UE2 – Servlets, Java Server Pages (25 Punkte)

In der ersten Übung haben Sie statische, valide, und barrierefreie Webseiten, sowie einfache Scripts für client-seitige Funktionalität erstellt. Ziel dieses Übungsbeispiels ist es, eine serverseitige, MVC2-basierten Web-Applikation zu implementieren, die das Quiz-Spiel realisiert. Diese soll auf Basis von Java-Technologien erstellt werden und eine klare Trennung zwischen Model (Java Beans), View (JSP) und Controller (Servlet) aufweisen.

Deadline der Abgabe via TUWEL[1]: **Sonntag, 27. April 2014 23:55 Uhr**

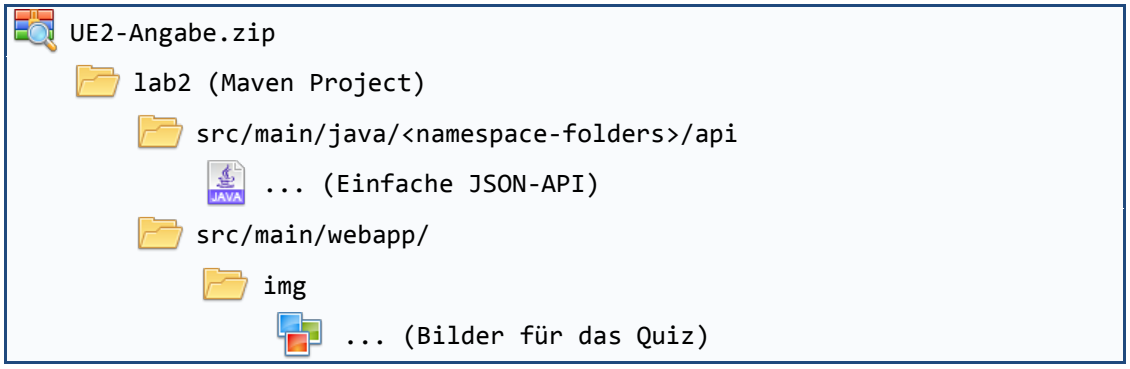
Quiz-Spiel

Beim "Quiz" handelt es sich um ein klassisches Fragespiel mit Mehrfachauswahl ("Multiple Choice"). Registrierte BenutzerInnen müssen während eines Spiels Fragen aus verschiedenen Kategorien beantworten und dabei gegen den Computergegner gewinnen.

