Fachhochschule Köln
University of Applied Sciences Cologne
Abteilung Gummersbach
Fachbereich Informatik

Diplomarbeit
(Drei-Monats-Arbeit)
zur Erlangung
des Diplomgrades
Diplom-Informatiker (FH)
in der Fachrichtung Informatik

„Web-Frameworks bei der Entwicklung von Web-Applikationen anhand von Apache Struts und JavaServer Faces“

vorgelegt am: 31.08.2005 von:

Name: Karl Schmid
Anschrift: Münchenerstr.16
51103 Köln
Telefon: 0221-5801607
E-Mail: schmid.karl@gmx.net
Matr.-Nr.: 11022723

Erstprüfer: Fr. Prof. Dr. Rer. Nat. Heide Faeskorn-Woyke
Zweitprüfer: Fr. Prof. Dr. Rer. Nat. Birgit Bertelsmeier

Inhaltsverzeichnis

Abbildungsverzeichnis

Stichwortverzeichnis

A		
	ActiveServer Pages	ASP
	American Standard Code for Information Exchange	ASCII
	Application Program Interface	API
B		
	Benutzersitzung des Clients	SESSION
C		
	Cascading Style Sheets	CSS
	Common Gateway Interface	CGI
	Create Remove Update Delete	CRUD
D		
	Document Type Definition	DTD
E		
	Enterprise JavaBean	EJB
	Enterprise Resource Planning	ERP
	Entity Realationship Diagramm	ERD
	Extensible Markup Language	XML
	EXtensible Stylesheet Language	XSL
	EXtensible User Interface Language	XUL
G		
	Graphical User Interface	GUI
H		
	Hypertext Markup Languagel	HTML
	Hypertext Transfer Transport Protokoll	HTTP
I		
	Integrated Development Environment	IDE
J		
	Java 2 Enterprise Edition	J2EE
	Java Database Connectivity	JDBC
	Java Message Service	JMS
	Java Naming and Directory Interface	JNDI
	Java Standard Tag Librar	JSTL
	Java Transaction API	JTA

1. Einleitung

Mit dem Internet hat sich eine Technologie etabliert, die Veränderungen in den unterschiedlichsten Bereichen nach sich gezogen hat. Sie ist im Rahmen der Globalisierung zu sehen und ist einer der Hauptakteure bei der Entstehung der Informationsgesellschaft. International und national agierende Unternehmen besitzen einen entscheiden Marktnachteil, wenn diese Technologie nicht gewinnbringend zur Unterstützung ihrer Geschäftsprozesse und deren globaler Vernetzung eingesetzt wird.

Einhergehend mit der Entwicklung des Internets sind zahllose Technologien entstanden. Wo es anfänglich nur möglich war, Informationen als feststehende Inhalte zu präsentieren, können und müssen heutige Internetpräsenzen Informationen anbieten, die Zugriff auf verteilt liegende Informationsquellen besitzen, die sich täglich ändern.

So sind Web-Applikationen entstanden, die die modernsten Technologien einsetzen, um Anwendern eine Zugangsmöglichkeit auf Informationsquellen über das Internet zu bieten. In Kapitel 2 werden die Anforderungen an Web-Applikationen und deren zugrunde liegende Architektur beleuchtet.

Da zahllose Programmiersprachen für die Entwicklung von Web-Applikationen verfügbar sind, wird in Kapitel 3 eine Auswahl an verfügbaren Technologien vorgestellt.

Um die Programmierung von Web-Applikationen zu vereinfachen sind im Laufe der letzten Jahre Web-Frameworks entstanden, die den Entwicklungsprozess stark beschleunigen und eine Art Bausatz anbieten, um die Programmierung zu vereinfachen. Der Schwerpunkt der vorliegenden Arbeit richtet sein Augenmerk auf 2 aktuelle Web-Frameworks, die jeweils theoretisch und praxisbezogen anhand einer Beispielapplikation beleuchtet werden.

In Kapitel 4 wird JavaServer Faces als erstes Web-Framework vorgestellt. Ausgehend von einer theoretischen Betrachtung wird ein Web-Shop als Beispielapplikation entwickelt und die damit in Zusammenhang stehenden Entwicklungswerkzeuge betrachtet.

Auf Struts, als zweites Web-Framework, wird in Kapitel 5 eingegangen und ähnlich wie in Kapitel 4 zu JavaServer Faces wird nach theoretischer Betrachtung des Web-

Frameworks eine Benutzerverwaltung als Beispielapplikation entwickelt, die die wesentlichsten Aspekte des Web-Frameworks zeigt. Auch hier werden Enticklungswerkzeuge vorgestellt, die den Umgang mit Struts unterstützen.

Schließlich findet in Kapitel 6 ein Vergleich zwischen JavaServer Faces und Struts statt, der mögliche Vor- und Nachteile des jeweiligen Web-Frameworks aufzeigt.

2. Architektur dynamischer Web-Applikationen

Die Architektur von Web-Applikationen hat sich innerhalb der letzten Jahre stark geändert. Verantwortlich für diesen Prozess ist der Siegeszug des Internets gewesen und mit ihm einhergehend neue Architekturmodelle, neue Programmiersprachen, neue Erfordernisse an eine Web-Anwendung.

In diesem Kapitel werden grundlegende Architekturmodelle beschrieben, die den aktuellen Standard widerspiegeln und aufzeigen, auf welchem Grundgerüst eine Web-Anwendung erstellt werden kann. Begonnen wird mit einem geschichtlichen Abriss, der zeigt wie sich Web-Anwendungen im Laufe der letzten Jahre geändert haben. Daran anschließend wird der grundsätzliche Aufbau mithilfe eines Schichtenmodells näher erläutert, um daran anknüpfend den Programmfluss einer Web-Anwendung zu beschreiben.

Im letzten Abschnitt wird das Model-View-Controller-Prinzip als wichtigstes Architekturprinzip für die Präsentationsschicht einer Web-Anwendung betrachtet und erläutert. Dieses bildet die Grundlage, auf der auch JavaServer Faces und Struts entstanden sind und in Kapitel 4 und Kapitel 5 ausführlich beschrieben werden.

2.1 Historischer Abriss

Web-Anwendungen der Vergangenheit setzten sich zum größten Teil aus statischen HTML[1]-Seiten zusammen, deren Inhalte fest in die Seite verwoben wurde. Verfügbar gemacht wurden solche Web-Seiten über Web-Server, die die Seiten an einer zentralen Stelle speichern konnten und diese auf Abruf über einen Web-Browser bereitstellten. Im Laufe der Zeit wuchsen die Anforderungen an eine Web-Seite. Da HTML als reine Darstellungssprache für Web-Inhalte gedacht ist, musste die Sprache erweitert werden. Sowohl für Client-Seite als auch für die Server-Seite entstanden die unterschiedlichsten Sprachen, die es ermöglicht haben, von einer Web-Seite mit statischen Inhalten eine nunmehr dynamische Web-Seite zu erstellen, die in der Lage ist auf Benutzeraktionen zu reagieren und ihre Inhalte aus den unterschiedlichsten Quellen zur Laufzeit zu

1 HTML: Hypertext Markup Language

beziehen. Dies schließt eine Datenbankanbindung ebenso ein wie die Anbindung an Applikationsserver, die auch erst in letzter Zeit entstanden sind und als Server für verteilte Applikationen dienen. Das führt zum Begriff der dynamischen Web-Applikation, welche in den folgenden Abschnitten näher beleuchtet werden soll.

