Dipl.-Inform. Thomas Ziemer
www.tfh-berlin.de/~ziemer
Stand: 5. Mai 2009

2 Kapitel

WIR BAUEN UNS EINE WEB-APPLIKATION (DRITTER TEIL)

2.10 Eine Benutzungsoberfläche mit JSF (Dritte Fingerübung)

Show

JavaServer Faces ist ein Framework, das Sie bei der Entwicklung der Benutzungsoberfläche einer Web-Applikationen unterstützt. Es bietet einige interessante Leistungsmerkmale:

- Einfache Webseitennavigation (*faces-config.xml*)
- Standard-Interaktionselemente (Eingabefeld, Schaltfläche, Verknüpfung etc.)
- Automatische Validierung von Benutzereingaben
- Fehlerbehandlung
- JavaBean-Management (*faces-config.xml*)
- Ereignisbehandlung
- Unterstützung mehrsprachiger Web-Anwendungen (*messages.properties*)

Wir werden unsere erste Fingerübung, in der wir ein einzelnes Servlet *index.jsp* erstellt haben, das „fest verdrahtet“ auf die Tabelle *Item* der Oracle-Datenbank zugegriffen hat, um eine „komfortable Benutzungsoberfläche“ erweitern. Auf einer ersten Webseite werden wir eine Tabelle auswählen können (aus den drei verfügbaren Tabellen *Order*, *Item* und *Inventory*), und auf der zweiten Webseite wird uns der Inhalt dieser Tabelle angezeigt.

Technische Voraussetzungen

JavaServer Faces besteht aus einigen JAR-Komponenten. Sie finden diese in dem Archiv *JavaServer Faces* auf der Manuskriptseite http://www.ziemers.de/tfh/se/manuskript2.html.

Packen Sie einfach alle Dateien, die in dem Zip-Archiv enthalten sind, in den Tomcat-Ordner *%CATALINA_HOME%\webapps\fingeruebung\WEB-INF\lib*.

2.11 Die generelle Struktur von JSF

Show

Das Servlet *JavaServer Faces* bearbeitet („rendert“) einzelne JSP-Dateien **.jsp*, in denen die Webseiteninhalte der Web-Applikation und zusätzliche Metainformationen abgelegt sind. Zwischen diesen Webseiten wird mittels Regeln navigiert, die in einer Datei namens *faces-config.xml* abgelegt werden.

In den Webseiten können ferner Namen-Werte-Paare verwendet werden, deren Werte einmalig in einer Eigenschaftendatei *messages.properties* definiert sind.

Show

Navigation zwischen den Webseiten

Die Regeln der Navigation zwischen den einzelnen Webseiten ist der Kern von JSF. Sie werden in der Datei *faces-config.xml* festgelegt. Legen Sie die Datei im Ordner *%CATALINA_HOME%\fingeruebung\WEB-INF* an:

```
01 <?xml version="1.0"?>
02 <!DOCTYPE faces-config PUBLIC
03  "-//Sun Microsystems, Inc.//DTD JavaServer Faces Config 1.1//EN"
04  "http://java.sun.com/dtd/web-facesconfig_1_1.dtd">
05 <faces-config>
06   <navigation-rule>
07     <from-view-id>/pages/choosetable.jsp</from-view-id>
08     <navigation-case>
09       <from-outcome>showtable</from-outcome>
10       <to-view-id>/pages/showtable.jsp</to-view-id>
11     </navigation-case>
12   </navigation-rule>
[…]
```

Diese Konfiguration bestimmt in einer Regel (Zeilen 06 bis 12), dass von der Webseite */pages/choosetable.jsp* (*from-view-id*, Zeile 07) zur Webseite */pages/showtable.jsp* (*to-view-id*, Zeile 10) navigiert wird, falls der Ausgang von *choosetable.jsp* den Namen *showtable* trägt (*from-outcome*, Zeile 09). Einen Ausgang definieren kann beispielsweise das *action*-Attribut eines Submit-Buttons. Dazu jedoch erst später mehr…

Show

Deklaration der JavaBeans

Wir verwenden eine weitere JavaBean namens *TableBean*, um den Oracle-Tabellenname, den der Benutzer in der ersten Webseite eingibt, über mehrere Webseiten hinweg zu speichern. Diese JavaBean fungiert als Schnittstelle (*bridge*) zwischen der Webseite und der eigentlichen Applikationslogik. Erstellen Sie dazu die Java-Datei *TableBean.java* im Ordner *%CATALINA_HOME%\webapps\fingeruebung\WEB-INF\classes\beans* (dort befindet sich bereits unsere JavaBean *DbBean* aus der ersten Fingerübung):

```
package beans;

public class TableBean {
  String tableName;

  public String getTableName() {
    return this.tableName;
  }

  public void setTableName(String tableName) {
    this.tableName = tableName;
  }
}
```

Vergessen Sie auch dieses Mal keinesfalls, den Namensraum der Klasse anzugeben!

