Fernfachhochschule Schweiz

JEE Semesterarbeit

Onlineshop mit AJAX Erweiterung

D. Bittante

Bachelor of Science in Informatik 2013

Dozent
Heinrich Zimmermann

Bern, 8. Dezember 2016

Abstract

Das vorliegende Dokument beschreibt die Semesterarbeit zum Fach Java Enterprise Edition (JEE). Eine Enterprise Applikation mit AJAX zu erweitern, steigert die User-Experience wesentlich indem die Applikation responsive bleibt auch wenn diese zeitintensiven Verarbeitungen durchführt.

Der unübersichtliche Code und das Fehlerhandling sind nur einige der Probleme die mit Java Script auftauchen. jQuery bietet ein ausgezeichnetes API um verschiedene Probleme entgegen zu wirken.

UML-Diagramme wie Klassendiagramme, Sequenzdiagramm und Anwendungsdiagramme wurden genutzt um die zu entwickelnden Objekte zu beschreiben.

Glassfish und Java Script wurden als Problemquellen analysiert. Die Entwicklung ohne Unterstützung einer IDE führt zu schwer lesbarem Code und Schwierigkeiten bei der Fehleranalyse. Architektur und Best-Practice, sind Themen die im Zusammenhang mit Java Script Mühe gemacht haben.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einführung

1.1 Fokus

Das Ziel der Semesterarbeit soll sein, einen vordefinierten Webshop zu erweitern und methodisch korrekt diese Erweiterung zu Dokumentieren und zu Implementieren. Den Fokus dieser Arbeit soll die Implementation von AJAX im Kontext einer JEE Applikation beinhalten. Als Ziel wurde gesetzt die folgenden aufeinander aufbauenden Arbeitspakete zu erledigen, testen, dokumentieren und zu präsentieren. Der Online Shop soll um folgende Funktionalitäten angereichert werden:

- Der User soll die Möglichkeit bekommen die Artikel zu bewerten
- Der User soll die Bewertungen eines Artikels erfahren können
- Der User soll immer benachrichtigt werden, wenn jemand den Artikel bewertet, den er gerade betrachtet.

1.1.1 Artikel bewerten

Der erste Anwendungsfall, der in dieser Arbeit betrachtet wird, ist ein Einstieg in AJAX und dazugehörende Framework.

Was sollte mit diesem Anwendungsfall erreicht werden?
Es soll ein Proof of Concept erstellt werden, mit dem man feststellen kann, wie einfach es ist mit AJAX Daten asynchron zu verarbeiten. Die Implementation wird mit Hilfe eines Frameworks namens jQuery entwickelt.

Worin bestand die Nützlichkeit der implementierten Funktionalität im Webshop?
Kundenbewertung ist wichtiger als ein Testbericht der Stiftung Warentest gibt Siemens und Bosch an. Darum ist diese Funktionalität von Produkten eine sinnvolle Erweiterung für einen Webshop. Unternehmen reagieren nach Angaben von Melissa Bohlsen in ihrem Artikel "Produktbewertungen als harte Währung"[Boh15] immer stärker auf Kundenbewertungen und nehmen Kontakt mit Kunden ausserhalb von herkömmlichen Customer-Service Kanälen auf. Direkte und interaktive Kommunikation mit dem Kunden wirkt sich positiv auf die Kundenbindung und Wertschätzung des Kunden dem Unternehmen entgegen aus.

1.1.2 Artikelbewertung laden

Die Bewertungen sollen hier mittels AJAX geladen werden, dabei wird JSON zum Einsatz kommen. Das wird dem Onlineshop mehr Dynamik verleihen.
Was sollte mit diesem Anwendungsfall erreicht werden? Es soll herausgefunden werden wie das Laden von Daten mittels AJAX funktioniert und welche Probleme dabei auftauchen können.
Worin bestand die Nützlichkeit der implementierten Funktionalität im Webshop?

Der User soll nicht immer erneut nach dem Produkt suchen müssen um die aktuellen Bewertungen zu sehen. Der User möchte, dass sich ihm jederzeit aktuelle Daten präsentieren.

1.1.3 Benachrichtigung bei Bewertungen

Dieser Anwendungsfall behandelt das Problem des Polling in einer Applikation. Eine Webseite kann nicht asynchron aufgerufen werden und muss somit immer wieder beim Backend nachfragen ob sich die Daten auf der Webseite noch up-to-date sind. Wie das funktioniert soll in diesem Anwendungsfall gezeigt werden. **Worin bestand die Nützlichkeit der implementierten Funktionalität im Webshop?**
Wie im Anwendungsfall "Artikelbewertung laden" wird die aktuelle Übersicht hier noch einen Schritt weiterentwickelt und die geladene Übersicht immer auf den aktuellen Stand gehalten.

Kapitel 2

Aufbau des Onlineshop und Infrastruktur

2.1 Onlineshop

Die Entwicklung des Onlineshops wurde nicht von Grund auf neu entwickelt. Aufbauend auf dem Onlineshop von A. Salvanos [Sal14] wurden die Anwendungsfälle weiterentwickelt. Die in diesem Kapitel beschriebene Struktur wurde vorgegeben.

2.1.1 Persistenz-Schicht

Das Datenbankschema das mit dem Buch zusammenpasst, muss mit folgenden SQL-Befehlen erstellt werden und soll hier der Vollständigkeit halber aufgenommen werden.

```
1
2  sqlplus /nolog
3
4  CONNECT / AS SYSDBA
5
6  begin
7  dbms_xdb.sethttpport('5050');
8  end;
9  /
10
11 CREATE TABLESPACE onlineshop_tablespace
12 DATAFILE 'C:\oraclexe\app\oracle\oradata\XE\ONLINESHOP.DBF'
13 SIZE 100M
14 AUTOEXTEND ON
15 MAXSIZE UNLIMITED;
16
17 CREATE USER onlineshop IDENTIFIED BY supergeheim_123
18 DEFAULT TABLESPACE onlineshop_tablespace;
19 CREATE USER onlineshop_user IDENTIFIED BY geheim_123
20 DEFAULT TABLESPACE onlineshop_tablespace;
21
22
23 CREATE ROLE onlineshop_role;
24 GRANT SELECT ON onlineshop.customer TO onlineshop_role;
25 GRANT SELECT ON onlineshop.item TO onlineshop_role;
```

```
26 GRANT onlineshop_role TO onlineshop_user;
27
28 GRANT CREATE SESSION TO onlineshop;
29 GRANT CREATE TABLE TO onlineshop;
30 GRANT UNLIMITED TABLESPACE TO onlineshop;
31 GRANT CONNECT TO onlineshop_user;
32
33
34
35 DROP TABLE item;
36 DROP TABLE customer;
37 CREATE TABLE customer (
38 id NUMBER(19) PRIMARY KEY,
39 email VARCHAR2(40) NOT NULL UNIQUE,
40 password VARCHAR2(10) NOT NULL
41 CHECK(LENGTH(password)>=6)
42 );
43 GRANT SELECT, INSERT, UPDATE, DELETE
44 ON customer TO onlineshop_user;
45 CREATE UNIQUE INDEX customer_index
46 ON customer(
47 email,
48 password
49 );
50 CREATE TABLE item (
51 id NUMBER(19) PRIMARY KEY,
52 title VARCHAR2(40) NOT NULL,
53 description VARCHAR2(1000) NOT NULL,
54 price NUMBER(12,2) NOT NULL,
55 foto BLOB,
56 seller_id NUMBER(19) NOT NULL,
57 buyer_id NUMBER(19),
58 sold TIMESTAMP(3),
59 CONSTRAINT fk_seller
60 FOREIGN KEY (seller_id) REFERENCES customer (id),
61 CONSTRAINT fk_buyer
62 FOREIGN KEY (buyer_id) REFERENCES customer (id)
63 );
64
65 GRANT SELECT, INSERT, UPDATE, DELETE
66 ON item TO onlineshop_user;
67 DROP SEQUENCE seq_customer;
68 CREATE SEQUENCE
69
70 CREATE OR REPLACE TRIGGER tri_customer
71 BEFORE INSERT ON customer
72 FOR EACH ROW
73 BEGIN :NEW.id := seq_customer.NEXTVAL;
74 END;
75 /
76 CREATE OR REPLACE TRIGGER tri_item
77 BEFORE INSERT ON item
78 FOR EACH ROW
```

```
79 BEGIN :NEW.id := seq_item.NEXTVAL;
80 END;
81 /
82 COMMIT;
```

2.1.2 Business-Schicht

Als Übersicht kann das Hypertext Schema dienen, dabei aufgezeichnet werden die einzelnen Bereiche die die Software umschließt. Die Applikation besteht hauptsächlich aus drei Bereichen: dem Produkt- , den Registrierungs- und Login-Bereich. D Im Folgenden wird auf diese weiter eingegangen.

