Übungsblatt 10

Willkommen zum Praktikum zu Web-basierte Anwendungen. Beachten Sie die Hinweise.

Hinweis 1. Dies ist die zweite bewertete Abgabe. Die Aufgaben von diesem Übungsblatt sind zu lösen und abzugeben.

- Abgabe: Spätester Abgabetermin ist **Freitag, 6. Juli 2012, 24:00 Uhr**. Packen Sie eine Zip-Datei, ein PDF der Dokumentation Ihrer Anwendung sowie eine auf Tomcat deploybare WAR-Datei mit! inkludierten Quellen. Laden Sie diese Zip-Datei hoch unter `read.mi.hs-rm.de`. Der Zugriff über Web-Browser mit Ihrer Anwendung gegen eine frisch gefüllte Datenbank muss möglich sein. Die Datenbank muss (entweder in `web.xml` oder `persistence.xml`) konfigurierbar sein.
- Gruppenarbeit: Es soll in Gruppen gearbeitet werden. Mindestens zwei und maximal vier Teilnehmer je Gruppe. Jeder Gruppenpartner muss sich in *allen* Aufgaben und deren Lösung genau auskennen. Bei berechtigten Zweifeln kann es Rücksprachen und Punktabzug geben.
- Kopien: Die Aufgaben sollen individuell und eigenständig gelöst werden. Identische Lösungen verschiedener Abgaben werden mit Punktabzug bestraft.

Hinweis 2. Die Daten einer Anwendung zum Ausleihen von Filmen (DVD-Verleih) ist gegeben. Die Datenbank basiert auf der Sakila-Datenbank[1] für PostgreSQL, allerdings ohne Sichten und Funktionen; also nur Tabellen und Sequenzen. Das relationale Schema liegt fest, die Datenbank ist gut gefüllt. Ihre Aufgabe ist basierend auf dieser Datenbank einige einfache Oberflächen und kurze Prozessschritte zu realisieren. Die Zielgruppe sind dabei hauptsächlich Ausleiher und Mitarbeiter.

Aufgabe 1. Zeigen Sie alle Filmdaten an. Achtung! Es gibt von 1.000 Filmen etwa 5000 Kopien. Machen Sie eine einfache Navigation möglich um in den Filmen zu browsen, zum Beispiel alphabetisch vor sortiert und dann eine begrenzte Anzahl von Filmdaten je Seite mit vor- und zurückblättern. Erlauben Sie die Einschränkung der Filme nach Kategorie oder Altersfreigabe. Erlauben Sie auch die Anzeige von Schauspielern und die Anzeige von Kunden.

Aufgabe 2. Benutzer sind daran interessiert schnell Daten von Filmen beziehungsweise Kunden zu finden. Es soll möglich sein in Eingabefeldern (*je Attribut ein Eingabefeld*) einen Teil der Daten eines Films/Kunden einzugeben und dann die passenden Filme/Kunden angezeigt zu bekommen, die den Kriterien entsprechen. Machen Sie es möglich bei der Anzeige der gefundenen Einträge nach einem beliebigen Kriterium abwechselnd aufsteigend/absteigend zu sortieren. Sie sollten die Anzahl der Suchergebnisse auf einer Seite begrenzen.

Aufgabe 3. Die gefundenen Filme/Kunden der letzten beiden Aufgaben sollen mit einem Klick auswählbar sein. Neben den Detaildaten eines Films sollen die Kopien des Films

[1] `http://dev.mysql.com/doc/sakila/en/sakila.html`
`http://cvs.pgfoundry.org/cgi-bin/cvsweb.cgi/dbsamples/dbsamples/sakila`

und die Schauspieler angezeigt werden. Falls eine Filmkopie ausgeliehen ist, dann soll ein Link auf den Ausleiher erscheinen. Bei einem Ausleiher (einem Kunden) sollen auch alle aktuell ausgeliehenen Filme erscheinen. Die Daten eines Kunden sollen auf Anfrage editierbar und speicherbar sein. Stellen Sie sicher, dass auf parallele Änderungen passend reagiert wird. Ausgeliehene Filmkopien sollen zurückgegeben werden können.

Aufgabe 4. Erstellen Sie eine einfache Anwendung, die es Kunden erlaubt Filmkopien auszuleihen. Diese Anwendung richtet sich an Kunden, nicht an Mitarbeiter, und soll daher eine unabhängige Oberfläche haben. Die Kundenidentifikation als auch das Passwort eines Kunden sei die Kundennummer. Stellen Sie sicher, dass nur ein Kunde eine Filmkopie ausleihen kann und nur aktuell nicht verliehene Filme ausgeliehen werden können.

Hinweis 3. Realisieren Sie alle Aufgaben außer Aufgabe 4 als JavaServer Faces Anwendung. Realisieren Sie Aufgabe 4 als JSP-Anwendung (mit JSTL, Servlets und Expression Language; Scriptlets sind verboten). Sie sollten die vorgegebene Realisierung für den Datenzugriff verwenden (empfohlen). Wer möchte (nicht empfohlen) darf sich auch selbst Datenbeans schreiben mit zum Beispiel direktem SQL-Zugriff über JDBC. In der Dokumentation muss ersichtlich sein wie man eine andere Datenbank (Host, Datenbankname, User, Passwort) anbindet.

Erstellen Sie eine für den Benutzer angenehme und einfach zu bedienende Oberfläche. Machen Sie sich Gedanken über die Interaktionslogik und eine einfache Bedienung. Dies ist jedoch unabhängig vom Aussehen und sie sollten auf aufwändiges Rendern verzichten. Wenn Ihrem persönlichem Empfinden nach notwendig, dann verwenden Sie CSS in separaten Style-Dateien und verwenden Sie bei den UI-Komponenten passende Attribute. Die Anwendung muss bei ausgeschaltetem CSS bedienbar sein.

Wenn die zur Verfügung stehenden Methoden zum Zugriff auf die Daten nicht passend erscheinen, dann schreiben Sie sich Ihre eigenen Beans, die sich die Daten passend holen und aufbereiten aufbauend auf den vorhanden Möglichkeiten. Es ist nicht erlaubt die vorhandene Persistenzschicht zu ändern; man darf sich jedoch weitere Zwischenschichten (mit Beans) erstellen. Versuchen Sie sauber zu unterscheiden zwischen von JSF-abhängigen Backing-Beans und für die Anwendungslogik notwendige und relevante Beans, die von JSF unabhängig sind. Viele, aber nicht unbedingt alle, dieser von JSF-unabhängigen Beans können die Beans der Datenschicht sein.

Hinweis 4. Verwenden Sie PostgreSQL an der Hochschule für Ihre Datenbank so wie in Übungsblatt 4 erklärt. Falls Sie JPA verwenden, dann müssen Sie am Ende noch zusätzlich das Skript `pgpb_sakila_jpa_changes.sql` ausführen. Wenn Sie sich für die einfach zu verwendende JPA-Persistenzschicht entscheiden (stark empfohlen), dann schauen Sie sich den Ordner `Persistenz` und `PersistenzSakila` im Praktikumsordner

```
/opt/share/praktika/WebAnw
```

an. Lesen Sie die dort vorhandene Dokumentation.

`http://www.mi.hs-rm.de/~barth/hsrm/webanw`