UE2 – Servlets, WebSockets, Java Server Pages (30 Punkte)

In der ersten Übung haben Sie statische, valide und barrierefreie Webseiten, sowie einfache Scripts für clientseitige Funktionalität erstellt. Ziel dieses Übungsbeispiels ist es, eine serverseitige, MVC Model 2 basierte Web-Applikation zu implementieren, die BIG Bid 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, 24. April 2016 23:55 Uhr**

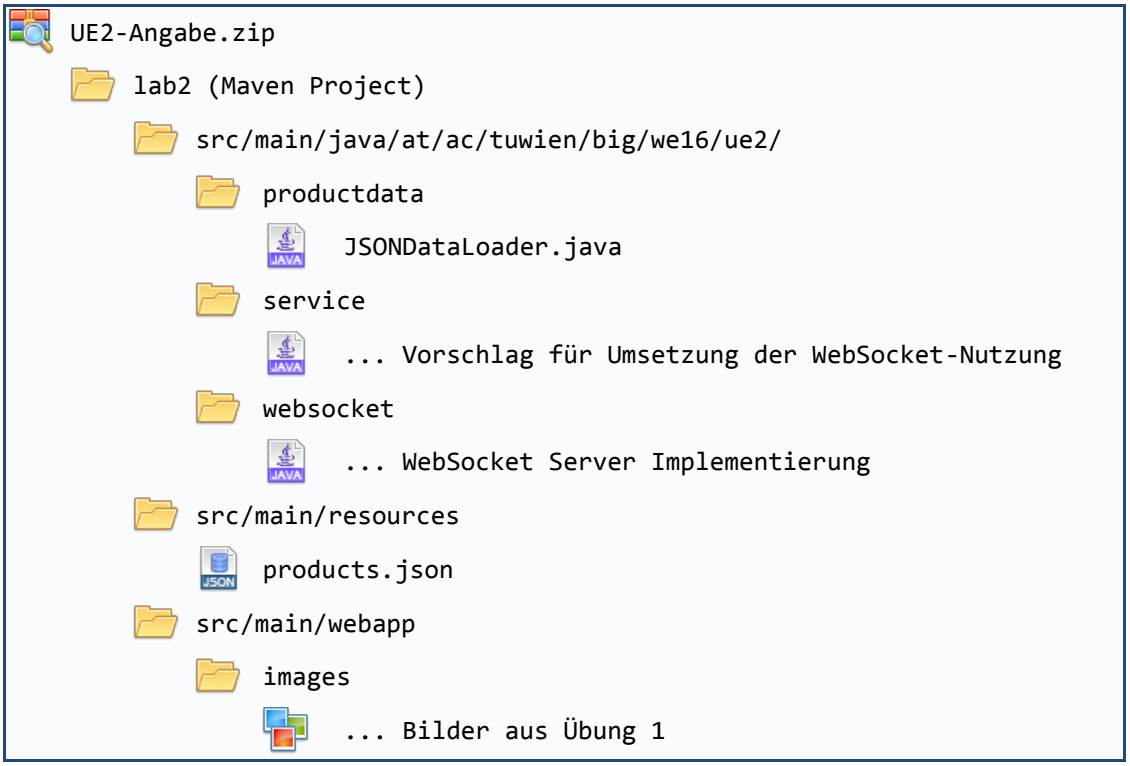
Nur ein Gruppenmitglied muss die Lösung auf TUWEL abgeben.