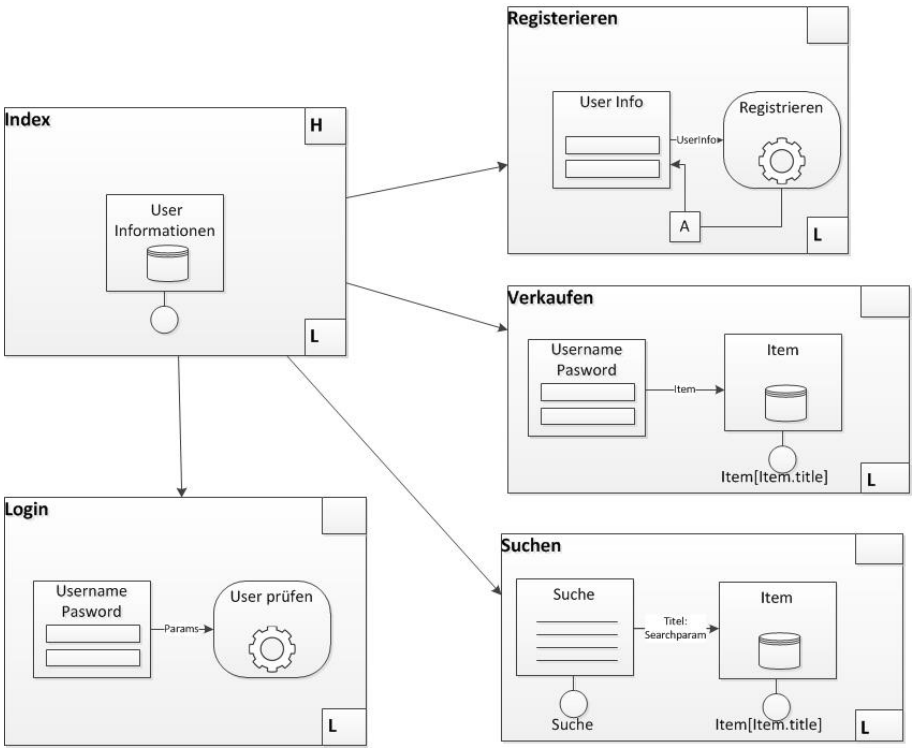

Abbildung 2.1: Hypertext Schema

Im Folgenden werden die Servlets und derer Funktion beschrieben. Der Onlineshop besteht aus sechs Servlets, die Controller, im MVC-Pattern, gleichzusetzen sind.

Registrierung - Servlet

Das Register-Servlet bearbeitet die Anfrage einen neuen User zu persistieren, es wird Username und Password ohne serverseitige Validierung niedergeschrieben.
Das ”Sign In”-Servlet übernimmt den Aufbau einer Session und überprüft die Login-Daten gegen die Registrierungs-Daten

Sell / Buy - Servlet

Das Sell-Servlet speichert die zu verkaufende Daten in die Datenbank, dabei wird das anzuzeigende Foto ebenfalls in der Datenbank als ByteStream persistiert.

Um ein Item zu kaufen, wird das Buy-Servlet eingesetzt, dieses mutiert einen bestehenden Artikel in der Datenbank.

Search / Foto- Servlet

Um die gespeicherten Artikel zu visualisieren braucht es eine Übersicht aller möglichen Artikel, der Standard-Onlineshop löste diese Anforderung mit einer Artikelsuche. Einen Suchbegriff sucht den Titel alle Artikel ab und stellt dies als eine Liste samt Details an.
Jeder Artikel besitzt nicht nur einen Titel und Beschreibung, sondern ebenfalls eine visuelle Repräsentation. Dieses Bild wird aus der Datenbank geladen mit Hilfe des Foto-Servelts. Die Search-Page greift dabei direkt auf das Bild zu als wäre diese im Foto-Verzeichnis, dies löst einen Servlet-Aufruf aus und das Foto wird als ByteStream verarbeitet und direkt in das Response-Objekt zurückgegeben.

2.1.3 Präsentation-Schicht

Die Präsentation-Schicht wurde mit der Web-Programmiersprache JSP umgesetzt. Diese erlaubt für diese Semesterarbeit nötige Java-Script einzubinden, und auszuführen.

Der Aufbau einer Seite ist einfach gehalten. Eine Seite besteht aus einem Header, Footer und einem Content. Definiert wurde eine Seite als JSP und der Header sowie der Footer als JSPF eine Java Server Page Fragment. Hier am Beispiel des Registrieren-Page illustriert

```
<%@ include file="head.jspf" %>
   <form action="register" method="post">
   <fieldset>
   <legend>Registrieren</legend>
   <table>
      <tbody>
      <tr>

       <!-- Web - Elemente-->
       ...
      </tr>
      <tr>
         <td/><td>
            <input type="submit">
            <input type="reset">
         </td>
      </tr>
   </tbody>
   </table>
   </fieldset>
   </form>
<%@ include file="footer.jspf" %>
```

2.2 Technische Infrastruktur

Für die Semesterarbeit wurden folgende Komponenten genutzt:

- Glassfish Open Source Edition v.4.1.1 (Web App Container)
- Oracle Express Edition 11g (DBMS)
- Java Enterprise Edition 1.8 (JDK)

Kapitel 3

Ajax - Theoretische Behandlung des Fokus

3.1 Was ist AJAX?

Das Acronym AJAX , steht für Asynchronous JavaScript and XML, beschreibt dabei klar was dieses Technologie-Erweiterung von Java-Script erreichen soll. Die Java-Script-Library wurde nicht erweitert und AJAX ist auf gar keinen Fall eine eigene Programmiersprache. Es soll hingehend der bestehenden Technologien auf die vom Browser bereits vorhandenen XMLHttpRequest Objekte benutzen und die Stärken von Java-Script, das DOM-Handling, ausgenutzt werden, um dynamische Webseiten zu gestalten, die nicht jedes mal von einem Backend vollständig geliefert werden sollen.

3.1.1 Asynchronität

Da das Laden einer Seite somit asynchron erfolgen kann, blockiert dies die Applikation nicht, was dem User erlaubt eine Interaktion mit dem Backend zu haben mit aktuellen Feedbacks, z.B. eines Fortschrittsbalken oder Statusanzeige, auch wenn die Verarbeitung länger dauern soll. Das Ganze wird mit Callbacks gesteuert und diese sind in den meisten Fällen Subfunktionen eines Aufrufs. Was dabei beeindruckend ist wie schnell sich diese anhäufen und schnell Spaghetti-Code entsteht. Eine weitere Schwierigkeit ist das Fehlerhandling das in diesem Fall sehr schnell vergessen gegangen geht. Erschwerend kommt hinzu, man kann bei Callbacks kein value mittels return zurückgeben, wohin den auch, und das mächtige Keyword ”throw” ist ebenfalls nicht einsetzbar. Wie Fehlerhandling dennoch bei asynchronen Aufrufen behandelt werden kann findet man in den nächsten Kapiteln.

