Bearbeitungszeitraum: 3 Wochen

Bearbeitungszeit: 19,5 Stunden (Nacharbeiten Vorlesung ist inbegriffen)

Zur Abgabe dieser und der folgenden Aufgabenblätter gehört unbedingt auch die Zeit, die Sie für die Lösung dieser Aufgaben benötigt haben. Notieren Sie die benötigte Zeit pro Aufgabe! Die Zeit soll auch die Zeit enthalten, die Sie für das Nachschauen / Nacharbeiten des Vorlesungsstoffs benötigt haben.

Achtung ubuntu User: beachten Sie Anhang

Aufgabe	a)	b)	c)
Zeit / min			

a) Wandeln Sie die in den vorigen Aufgabenblättern erarbeitete Lehrveranstaltungsverwaltung in eine Web Applikation um. Der Funktionsumfang soll dabei dem Funktionsumfang der Lehrveranstaltungsverwaltung aus Aufgabe 3 entsprechen.

- Technologie:
  Verwenden Sie Java Server Faces wie in der Vorlesung beschrieben.
- Benutzen Sie die localhost URL

Randbedingungen:

- Sie können davon ausgehen, dass jeweils nur ein Sachbearbeiter oder Admin gleichzeitig mit der Anwendung arbeitet.
- Sie müssen folgende in der Vorlesung genannten Möglichkeiten von Java Server Faces in Ihrer Anwendung ausprobieren:
  - KomboBox
  - RequestScoped
  - ApplicationScoped
  - Prüfung auf leere Eingaben
  - PanelGrid

- Die MVC Struktur Ihres bisherigen Programms aus den vorigen Aufgabenblättern soll erhalten bleiben. Versuchen Sie die Entitätsklassen und die Kontrollklassen so unverändert wie möglich wieder zu verwenden.
  - Gelingt das?
  - Wenn nein, was hätte man von Anfang an besser machen sollen?

Zur Vorbereitung dieser Aufgabe kopieren Sie zunächst wie gewohnt das Projekt aus Aufgabe 3 in ein neues Projekt mit Namen LehrveranstaltungsVerwaltung4 kopieren.

Tips:

Die Installation der Eclipse Umgebung für JSF wird in den Vorlesungsfolien beschrieben

Die Schnittstellenschicht der Swing GUI Aufgabe hatte 3 Aufgaben:

1. Navigation von Dialog zu Dialog
2. Komposition der GUI Komponenten
3. Datentransfer von und zu der Kontrollschicht

In einer JSF Anwendung arbeiten xhtml Dateien mit Java Klassen (ManagedBeans) zusammen. Die Komposition muss von den xhtml Dateien durchgeführt werden während der Datentransfer von den ManagedBeans vorgenommen wird. Die Navigation wird von den xhtml Dateien alleine oder in Kombination mit den ManagedBeans bewerkstelligt.

1. Navigation

Das JSF commandLink Tag ermöglicht es, im action Attribut eine neue Seite anzugeben, zu der hin navigiert werden kann. Damit kann das AdministratorAS oder SachbearbeiterAS Menü erstellt werden. Dadurch können die Klassen AdministratorAS.java und SachbearbeiterAS.java durch zwei xhtml Dateien AdministratorAS.xhtml und SachbearbeiterAS.xhtml ersetzt werden, die jeweils die Links zu den Anwendungsfällen des Akteurs enthalten. Die Links weisen auf die xhtml Seiten der Anwendungsfälle. Dafür hat jeder Anwendungsfall eine eigene xhtml Datei mit dem Namen <Akteur><Anwendungsfall>AAS.xhtml.

Wie kommt man z.B. nach erfolgreicher Durchführung des Anwendungsfalls AdminSachbearbeiterBearbeiten wieder zur Seite mit dem Administrator Menü? In dem commandButton Tag des OK Kopfs wird hinter dem action Attribut eine Methode aufgerufen, die die Datenverarbeitung vornimmt (siehe weiter unten). Diese Methode kann einen String zurückgeben. Dieser String soll der Pfad zu der Seite AdministratorAS.xhtml sein. Dadurch erscheint bei erfolgreicher Durchführung der Datenverarbeitung wieder die Menü Seite AdministratorAS.xhtml.