BIG Bid

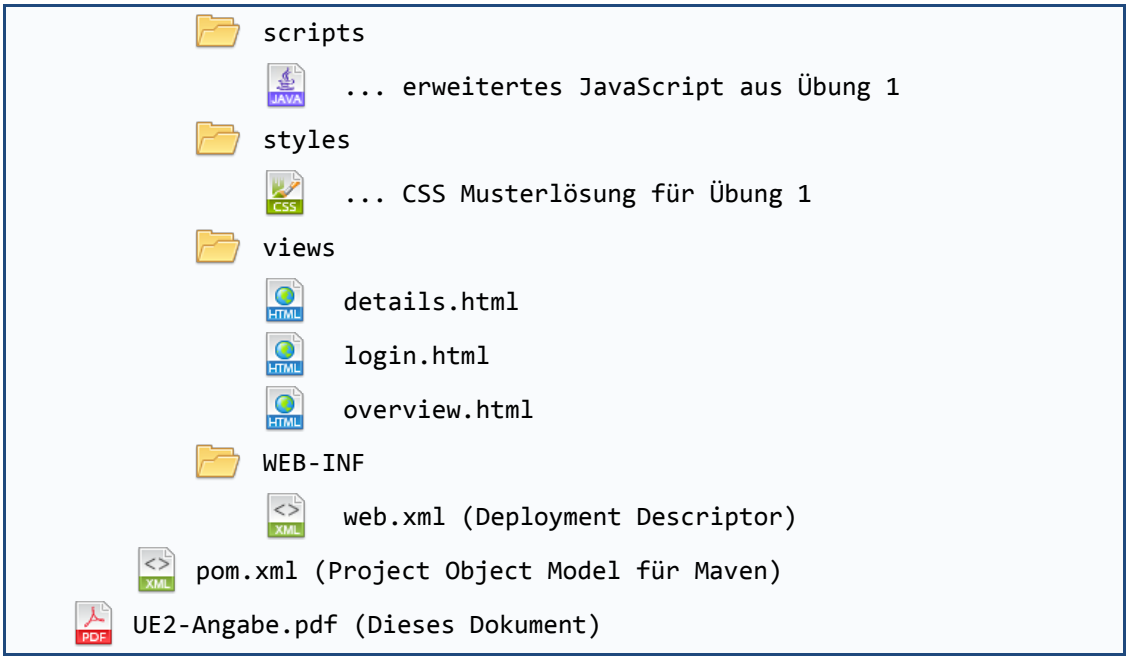
BIG Bid ist eine Online-Plattform, auf der registrierte Benutzerinnen und Benutzer Bücher, CDs und DVDs ersteigern können. Um sicherzustellen, dass die Benutzerinnen und Benutzer nur Gebote abgeben, die sie sich auch tatsächlich leisten können, müssen sie vor der Teilnahme an Auktionen Guthaben bei BIG Bid kaufen. Neue Gebote werden dann sofort von diesem Kontostand abgezogen. Wenn eine Benutzerin oder ein Benutzer überboten wird, dann wird ihr oder ihm der Betrag wieder gutgeschrieben. Nur volljährige Personen können sich bei BIG Bid registrieren.

Angabe

Diese Angabe umfasst folgende Dateien:



[1] https://tuwel.tuwien.ac.at/course/view.php?id=7423



Es steht Ihnen frei, alle von uns zur Verfügung gestellten Klassen zu verändern und zu erweitern.

Implementieren Sie eine auf MVC Model 2 basierende Web-Applikation, welche die webfähige BIG Bid Plattform realisiert. Berücksichtigen Sie, soweit mit Servlets und Java Server Pages möglich, eine Trennung von Logik (eigens zu entwickelnde API), Benutzeroberfläche (JSP), Spielflusskontrolle (Servlet) und Daten (Java Beans).

Verwenden Sie als Benutzeroberfläche den von uns zur Verfügung gestellten HTML5- und CSS-Code aus den Angaberessourcen. Es liegt an Ihnen, die daraus erstellten JSP-Seiten entsprechend anzupassen und zu erweitern. Das Aussehen der Seiten soll allerdings so gut es geht erhalten bleiben. Um möglichst wenig Code zu duplizieren, können Sie Teile des HTML-Codes in neue JSP-Dateien auslagern.

Hauptanforderungen an Ihre Implementierung

- *Seitenfluss*: Ruft der Benutzer oder die Benutzerin die Seite zum ersten Mal auf, soll das Anmelde-Formular angezeigt werden. Eine Überprüfung der Benutzerdaten erfolgt in dieser Übung noch nicht. Nach dem Anmelden wird eine Übersicht aller Produkte angezeigt. Durch Klicken auf ein Produkt gelangt man auf die Detailseite der Auktion. Dort kann ein Gebot abgegeben werden, falls die Auktion noch nicht abgelaufen ist. In der Übersicht sollen Produkte, auf welche man der oder die Höchstbietende ist, mittels der CSS-Klasse `.highlight` mit einem orangefarbenen Rahmen gekennzeichnet werden. Auf jedes Produkt kann nur eine bestimmte Zeitspanne geboten werden. Ist die Zeit eines Produkts abgelaufen, hat die oder der aktuell Höchstbietende diese Auktion gewonnen. Abgelaufene Auktionen sollen in der Übersicht die Klasse `.expired` erhalten, damit diese ausgegraut dargestellt werden; außerdem soll in der Detailansicht anstelle des Bieten-Formulars ein Text ausgegeben werden, wer das Produkt zu welchem Preis ersteigert hat.
- *Kaufen von Guthaben*: Das Kaufen von Guthaben ist nicht Teil der Übung. Stattdessen erhält jede Benutzerin und jeder Benutzer beim Start der Applikation ein Startguthaben von 1.500 €.