3.2 jQuery

Nachfolgend soll aufgezeigt werden, dass die Benutzung eines Frameworks auch und gerade in Zusammenhang mit Java Script eine deutliche Vereinfachung darstellen kann und den Overhead von AJAX und deren Handhabung vereinfacht. Als Beispiel wurde hier eine ladende Funktion dargestellt die Asynchron einen Inhalt von einem Server bezieht nachdem ein Event die Funktion loadDoc aufruft. Die erste Aufstellung wurde mit der purem JavaScript implementiert.

```
1  <button type="button"
2  onclick="loadDoc('ajax_info.php', myFunction)">Change Content</button>
3  </div>
4
5  <script>
6  function loadDoc(url, cFunction) {
7    var xhttp;
```

```
  xhttp=new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      cFunction(this);
    }
  };
  xhttp.open("GET", url, true);
  xhttp.send();
}
function myFunction(xhttp) {
  document.getElementById("demo").innerHTML =
  xhttp.responseText;
}
</script>
```

Dieses Native-Ajax Beispiel ruft nach erfolgreicher Datenabfrage am Server eine weitere Funktion namens myFunction auf. Dieses Callback-Beispiel, zeigt wie einfach die Handhabung eines solchen Anwendungsfall mit jQuery ist, denn sogar für GET und POST Methoden hat jQuery eigene Funktionen implementiert. Beispielsweise werden GET mit jQuery.get() aufgerufen, das nachfolgende Beispiel zeigt einen Anwendungsfall.
Das oben genannte Skript kann wie folgt umgeschrieben werden:

```
<button type="button" onclick="loadDoc()">Change Content</button>
</div>

<script>
$.get( "ajax_info.php", function( data ) {
  $('demo').html(data);
});
</script>
```

3.2.1 Fehlerhandling

JQuery bietet gute Schnittstellen an, um die Fehler die beim Laden von Daten auftauchen zu behandeln. Wie zum Beispiel bietet die load()-Funktion an den Fehler über den Parameter ”status” abzufragen, siehe dazu die folgende Auflistung.

```
<script>
$( "#success" ).load( "/not-here.php", function( response, status, xhr ) {
  if ( status == "error" ) {
    var msg = "Entschuldigt, da gabs wohl einen Fehler: ";
    $( "#error" ).html( msg + xhr.status + " " + xhr.statusText );
  }
});
</script>
```

3.2.2 Laden von Daten

Die Interpretation des Acronym ”AJAX”, dass AJAX nur mit XML umgehen kann ist eine falsche Annahme, die AJAX benutzt das XMLHttpRequest Objekt aber die Request und Responses können mit verschiedensten Datentypen verarbeiten und aufbereiten werden. Da JSON eine einfache strukturierte Datenstruktur ist um Daten zu verarbeiten, wollen wir im nächsten Unterkapitel kurz

eingehen.
http://stackoverflow.com/questions/2076642/difference-between-id-load-and-ajax

JSON

Auch in dieser Arbeit wurde die Übermittlung der Daten mittels, eines vom Autor definiertem, JSON-Model umgesetzt. Die Struktur besteht aus den Felder aus denen sich eine Bewertung zusammensetzt. Wenn die jQuery.getJSON() Funktion genutzt wird können die Elemente gleich verarbeitet werden. Ein weiteres Beispiel soll aufzeigen wie.

```
1
2  var commentServlet =
       "http://localhost:8081/onlineshop-war/comments?productId="+product_id;
3  $.getJSON( commentServlet, function() {
4
5      console.log( "success" );
6
7  } ) .done(function( data ) {
8
9      $.each( data.items, function( i, item ) {
10
11           <!-- Verarbeitung -->
12         console.log (item.comment);
13
14     });
15 });
```

Kapitel 4

Analyse

4.1 Anwendungsfälle

In diesem Kapitel werden die relevanten Anwendungsfälle dokumentiert und deren Bedingungen für eine erfolgreiche Durchführung festgehalten. Der Onlineshop besitzt weitere Anwendungsfälle wie z. B. ein Produkt zu erfassen oder ein Produkt zu kaufen, diese werden hier nicht aufgenommen da sie nicht im Fokus dieser Semesterarbeit stehen.

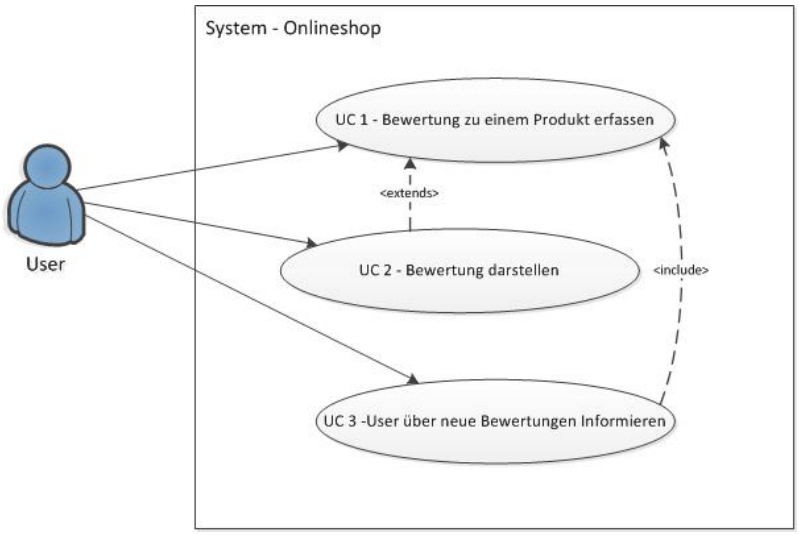

Abbildung 4.1: Systemanwendungsfälle

UC-1	Bewertung zu einem Produkt erfassen
Kurzbeschreibung	Benutzer kann Bewertungen erfassen, ein Produkt mit einem Text und Sternen bewerten.
Akteure	Benutzer
Auslösendes Ereignis	Benutzer erfasst eine Bewertung und drück Bewertung abgeben
Vorbedingung	Produkt gesucht
Eingehende Informationen	Bewertung des Benutzers
Ablauf (essentielle Schritte)	1. User loggt sich ein 2. Benutzer sucht Produkt 3. Benutzer erfasst und speichert neue Bewertung zu einem Produkt
Nachbedingungen	Bewertung ist auf der DB gespeichert
Kommentare	Die Bewertung kann nach dem Speichern nicht mehr geändert werden.

Tabelle 4.1: UC-1 Bewertung zu einem Produkt erfassen

UC-2	Bewertung darstellen
Kurzbeschreibung	Benutzer kann die Bewertungen zu einem Produkt anzeigen
Akteure	Benutzer
Auslösendes Ereignis	Benutzer klickt auf Bewertung auf der Suchen-Seite
Vorbedingung	Produkt gesucht
Eingehende Informationen	Bewertung aller Benutzers samt Sternen-Bewertung
Ablauf (essentielle Schritte)	1. User loggt sich ein 2. Benutzer sucht Produkte 3. Benutzer klickt Bewertungen anzeigen
Nachbedingungen	Alle Bewertungen werden angezeigt
Kommentare	Bewertungsdurchschnitt eines Produktes wird initial beim Suchen berechnet dieser wird nicht laufend angepasst.

Tabelle 4.2: UC-2 Bewertung darstellen

UC-3	User über neue Bewertungen Informieren
Kurzbeschreibung	Benutzer wird über die eingehenden Bewertungen für die gerade eingesehenen Produkte benachrichtigt
Akteure	System
Auslösendes Ereignis	Ein Benutzer bewertet ein Produkt
Vorbedingung	Produkt gesucht
Eingehende Informationen	Neueste Bewertungen aller Benutzers ausser eingeloggtem User
Ablauf (essentielle Schritte)	1. User loggt sich ein 2. Benutzer sucht Produkte 3. Ein anderer Benutzer erfasst und speichert eine Bewertung 4. Benutzer sucht während dieser Phase kein neues Produkt
Nachbedingungen	Bewertung eines anderen Users wird beim Produkt ersichtlich
Kommentare	Neue Bewertungen werden alle x Sekunden neu geladen, darum kann es bei einer neuen Suche sein, dass die Benachrichtigung nicht erscheint

Tabelle 4.3: UC-3 User über neue Bewertungen Informieren

4.2 Fachklassenmodell

Im Folgenden werden die Fachklassen mittels UML-Notation aufgezeigt nach Definition von Oesterreich [OB12]. Mittels diesen Klassen soll das Verständnis des behandelnden Objekts erleichtern.