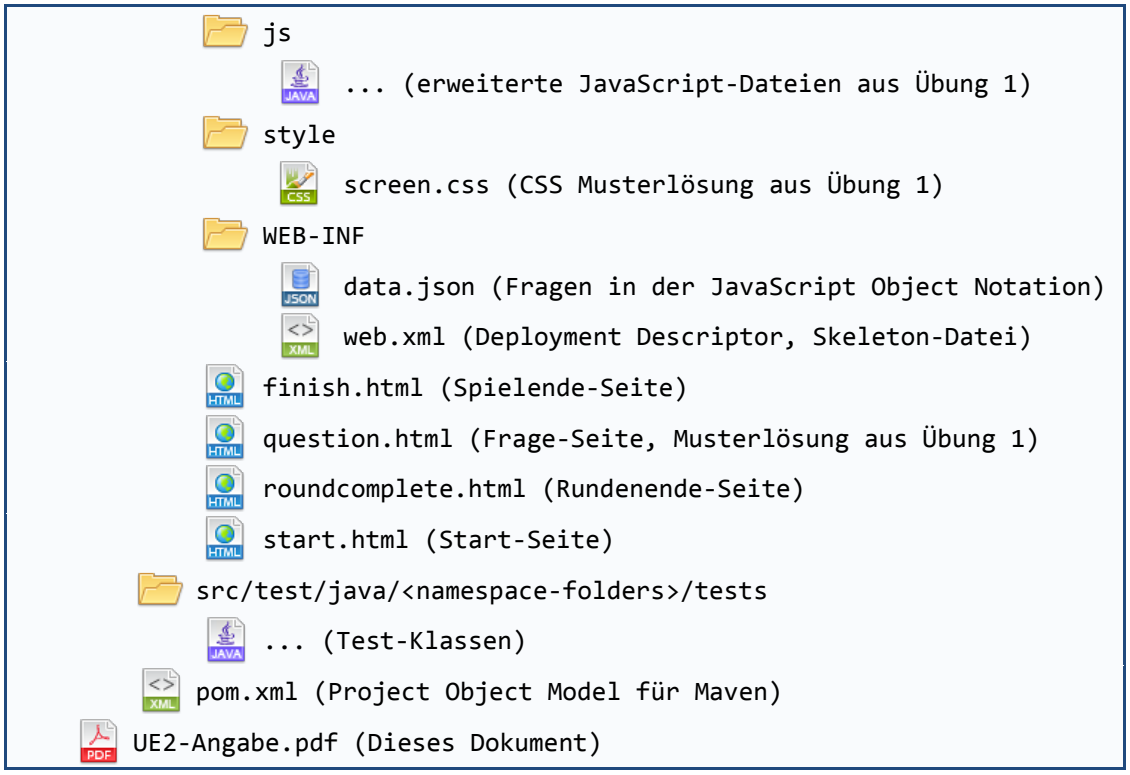
Ein Spiel besteht aus fünf Runden und jede Runde aus drei Fragen, die alle einer bestimmten Kategorie zugewiesen sind. Eine Frage kann beliebig viele, aber mindestens zwei, Antwortmöglichkeiten haben, wobei mindestens eine dieser Antwortmöglichkeiten die korrekte Lösung auf die gestellte Frage ist. Eine 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 aktuelle Auswahl des Spielers als Antwort gewertet und ggf. die nächste Frage gestellt. Am Ende jeder Runde (= nach drei gestellten Fragen aus einer Kategorie), wird der/die RundensiegerIn ermittelt und angezeigt. Der/Die RundensiegerIn ist jene/jener SpielerIn (Mensch oder Computer), welche die meisten Fragen in dieser Runde richtig beantwortet hat. Bei Gleichstand entscheidet die Zeit, die benötigt wurde, um die Fragen zu beantworten (je weniger desto besser). Haben beide SpielerInnen dieselbe Zeit gebraucht um die Fragen zu beantworten, geht die Runde unentschieden aus. Handelt es sich bei der Runde nicht um die letzte Runde, wird eine neue Runde mit einer zufällig gewählten Kategorie, die bisher noch nicht gespielt wurde, gestartet. Ist die letzte Runde abgeschlossen, wird der/die SpielsiegerIn durch die Anzahl der gewonnenen Runden ermittelt und angezeigt. Herrscht Gleichstand geht das gesamte Spiel unentschieden aus, ansonsten gewinnt der/die SpielerIn mit den meisten Rundensiegen. Nach der Siegesanzeige (= Spielende) kann der/die BenutzerIn ein neues Spiel starten.

Angabe

Diese Angabe umfasst folgende Dateien:



[1] https://tuwel.tuwien.ac.at/course/view.php?id=5399



Implementieren Sie eine MVC2-basierte Web-Applikation, welche mit einer eigens zu entwickelnden Quiz-API ein webfähiges Quiz-Spiel realisiert. Berücksichtigen Sie, soweit mit Servlets und Java Server Pages möglich, eine Trennung von Logik (eigens zu entwickelnde API), Benutzeroberfläche (JSPs), Spielflusskontrolle (Servlet) und Daten (Java Beans).

Verwenden Sie als Benutzeroberfläche den XHTML5- und CSS-Code der von uns zur Verfügung gestellten Musterlösung für Übung 1 (siehe Angaberessourcen). Achten Sie darauf, alle IDs der Elemente aus den statischen XHTML5-Seiten in die Elemente der dynamischen JSP-Seiten zu übernehmen, da dies ist für die automatisierte Validierung Ihrer Lösung erforderlich ist. Ansonsten liegt es an Ihnen, die JSP-Seiten entsprechend anzupassen und zu erweitern. Beispielsweise dient der Klick auf einen Link bei statischen Seiten (zB "Quiz starten") nur zur Navigation, während derselbe Klick bei dynamischen Seiten den Zustand am Server verändern kann und deswegen das ursprünglich verwendete Element (`a`) eventuell nicht mehr geeignet ist. Das Aussehen der Seiten soll allerdings so gut es geht erhalten bleiben.

Hauptanforderungen an Ihre Implementierung:

- *Seitenfluss*: Ruft der Benutzer die Seite zum ersten Mal auf, soll er auf der Start-Seite landen. Von dieser aus, kann er dann ein Spiel starten. Ein gestartetes Spiel zeigt die Frage-Seite für jede Frage, die Rundenende-Seite, um den Rundensieger anzuzeigen und die Spielende-Seite um den Spielsieger anzuzeigen. Auf der Spielende-Seite hat man zusätzlich die Möglichkeit ein neues Spiel zu beginnen. In diesem Fall wird man direkt auf die Frage-Seite weitergeleitet und nicht auf die Start-Seite.
- *Spielablauf*: Das Spiel muss den am Anfang dieses Dokuments beschriebenen Regeln folgen (siehe Quiz-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.).