2.2 Schichtenmodelle

Ansätze für eine Modellierung zur Beschreibung von Software gibt es viele. Das Modell, welches sich am stärksten durchgesetzt hat, zerlegt die Anwendung in Schichten, die für jeweils bestimmte Aufgaben zuständig ist. Ferner wird zusätzlich für jede Schicht empfohlen, den Kommunkationsfluss nur von einer Schicht zur nächst niedrigeren Schicht zuzulassen.

Als Folge davon erhält man Software, die klar strukturiert ist und dadurch verschiedene Vorteile mit sich bringt. Dadurch, dass Software sehr komplexe Formen annehmen kann, wird durch eine geschichtete Architektur erreicht, dass sich der Komplexitätsgrad einer Anwendung verringert, da die Aufgaben auf jeweils eine Schicht verteilt werden.

Das verbessert die Testbarkeit, Wartbarkeit, logische Struktur und führt auch dazu, dass sich bessere Integrationsmöglichkeiten zu einer anderen Software eröffnen. Man kann so auch viel leichter und schneller bestimmte Schichten einer Anwendung austauschen und wiederverwenden.

Die Nachteile, die durch eine erhöhte Strukturierung mit einhergehender anfänglicher Mehrarbeit verbunden ist, werden in der späteren Entwicklungs- bzw. Wartungsphase ausgeglichen und durch die Vorteile eines Schichtenmodellansatzes übertroffen.

Das Schichtenmodell gilt auch für Moderne Web-Applikationen, bei denen häufig auch eine Verteilung der einzelnen Schichten auf unterschiedliche Server erfolgt. Beide Modelle werden nachfolgend anhand von Abbildung 1 und 2 illustriert.

Verteilung innerhalb der Anwendung:

Präsentationsschicht:	Die Darstellung von Inhalten und gegebenenfalls bereitzustellende Interaktionsmöglichkeiten für den Benutzer.
Geschäftsschicht:	Sie beinhaltet die eigentliche Logik der Anwendung und stellt den Kern der Anwendung dar.
Persistenzschicht:	Die Anbindung einer Datenbank, mit deren Hilfe die Geschäftsobjekte der Geschäftsschicht permanent gespeichert werden können.
Datenbankschicht:	Abbildung der Geschäftsobjekte und ihrer Beziehungen untereinander auf Tabellen.

Kommunikationsfluss

Abbildung 1: *Schichten-Architektur (Verteilung innerhalb der Anwendung)* [2]

Verteilung auf unterschiedliche Server:

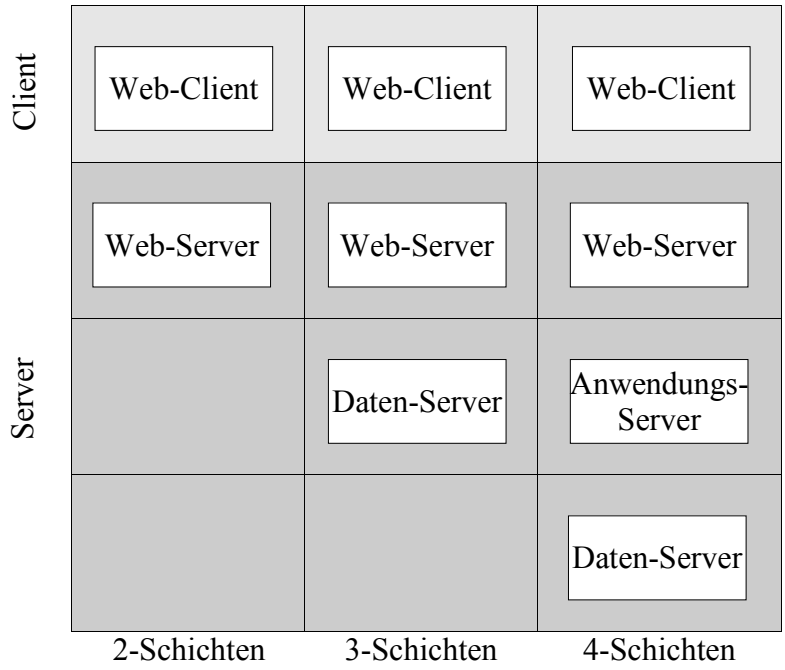


Abbildung 2: *Schichten-Architektur (Verteilung auf verschiedene Server)* [3]

2 Dieses Modell wurde keinem Buch entnommen, findet sich aber in dieser oder ähnlicher Form in jedem Buch zur Softwaretechnik oder Softwarearchitektur. Beispielsweise in [FOW01] Seite 20

3 In Anlehnung an [BAL01] Seite 706

Wie aus der Grafik ersichtlich wird, gibt es eine unterschiedliche Anzahl an Schichten[4] (auch als n-Tier bezeichnet), die je nach Komplexitätsgrad und Anforderungen an eine Web-Applikation variieren.

Die einfachste Lösung besteht in einer 2-Schichten-Lösung, die aus einem Client und einem Web-Server besteht. Dieser Ansatz ist in vielen Fällen ausreichend, da die Funktionalität eines Web-Servers viele Möglichkeiten beinhaltet, die es auch ermöglichen dynamische Web-Seiten zu generieren.

Das 3-Schichten-Modell sieht die Anbindung von Datenbankservern vor und erweitert das anfängliche Modell um eine weitere Schicht. Dieser Ansatz ist der am meisten verbreitetste und es lassen sich damit so genannte CRUD[5]-Anwendungen erstellen.

Kommt dazu noch ein Applikationsserver wird von einer 4-schichtigen Architektur gesprochen, die es noch stärker ermöglicht, die Anwendung auf unterschiedliche Server zu verteilen. Der Anwendungsserver, meist ein J2EE[6] konformer Applikationsserver, bietet Lösungen für die komplexesten Probleme einer Web-Applikation, indem durch den Server viele unterschiedlich nutzbare Dienste bereitgestellt werden, welche in Kapitel 3.7 näher beschrieben werden.

2.3 Web-Architektur im Detail

Die nachfolgenden Ausführungen grenzen eine moderne dynamische Web-Applikationen ein anhand eines fiktiven Durchlaufes durch eine Web-Applikation.

Dadurch offenbart sich der Aufgabenkatalog einer Web-Applikation, der mit Hilfe der Abbildung 3 beschrieben wird. Durch diesen Programmfluss ergeben sich auch die notwendigen Anforderungen an eine Web-Applikation.

Angefangen bei einem Request[7] durch den Browser wird gezeigt, welche weiteren Schritte vonnöten sind, bevor dem Browser das Ergebnis zurückgesendet werden kann.

