ZT IT-KOLUMNE

Einstieg in die Programmierung von Computern – Teil XIV

Im Teil 14 werden wir uns mit dem Thema „Java Server Faces (JSF)“ beschäftigen. JSF ist aktuell der Standard für dynamische Webseiten mit der Programmiersprache Java und ermöglicht die Entwicklung von komplexen und umfangreichen Webpräsenzen.

In den letzten Artikeln wurde das Thema „Java Servlets“ sehr ausführlich beschrieben. Wir haben gesehen, dass der Server HTML-Seiten an den Client (Internetbrowser) zurückliefert, so weit, so gut. Dem konzentrierten Leser wird aber aufgefallen sein, dass ein Java Servlet zur Erzeugung von HTML immer Java-Code Befehle einbinden muss. Gezwungenermaßen bedeutet das letztendlich der Aufruf einer „println-Funktion“, also

```
println("<html>Hier steht der Text
</html>");
```

mittels eines OutputStreams. Dadurch wird eine etwas umfangreichere Webseite sehr schwer lesbar und es wird nicht zwischen Daten (Model), Optik (View) und der Steuerung (Controller) unterschieden, da alles in einem Code vermischt ist. Man kann sich vorstellen, dass damit große Java-Webprojekte nur sehr schwer zu realisieren sind. Aus diesem Grund wurde die neue Java-Technologie „JavaServer Pages (JSP)“ entwickelt. Bei JSP ist HTML im Fokus bzw. HTML ist die treibende Kraft. Zwischen den einzelnen HTML-Tags werden für die dynamischen Anteile bei JSP sogenannte „Scriptlets“ eingebunden, d. h. zwischen den HTML-Tags steht Java-Code. Prinzipiell war das schon einmal eine deutliche Verbesserung im Vergleich zu den Java Servlets. Aber auch bei der Verwendung von JSP kann man sich vorstellen, dass nun mittels Scriptlets in einer HTML-Seite viel Java-Code eingebunden wird und so der Quellcode, also HTML, Javascript und Java ebenfalls sehr unleserlich wird. Im Kasten 1 sehen Sie ein einfaches JSP Code-Beispiel.

Sogenannte „Webframeworks“ hielten die Softwareentwickler dazu an, Layoutanteile in der Seitendeklarationssprache JSP zu erstellen und die Java-Anwendungssoftware in die Scriptlets einzubetten. Eine neue Technologie musste also her, um all die beschriebenen Nachteile zu beseitigen. Außerdem musste unbedingt eine Trennung zwischen „Optik, Daten und Steuerung (Model, View und Controller, abgekürzt **‚MVC‘**)“ geschaffen werden. Ein weiteres wichtiges Feature für eine neue Java-Technologie war dabei die Gestaltung von Webseiten mittels sogenannter „Templates“, also Vorlagen, die eine Wiederverwendung bestimmter Layoutteile einer Seite ermöglichen. Im Jahre 2004 konnte dann die neue **„Javaserver Faces (JSF)“-Technologie** der Öffentlichkeit präsentiert werden. JSF beherrscht seitdem als neue Java-Seitendeklarationssprache die Entwicklung von Webseiten in Java und stellt dem Entwickler komplexe und leistungsfähige Funktionen bereit. Im Folgenden eine kurze Auflistung der wichtigsten JSF-Features:

- **Facelets** als neue Seitendeklarationssprache in der View-Technologie (löst JSP ab).
- **MVC:** Vollständige Trennung der View von der Anwendungssoftware.
- **Ajax:** Ermöglicht das asynchrone Laden von Server-Daten. Es werden nur die Teile der Webseite neu geladen, in denen auch eine Datenänderung stattgefunden hat. Dadurch wird ein erneutes Laden der gesamten Webseite verhindert, was zu einer deutlichen Geschwindigkeitssteigerung führt.
- **Standardisiertes Vorgehen bei der Einbindung von Ressourcen:** wie z.B. Skripte und Cascading Stylesheets.
- **Systemevents:** Es kann endlich auf Ereignisse im Lebenszyklus reagiert werden.
- **Implizite und bedingte Navigation:** Vereinfacht die Navigation zwischen den einzelnen Ansichten.
- **Validieren von Benutzereingaben:** Die vom Benutzer eingegebenen Daten können mit sogenannten Validatoren überprüft werden.
- **Einsatz von Managed Beans:** Von der View aus können sogenannte „Managed Beans“ angesprochen werden, die als einfache Java-Klassen deklariert sind. Diese müssen dem Java-Beans Standard genügen.
- Und vieles mehr …

```
<%@ page language="java" %>
<html>
   <head>
      <title> <%=request.getParameter("ueberschrift");%> </title>
   </head>
   <body>
      <%=request.getParameter("bodytext");%>
   </bod<>
</html>
```

Kasten 1: Einfaches JSP-Beispiel mit eingebundenen Scriptlets.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
   PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
        xmlns:f="http://java.sun.com/jsf/core"
        xmlns:h="http://java.sun.com/jsf/html">
<head>
   <title>MyGourmet - Show Customer</title>
</head>
<body>
   <h1><h:outputText value="MyGourmet"/></h1>
   <h2><h:outputText value="Show Customer"/></h2>
   <h:form id="form">
      <h:panelGrid id="grid" columns="2">
        <h:outputLabel value="Vorname:" for="firstName"/>
        <h:inputText id="firstName" value="#{customer.firstName}"/>
        <h:outputLabel value="Nachname:" for="lastName"/>
        <h:inputText id="lastname" value="#{customer.lastName}"/>
         <h:commandButton id="save" action="#{customer.save}" value="Speichern"/>
      </h:panelGrid>
   </h:form>
</body>
</html>
```

Kasten 2: Einfaches JSF-Beispiel „bearbeitenKunde.xhtml“.