2. Komposition

Die JSF xhtml Dateien komponieren mit Tags die graphische Bedienoberfläche. Beispielsweise können mit outputText und inputText Tags bestimmte Formulare erstellt werden. Jedem Anwendungsfall entspricht eine xhtml Seite, die diesen Anwendungsfall bearbeitet (siehe oben). Die xhtml Dateien können nicht in das Java src Verzeichnis einsortiert werden, sondern gehören in das WebContent Verzeichnis. Man erstellt dort die gleiche Verzeichnisstruktur wie im src Ordner z.B. das Verzeichnis administratorDK, adminSachbearbeiterEditierenDK usw.

Viele Anwendungsfälle haben gemeinsame Elemente. Diese gemeinsamen Elemente wurden in der Swing Anwendung des letzten Aufgabenblattes in Sicht Klassen wie SachbearbeiterS ausgelagert. Dies Prinzip zur Redundanzvermeidung können wir beibehalten. Im Fall von Swing war SachbearbeiterS im Wesentlichen ein JPanel mit Textfeldern für Benutzername und Passwort. Dieses JPanel konnte dann den JFrames von SachbearbeiterSachbearbeiterEditierenAAS, AdminSachbearbeiterEditierenAAS sowie LoginAS hinzugefügt werden. Ein ähnliches Prinzip können wir mit der include Anweisung bei JSF verwirklichen. Wir schaffen eine eigene xhtml Datei für Benutzername und Passwort (BenutzernamePasswort.xhtml). Dazu kommt eine eigene xhtml Datei für die Checkbox für die Berechtigung (Berechtigung.xhtml). Jetzt können wir z.B. AdminSachbearbeiterEditierenAAS.xhtml schreiben, indem wir Berechtigung.xhtml und Berechtigung.xhtml inkludieren. Zusätzlich fügen wir noch einen commandButton für OK und für Abbruch hinzu. Für AdminSachbearbeiterEditierenAAS sowie LoginAS gehen wir analog vor. Einem action Attribut eines commandButtons für Abbruch geben wir direkt den Pfad zu dem Menü, zu dem zurückgekehrt werden soll.

4. Datentransfer

In einem Browser können die Formulareinträge mit Hilfe des commandButton Tag per Knopfdruck als HTTP POST Befehl abgeschickt werden. Der Datentransfer geschieht dabei mit value expressions der expression language. Die Daten kommen in einer oder mehreren ManagedBeans an und können dort verarbeitet werden. Die ManagedBeans sind Java Klassen. Welche von unseren Klassen aus den vorhergehenden Aufgabenblättern sollten wir als ManagedBeans wieder verwenden? Entsprechend unserer Schichtenarchitektur können wir nur Klassen der Schnittstellenschicht als ManagedBeans verwenden, da die anderen Schichten nicht durch Besonderheiten des verwendeten GUI Framework (hier JSF) verschmutzt werden sollen. Hier müssen wir jedoch einen (kleinen) Bruch dieser Vorschrift in Kauf nehmen. Damit die Sachbearbeiter Entitäten über einen Request/Response Zyklus hinaus leben, müssen auch sie ManagedBeans werden (mit welchem Scope?). Die AAS Klassen dienen als Zwischenspeicher z.B. für Benutzername und Passwort und tauchen in EL Expressions bei den entsprechenden inputText Tags auf.

5. Validierung / Fehlerbehandlung

Wir überprüfen die Eingaben aus dem Formular in der EL Methode, die durch das action Attribut des commandButton Tags bezeichnet wird. Folgende Namenskonvention soll gelten: die action Methoden heißen wie der zugehörige Anwendungsfall. Die action Methode des OK command Buttons von AdminSachbearbeiterBearbeiten heißt also `sachbearbeiterBearbeiten`.

Aufgrund des Lebenszyklus einer Request/Response Anfrage sind zum Zeitpunkt der Ausführung einer solchen action Methode bereits alle Werte in die ManagedBeans eingetragen. Die action Methoden entsprechen den actionPerformed Methoden, die in einer Swing Anwendung durch Knopfdruck auf einen JButton ausgeführt werden. Sie greifen auf die in den AAS Klassen gespeicherten Werte folgendermaßen zu:

```
SachbearbeiterS sachbearbeiterS = (SachbearbeiterS)
util.JSFUtils.getELBean("sachbearbeiterS");

String neuerBenutzername = sachbearbeiterS.getbenutzername();
```

In die Methode `getELBean` muss der EL Name der Klasse als String eingegeben werden. Nun kann die Fehleingabenanalyse erfolgen. Im Falle, dass kein Fehler eintritt, werden die neuen Daten mittels der K-Klassen in die Entitätsschicht eingetragen. Die action Methoden liefern als Rückgabewert eine String mit dem Pfad zur Seite, zu der als nächstes navigiert werden soll.