- *Computergegner*: Es muss ein Quiz-Spiel realisiert werden, bei dem ein Spieler gegen einen computergesteuerten Gegner spielt. Das Verhalten des Computergegners können Sie frei gestalten (zB 50% Chance, korrekt zu antworten), es muss jedoch auf jeden Fall möglich sein, dass sowohl der menschliche Spieler als auch der computergesteuerte Gegner gewinnen können.
- *Gleichzeitiges Spielen*: Es muss möglich sein, dass mehrere Spieler je ein eigenes Spiel am selben Server starten können. Diese dürfen sich nicht gegenseitig beeinflussen. Testen Sie dies mit unterschiedlichen Browsern. (Hinweis: Mehrere Tabs innerhalb eines Browsers benutzen dieselbe Session.)
- *Dynamische Inhalte*: Die Spielinformationen (bspw. Kategorie, Fragen, Antwortmöglichkeiten Spielernamen, Spielstand, Rundensieger und Spielsieger) müssen dynamisch ausgegeben werden.
- *Local Storage*: Nutzen Sie den Local Storage von HTML5 um nach einem Spiel das aktuelle Datum und die aktuelle Uhrzeit zu speichern. Diese Information soll auf der Frageseite entsprechend angezeigt werden ("Letztes Spiel"). Wurde noch kein Spiel gespielt, ist "Nie" anzuzeigen. Das JavaScript-Framework (`framework.js`) wurde um die Funktion `supportsLocalStorage` erweitert, welche Sie verwenden können, um zu prüfen, ob der Browser Local Storage unterstützt. Ist dies nicht der Fall, muss die Information auch nicht aktualisiert werden.
- *Standards und Barrierefreiheit*: Das User Interface muss den Anforderungen von XHTML5 sowie WCAG-AA gerecht werden.
- Registrierung und Login müssen in dieser Übung noch nicht implementiert werden!

Zum Ausprobieren Ihrer Implementierung können Sie die von uns zur Verfügung gestellten Fragen in der JavaScript Object Notation (`data.json`) verwenden. Im Package `at.ac.tuwien.big.we14.lab2.api` befinden sich alle Klassen, die Sie zum Einlesen dieser Daten in Java-Objekte benötigen können. Eine Verwendung ist wie folgt:

```
// ServletContext coming from javax.servlet.GenericServlet or subclass
ServletContext servletContext = getServletContext();
QuizFactory factory = ServletQuizFactory.init(servletContext);
QuestionDataProvider provider = factory.createQuestionDataProvider();
List<Category> categories = provider.loadCategoryData();
// category has name and holds questions
// questions have attributes and choices
```

Dabei ist es wichtig, dass Sie den Speicherort der Datei (`WEB-INF`) nicht verändern.

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.

Entwicklungsumgebung

Es ist Ihnen freigestellt, welche Entwicklungsumgebung Sie für diese Übung verwenden. Wir empfehlen den Einsatz der `Eclipse IDE for Java EE Developers`[2], da diese bereits Maven beinhaltet und auch Unterstützung für das Herunterladen eines Webservers anbietet. Achten Sie auf jeden Fall darauf, dass Ihr abgegebenes Projekt mit dieser Eclipse-Version geöffnet werden kann, da diese auch bei den Abgabegesprächen zum Einsatz kommt.

Maven Projekt

In den Angaberessourcen befindet sich innerhalb des Projekts das so genannte Project Object Model (`pom.xml`), welches alle notwendigen Informationen über das Maven Projekt beinhaltet. Unter anderem betrifft das auch die Abhängigen zu bspw. externen Bibliotheken, die automatisch nachgeladen werden können. Sollten Sie nicht die oben empfohlene Entwicklungsumgebung verwenden, benötigen Sie vermutlich ein Maven-Plugin wie bspw. `m2e`[3] für Nicht-J2EE-Eclipse-Versionen. Ansonsten können Sie das Projekt einfach als Maven Projekt importieren.

Web Server

Die Informationen zum Deployen der Anwendung innerhalb eines Servlet-Containers befinden sich im Deployment Descriptor[4] (`web.xml`). Diese Informationen beinhalten unter anderem die Informationen zum Servlet selbst (`servlet`), zum Mapping von URL-Pattern auf Servlets (`servlet-mapping`) und zu Default-Dateien, die gesucht werden, falls kein Dateiname in der URL angegeben ist (`welcome-file-list`).