4 Als andere Bezeichnung für die einzelnen Schichten wird auch gerne der Begriff n-Tier benutzt, wobei n für die Anzahl der Schichten steht.
5 CRUD: Create Remove Update Delete (Standardoperationen einer Datenbank)
6 J2EE: Java 2 Enterprise Edition
7 Request: Anfrage eines Browsers über das HTTP-Protokoll, das für den Datenverkehr zwischen Web-Browser und Web-Server zuständig ist

Die einzelnen Schritte sollen einzeln beschrieben werden und stellen somit auch einen Grundkatalog an Erfordernissen dar, die bestenfalls durch ein passendes Web-Framework umgesetzt bzw. unterstützt werden sollte. In Kapitel 4 und Kapitel 5 soll anhand von 2 Frameworks gezeigt werden, inwieweit es geschafft wurde, diese hier in Grundzügen aufgezeigten Erfordernisse, umzusetzen. Nachfolgende Ausführungen sind in Anlehnung an [SHR01][8] entstanden.

Abbildung 3: *Verarbeitungsschritte einer Web-Applikation*[9]

Verarbeitung und Routing eingehender Requests:

Nachdem ein Client eine bestimmte URL[10] aufgerufen hat, muss der angesprochene Web-Server die eingehenden Informationen verarbeiten. Diese Informationen werden über das HTTP[11]-Protokoll übertragen. Die meisten Web-Server bieten die Möglichkeit, anhand der File-Endung festzulegen, welcher Prozess für die Verarbeitung zuständig ist. Auf diese Weise lassen sich z.B. Perl-Skripte, ASP-Seiten, PHP-Skripte, JavaServlets

8 [SHR01] Seite 203 ff
9 in Anlehnung an [SHR01] Seite 204
10 URL: Uniform Resource Locator
11 HTTP: Hypertext Transfer Transport Protokoll (Transport-Protokoll des Internets)

oder JSP-Seiten ausführen. Ein weiterer wichtiger Punkt ist das Speichern des Zustandes der Web-Applikation, wodurch es möglich wird, über mehrere Requests hinweg, Informationen zu speichern. Dies kann entweder auf Client-Seite durch Cookies[12] erfolgen, oder auf Server-Seite mithilfe einer Sessionverwaltung[13], welches durch den Web-Server selbst oder durch ein Web-Framework bereitgestellt wird.

Security:

Dadurch dass viele Web-Applikationen über das Internet zugänglich sind und diese weltweit benutzt werden können, besteht häufig die Notwendigkeit den Zugang zu beschränken. Eine Bankanwendung, die Zugriff auf Kontoinformationen bietet, muss z.B. sicherstellen, dass nur die User Zugang auf ihr Konto haben, die auch tatsächlich ein Konto bei der Bank haben. Der Zugang zur Web-Applikation wird durch eine Authentifizierung und Authorisierung bewerkstelligt.

Authentifizierung gibt Usern die Möglichkeit, sich beispielsweise durch Name und Passwort zu identifizieren und dadurch Zugriff auf die Web-Applikation zu bekommen.

Authorisierung erteilt nach geglückter Authentifizierung Rechte an den User, die ein fein abgestimmtes Zugriffsmodell auf die Web-Applikation bietet. Dadurch können bestimmte Rollen festgelegt werden, z.B. Admin für unbeschränkten Zugriff, Angestellter für Zugriff auf bestimmte Funktionen. Anschließend wird jedem berechtigten User eine Rolle zugeordnet, die es ihm erlaubt, seiner Rolle entsprechend Funktionen der Web-Applikation auszuführen.

Zugriff auf Datenspeicher:

Die Inhalte einer Web-Applikation können durch die verschiedensten Quellen gespeist werden. Je nach vorhandener Infrastruktur sind z.B. flache Dateien, Datenbankserver, Fileserver, Applikationsserver, Web-Services oder auch Legacy-Systeme denkbar als Resourcenquellen. Nicht selten sollen die schon vorhandenen Resourcen über die Web-Applikation einem breiteren Kreis an Usern zur Verfügung gestellt werden.

12 Cookies: Web-Browser bieten die Möglichkeit über Cookies Informationen auf der Festplatte des Clients abzuspeichern.
13 Session: eine Benutzersitzung des Clients

Je nach Datenspeicher, muss mit sehr verschiedenen Technologien gearbeitet werden. Nachfolgend werden die wichtigsten kurz beschrieben werden.

- *ASCII*[14]*:* Texte, die als flache Dateien gespeichert werden
- *statische HTML-Seiten:* normale Web-Seiten programmiert mit der Programmiersprache HTML
- *Datenbankserver:* Dient der Datenhaltung von großen Mengen an Daten, die mit Hilfe der Abfragesprache SQL[15] angesprochen werden.
- *Applikationsserver:* Hierbei handelt es sich um eine besondere Form von Anwendung, die über spezielle Schnittstellen angesprochen werden kann und in verteilten Anwendungen zum Einsatz kommt. Spezielles Detailwissen ist nötig, um Daten, die dort zur Verfügung gestellt werden zu beschaffen.
- *Web-Services:* Diese Technologie ist erst seit einigen Jahren auf dem Markt und bietet Anwendungen die Möglichkeit auf Grundlage von XML[16] Daten auszutauschen.
- *Enterprise Resource Planning (ERP) – Systeme:* Anwendungen, die sämtliche Unternehmensprozesse unterstützen und über spezielle Schnittstellen ansprechbar sind.

Weiterverarbeitung der Daten:

Sowohl eingehende als auch ausgehende Daten unterliegen der Verarbeitung durch die Anwendung. In ein Formular eingetragene Benutzerdaten müssen überprüft werden, bevor sie z.B. in eine Datenbank geschrieben werden. Abgerufene Daten aus einer Datenbank werden ausgewertet und müssen zwischengespeichert werden. Daraus entsteht eine Notwendigkeit vorhandene Objekte bzw. Zustände zu aktualisieren, indem diese gegebenenfalls gelöscht, neu angelegt oder ein Update durchgeführt wird.

14 ASCII: American Standard Code for Information Exchange
15 SQL: Structured Query Language
16 XML: Extensible Markup Language

Vorbereitung für die Darstellung:

Nachdem alle benötigten Daten beschafft und die jeweilige Anwendung aktualisiert worden ist, muss jetzt das Ergebnis für die Darstellung im Browser vorbereitet werden. Die richtige Wahl des Ausgabeformats wird durch die verwendeten Technologien vorgegeben. Im nächsten Kapitel werden die wichtigsten Technologien beschrieben, die für die Entwicklung der Präsentationsschicht zur Verfügung stehen.

Weiterleitung an den Client-Browser:

Im letzten Schritt wird das Ergebnis meist in Form von HTML an den User zurückgeschickt und dieser ist nun in der Lage das Ergebnis im Browser zu sehen.

2.4 MVC Architektur

Dieses äußerst beliebte Architekturmuster ist sehr häufig in der Präsentationsschicht einer Anwendung anzutreffen. Beispielsweise ist Java Swing darauf aufgebaut. Daneben kann es als das wichtigste Muster vieler Web-Frameworks betrachtet werden. Struts, JavaServer Faces, oder auch WebWork[17] sind um dieses Muster herum aufgebaut.

Das Modell-View-Controller(MVC)-Muster hat seinen Ursprung in der Sprache Smalltalk und sorgte für die Trennung von Präsentation und Logik. Dieses altbewährte Muster wurde von vielen Sprachen in der einen oder anderen Form adaptiert.