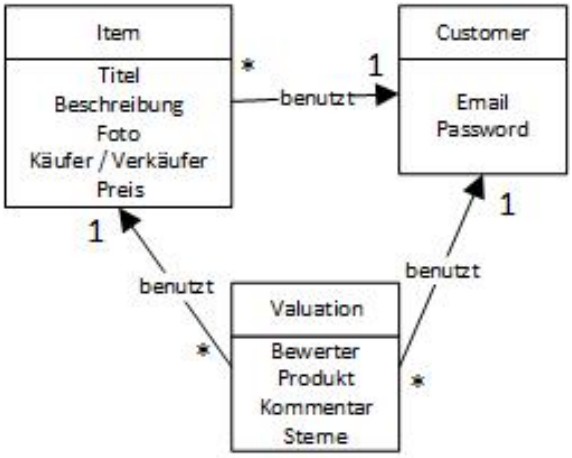

Abbildung 4.2: Fachklassenmodell

Es wird definiert, dass die Bewertung ohne Titel nur mit Kommentar und einer Sternen-Skala gemacht wird. Dabei werden die Benutzer und Items benutzt.

Kapitel 5

Entwurf

5.1 Klassendiagramme

Das Klassenmodell beschreibt die statische Struktur der Klassen in einem System sowie die Beziehungen untereinander. Zusätzlich werden Attribute und Methoden jeder Klasse hinzugefügt welche die Klasse beschreiben.

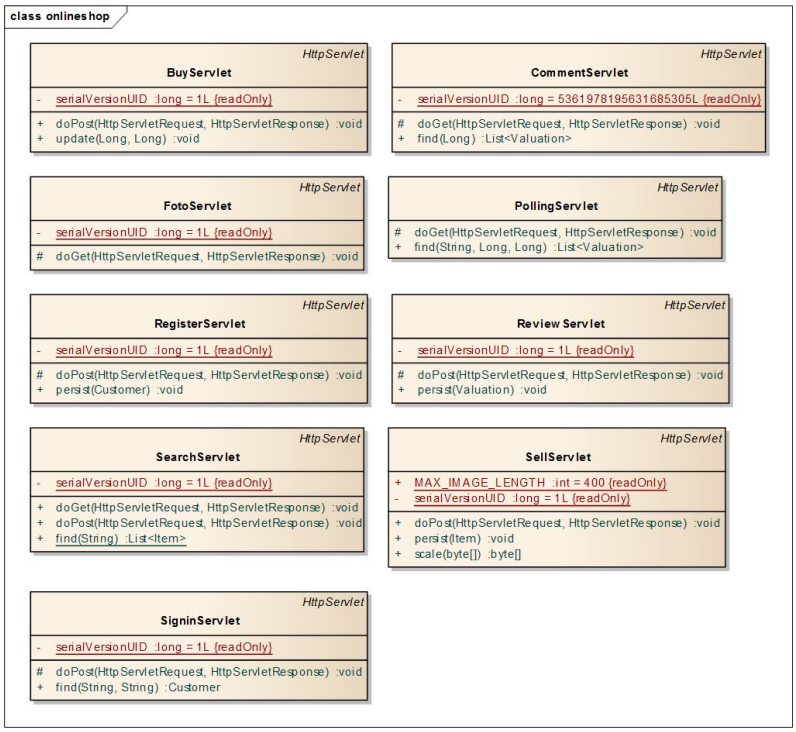

Abbildung 5.1: Servlets Klassendiagramm

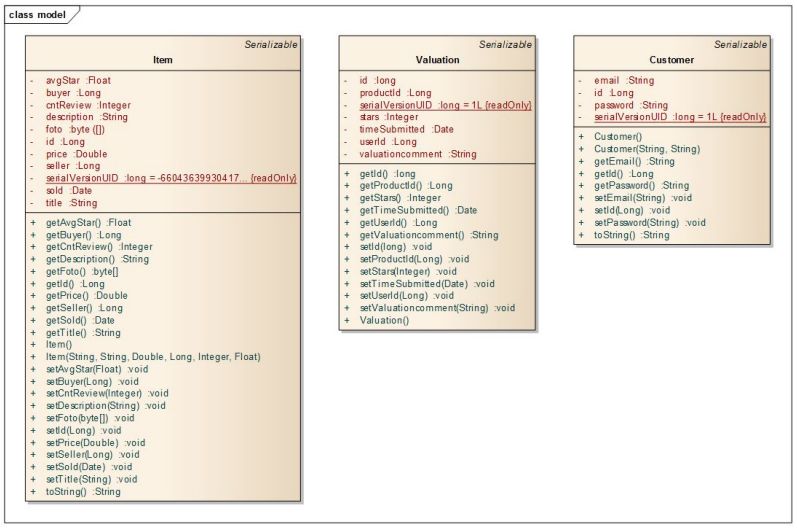

Abbildung 5.2: Model Klassendiagramm

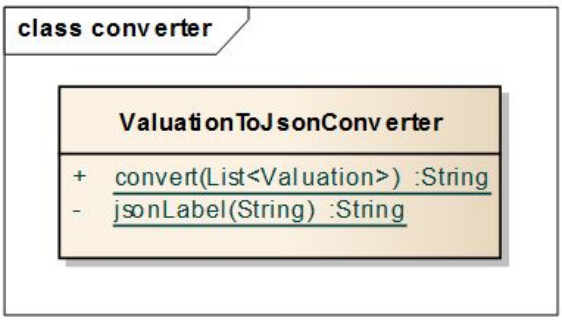

Abbildung 5.3: Converter Klassendiagramm

5.2 Datenmodell

Das folgende Datenmodell zeigt auf wie die Daten gespeichert werden sollen.

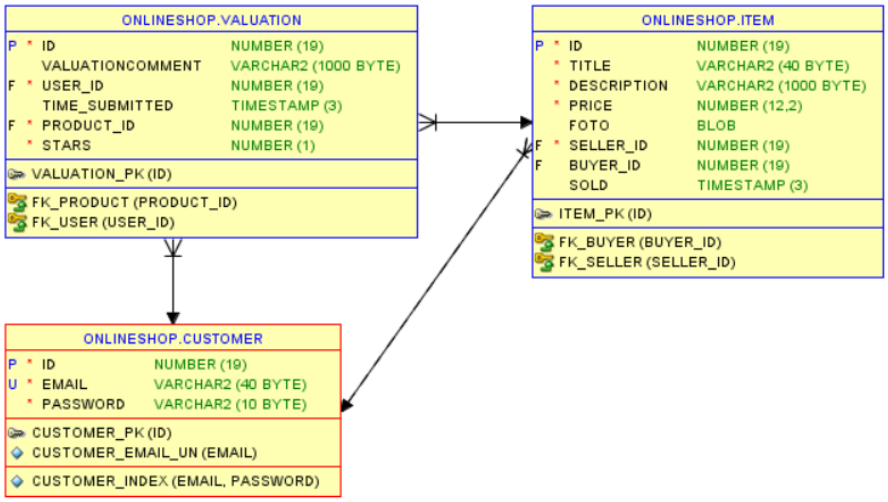

Abbildung 5.4: Datenschema UC1

Die Entität ”Valuation” beinhaltet den Kommentar zu einem Produkt der bereit von einem User erfasst worden ist, ebenfalls werden Zeitstempel und Erfasser protokolliert. Ein Rating von 1 bis 5 wird dem User vorgeschlagen und ebenfalls persistiert.

5.3 Sequenzmodell

Das folgende Sequenzmodell soll aufzeigen wie das Polling implementiert werden soll und welche Komponenten dabei betroffen sind.