Wenn nach Betätigen des OK Buttons ein Fehler auftritt muss die action Methode des Buttons null zurückliefern. Dadurch wird wieder dieselbe Seite z.B. mit Benutzername und Passwort angezeigt. Gleichzeitig soll eine Fehlermeldung ausgegeben werden. Dafür sehen wir den Tag <h:messages /> am Ende jeder Anwendungsfallseite vor. Dies führt dazu, dass eventuelle Fehlermeldungen am Ende der Seite angezeigt werden. Das eigentliche Instrument von JSF zur Entdeckung von Fehlern ist die Verwendung von Standard- oder Benutzerdefinierten Validatoren. Die soll aber nur für die Entdeckung von leeren Eingaben benutzt werden. Da wir die Fehlerbehandlung aus den früheren Aufgabenblättern schon vorliegen haben, müssen wir uns nur noch Gedanken machen, wie wir den JOptionPane als GUI Komponente ersetzen. Dafür gibt es folgenden Codeschnipsel:

```
FacesMessage msg = new FacesMessage(FacesMessage.SEVERITY_INFO,
                                    "Sachbearbeiter / Admin mit diesem
Benutzernamen existiert schon", null);
      FacesContext fc = FacesContext.getCurrentInstance();
      fc.addMessage(null, msg);
```

Die entsprechende Meldung wird dann an der Stelle von <h:messages /> angezeigt.

6. Auswahl

Bei den Anwendungsfällen, die mit Editieren oder Löschen zu tun haben, ist die Auswahl eines Sachbearbeiters erforderlich. Mein Vorschlag dazu ist: <h:selectOneMenu>. Das zugehörige value Attribut wird mit dem Attribut `ausgewaehlterSachbearbeiter` einer ManagedBean durch einen Value Expression verknüpft. Es zeigt, nachdem der Benutzer ein Item der Combo Box ausgesucht hat, auf die String Repräsentation dieses Items. Das <h:selectOneMenu> Tag kann in eine eigene xhtml Datei mit Namen SachbearbeiterAuswahl.xhtml geschrieben werden. Diese Datei wird in die für den jeweiligen Anwendungsfall zuständige Datei inkludiert. In der action Methode des OK Buttons des Anwendungsfalls kann man auf die ManagedBean die zur ComboBox gehört wie oben gezeigt zugreifen, um den ausgewählten Sachbearbeiterstring zu finden.

b) Durch die Portierung unseres Programms auf eine Server Client Architektur ist es jetzt möglich, dass mehrere Benutzer gleichzeitig auf das Programm zugreifen.

Wo kann es bei unserem Programm dadurch Schwierigkeiten geben?

Wie könnte man das Programm auf die Verwendung durch mehrere Benutzer einstellen?

c) Sicherheit: es ist möglich, durch Eingeben des Pfades z.B. zu AdministratorAS.xhtml in die Browserzeile den Login zu umgehen. Wie kann man das verhindern?

Anhang für Ubuntu User

Tomcat 7 ubuntu 13.04 and eclipse Kepler problem to run

To fix the error ‘Cannot create a server using the selected type’ ‘Could not load the Tomcat server configuration at /usr/share/tomcat7/conf. The configuration may be corrupt or incomplete /usr/share/tomcat7/conf/catalina.policy (No […]

To fix the error ‘Cannot create a server using the selected type’

‘Could not load the Tomcat server configuration at /usr/share/tomcat7/conf. The configuration may be corrupt or incomplete /usr/share/tomcat7/conf/catalina.policy (No such file or directory)’

run the following:

```
cd ~/workspace/.metadata/.plugins/org.eclipse.core.runtime/.settings/
rm org.eclipse.jst.server.tomcat.core.prefs
rm org.eclipse.wst.server.core.prefs
cd /usr/share/tomcat7
sudo service tomcat7 stop
sudo update-rc.d tomcat7 disable
sudo ln -s /var/lib/tomcat7/conf conf
sudo ln -s /etc/tomcat7/policy.d/03catalina.policy conf/catalina.policy
sudo ln -s /var/log/tomcat7 log
sudo chmod -R 777 /usr/share/tomcat7/conf
sudo ln -s /var/lib/tomcat7/common common
sudo ln -s /var/lib/tomcat7/server server
sudo ln -s /var/lib/tomcat7/shared shared
```

Restart eclipse delete all servers and once again add it entering /usr/share/tomcat7 as a tomcat install dir.