Der Controller wertet Benutzereingaben aus und greift auf das Model zu, welches dafür verantwortlich ist, beispielsweise Berechnungen durchzuführen oder auf Datenspeicher zuzugreifen. Nach der Verarbeitung wird das Ergebnis repräsentiert durch die View an den User weitergeleitet.

17 http://www.opensymphony.com/webwork

Abbildung 4 veranschaulicht das MVC-Prinzip.

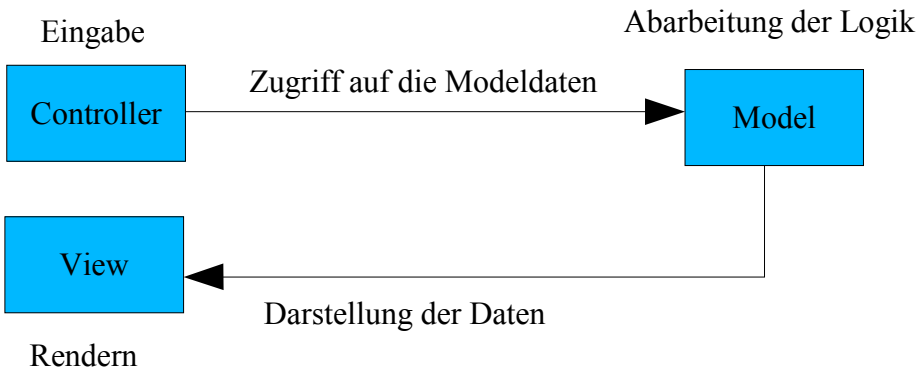


Abbildung 4: *MVC-Architektur* [18]

18 in Ahnlehnung an [NAS01] S.316

3. Programmierung von Web-Applikationen

Da in Kapitel 2 auf die Architektur von Web-Applikationen eingegangen wurde und der sich immer ähnelnde Grundaufbau beleuchtet wurde, soll in diesem Kapitel gezeigt werden, welche Varianten es für eine Umsetzung mithilfe von Programmiersprachen gibt. Angefangen bei den Basissprachen HTML und XML, die in praktisch jeder Web-Applikation zum Einsatz kommen, werden daran anschließend unterschiedliche Sprachen und Modelle vorgestellt, die sich von Abschnitt zu Abschnitt auf jeweils einem höheren Abstraktionsniveau befinden und mit der Vorstellung von Web-Frameworks ihr Ende findet. In Kapitel 3.7 wird zum Abschluss gezeigt, auf welche Weise man die erstellte Web-Applikation übers Internet verfügbar machen kann.

3.1 Sprachen für die Basis

HTML und XML bilden die Grundlage einer jeden Web-Applikation. Diese 2 Sprachen sind die wichtigsten Sprachen, ohne die die Entwicklung von Web-Applikationen undenkbar wäre. Deswegen werden sie als Sprachen für die Basis bezeichnet und in den nächsten 2 Unterkapitel vorgestellt.

3.1.1 HTML

Hypertext Markup Language[19] (HTML) ist die Basissprache des World Wide Web, mit der der Siegeszug des Internets begründet wurde. 1990 entwickelte der Forscher Tim Berners-Lee vom Hochenergieforschungszentrum CERN in Genf eine Möglichkeit, neben reinen Texten auch Formeln, Zeichnungen und Grafiken durch einfache Positionsangaben in Texten zu übermitteln. Das war die Geburtsstunde von HTML, das auf SGML[20] basiert.

19 http://www.w3.org/MarkUp
20 SGML: Standard Generalized Markup Language ist eine Metasprache, mit deren Hilfe man verschiedene Auszeichnungssprachen für Dokumente definieren kann.

HTML ist eine Sprache, die die einzelnen Bestandteile eines Dokuments beschreibt. Sie können also HTML dazu benutzen, Überschriften zu definieren, Tabellen festzulegen oder Absätzen eine bestimmte Form oder Farbe zu geben.

Eine der wichtigsten Fähigkeiten von HTML, besteht darin, Dokumente mithilfe von Hyperlinks zu verknüpfen. Dadurch können Dokumente, die sich auf unterschiedlichen Servern in unterschiedlichen Ländern befinden, miteinander verlinkt werden. Dies macht zu keinem kleinen Teil die Faszination des Internets aus und sorgt gleichzeitig für eine lose Strukturierung der Web-Inhalte.

Für das Anzeigen einer HTML-Seite kann ein Web-Browser verwendet werden, welcher als Standardsoftware den meisten Betriebssystemen beiliegt.

3.1.2 XML

Exentensible Markup Language[21] (XML) hat sich innerhalb der letzten Jahre als Standard für verschiedene Anwendungen etabliert. Mit XML wurde eine Auszeichnungssprache geschaffen, mit deren Hilfe sich sehr leicht Baumstrukturen darstellen lassen, daher eignet es sich hervorragend als Austauschformat für unterschiedliche Anwendungen, die damit ein einheitliches Format für den Austausch von Informationen an die Hand bekommen.

Neben der Form als Austauschformat hat sich XML auch für Konfigurationsdateien durchgesetzt, so sind beispielsweise Konfigurationsdateien für Web-Server, für Frameworks und ähnliches als XML-Datei kodiert.

Rund um XML sind eine ganze Reihe anderer Sprachen entstanden :

- *XSL/XSLT:* Vergleichbar zu Cascading Style Sheets(CSS), die der Formatierung von HTML-Elementen dienen, wurde speziell für XML die eXtensible Stylesheet Language(XSL) entworfen, um XML-Dateien zu formatieren und für die richtige Darstellung sorgen zu können. XSL Transformations (XSLT) wird verwendet, um

21 http://www.w3.org/XML

XML-Dateien in andere Formate zu konvertieren. Möglich wäre z.B. die Ausgabe als PDF- oder HTML-Dokument.

- *XPATH:* Xpath wird benötigt, um innerhalb einer XML-Datei die richtigen Datensätze zu finden und auszufiltern.
- *WML:* Wireless Markup Language (WML) ist die Sprache, die verwendet wird, um über das Wireless Application Protocoll (WAP) Mobiltelefone über das Internet anzusprechen.
- *Xschema:* Damit lässt sich die Struktur, Semantik und Inhalt von XML-Dokumenten beschreiben. Sie kann als Nachfolger der Document Type Definition (DTD) gesehen werden, die für die gleichen Aufgaben vorgesehen war, sich aber als nicht flexibel genug herausstellte. Trotzdem ist sie noch häufig anzutreffen, wobei davon auszugehen ist, dass Xschema die DTD in Kürze ersetzen wird.
- *SMIL:* Synchronized Multimedia Integration Language (SMIL) bietet die Realisierung von Multimediaanwendungen. Einige Software-Hersteller ,wie beispielsweise RealNetworks , haben SMIL als Grundlage für die Erstellung von Multimediaanwendungen eingesetzt. Vom gleichen Hersteller gibt es auch eine Entwicklungsumgebung, die eine schnelle Umsetzung von Multimediainhalten für das Internet ermöglicht. So erstellte Dateien lassen sich dann über den frei verfügbaren REALPlayer abspielen.
- *XUL:* eXtensible User Interface Language (XUL) ist ein Ableger der Mozilla-Entwicklung, mit deren Hilfe sogenannte Fat-Clients erstellt werden können. Fat-Clients im Gegensatz zu Thin-Clients sind Clients, die ähnlich zu Java-Swing die Option bieten eine GUI zu erstellen, die unabhängig vom Browser benutzt werden kann.
- *XHTML:* HTML, welches den XML-Grundsätzen folgt, wird als XHTML bezeichnet.
- *S*OAP: Bei SOAP handelt es sich um ein Protokoll, dessen Einsatzbereich im Zusammenhang mit Web-Services eine grosse Bedeutung zukommt. SOAP[22] setzt auf dem HTTP-Protokoll auf und wird eingesetzt, um XML-Dateien über das Internet zu übertragen.

