklungsumgebung (IDE) frei wählen, wir empfehlen jedoch NetBeans 7.3 Java EE (<https://netbeans.org/downloads/>) mit Tomcat 7 und JDK 7 zu verwenden. Bedenken Sie, dass Projekt auf Tomcat 7 laufen muss.

Verwenden Sie zur Validierung Ihrer XHTML-Seiten das vom W3C zur Verfügung gestellte Validation Service unter <http://validator.w3.org/>. Zur Überprüfung der WAI-Tauglichkeit steht Ihnen eine Vielzahl von Services im Internet zur Verfügung. Einige Überprüfungsmethoden und Hilfsmittel wurden in der Vorlesung vorgestellt (siehe Vorlesungsfolien).