- *Bieten*: Gebote können nur über die Bieten-Funktion auf der Detailseite abgegeben werden. Dieses Formular soll über Ajax[2] abgeschickt werden. Wird ein Gebot abgegeben soll am Server validiert werden, ob es höher als das aktuelle ist. War die Prüfung negativ soll eine Fehlermeldung auf der Detailseite ausgegeben werden. Bei einer erfolgreichen Überprüfung werden der Kontostand und der Zähler der laufenden Auktionen aktualisiert und inklusive dem neuen Gebot am Server gespeichert. Der Zähler der laufenden Auktionen soll nur bei der ersten Teilnahme an einer Auktion erhöht werden und wird erst nach dem Ablauf einer Auktion wieder verringert. Danach wird entweder der Zähler der gewonnen oder der verlorenen Auktionen erhöht. Die Daten (Anzahl laufender Auktionen, Kontostand) sollen als Antwort auf den Ajax-Request an den Client zurückgeschickt und auf dessen Detailseite aktualisiert werden. Zusätzlich sollen der Name des oder der Höchstbietenden und sein oder ihr Gebot über den WebSocket an alle Clients gesendet werden, um die angezeigten Daten live zu aktualisieren. Hat es zuvor ein gültiges Gebot eines anderen Bieters oder einer anderen Bieterin gegeben, muss demjenigen oder derjenigen der Betrag wieder gutgeschrieben und die Gutschrift am Server gespeichert werden. Auch diese Änderung soll sofort per WebSocket an den relevanten Client übermittelt und in dessen Sidebar aktualisiert werden. Dies kann beispielsweise Auftreten, wenn das eigene Gebot vom Computerbenutzer überboten wird.
- *Computerbenutzer*: Zur Simulation der realen Umgebung der Applikation mit mehreren konkurrierenden Benutzerinnen und Benutzern soll zumindest ein Computerbenutzer implementiert werden, der im Abstand von 10 Sekunden auf jedes Produkt mit einer Wahrscheinlichkeit von 30 % das aktuell höchste Gebot überbietet[3].
- *WebSocket*: Um die im Browser angezeigten Daten aktuell zu halten, soll der Server Informationen über neue Gebote und abgelaufene Auktionen mittels WebSocket an verbundene Clients senden. Die zu übertragenden Daten sind:
  - Wenn eine Auktion abgelaufen ist, dann wird allen angemeldeten Benutzerinnen und Benutzern mitgeteilt, welche Auktion abgelaufen ist, was der aktuelle Kontostand ist und wie viele Auktionen derzeit laufen bzw. verloren oder gewonnen wurden. Mit Hilfe dieser Daten wird die abgelaufene Auktion auf der Übersichtsseite als solche markiert (mit der CSS-Klasse `.expired`), das Bieten-Formular der Detailseite durch einen Informationstext ersetzt und die Sidebar aktualisiert.
  - Wenn ein neues Gebot abgegeben wurde, dann wird an alle angemeldeten Benutzerinnen und Benutzer eine Nachricht gesendet, welche die Produkt-ID, den Namen des neuen Höchstbietenden, und den Betrag des neuen Gebots enthält. Mit Hilfe dieser Daten werden die angezeigten Informationen in der Übersicht und auf der Detailseite aktualisiert.
  - Wenn ein Benutzer oder eine Benutzerin überboten wurde, dann wir ihm oder ihr via WebSocket mitgeteilt, wie hoch der Kontostand nach der erfolgten Gutschrift ist, damit der alte Kontostand in der Sidebar ersetzt werden kann.

  Beachten Sie, dass der Inhalt der ersten und dritten Nachricht vom Empfänger der Daten abhängt, während der Inhalt der zweiten Nachricht für alle verbundenen Clients gleich ist.

[2] Tipp: Mit jQuery lässt sich das Abschicken eines Formulars mittels Ajax deutlich einfacher implementieren als mit den nativen Möglichkeiten von JavaScript.

[3] Den Computerbenutzer implementieren Sie am besten mit einem `ScheduledExecutorService`, wie es auch im `NotifierService` der Angabe benutzt wird.