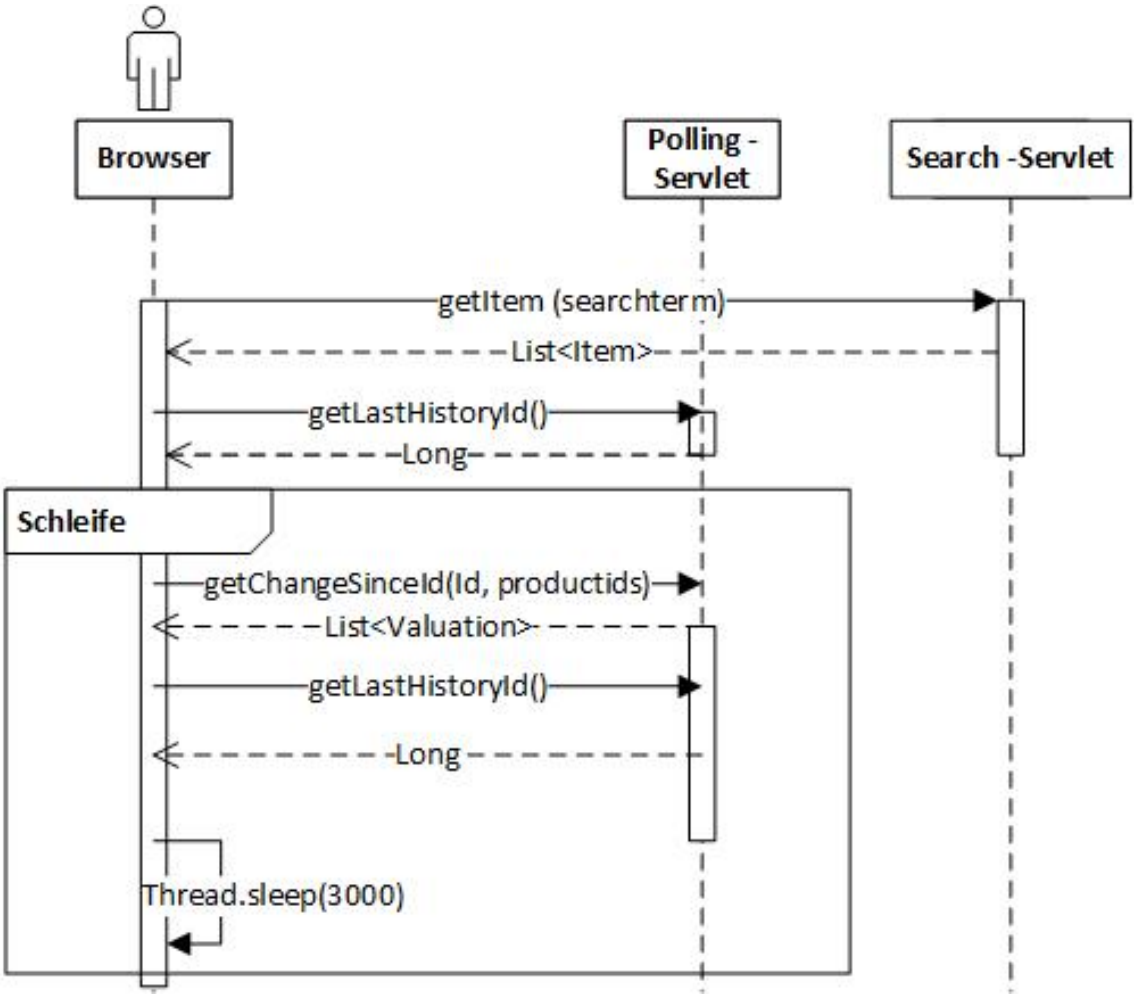

Abbildung 5.5: Mockup

Der erste Aufruf soll die zu kaufende Produkte darstellen. Anschliessend wird über das Polling-Servlet die höchste History-Id gesucht. Sobald ein neuer Eintrag geschrieben wird hätte es eine höhere History-Id in der Datenbank. Sobald dann gesucht wird ob es einen neuen Beitrag existiert wird dieser geliefert und dem User angezeigt. Anschliessend wird wieder die Höchste Id geholt und zwischengespeichert. Anschliessend werden 3 Sekunden gewartet bis zur nächsten Anfrage, damit der Anwendungsserver nicht die ganze Zeit befragt wird.

5.4 Testkonzept

Es wurde ein Testkonzept erarbeitet um die umgesetzten Anwendungsfälle auf der Gui zu testen.

Testfallnummer	**JEE-01**	
Name	Bewertung speichern	
Testschritte	**Daten**	**Erwartetes Resultat**
1 Produkt suchen		
2 Kommentar und Sterne angeben		
3 Bewertung abgeben klicken		Nachricht erscheint für Erfolgreiches Speichern

Tabelle 5.1: Testfall - Bewertung speichern

Testfallnummer	**JEE-02**	
Name	Bewertung laden	
Testschritte	**Daten**	**Erwartetes Resultat**
1 Produkt suchen		
2 Bewertung anzeigen klicken		Bewertung werden angezeigt
3 Bewertung anzeigen klicken		Bewertungen werde wieder versteckt
4 Bewertung anzeigen klicken		Bewertungen werden angezeigt immer gleich viele

Tabelle 5.2: Testfall - Bewertung laden

Testfallnummer	**JEE-03**	
Name	Durchschnitt Produktbewertung	
Testschritte	**Daten**	**Erwartetes Resultat**
1 Produkt suchen		
2 Produkt mit 1 bewerten		
3 Produkt mit 5 bewerten		
4 Produkt erneut suchen		Bewertung steht 3 weil 1 plus 5 und durch zwei gleich 3 ist

Tabelle 5.3: Testfall - Durchschnitt Produktbewertung

Testfallnummer	**JEE-04**	
Name	Benachrichtigung vom User	
Testschritte	**Daten**	**Erwartetes Resultat**
1 Produkt suchen	(Session1 S1)	Produkte werden angezeigt
2 Produkt suchen	(Session1 S2)	Produkte werden angezeigt
3 Bewertung einfügen und speichern	S1	Name wird richtig dargestellt
4	S2	Bewertung erscheint in Bereich Bewertungen vom Produkt
5	S1	Bewertung erscheint nicht da vom gleichen Benutzer erstellt

Tabelle 5.4: Testfall - Benachrichtigung vom User

5.5 Spezifikation der Bedienoberflächen

Um die Bewertungen einzelner User und die Bewertung von Produkten zu ermöglichen, wurde folgender Mockup erstellt welcher als Prototyp dienen soll.

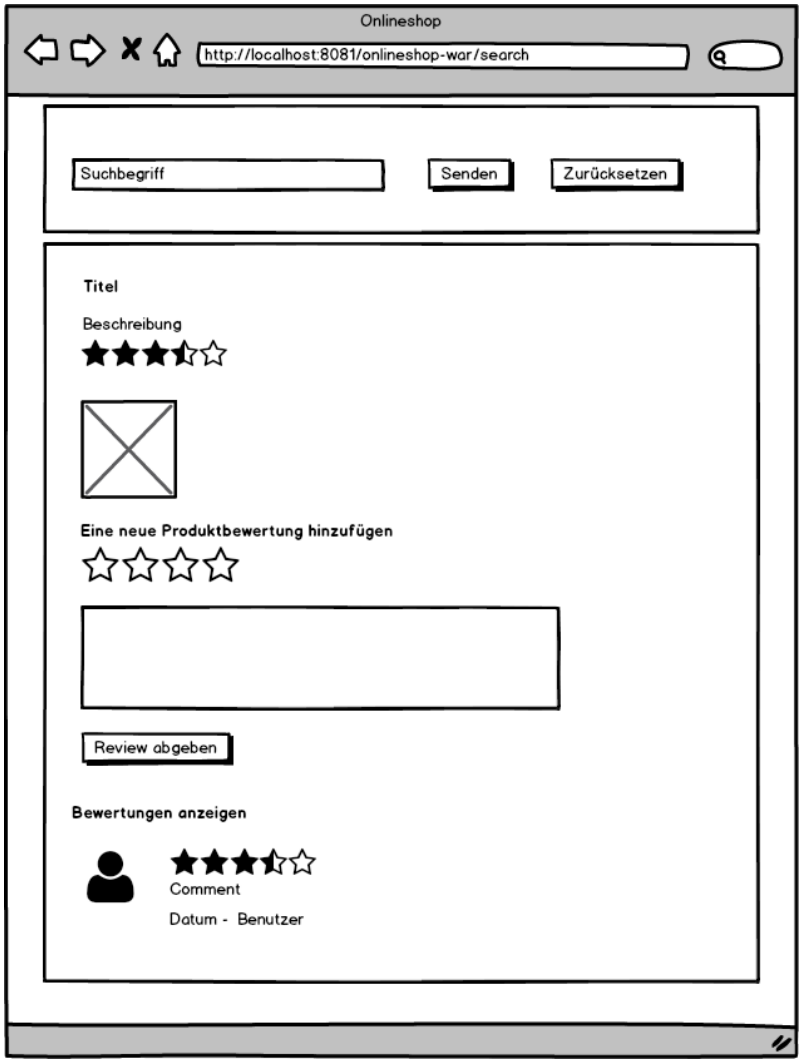

Abbildung 5.6: Mockup

Kapitel 6

Implementation

In diesem Kapitel werden die Spezialitäten der Implementation aufgezeigt. Es soll eine Hilfe darstellen um die entwickelte Applikation zu verstehen. Ebenfalls in diesem Kapitel wurden Stolperfallen und Probleme aufgenommen die während der Entwicklung aufgefallen sind.