Und bitte beachten Sie: der spätere Feldname in der JSP-Datei muss exakt mit dem Attributnamen in der Klasse übereinstimmen.

Show

Der zweite Teil der *faces-config.xml* beschreibt sämtliche JavaBeans, die in der Web-Applikation verwendet werden. Erweitern Sie die Datei um folgenden Inhalt:

```
[…]
01   <managed-bean>
02     <managed-bean-name>tableBean</managed-bean-name>
03     <managed-bean-class>beans.TableBean</managed-bean-class>
04     <managed-bean-scope>request</managed-bean-scope>
05   </managed-bean>
06   <managed-bean>
07     <managed-bean-name>dbBean</managed-bean-name>
08     <managed-bean-class>beans.DbBean</managed-bean-class>
09     <managed-bean-scope>request</managed-bean-scope>
10   </managed-bean>
11 </faces-config>
```

In den Zeilen 01 bis 05 wird die soeben erstellte Bridge-JavaBean *TableBean* deklariert. In den Zeilen 06 bis 10 wird die bereits in der ersten Fingerübung erstellte Datenbankzugriffs-JavaBean *DbBean* deklariert.

2.12 Die Eigenschaftsdatei (*messages.properties*) erstellen

Show

Die Eigenschaftsdatei *messages.properties* enthält Namen-Werte-Paare. Nur die Namen werden in den Webseiten verwendet; die korrespondierenden Werte hingegen nur einmalig in der Eigenschaftsdatei definiert.

Die Werte getrennt von den JSP-Dateien aufzubewahren, erlaubt es, sie beliebig oft verwenden und schnell modifizieren zu können – beispielsweise für mehrsprachige Webseiten - , ohne die JSP-Dateien jedes Mal bearbeiten zu müssen.

JavaServer Faces wertet die Sprache der angeforderten Webseite automatisch aus und verwendet die korrekte Eigenschaftsdatei.

Erstellen Sie den Ordner *%CATALINA_HOME%\webapps\fingeruebung\classes\bundle* und darin die Datei *messages.properties* mit folgendem Inhalt:

```
choosetable_header=Dritte Fingerübung JSF
prompt=Bitte Tabellenname eingeben:
showtable_txt=Inhalt der Tabelle
button_txt=Tabelle zeigen
sign=!
```

2.13 Die JSP-Dateien bearbeiten

JSP-Datei *choosetable.jsp*

Show

Erstellen Sie mit Ihrem Lieblingseditor als erstes die neue Webseite *choosetable.jsp* im Ordner *%CATALINA_HOME%\webapps\fingeruebung\pages* mit folgendem Inhalt:

```
01 <%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
02 <%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
03 <f:loadBundle basename="bundle.messages" var="msg"/>
04 <html>
05   <head>
06     <title>Tabellenname eingeben</title>
07   </head>
08   <body>
09     <f:view>
10       <h1>
11         <h:outputText value="#{msg.choosetable_header}"/>
12       </h1>
13       <h:form id="indexForm">
14         <h:outputText value="#{msg.prompt}"/>
15         <h:inputText value="#{tableBean.tableName}" required="true"/>
16         <h:commandButton action="showtable" value="#{msg.button_txt}"/>
17       </h:form>
18     </f:view>
19   </body>
20 </html>
```

Die Zeilen 01 und 02 sind Direktiven, die Tags für HTML-Elemente und Kern-Elemente von JSF definieren. Zeile 03 lädt die im vorangegangenen Abschnitt definierte Eigenschaftsdatei.

Zeile 11 verweist in das Resource Bundle *msg*, das in Zeile 03 geladen wurde (es handelt sich um unsere Eigenschaftsdatei *message.properties*). Daraus soll der Wert des Namens *index_header* ausgegeben werden.

In den Zeilen 13 bis 17 wird ein HTML-Formular definiert, in dem ein Prompt ausgegeben (Zeile 14) wird, sowie ein HTML-Eingabefeld (Zeile 15) und ein Submit-Button (Zeile 16) angezeigt werden.

Der Wert des Eingabefeldes verweist auf das Attribut *tableName* der JavaBean *TableBean*. Das HTML-Attribut *action* des Buttons bestimmt den Ausgang (*outcome*) der Webseite.

Automatische Validierung der Benutzereingaben

Das Eingabefeld in Zeile 15 ist ein Muss-Feld (*required="true"*). Die Webseite kann nicht verlassen werden, ohne dass ein Tabellennamen eingegeben wurde.