```
/**
 * Datei: Customer.java
 *
 * Dient als Managed Bean und kommuniziert mit
 * der View „bearbeitenKunde.xhtml“.
 */
package oemus.com.jsfbeipiel.gui.page;

import java.io.Serializable;
import javax.enterprise.context.SessionScoped;
import javax.inject.Named;

@Named
@SessionScoped
public class CustomerController implements Serializable {

  private String firstName;
  private String lastName;

  public String getFirstName() {

    return firstName;
  }

  public void setFirstName(String firstName) {

    this.firstName = firstName;
  }

  public String getLastName() {

    return lastName;
  }

public void setLastName(String lastName) {

    this.lastName = lastName;
  }
}
```

Kasten 3: Die Backing-Bean Klasse „CustomerController.java“.

Ein erstes JSF-Beispiel

Am Anfang wollen wir uns ein einfaches JSF-Beispiel anschauen, mit dem wir das Grundkonzept von JSF kennenlernen werden. Hier geht es um die Grundlagen. Im Kasten 2 sehen Sie das erste einfache JSF-Beispiel „bearbeitenKunde.xhtml“. Die JSF-Dateien haben die Dateiendung **.xhtml** und stellen die View im MVC-Design Muster dar. Im Kasten 3 sehen Sie die zur View gehörige Controller Java-Klasse „CustomerController.java“ als „Backing Bean“.

Erklärung zu der Facelet-Datei „bearbeitenKunde.xhtml“:
Diese View-Datei besteht aus einem Head-Bereich, in dem bestimmte JSF-Informationen abgelegt werden müssen. Die bekannten HTML-Tags sind bei einer JSF-Datei durch eigene JSF-Tags ersetzt worden (z.B. <form …> <h:form …>), die notwendig sind, um das Erzeugen (rendern) der HTML-Seite von der Java-Laufzeitumgebung zu ermöglichen. Mit den speziellen Anweisungen, der sogenannten „Expression-Language (EL)“ kann auf Backing Bean-Methoden zugegriffen werden:

```
<h:inputText id="lastname" value
="#{customer.firstName}"/>
```

Erklärung dazu siehe nächsten Abschnitt!

Erklärung zu der Klasse „CustomerController.java“
Backing Bean bedeutet, dass der Applikationsserver diese Klasse verwaltet und einem Facelet zur Verfügung stellt. Diese Klasse arbeitet im Hintergrund und soll die eigentliche Arbeit verrichten, deshalb ist diese Klasse eine sogenannte Controller-Klasse, die immer zu einer bestimmten View-Klasse gehört. Die Backing Bean-Klasse bzw. Controller-Klasse kann weitere Klassen wie z. B. Model-Klassen zur Datenspeicherung verwenden. Wie man leicht erkennen kann, befinden sich in der Controller-Klasse Methoden, die direkt vom Facelet, also der View-Datei „bearbeiten-Kunde.xhtml“ aufgerufen werden können. Dazu gleich mehr.

Was bedeuten die beiden Code-Zeilen „@Named“ und „@SessionScoped“?
Diese beiden Code-Zeilen sind in Java sogenannte **„Annotationen“** und müssen immer vor die Klassen-Deklaration geschrieben werden.
Mit der **@Named**-Annotation wird eine Bean gekennzeichnet, die in einem Facelet über ihren Klassennamen (aber kleingeschrieben) referenziert werden kann:
In der Datei bearbeitenKunde.xhtml —›

```
#{customer.firstName}
```

Damit wird im Facelet direkt die Methode „setFirstName“ in der Klasse „Customer“ aufgerufen.
Mit der **@SessionScoped**-Annotation wird der Sichtbarkeitsbereich und somit auch die Lebensdauer der Backing Bean festgelegt. Das bedeutet, dass alle gespeicherten Daten in der Backing Bean unabhängig von der aktuellen View für die gesamte Sitzung (Session) gespeichert werden.
Jetzt haben wir einen ersten kleinen Eindruck über Javaserver Faces (JSF) gewonnen, mit dem hoffentlich der Einstieg in Javaserver Faces (JSF) gelungen ist. ZT

Infos zum Autor

ZT Adresse

Thomas Burgard Dipl.-Ing. (FH)
Softwareentwicklung
& Webdesign
Bavariastraße 18b
80336 München
Tel.: 089 540707-10
info@burgardsoft.de
www.burgardsoft.de
burgardsoft.blogspot.com
twitter.com/burgardsoft

NEU

Programat® P510

Der intelligente Brennofen

Effizient
zu ausgezeichneten
Brennresultaten.
Mit Wärmebildkamera.

INFRARED TECHNOLOGY

Mit Infrarot-Technologie zu ausgezeichneten Ergebnissen

- **Programat-Infrarot-Technologie** für bis zu 20 % schnellere Vortrocknungsprozesse
- **Einfache Bedienung** dank ausgeklügelter Kombination aus farbigem Touchscreen und bewährter Folientastatur
- **Homogene Wärmeverteilung und ausgezeichnete Brennresultate** dank QTK2-Muffeltechnologie mit SiC-Bodenreflektor

www.ivoclarvivadent.de/programat-p510

www.ivoclarvivadent.de
Ivoclar Vivadent GmbH
Dr. Adolf-Schneider-Str. 2 | D-73479 Ellwangen, Jagst | Tel. +49 7961 889 0 | Fax +49 7961 6326

ivoclar vivadent®
passion vision innovation