6.1 Datenbankerweiterung

Man hat sich für eine Produktbewertung mit Sternen von 1-5 und Kundenkommentar entschieden, dafür wurde die Datenbank wie in folgt erweitert.

```
1  DROP TABLE onlineshop.valuation;
2
3  CREATE TABLE onlineshop.valuation (
4  id NUMBER(19) PRIMARY KEY,
5  valuationcomment VARCHAR2(1000) NOT NULL,
6  user_id NUMBER(19) NOT NULL,
7  time_submitted TIMESTAMP(3),
8  product_id NUMBER(19) NOT NULL,
9  stars NUMBER (1) NOT NULL,
10 CONSTRAINT fk_user
11 FOREIGN KEY (user_id) REFERENCES onlineshop.customer (id),
12 CONSTRAINT fk_product
13 FOREIGN KEY (product_id) REFERENCES onlineshop.item (id)
14 );
15
16 GRANT SELECT ON onlineshop.valuation TO onlineshop_role;
17 GRANT SELECT, INSERT, UPDATE, DELETE
18 ON onlineshop.valuation TO onlineshop_user;
19
20 DROP SEQUENCE onlineshop.seq_valuation;
21 CREATE SEQUENCE onlineshop.seq_valuation;
22
23 CREATE OR REPLACE TRIGGER tri_valuation
24 BEFORE INSERT ON onlineshop.valuation
25 FOR EACH ROW
26 BEGIN :NEW.id := onlineshop.seq_valuation.NEXTVAL;
27 END;
28 /
29
```

```
30 COMMIT;
```

Die Erweiterung umfasst die Bewertung eines Produkts abhängig vom User und Produkt, was bedeutet, es kann keine Bewertung ohne erfolgreiches Login durchgeführt werden. Die Session Informationen mussten ausgewertet werden um den kommentierenden User ausfindig zu machen.

6.2 Gui

Die Umsetzung die Webseite ist wie folgt Implementiert und orientiert sich an das Mockup welches zuvor definiert wurde.

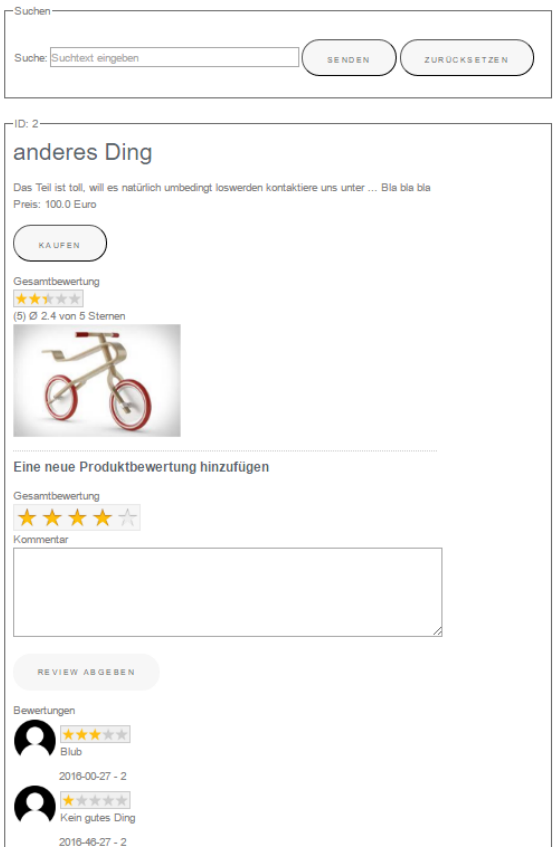

Abbildung 6.1: Gui-Resultat

6.3 jQuery-Funktionen

6.3.1 Sterne animieren

Um die Sterne zu animieren und damit diese sich gelb färben werden Klassen den einzelnen Elementen angehängt und wieder gelöscht. Die Klasse ”selected” markiert die schlussendlich gewählten Bewertung.

```
1
2
3   $('.rate-btn').hover(function() {
4         $('.rate-btn').removeClass('rate-btn-hover');
```

```
5           var therate = $(this).attr('id');
6           for (var i = therate; i >= 0; i--) {
7              $('.rate-btn-' + i).addClass('rate-btn-hover');
8           }
9           ;
10       });
11
12       $('.rate-btn').click(function() {
13          var therate = $(this).attr('id');
14
15          $('.rate-btn').removeClass('selected');
16          $('.rate-btn-' + therate).addClass('selected');
17
18          $('.rate-btn').removeClass('rate-btn-active');
19          for (var i = therate; i >= 0; i--) {
20             $('.rate-btn-' + i).addClass('rate-btn-active');
21          }
22          ;
23       });
```

6.3.2 Review speichern mit Feedback

Ein Review mit jQuery zu speichern ist ziemlich simpel. Die URL definiert im wesentlichen wohin die Daten gesendet werden sollen. Die Methoden ”success” und ”error” sind dazu da auf den Fall zu reagieren wenn der Aufruf misslungen oder erfolgreich war. Hier hat man sich entschieden dem User eine Meldung darzustellen.

```
1
2    $.ajax({
3        type : "POST",
4        url : "http://localhost:8081/onlineshop-war/review",
5        data : dataRate,
6        success : function() {
7            $('#review' + product_id).hide()
8            $('#log' + product_id).html(
9                '<div class="info"> Vielen Dank fuer Ihre Bewertung<//div>')
10           },
11       error : function() {
12           $('#review' + product_id).hide()
13           $('#log' + product_id).html('<div class="error"> Ach, nee irgendwas
                 ist in die Hose... )</div>')
14           }
15       });
16    });
```

6.3.3 Polling

Für das Polling wurde schlussendlich eine Lösung gefunden die gegen einen separaten Service sprach um die letzte History-Id zu laden, sondern gegen das Konstrukt wie im Sequenzdiagramm dargestellt wurde der Response direkt ausgewertet und in die globale Variable ”lastid” zwischengespeichert. Damit nicht gleich beim ersten Mal die Bewertungen dargestellt werden, wird geschaut

ob ”lastid” = 0 also die erste Durchführung ist, wenn ja wird keine Bewertung dargestellt. Anschliessend wartet die Anwendung drei Sekunden für den nächsten Durchlauf.

```
function doPoll(){
      var prodids = $("#loadedProducts").html();

      var pollingServlet =
          "http://localhost:8081/onlineshop-war/polling?productids="+prodids
          +"&lastId="+lastid;
        $.getJSON( pollingServlet, function() {
           console.log( "success" );

        } ).done(function( data ) {
            $.each( data.items, function( i, item ) {

              if (lastid != 0){
                 addComment(item);
              }
              if (item.id > lastid){
                 lastid = item.id;
              }
              $("#bewertungen"+ item.productId).show("slow");

            });

         });
           setTimeout(doPoll,3000);
   }
   });
```

Das Polling stellt nicht alle neuen Bewertungen dar. Es werden folgende Kriterien berücksichtigt:

- Der User wird nicht seine eigenen Bewertungen laden sehen.
- Spam wird nicht dargestellt. Spam ist wenn weder Kommentar noch einen Stern vergeben wurde.
- Der letzte Eintrag muss später erstellt sein als die ”lastid”.