- *Gleichzeitige Benutzung der Plattform*: Es muss möglich sein, dass am selben Server mehrere Benutzerinnen und Benutzer gleichzeitig angemeldet sind und mehrere Auktionen gleichzeitig laufen. Testen Sie dies mit unterschiedlichen Browsern oder mit Hilfe des Incognito-Modus Ihres Browsers. (Hinweis: Mehrere Tabs innerhalb eines Browsers benutzen dieselbe Session.)
- *Dynamische Inhalte*: Die Produktdaten und Informationen über die Benutzerin oder den Benutzer (beispielsweise Produktname, aktueller Preis, Höchstbietende oder Höchstbietender, Kontostand, Anzahl der laufenden/gewonnenen/verlorenen Auktionen) müssen dynamisch ausgegeben werden.
- *Local Storage*: Nutzen Sie den Local Storage von HTML5 um jene Produkte zu speichern, die sich eine Benutzerin oder ein Benutzer zuletzt angesehen hat. Zeigen Sie die Links zu diesen Produkten in der rechten Spalte an. Wenn noch keine Produkt-Detailseiten aufgerufen wurden oder der Browser kein Local Storage unterstützt, dann sollen die Überschrift “Zuletzt angesehen” und die Liste ausgeblendet werden. Um zu überprüfen, ob Local Storage verfügbar ist, können Sie die Funktion `supportsLocalStorage` aus framework.js benutzen.
- *Standards und Barrierefreiheit*: Das User Interface muss den Anforderungen von HTML5 sowie WCAG-AA gerecht werden.
- *Speichern von Daten*: Da Sie in dieser Übung keine Datenbankanbindung implementieren müssen, speichern Sie die Daten zu Benutzern, Auktionen und Geboten in Java-Variablen. Das heißt, die Daten gehen verloren, sobald die Applikation am Server beendet wird.
- Eine Überprüfung des Passworts beim Login muss in dieser Übung nicht implementiert werden.

Testdaten

Zum Ausprobieren Ihrer Implementierung können Sie die von uns zur Verfügung gestellten Produkte in der JavaScript Object Notation (`products.json`) verwenden.

Im Package `at.ac.tuwien.big.we16.ue2.productdata` befinden sich alle Klassen, die Sie zum Einlesen dieser Daten in Java-Objekte benötigen. Die Verwendung ist wie folgt:

```
// Loads all albums from the products.json
JSONDataLoader.Music[] music = JSONDataLoader.getMusic();

// Loads all movies from the products.json
JSONDataLoader.Movie[] movies = JSONDataLoader.getFilms();

// Loads all books form the products.json
JSONDataLoader.Book[] books = JSONDataLoader.getBooks();
```

Dabei ist es wichtig, dass Sie den Speicherort der JSON-Datei (`src/main/resources`) nicht verändern.

Hinweise

Validierung

Verwenden Sie zur Validierung Ihrer HTML 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 (zum Beispiel http://achecker.ca/). Nähere Infos dazu finden Sie in den Folien beziehungsweise in TUWEL.

Entwicklungsumgebung

Es ist Ihnen freigestellt, welche Entwicklungsumgebung Sie für diese Übung verwenden. Achten Sie auf jeden Fall darauf, dass Ihr abgegebenes Projekt mit *Eclipse IDE for Java EE Developers* 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 externen Bibliotheken, die automatisch nachgeladen werden können. In den meisten Entwicklungsumgebungen können Sie das Projekt einfach als Maven Projekt importieren.

Web Server

Verwenden Sie für diese Übung *Java 8* und *Apache Tomcat 8.0*.