22 SOAP: Simple Object Access Protokoll

- *XMI:* XML Metadata Interchange (XMI) ist ein von der Object Management Group (OMG) erarbeiteter Standard, der den Austausch zwischen verschiedenen objektorientierten Modellierungssprachen ermöglichen soll.

3.2 erste Ansätze für serverseitige Sprachen

Prinzipiell können sowohl auf Client-Seite, als auch auf Server-Seite Programmiertechniken zum Einsatz kommen. Auf der Client-Seite werden hauptsächlich Techniken eingesetzt, die das Design der Web-Seiten betreffen. Hierbei handelt es sich um den Aufgabenbereich von Web-Designern, die dafür meist HTML, CSS und JavaScript einsetzen. JavaScript ist eine einfache Skriptsprache mit den Grundbestandteilen einer Programmiersprache.

Da sich die Clientseitige Programmierung als nicht flexibel genug erweist, sind viele Techniken entstanden, die auf der Server-Seite eingesetzt werden. Damit lassen sich alle Anforderungen an moderne Web-Applikationen abdecken. Sämtliche nachfolgende Ausführungen beziehen sich auf die Serverseitige Programmierung.

Während der Anfangszeit des Internet gab es noch keine Web-Frameworks, wie man sie heute kennt, sondern es gab erste Versuche, die Funktionalität der entsprechenden Web-Server durch zusätzliche Module zu erweitern. Dadurch wurden die ersten dynamischen Web-Applikationen ermöglicht, die neben reinem HTML auch andere Sprachen für die Ausführung der Web-Anwendung miteinbezogen. Diese ersten Ansätze werden nachfolgend beschrieben.

3.2.1 CGI

Mit einer der ersten Ansätze zur Erweiterung der Web-Server-Funktionalität, um das statische Konzept normaler HTML-Seiten aufzusprengen, war das Common Gateway Interface (CGI). CGI unterstützt als allgemeine Schnittstelle viele unterschiedliche Sprachen, wobei man hauptsächlich PERL[23] als Programmiersprache benutzt, um CGI

23 PERL: Programmiersprache zur einfachen Bearbeitung von Texten, Dateien und Prozessen

zu nutzen. Dadurch wurde die Möglichkeit geschaffen, dynamische Web-Inhalte zu erzeugen, indem beispielsweise Einzelheiten des HTTP-Stroms ausgelesen werden konnten. Formulareingabedaten können an ein PERL-Script geleitet werden, um dieses auszuwerten und anschließend das passende Ergebnis an den Aufrufer zurückzuschicken.

Ein häufiges Problem bei CGI-Anwendungen besteht darin, dass für jede Anfrage ein schwergewichtiger Betriebssystemprozess gestartet wurde, was sich natürlich sehr negativ auf die Performance solcher Anwendungen auswirkte.

3.2.3 JavaServlet API

SUNS's Antwort auf CGI nennt sich JavaServlet API[24]. Ähnlich zum CGI können auch über die JavaServlet API sämtliche Informationen aus dem HTTP-Strom gewonnen werden mit dem Unterschied, dass nun die ganze Mächtigkeit von Java genutzt werden kann. Da die JavaServlet API mittlerweile in der Version 2.4 vorliegt handelt es sich um eine ausgereifte Programmierschnittstelle für Web-Anwendungen, auf der viele andere Technologien aufbauen.

Das Problem bei JavaServlets besteht darin, dass der HTML-Output mühselig innerhalb des Servlets erzeugt werden muss, was dazu führt, dass der Präsentationscode schwer zu erzeugen und noch schwieriger zu warten ist. Aus diesem Grund sind JavaServer Pages entstanden, die im nächsten Kapitel beschrieben werden.

Viele der nachfolgend beschriebenen Ansätze basieren auf der JavaServlet API, wie JavaServer Faces, Struts, JavaServer Pages oder auch Tapestry und sind daher ein Grundbestandteil vieler Web-Applikationen.

24 http://java.sun.com/products/servlet/reference/api/index.html

3.3 Hybride Ansätze

Da CGI, als auch Servlets einen grossen Fortschritt in Richtung serverseitige flexible Programmierung bot, sind darin doch einige Schwächen zu sehen, für die man Lösungsmöglichkeiten suchte.

Gemeinsam ist diesen Ansätzen eine Verflechtung von HTML mit der jeweiligen Programmiersprache, welche zwar einen Schritt in Richtung erweiterter Funktionaliät von Web-Applikationen führt, jedoch auch Nachteile mit sich bringt. Durch die Verflechtung von HTML-Elementen mit Elementen der jeweiligen Programmiersprache kommt es häufig zu einer schwer verständlichen Mischung aus Präsentationselementen mit Geschäftslogik, die bei Erweiterung bzw. Wartung erhebliche Nachteile mit sich bringt, da dadurch die Rollenverteilung bei der Entwicklung nur unzureichend unterstützt wird und ein Web-Designer entweder solide Java-Kenntnisse mitbringen muss oder der Java-Programmierer gleichzeitig gute Web-Design-Fähigkeiten mitbringen sollte.

Diese als hybrid bezeichneten Ansätze werden in den nachfolgenden Abschnitten beleuchtet.

3.3.1 JavaServer Pages

Die JavaServer Pages (JSP)[25] stellen eine Erweiterung und eine Anwort auf die Schwächen der JavaServlet API dar. Bei Nutzung von Servlets müssen zum Rendern von HTML-Seiten sämtliche Befehle in den Ausgabestrom geschrieben werden, was zu einer unübersichtlichen Ansammlung von print-Befehlen führt, die eine Wartbarkeit und Erweiterbarkeit der Anwendung erschweren.

Ferner kommt dazu, dass sowohl das Web-Design als auch die Web-Programmierung in den Händen einer Person liegt, was meistens zu keinem guten Ergebnis führt, es sei denn der Java-Programmierer ist gleichzeitig ein talentierter Web-Designer.

Um diese Probleme zu beseitigen bzw. einzudämmen wurden die JavaServer Pages

25 http://java.sun.com/products/jsp/reference/api/index.html

geschaffen, die dafür gedacht sind die erforderliche Logik bei Bedarf in eine HTML-Seite einzubetten, die in diesem Zusammenhang dann JavaServer Page heißt, und von einem Pre-Prozessor automatisch in ein Servlet übersetzt wird.