Show

Möchte man eine minimale oder maximale Eingabelänge eines Feldes sicherstellen, könnte folgende Ergänzung verwendet werden:

```
[…]
  <h:inputText value="#{tableBean.tableName}" required="true">
    <f:validateLength minimum="2" maximum="10"/>
  </h:inputText>
[…]
```

Scheitert die Validierung, werden von JSF automatisch Meldungen generiert und ausgegeben. An welcher Position in der Webseite die Meldung sichtbar ist, legt folgende Angabe fest:

```
[…]
  <h:messages style="color: darkred;"/>
[…]
```

Show

JSP-Datei *showtable.jsp*

Die Webseite *showtable.jsp* zeigt den Inhalt der ausgewählten Datenbanktabelle. Erstellen Sie die neue Datei *showtable.jsp* in dem Ordner, in dem Sie soeben bereits die Webseite *choosetable.jsp* erstellt haben:

```
01 <%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
02 <%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
03 <f:loadBundle basename="bundle.messages" var="msg"/>
04 <html>
05   <head>
06     <title>Tabelleninhalt</title>
07   </head>
08   <body>
09     <f:view>
10       <h3>
11         <h:outputText value="#{msg.showtable_txt}"/>
12         <h:outputText value="#{tableBean.tableName}"/>
13         <h:outputText value="#{msg.sign}"/>
14       </h3>

15       <%
16         dbBean.open("system", "<password>");
17         dbBean.query("SELECT name FROM " +
                                    tableBean.getTableName());
18         while(dbBean.next()) {
19           out.println(dbBean.getString(1) + "<br/>");
20         }
21         dbBean.close();
22       %>

23     </f:view>
24   </body>
25 </html>
```

Vorsicht: „name“ gibt es natürlich nicht in allen Tabellen!

Show

JSP-Datei *index.jsp*

Benennen Sie die Webseite *%CATALINA_HOME%\webapps\fingeruebung\index.jsp* (aus der ersten Fingerübung) um in *index1.jsp*. Wir werden nun eine neue Einstiegs-Webseite der Web-Applikation erstellen. Diese benutzt das JSP-Tag *forward*, um zur Webseite *choosetable.jsp* zu navigieren. Erstellen Sie die neue *index.jsp*. Beachten Sie, dass diese *nicht* im Ordner *pages* erstellt wird wie die Webseiten *choosetable.jsp* und *showtable.jsp*:

```
<html>
  <body>
    <jsp:forward page="/pages/choosetable.jsf"/>
  </body>
</html>
```

Beachten Sie, dass die Webseite, auf die verwiesen wird, mit der Namenserweiterung *.jsf* endet, und nicht mit *.jsp*, wie wir eigentlich erwarten würden. Diese Namenserweiterung wird verwendet, um Tomcat zu signalisieren, dass diese Webseite vom Servlet *JavaServer Faces* bearbeitet werden soll.

2.14 Tomcat konfigurieren

Show

Die Abbildung der Dateierweiterung *.jsf* auf die Bearbeitung durch das Servlet *JavaServer Faces* wird in der generellen Tomcat-Konfigurationsdatei *web.xml* konfiguriert. Erstellen Sie sie im Ordner *%CATALINA_HOME%\webapps\fingeruebung\WEB-INF*:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
      http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
    version="2.4">

  <context-param>
    <param-name>javax.faces.STATE_SAVING_METHOD</param-name>
    <param-value>server</param-value>
  </context-param>

  <context-param>
    <param-name>javax.faces.CONFIG_FILES</param-name>
    <param-value>/WEB-INF/faces-config.xml</param-value>
  </context-param>

  <listener>
    <listener-class>com.sun.faces.config.ConfigureListener</listener-class>
  </listener>

  <!-- Faces Servlet -->
  <servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <!-- Faces Servlet Mapping -->
  <servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>*.jsf</url-pattern>
  </servlet-mapping>
</web-app>
```

Die blau markierten Zeilen übernehmen die oben erwähnte JSF-Abbildung.

2.15 Überprüfung der Web-Applikation

Übersetzen Sie die TableBean. Wechseln Sie dazu in einer Kommando-Shell in den Ordner, der die Java-Datei enthält, und führen Sie folgendes Kommando aus:

```
javac TableBean.java
```

Nun können Sie die Anwendung im Webbrowser starten:

```
http://<hostname>:8081/fingeruebung/index.jsp
```

Ersetzen Sie <*hostname*> durch den Namen des Tomcat-Servers.

Sollte es zu Fehlern kommen, lohnt häufig ein Blick in die Log-Dateien des Tomcat-Servers, die sie im Ordner *%CATALINA_HOME%\logs* finden.

2.16 Erweiterung der Funktionalität

Die Funktionalität der Web-Applikation beschränkt sich augenblicklich darauf, einen Tabellennamen entgegenzunehmen, und diesen Namen auf der nachfolgenden Webseite wieder auszugeben. Bitte erweitern Sie die Anwendung dahingehend, von allen Datensätzen der vom Benutzer ausgewählten Tabelle ein geeignetes alphanumerisches Attribut anzuzeigen.

Bitte führen Sie diese gesamte dritte Fingerübung im Rahmen einer Übungsveranstaltung der Lehrveranstaltung *Software Engineering II* durch.