Das Deployen der Anwendung selbst hängt von der gewählten Entwicklungsumgebung ab. Dazu ist es notwendig ein Web Application Archive (`war`-Datei) zu erzeugen, das unter anderem die erstellten JSP-Dateien sowie das Servlet und die Konfigurationsdatei beinhaltet. In Eclipse können Sie ein entsprechendes Projekt als WAR-Archiv exportieren. Dieses Archiv muss dann auf einem entsprechenden Webserver deployt werden. Verwenden Sie hier einen Apache Tomcat v7.0.

Verwenden Sie die oben empfohlene Entwicklungsumgebung, so können Sie Ihre Anwendung auch direkt auf einem Server deployen. Klicken Sie dazu einfach mit rechter Maustaste auf das Projekt und wählen Sie `Run As... > Run on Server` aus. Im darauf folgenden Dialog können Sie unter Apache den `Tomcat v7.0 Server` auswählen. Klicken Sie auf Next und wählen das entsprechende Installationsverzeichnis aus oder laden Sie den Server direkt in das angegebene Verzeichnis. Wählen Sie `Next` und klicken Sie auf `Finish`. Die Applikation sollte nun auf den Webserver deployt werden und kann üblicherweise unter http://localhost:8080/<project-name>/ abgerufen werden. Achten Sie darauf, dass dieser Aufruf nicht ins Leere geht!

Testfall

Um Sie bei der Entwicklung zu unterstützen, ist in den Angaberessourcen ein JUnit-Testfall (`SeleniumTest.java`) enthalten, mit dem Sie Ihre Webseite auf grundlegende Probleme überprüfen können. Dieser Testfall deckt jedoch nicht die gesamte Funktionalität ab, die in dieser Übung umzusetzen ist bzw. behandelt auch nicht alle qualitativen Aspekte. Ein erfolgreicher Testdurchlauf ist daher als guter Start zu sehen, garantiert jedoch nicht die volle Punkteanzahl!

---

[2] http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/keplersr2
[3] http://www.eclipse.org/m2e/
[4] http://wiki.metawerx.net/wiki/Web.xml, https://developers.google.com/appengine/docs/java/config/webxml

Zum Ausführen klicken Sie einfach mit der rechten Maustaste auf die entsprechende Datei und wählen `Run As > JUnit Test`. Sollte der Testfall nicht ordnungsgemäß laufen, so prüfen Sie zuerst, ob Sie die IDs aus den angegebenen HTML-Dateien auch wirklich in die JSP-Dateien übernommen haben und stellen Sie sicher, dass der WebServer läuft. Außerdem sollten die ursprünglichen Fragen aus der `data.json`-Datei nicht verändert werden.
Beachten Sie, dass der Testfall derzeit nur für die angegebene Entwicklungsumgebung konfiguriert ist. Nähere Informationen zum Testfall finden Sie in den JavaDoc-Kommentaren, entweder direkt im Code oder in Eclipse auch in einer eigenen View unter `Window > Show View > Javadoc`.

Abgabemodalität

Beachten Sie die allgemeinen Abgabemodalitäten des TUWEL-Kurses[5]. Zippen Sie Ihre Abgabe, sodass sie die folgende Struktur aufweist:

```
UE2-AG_<GruppenNr>_gruppe.zip
    lab2 (Maven Project)
        src/main/java/<namespace-folders>/api
            ... (Einfache JSON-API, erweitert um Modellklassen, Spiellogik, ...)
        src/main/java/<namespace-folders>/servlet
            BigQuizServlet.java (Servlet, selbst implementiert)
        src/main/webapp/
            img
                ... (Bilder für das Quiz, unverändert)
            js
                ... (erweiterte JavaScript-Dateien aus Übung 1, unverändert)
            style
                screen.css (CSS Musterlösung aus Übung 1, unverändert)
            WEB-INF
                data.json (Fragen in der JavaScript Object Notation, unverändert)
                web.xml (Deployment Descriptor, vervollständigt)
            finish.jsp (Spielende-Seite, als JSP realisiert)
            question.jsp (Frage-Seite, als JSP realisiert)
            roundcomplete.jsp (Rundenende-Seite, als JSP realisiert)
            start.jsp (Start-Seite, als JSP realisiert)
        src/test/java/<namespace-folders>/tests
            ... (Test-Klassen, unverändert)
        pom.xml (Project Object Model für Maven, unverändert)
```

Alle Dateien müssen UTF-8 codiert sein!
ACHTUNG: Wird das Abgabeschema nicht eingehalten, so kann es zu Punkteabzügen kommen!

[5] https://tuwel.tuwien.ac.at/course/view.php?id=5399