Dieser Übersetzungsvorgang erfolgt automatisch und wird vor dem Entwickler verborgen. Das ermöglicht eine bessere Aufgabenteilung zwischen Web-Designer und Web-Programmierer, da der Web-Designer nun in der Lage ist seine Tools für die Erstellung der Seite zu benutzen und der Web-Programmierer die Logik in die Seite einbetten kann, ohne die Arbeit des Web-Designers zu beeinträchtigen.

Allerdings sieht die Realität doch meistens anders aus, da es auch bei diesem Ansatz zu einer unübersichtlichen Vermischung von HTML-Tags und unübersichtlichen Java-Scriptlets führt, die das Lesen einer solchen Seite erschwert und ähnliche Nachteile nach sich zieht wie bei PHP.

Einen möglichen Ausweg aus diesem Dilemma bieten die Template basierten Sprachen oder auch die Web-Frameworks, die in den nachfolgenden Kapiteln näher beschrieben werden.

3.3.2 PHP

PHP[26] wird seit ungefähr 10 Jahren entwickelt und erfreut sich stetig zunehmender Beliebtheit bei den Entwicklern von dynamischen Web-Seiten. PHP ist eine Scriptsprache, die serverseitig ausgeführt wird, allerdings unter der Voraussetzung, dass ein entsprechendes PHP-Modul für den Web-Server installiert wurde.

PHP-Code wird in normale HTML-Seiten eingebettet und wird zur Laufzeit von dem entsprechenden Web-Server-Modul übersetzt, welches im Anschluss daran die Ergebnisse an den Browser in Form von reinem HTML zurückschickt.

Eine besondere Stärke von PHP liegt in der relativ einfachen Anbindungsmöglichkeit unterschiedlichster Datenbanken. Durch die Vermischung von PHP-Code und HTML-Tags kommt es allerdings zu einer Mischung von Präsentationselementen und der dahinterliegenden Logik, was zu schwer wartbarem Code führen kann.

26 http://www.php.net/

3.4 Template basiert

Unter einem Template soll eine Vorlage verstanden werden, in welcher Basiskonstrukte, wie Schleifen und Verzweigungen verwendet werden können, um den programmiertechnischen Teil einer Web-Anwendung auszulagern und die HTML-Seite, welche dann nur noch der Darstellung der Inhalte dienen soll, von Codebestandteilen möglichst frei zu halten.

Alle Template-Sprachen benutzen dafür eine simple Scriptsprache und haben sich auf ihre Fahnen geschrieben die Arbeit zwischen Web-Designer und Web-Programmierer so aufzuteilen, dass jeder nur noch für seinen Teil zuständig ist. Der Web-Designer soll ein gutes Design erstellen und der Web-Programmierer die notwendige Logik beispielsweise in Form einer Datenbankanbindung bereitstellen. Dies ist ein absolut wichtiges Kriterium bei der Entwicklung von Web-Anwendungen, da nicht jeder Web-Designer auch gleichzeitig ein guter Programmierer ist und umgekehrt.

In den folgenden Abschnitten soll dieser Ansatz anhand von unterschiedlichen Umsetzungen näher beleuchtet werden, wobei die Auswahl sich auf einige wenige beschränken muss und keinesfalls eine vollständige Darstellung aller am Markt befindlichen Template-Sprachen bietet.

3.4.1 WebMacro

WebMacro basiert auf folgenden Annahmen[27]:

- We think it is wrong to use markup for a scripting lnaguage
- We think it is wrong to embed programs on a web pages
- We think it is wrong for web scripts to look like hard programming
- We feel that an API like WM should make it easy to do the things you need to do when designing and deploying applications

27 [JOH01] Seite 545

- We believe that programming and graphical page designs are seperate tasks

Da die Konzepte von WebMacro[28] sich in Velocity spiegeln und nur die Syntax etwas anders ist, soll daher im nächsten Kapitel darauf eingegangen werden.

3.4.2 Velocity

Velocity[29] von Jakarta kann als OpenSource-Version von WebMacro angesehen werden, welche die Grundkonzepte um einige weitere Funktionalitäten erweitert. Velocity versteht sich als Lösung, die HTML-Seite von jeglicher Logik zu befreien, wodurch Web-Designer und Java-Programmierer gemeinsam an einem Projekt arbeiten können.

Ein Beispiel soll dies verdeutlichen:

```
<html>
        <body>
                #set ( $hello = „Hello World“ )
                $hello !
        </body>
</html>
```

Anhand des Beispiels wird die grundsätzliche Funktionsweise von der VTL klar. Variablen werden durch #-Zeichen und das $-Zeichen kann als Referenz darauf verwendet werden. Die Scriptsprache beinhaltet alle gängigen Standardprogrammierkonstrukte wie z.B. Schleifen, Verzweigungen ,Vergleiche und logische Operatoren, wodurch natürlich nicht die Mächtigkeit „echter“ Programmiersprachen erreicht werden soll. Einfach und auch für Web-Designer verständlich soll es sein.

28 http://sourceforge.net/projects/webmacro
29 http://jakarta.apache.org/velocity

Die Stärke von Velocity liegt in der Auslagerung von Code auf sogenannte Velocimacros, die es möglich machen, Codeteile extern auszulagern, um sie anschliessend in der HTML-Seite über einen VTL-Befehl einzubinden. Das ausgelagerte Macro kann wiederum alle VTL-Möglichkeiten voll ausschöpfen.

Beliebtes Einsatzgebiet von Velocity ist das Ersetzen von JSP-Seiten in Struts- oder JavaServer Faces Anwendungen, bzw. generall als View-Komponente für Servlet basierte Programmierung.

3.4.2 JSTL

Die Java Standard Tag Library[30] (JSTL) ist eine Erweiterung für die JSP-Technologie und ist derzeit in der Version 1.1 verfügbar. Die JSP-Technologie unterstützt das Erstellen eigener Tags, welche den Zweck haben, die Logik fernab der HTML-Seite in einer normalen Java-Klasse unterzubringen und diese Funktionalität über ein entsprechendes Tag der HTML-Seite zugänglich zu machen.

Dadurch soll auch ein Web-Designer in der Lage sein dieses Tag zu benutzen ohne die dahinter liegende Programmierlogik zu kennen. Das Ziel der JSTL besteht darin, durch vordefinierte Tags für unterschiedliche Bereiche, eine vorgefertigte Tag-Library zu liefern, die sofort einsetzbar sein soll.

Die JSTL gliedert sich in folgende Teilbibliotheken:

- *core:* für Ein- und Ausgaben
- *fmt:* für die Formatierung von Zahlen
- *sql:* bietet direkte Zugriffe auf die Datenbank
- *xml:* ist für das Handling mit XML- und XSLT-Dateien zuständig
- *fn:* enthält verschiedene Funktionen

30 http://jcp.org/aboutJava/communityprocess/final/jsr052/index2.html

3.5 Komponenten

Um den Begriff der Komponente zu erläutern, werde ich mich an Helmut Balzert[31] orientieren, der folgende Aussagen über Komponenten macht.

„Eine Komponente ist ein geschlossener, binärer Software-Baustein, der eine anwendungsorientierte, semantisch zusammengehörende Funktionalität besitzt, die nach außen über Schnittstellen zur Verfügung gestellt wird."