Diese Kriterien sind im SQL ersichtlich der diese Daten abholt:

```
select val.*, (cust.email) as email from onlineshop.valuation val join
    onlineshop.customer cust
       on val.user_id = cust.id
       where  val.product_id in (?)
             and (length(val.valuationcomment) > 0 or val.stars >0)
             and val.user_id != ?
             and val.id > ?
     order by val.TIME_SUBMITTED
```

6.4 Probleme

SQL Developer UTF-8
Für den SQL Developer von Oracle muss darauf geachtet werden, dass dieser auf UTF-8 umge-

schaltet ist.
Extras > Voreinstellung> Umgebung > Codierung
Standardmässig ist der SQL Developer auf Cp1252 eingestellt und die Umlaute werden dementsprechend interpretiert.

ORA-00020: maximum number of processes
Dabei konnte man sich dann mit dem SQL Developer nicht mehr verbinden und die Verbindungen wurden irgendwo benutzt. Das Problem kann behoben werden durch einen Neustart der Maschine oder indem man von irgendwo her eine Verbindung kappt z.B. war bei einer Eclipse-Instanz mit Datenbankverbindung.

Anschließend kann dieser Befehl ausgeführt werden um die Verbindungsanzahl zu erhöhen:

```
1 alter system set processes=200 scope=spfile;
```

Probleme beim Starten von Glashfish
Der Applikationsserver hat plötzlich ohne Vorwarnung nicht mehr gestartet, es stoppt bei 69 % beim Laden der JMX Konfiguration. Das Problem ist in der Comunity bekannt siehe StackOverflow. Die Lösung ist besteht darin den Anwendungsserver aus dem Eclipse zu entfernen und anschliessend wiedereinzurichten. Ein weiteres Problem ist die gleichzeitige Belegung der Ports von Skype und Glassfish. Dieses Problem wurde behoben indem die Skype-Einstellungen angepasst wurden.

Kapitel 7

Test

7.1 Testprotokoll

Tabelle 7.1: Testergebnis JEE-01

Testfall ID	JEE-01 Bewertung speichern
Tester	Denis Bittante
Datum Testdurchführung	17.11.2016
Testergebnis	PASS
Fehlerbeschreibung	Eine Unschönheit bei den Sternen: nachdem der Cursor sich von den Sternen wegbewegt, bleibt die Skale am letzten berührten Stern hängen, nicht am selektierten Stern.

Tabelle 7.2: Testergebnis JEE-02

Testfall ID	JEE-02 Bewertung laden
Tester	Denis Bittante
Datum Testdurchführung	17.11.2016
Testergebnis	PASS
Fehlerbeschreibung	Unschönheit, die Zeilenumbrüche werden nicht dargestellt.

Tabelle 7.3: Testergebnis JEE-02

Testfall ID	JEE-02 Bewertung laden
Tester	Denis Bittante
Datum Testdurchführung	25.11.2016
Testergebnis	PASS - Zeilenumbrüche werden nun dargestellt. Weitere unschönheit UTF8 wird nicht richtig dargestellt. Umlaute werden durch Rombusse ersetzt.
Fehlerbeschreibung	

Tabelle 7.4: Testergebnis JEE-03

Testfall ID	JEE-03 Durchschnitt Produktbewertung
Tester	Denis Bittante
Datum Testdurchführung	17.11.2016
Testergebnis	PASS
Fehlerbeschreibung	

Tabelle 7.5: Testergebnis JEE-04

Testfall ID	JEE-04 Benachrichtigung vom User
Tester	Denis Bittante
Datum Testdurchführung	17.11.2016
Testergebnis	PASS - Die Bewertung wird automatisch geladen, Unschön es ist nicht so ”ersichtlich” ohne Animation
Fehlerbeschreibung	

7.2 JUnit-Test

Der einzige Unit-Test der genutzt wurde, war um zu testen ob der Konverter ein korrektes JSON-Objekt erstellt. Dafür wurde ein Valuation-Objekt gefüllt und mittel String-Compare die Inhalte verglichen.

```
1
2  @Test
3     public void test() {
4        Date timeSubmitted = new Date(Long.valueOf("1480041257521"));
5
6        ValuationToJsonConverter testee = new ValuationToJsonConverter();
7
8        ArrayList<Valuation> arrayList = new ArrayList<>();
9        Valuation e = new Valuation();
10       e.setId(Long.valueOf(10));
11       e.setProductId(Long.valueOf(2));
12       e.setStars(2);
13       e.setTimeSubmitted(timeSubmitted);
14       e.setUserEmail("a@b.ch");
15       e.setUserId(Long.valueOf("1"));
16       e.setValuationcomment("ist ja toll");
17       arrayList.add(e);
18       String convert = testee.convert(arrayList);
19
20       String[] split = convert.split("\n");
21       Assert.assertEquals("{ \"items\": [ {", split[0]);
22       Assert.assertEquals("\"id\" : 10,", split[1]);
23       Assert.assertEquals("\"productId\" : 2,", split[2]);
24       Assert.assertEquals("\"stars\" : 2,", split[3]);
25       Assert.assertEquals("\"time\" : \"2016-11-25 03:34:521\",", split[4]);
26       Assert.assertEquals("\"comment\" : \"ist ja toll\",", split[5]);
27       Assert.assertEquals("\"userName\" : a@b.ch}] }", split[5]);
28
29    }
```

Kapitel 8

Evaluierungsbericht

8.1 Zusammenfassung

Anfangsschwierigkeiten mit der Infrastruktur, der GlassFish war etwas schwierig hat dennoch mit den Ressourcen alles geklappt. Der Einsatz von jQuery hat sich schlussendlich einfacher präsentiert als gedacht, die ersten Schritten waren nicht ganz einfach, doch mit etwas Übung wurde die Syntax für die Dokument-Aufrufe klar.

Etwas streng wurde es am Schluss um die Dokumentation auf den gewünschten Stand zu bringen, da diese in der Struktur nicht den gewünschten Vorgaben entsprach, wurde die Arbeit nochmals auf den Kopf gestellt.

8.2 Schwierigkeiten

Es gab Schwierigkeiten mit den Einstellungen der Entwicklungsumgebung und der Lektüre die etwas umfangreicher Anfiel als angenommen. Querlesen brachte immer wieder Lücken auf die, wenn das Buch von Deckel bis Rücken durchgearbeitet wird, nicht aufkommen. Vor allem wollte man den Onlineshop schnell auf die Beine bringen und man hat schnell mal einige Kapitel dafür übersprungen.

8.3 Fazit

Die Entwicklung seitens Java-Backend, hat keine grosse Mühe bereitet, die Entwicklung wurde sehr gut von Eclipse unterstützt und bei Problemen einfach kurz den Debug-Modus eingeschaltet konnte den Fehler schnell gefunden werden. Das Front-End konnte ebenfalls schnell verändert und angepasst werden. Das Mockup konnte gut implementiert werden. Ebenfalls gut war das Debuggen von Response-Informationen, Chrome bietet ganz gute Entwicklertools an um den Netzwerkverkehr einzusehen und dabei Fehler zu finden ist ebenfalls einfach, z.B.: Parset Chrome JSON Objekte automatisch und wenn diese fehlerfrei sind werden sie in einer Baumstruktur dargestellt.
Was etwas weniger gut verlief war die Implementation mittels Java-Script. Die Umsetzung wurde etwas blind durchgeführt da das Intellisense auch von Chrome und Firefox nicht immer zu 100% funktionierte und darum immer wieder eine Variable falsch geschrieben wurde und dann fing das Suchen an, speziell war produktId, produktIds, productid und produkt_id konsistent zu halten.

Was ebenfalls mühe machte war das stetige Deployment, das je nach Lust und Laune des GlassFish durchgeführt wurde. Es war etwas schwierig nachzuvollziehen wann nun Java-Code, nur die JSP-Seiten oder beide auf dem Server publiziert wurden. Somit wurde immer wieder gleich die ganze Maschine von neuem gestartet.

Die Entwicklung mit jQuery hat ziemlich gut geklappt leider aber fehlt die Struktur wie mit Java-Script entwickelt werden soll. Fragen wie diese haben mich etwas verunsichert:

- Wann sollte man ein eigenes JavaScript-File erstellen?
- Wie modelliert man ansprechende Modelle und wo legt man diese ab?
- Ist es besser einen Dokumenten-Knoten neue Elemente hinzuzufügen oder HTML-Strukturen als String?
- Wie validiert man mit jQuery?
- Wie testet man mit dem Framework QUnit?
- Wie validiert man JSON?