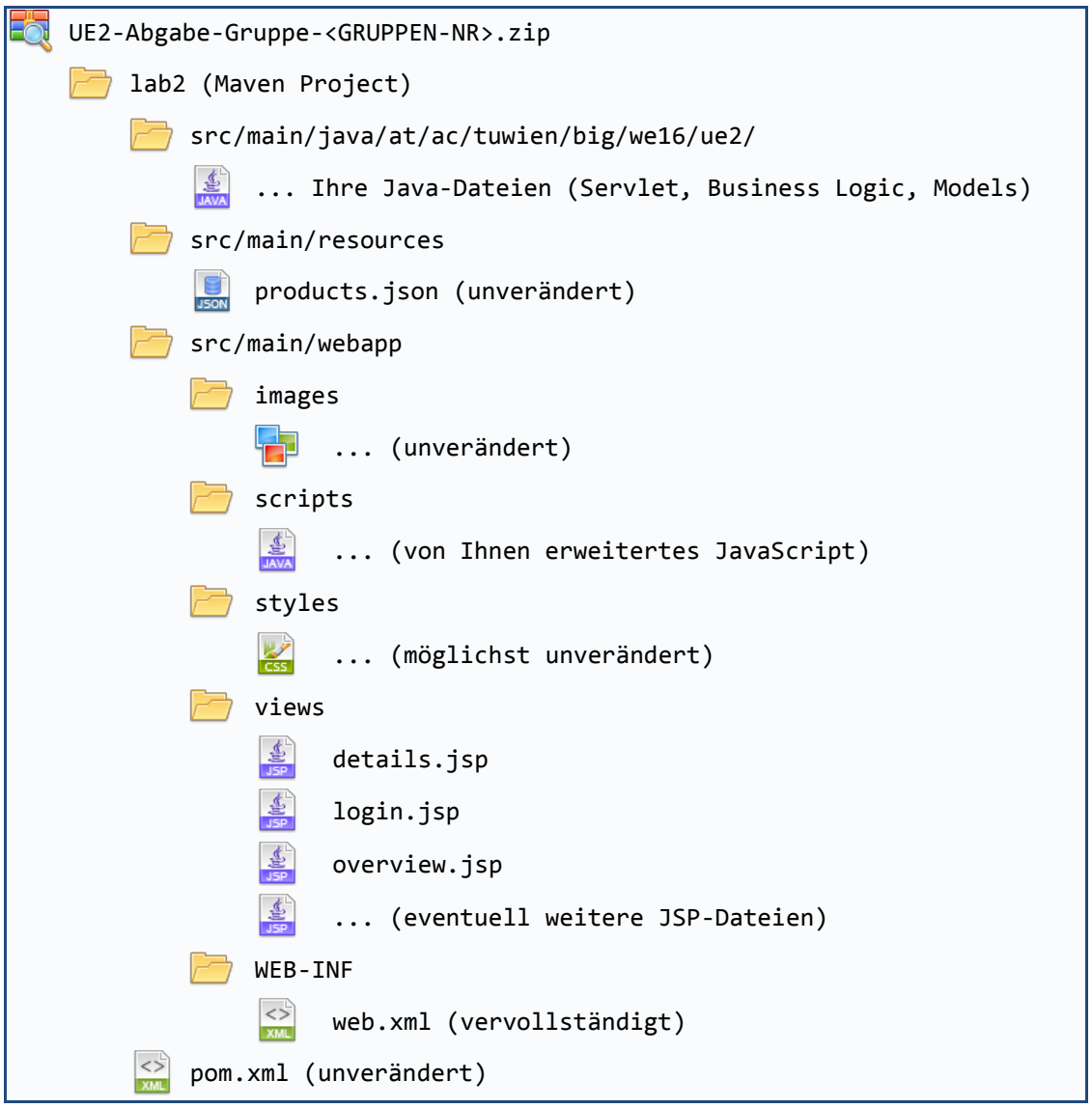
Die Informationen zum Deployen der Anwendung innerhalb eines Servlet-Containers befinden sich im Deployment Descriptor (`web.xml`). Diese Datei enthält unter anderem Informationen zum Servlet selbst (`servlet`), zum Mapping von URL-Pattern auf Servlets (`servlet-mapping`) und zu Default-Dateien, die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

Das Deployment der Anwendung funktioniert mit Hilfe eines Web Application Archive (`war`-Datei). Dieses Archiv enthält alle Dateien, die der Server benötigt, um die Applikation zur Verfügung zu stellen. Das sind unter anderem JSP-Dateien, das Servlet und der Deployment Descriptor. Das Archiv wird in der Regel auf einen Webserver deployt, indem die `war`-Datei in ein spezielles Verzeichnis kopiert wird (siehe Vorlesungsfolien). Achten Sie darauf, dass nicht nur Admin- bzw. root-User Schreibrechte auf diesen Ordner haben. Die Applikation kann dann üblicherweise unter http://localhost:8080/ oder http://localhost:8080/<project-name>/ aufgerufen werden. Stellen Sie sicher, dass einer dieser Aufrufe nicht ins Leere geht!

Wenn Sie eine Entwicklungsumgebung wie *Eclipse IDE for Java EE Developers* oder *IntelliJ IDEA* benutzen, dann können Sie Ihre Anwendung auch direkt aus Ihrer IDE auf den Server deployen.

Abgabemodalität

Beachten Sie die allgemeinen Abgabemodalitäten des TUWEL-Kurses[4]. Zippen Sie Ihre Abgabe, sodass sie die folgende Struktur aufweist:



Alle Dateien müssen UTF-8 codiert sein!

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

[4] https://tuwel.tuwien.ac.at/course/view.php?id=7423