„In der industriellen Produktion ist es üblich und insbesondere wirtschaftlich, ein Produkt auf Halbfabrikaten aufzubauen. Die Halbfabrikate werden in der Regel von spezialisierten Herstellern bezogen."

„Eine komponentenbasierte Software-Entwicklung erlaubt eine einfachere, schnellere und preiswertere Herstellung von Anwendungen mit Hilfe von vorgefertigten Komponenten."

3.5.1 JavaBeans

Das Standardkomponentenmodell von Java ist die JavaBeans-API[32], welches die Erstellung wiederverwendbarer Komponenten ermöglicht und in jeder Java-basierten Web-Applikation verwendet wird. Ausgereifte Web-Frameworks, wie JavaServer Faces und Struts haben dieses Komponentenmodell integriert und mit neuen Fähigkeiten ausgestattet. Bei Struts spricht man von ActionForms und bei JavaServer Faces werden sie als Managed Beans bezeichnet.

Die JavaBeans dienen dazu, bestimmte Funktionen über vordefinierte Schnittstellen bereitzustellen. Durch einzuhaltende Design-Vorschriften bei der Programmierung von JavaBeans sind entsprechende Werkzeuge in der Lage, Informationen über diese speziellen Klassen einzuholen und diese dann zu bearbeiten, ohne dass der Programmierer explizit Code schreiben muss.

31 [BAL01] Seite 856
32 http://java.sun.com/products/javabeans/docs/spec.html

Die Swing-Bibliothek von Java demonstriert diesen Gebrauch von JavaBeans, in der beispielsweise die visuellen Komponenten, Table, TextField, Panel etc., durch ein visuelles Entwicklungstool erstellt und ihre Eigenschaften verändert werden können. Dadurch lassen sich schnell Benutzeroberflächen erstellen, ohne den jeweiligen Code dafür selbst zu erstellen.

Bei den Web-Frameworks wird diese Form von komponentenorientierter Programmierung ebenfalls umgesetzt. Tapestry, welches noch ausführlicher beschrieben wird, ist komponentenorientiert ausgerichtet und stellt diese dem Entwickler zur Verfügung, der diese nur noch nach seinen Bedürfnissen anpassen muss.

JavaServer Faces hat dieses Komponentenmodell ebenfalls integriert, sowohl im visuellen Bereich, als auch im Bereich der Datenhaltung. In Kapitel 5 wird dies ausführlich beschrieben.

3.5.2 Enterprise JavaBeans

Enterprise JavaBeans (EJB)[33] ermöglichen das Erstellen einer verteilten Anwendung, mit deren Hilfe man entfernte Objekte in Form von Enterprise JavaBeans über ein Netzwerk ansprechen kann und dies eine Nutzung ermöglicht, als wären sie lokal auf dem Rechner vorhanden. Sie wurden von SUN entwickelt und bilden einen wichtigen Bestandteil der J2EE.

3.6 Web-Frameworks

Hat man sich für die Entwicklung einer Web-Applikation entschieden, stellt sich die Frage, ob man das System komplett neu entwirft, oder auf bewährte Muster zurückgreift, in Form von Frameworks oder Komponenten. Diese Frage ist nicht leicht zu beantworten, da viele Faktoren darauf Einfluss nehmen.

33 http://java.sun.com/products/ejb

Eine Auswahl an Fragen, die geklärt werden müssen, bevor eine Entscheidung gefällt werden kann, wie die Web-Applikation entwickelt wird:

- Welche Sprache soll eingesetzt werden ?
- Gibt es erfahrene Programmierer ?
- Gibt es intern schon Komponenten, die wiederbenutzt werden können ?
- Wie komplex wird das entstehende System ?
- Wie hoch ist das Budget für das Projekt ?

Dies ist nur eine Auswahl an Fragen, die geklärt werden müssen, bevor eine Entscheidung gefällt werden kann, auf welcher Basis die Web-Applikation entwickelt wird.

Welche Vor- und Nachteile können durch den Einsatz von Frameworks erreicht werden?

Ins Deutsche kann dieses Wort als Gerippe oder auch Rahmen übersetzt werden. Ein Framework kann allgemein als Lösungsvorschlag für unterschiedliche Problemkreise beim Erstellen einer Web-Applikation verstanden werden.

Ein wichtiges Merkmal ist das Vorhandensein eines gewissen Designs bzw. eines Architekturentwurf, der benutzt werden kann, und dadurch meistens eine bessere Architektur erzielt wird, als das bei einer Neuentwicklung des Softwaredesigns der Fall wäre. Frameworks bieten eine mehr oder weniger feste Vorgabe für ein Problemfeld, sei es beispielsweise ein Modell für Persistenzstragien oder ein Modell für das Erstellen der Präsentationsschicht einer Web-Applikation.

Da die Einarbeitung in ein Framework Zeit und KnowHow von den Entwicklern erfordert, muss abgewägt werden, ob man nicht schneller ohne die Hilfe eines Frameworks entwickeln kann. Allerdings zahlt sich die Einarbeitung in den meisten Fällen nach der Einarbeitungsphase aus, da dadurch bessere Softwareentwürfe

entstehen, die sich auf lange Sicht auch besser warten lassen. Außerdem kann man davon ausgehen, dass in den Frameworks jeweils die Best Practices im Bereich Softwareentwurf angewandt wurden und der Entwickler sich mehr um die tatsächliche Problemstellung kümmern kann, als das Rad permanent neu erfinden zu müssen.

Durch Frameworks wird eine gemeinsame Sprache gefördert, wodurch sich die Kommunikation unter den Entwicklern entscheidend verbessern kann und dies Einfluss auf die Produktivität des Teams hinterlässt.

Da es unzählige Frameworks für die unterschiedlichsten Anwendungen gibt, kann hier nur eine kleine Auswahl näher beschrieben werden.

Da im Rahmen dieser Arbeit der Schwerpunkt auf 2 Frameworks gesetzt wurde, die Lösungshilfen für die Präsentationsschicht bieten, bezieht sich auch die Auswahl der nachfolgend beschriebenen Frameworks auf die Präsentationsschicht.

3.6.1 Tapestry

Bei Tapestry[34] handelt es sich im ein OpenSource-Framework zur Entwicklung von Web-Applikationen, das seit seinem Erscheinen steigende Popularität genießt. Es ist mittlerweile ein Projekt von Jakarta geworden und wird in der Entwicklerlizenz hoch angesehen, da es einige neue Ansätze verfolgt, die man von Java Server Faches oder Strunks nicht kennt.

Eines der Hauptprobleme, die während der Entwicklung auftreten, besteht darin, wie die Rollen innerhalb eines Teams zu trennen sind und sich dies auch auf die Arbeitsweise bei der Entwicklung niederschlagen soll. Der HTML-Designer soll sich um das Design der Anwendung kümmern und der Programmierer soll die Logik implementieren.