Diese Wissenslücken sind für mein persönliches Studium wichtig und werden hoffentlich in einer anderen Arbeit vertieft.

Selbstständigkeitserklärung

Die Verfasser erklären, dass sie die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt haben. Die aus fremden Quellen (einschließlich elektronischer Quellen) direkt oder indirekt übernommenen Gedanken sind ausnahmslos als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden.

Bern, 8. Dezember 2016

Denis Bittante

Kapitel 9

Anhang

9.1 Statusbericht 1

9.1.1 Übersicht

Bereich	Status	Problem	Massnahmen
Gesamtbeurteilung	Grün		
Infrastruktur	Gelb	JDBC Ressource - GlassFish zeitintensiv	keine
Ergebnisse	Gelb	Template konnte nicht gestartet werden	Troubleshooting
Forumsbeitrag	Grün		
Semesterziel	Grün		

Tabelle 9.1: Statusbericht 1 Übersicht

Legende: Grün: Semesterarbeit auf Kurs, Gelb: Erfolg der Semesterarbeit gefährdet, Rot: Erfolg der Semesterarbeit stark gefährdet

9.1.2 Gesamtbeurteilung

Die Infrastruktur für die Semesterarbeit konnte aufgebaut werden. Das Template von der Homepage wurde heruntergeladen und im Eclipse-Workspace geladen. Oracle wurde installiert und anhand der Installationsanleitung im Kapiel 6 eingerichtet. GlassFish wurde installiert, Domänen installiert und in Betrieb genommen. Installation der JDBC-Connection Pools wurde eingerichtet, kostete aber etwas Zeit (Forumsbeitrag von Marc Sallin war hilfreich). Das Fokus Thema ist dokumentiert und zur Prüfung an Herrn Zimmermann mit Statusbericht 1 eingericht siehe Kapitel 'Semesterarbeit Fokus'. Forumsbeitrag bezüglich Datenbank-Tool SQL Developer erstellt und freigegeben. Das Template der Arbeit High-Level (JFS) konnte noch nicht gestartet werden, ist aktuell in der Troubleshooting-Phase. Darum ist der Punkt Gelb markiert.

9.2 Statusbericht 2

9.2.1 Übersicht

Bereich	Status	Problem	Massnahmen
Gesamtbeurteilung	Grün		
Infrastruktur	Grün		keine
Ergebnisse	Gelb	AJAX Einsatz	JQuery eingesetzt
Forumsbeitrag	Grün		
Anwendungsfall 1	Gelb	Fehlende Zeit	Fach höher priorisiert
Semesterziel	Grün		

Tabelle 9.2: Statusbericht 2 Übersicht

Legende: Grün: Semesterarbeit auf Kurs, Gelb: Erfolg der Semesterarbeit gefährdet, Rot: Erfolg der Semesterarbeit stark gefährdet

9.2.2 Gesamtbeurteilung

Es wurde der Anwendungsfal l in Angriff genommen das Datenbankschema ist angepasst. Die Produkte können nun bewertet werden mit einer Skala von 0-5. Es wurden die Anzahl Kommentare ebenfalls persistiert um den Durchschnitt zu berechnen. Es wurde begonnen mit AJAX zu implementieren und Tutorial durchzuspielen. Es wurde in Betracht gezogen das Framework JQuery zu benutzen um die Aufrufe zu handeln. Die JSP Seiten konnten noch nicht angepasst werden. Es wurde entschlossen das Low-Level-Template zu nutzen, weil dadurch die Komplexität sich im Rahmen hält um die Semesterarbeit im gesetzten Zeitrahmen zu beenden. Das zu bearbeitende Kapitel im Buch ist anspruchsvoll und lange, dabei ist es schwierig ebenfalls viel an der Semesterarbeit zu Implementieren. Es soll dennoch machbar sein diese fertig zu stellen. Der Forumsbeitrag wurde erstellt.

9.3 Statusbericht 3

9.3.1 Übersicht

Bereich	Status	Problem	Massnahmen
Gesamtbeurteilung	Grün		
Infrastruktur	Grün		
Ergebnisse	Grün		
Forumsbeitrag	Grün		
Anwendungsfall 1	Grün		
Anwendungsfall 2	Grün	In Arbeit	keine
Semesterziel	Grün		

Tabelle 9.3: Statusbericht 2 Übersicht

Legende: Grün: Semesterarbeit auf Kurs, Gelb: Erfolg der Semesterarbeit gefährdet, Rot: Erfolg der Semesterarbeit stark gefährdet

9.3.2 Gesamtbeurteilung

Die Umgebung konnte vollständig aufgebaut werden und die Umsetzung des ersten Anwendungsfalls abgeschlossen werden. Der User kann somit nun einen Beitrag mit samt Bewertung zu einem Produkt erstellen und speichern. Anwendungsfall 2 - die Darstellung der Bewertungen - ist nun in Arbeit. Entschieden wurde die Arbeit auszuweiten und ein weiteres Framework einzubinden. D3 bietet verschiedene grafische Daten basierte Auswertungen von statisch bis hoch interaktiv. Da diese Arbeit die grafische Erarbeitung nicht im Mittelpunkt gestellt bietet sich eine solche Library förmlich an.
Die Arbeit ist soweit noch im grünen Bereich, es muss etwas mehr Zeit in die Dokumentation gesteckt werden und die theoretische Erarbeitung des Fokus ist noch nicht vollständig ist aber in Arbeit.
Das Moodle-Forum ist mit einem Problem, dass mich etwas aufgehalten hat, bestückt. Die Installation der Datenbank hat funktioniert, leider aber waren die Anzahl Verbindungen sehr limitiert gewesen und haben dazu geführt das meine Verbindung mit dem SQL Developer nicht mehr möglich waren und von der DBMS abgewiesen wurden.

9.4 Auftragsbeschreibung

Ausgehend von der Beispielanwendung aus dem Lehrbuch entwickelt jeder Studierende einen eigenen Onlineshop. Dabei bearbeitet er ein individuelles Fokus-Thema, welches er in Absprache mit dem Dozenten wählt. Der Onlineshop muss insgesamt funktionsfähig und dokumentiert sein. Das Hauptgewicht der Semesterarbeit liegt auf der theoretischen Behandlung, der Analyse, dem Design, der Implementation und dem Test des ausgewählten Fokus-Themas.

9.4.1 Rahmenbedingungen

Jeder Studierende erstellt monatlich einen Statusbericht zuhanden des Dozenten über den Fortgang seiner Semesterarbeit. Er lädt diesen jeweils am Mittwoch vor der zweiten, dritten und vierten Präsenzveranstaltung in Moodle hoch. Die Studieren präsentieren und diskutieren ihre Erkenntnisse aus der Semesterarbeit in einem dafür vorgesehenen Forum. Dabei erstellt jeder Studierende vor der zweiten, dritten und vierten Präsenzveranstaltung mindestens je einen Beitrag. Die Semesterarbeit muss vor der fünften Präsenzveranstaltung abgegeben werden. Abzugeben sind die schriftliche Arbeit und der selbst erstellte Quellencode

9.4.2 Beurteilungskriterien

Die Gesamtbeurteilung ergibt sich aus folgenden Teilnoten:

- Statusberichte (10%)
- Forumsbeiträge (10%)
- Dokumentation des Onlineshops (10%)
- Theoretische Behandlung des Fokus (20%)
- Dokumentation von Analyse, Entwurf, Implementation und Test des Fokus (20%)
- Dokumentation des persönlichen Erkenntnisgewinns (20%)
- Mündliche Präsentation (10%)

Literaturverzeichnis

[Boh15] Melissa Bohlsen. Produktbewertungen als harte währung, 2015.

[OB12] B. Oestereich and S. Bremer. Analyse und Design mit der UML 2.5: Objektorientierte Softwareentwicklung. Oldenbourg Wissenschaftsverlag, 2012.

[Sal14] Alexander Salvanos. Professionell entwickeln mit Java EE 7 - Das umfassende Handbuch. Rheinwerk Verlag GmbH, Bonn, 1. aufl. edition, 2014.