Web-Frameworks versuchen dieses Problem mehr oder weniger erfolgreich zu lösen, wobei es trotzdem häufig zu einer Mischung der Rollen kommt, die sich negativ auf das entstehende Produkt auswirken kann. Wie bereits im Kapitel über hybride Ansätze besprochen, besteht eine solche Web-Seite, die die Rollen vermischt, aus einer Mischung aus HTML und Java-Scriptlets im Falle von JSP-Seiten. Dazu kommt, dass

34 http://jakarta.apache.org/tapestry

die HTML-Seiten, die vom HTML-Designer erstellt wurden, in JSP-Seiten konvertiert werden müssen, wodurch sich eine anschließende Veränderung durch den Web-Designer erschwert, da dieser ja auch weiterhin seine Programmierumgebung nutzen möchte.

Professionelle IDE's[35] bieten nicht die gleiche Ausstattung, wie professionelle HTML-Tools zur Web-Seitenerstellung, wodurch sich die Arbeit zwischen Web-Designer und Programmierer erschwert. Tapestry bietet für dieses Problem einen Lösungsansatz, der sich von den anderen hier beschriebenen Frameworks deutlich unterscheidet und einen neuen Weg beschreitet.

Ein Problem stellt allerdings das Fehlen von Literatur über Tapestry dar. Einziges erhältliches Buch stammt von dem Entwickler von Tapestry, Howard M. Lewis Ship[36], welches allerdings Tapestry in allen Aspekten darstellt und für dieses Kapitel von unschätzbarem Wert war.

Eine Tapestry-Applikation besteht aus folgenden 3 Teilen:

- *HTML-Vorlage:* Diese Vorlage wird vom Web-Designer mit dem Tool seiner Wahl erstellt und stellt die grafische Präsentation der Anwendung dar. Diese wird bereichert um Tapestry-Components, die im Tapestry-Jargon als JavaWeb Components bezeichnet werden und durch einen kleinen Zusatz innerhalb der HTML-Tags gekennzeichnet werden.
- *Page-Spezifikation:* Zu jeder HTML-Vorlage wird eine Page-Spezifikationsdatei angelegt. Hierbei handelt es sich um eine XML-konforme Datei, in der die Zuordnung zu einer Page-Class vorgenommen wird. Ferner können hier die auf der HTML-Seite verwendeten Eingabefelder entsprechenden Datentypen zugeordnet werden.
- *Page-Class:* In der Page-Class wird die eigentliche Logik der Anwendung codiert.

35 IDE: Integrated Development Environment (Bezeichnung für Entwicklungsumgebungen)
36 [SHI01]

3.6.2 Struts

Struts[37] ist ein OpenSource-Produkt von der Apache Software Foundation, welches als Version 1.0 im Juni 2001 den Markt betrat. Einer der Hauptarchitekten und Lead Developer von Struts, Craig McClanahan, trug maßgeblich zum Erfolg von Struts bei. Er ist auch ein wichtiger Akteur bei der Entwicklung von JavaServer Faces, wodurch klar wird, warum sich beide Frameworks stark ähneln. Struts liegt mittlerweile in der Version 1.2 vor und kann als ausgereiftes und anerkanntes Framework bei der Entwicklung von Web-Applikationen bezeichnet werden.

Struts wird ausführlich in Kapitel 5 beleuchtet.

3.6.3 JavaServer Faces

JavaServer Faces (JSF)[38] von SUN kann als Antwort auf ASP.NET von Microsoft gesehen werden, da es auf ähnlichen Konzepten beruht und sich auch als ereignisgesteuertes Web-Framework präsentiert. Im März 2004 wurde die Version 1.0 freigegeben und liegt nun 2005 in der Version 1.1.01 vor, welche viele Mängel der Vorgängerversion beseitigt hat. Konzepte, wie die Trennung der Daten von der Präsentation, als auch ein ereignisgesteuertes Event-Modell, welche sich in der GUI-Programmierung durchgesetzt haben, zeichnet JavaServer Faces aus.

SUN versucht wie Microsoft ihre Technologie über ein Entwicklungswerkzeug zu vermarkten. Allerdings ist der Java Studio Creator nur auf die Entwicklung von Web-Anwendungen beschränkt und kann somit mit dem Visual Studio .NET auf keinen Fall mithalten, da dieses eine Lösung für alle Bereiche der Programmierung unterstützt.

Die JavaServer Faces Technologie und ihre Entwicklungswerkzeuge werden in Kapitel 4 ausführlich dargestellt.

37 http://struts.apache.org/
38 http://java.sun.com/j2ee/javaserverfaces/reference/api/index.html

3.6.4 ASP.NET

ASP.NET[39] aus dem Hause Microsoft kann als Nachfolgeprodukt zu den Active Server Pages (ASP) gesehen werden. Ein grosser Nachteil bei der Programmierung von ASP-Seiten bestand darin, dass die Seiten in einer Scriptsprache, Jscript oder VBScript, geschrieben werden mussten. Dadurch kam es zu einer unangenehmen Mischung aus HTML-Code und Script-Code. Weitere Probleme gab es bei der Ausführung der ASP-Seiten, da die Scriptsprache nur zur Laufzeit ausgeführt wurde, was Einbußen bei der Performance hatte. All diese Probleme sind durch ASP.NET gelöst worden, wodurch die Programmierung von Web-Seiten nun wesentlich einfacher geworden ist.

ASP.NET ist eingebettet in das .NET Framework, welches als Gesamtpaket, auch die Grundarchitektur für die .NET-Server bildet. Da Microsoft ein eigenes Entwicklungswerkzeug, das Visual Studio .NET, bereitstellt, erfolgt die ASP.NET-Programmierung komplett Tool-unterstützt.

Wie auch JavaServer Faces arbeiten die ASP.NET-Seiten nach dem Ereignismodell, wonach das Programm auf Benutzerevents reagiert, und passende Methoden aufruft. Ein reichhaltiges Angebot an Server-Controls, die neben den Standard-Komponenten, auch beispielsweise Controls für die Darstellung von XML-Inhalten beinhaltet. Die Trennung der View vom Model wurde durch sogenannte Code-Behind-Files realisiert, die das Rückgrat der HTML-Seite bilden und darüber ein Ansprechen der Server-Controls auf der ASP.NET-Seite möglich wird. HTML-Designer und ASP.NET-Programmierer können nun einfacher miteinander arbeiten. Ferner sind noch Validierungsmöglichkeiten in das Framework eingebaut.

Ein Nachteil von ASP.NET ist das Fehlen eines Navigationssystems für die Web-Seiten, wie das z.B. bei Struts oder JavaServer Faces vorhanden ist.

Der größte Vorteil von ASP.NET besteht darin, dass sich problemlos andere Kernkonzepte von .NET, wie ADO.NET für die Datenbankanbindung oder die Web-Services einbauen lassen.

39 http://www.asp.net/

3.7 Web-Application Server

Web-Server sind diejenigen Funktionseinheiten, die benötigt werden um eine Anwendung über ein Netzwerk zur Verfügung zu stellen. Dieser Server-Typ richtet seinen Schwerpunkt auf die Bereitstellung von statischen HTML-Seiten. Meist bieten moderne Web-Server ein breites Spektrum an zusätzlichen Möglichkeiten an, um dynamische Web-Seiten zu generieren. So ergibt sich je nach Server, die Möglichkeit auch ASP-Seiten, JSP-Seiten, Servlets und CGI-Skripte auszuführen.

Da im Rahmen dieser Diplomarbeit der Schwerpunkt auf die Java-Programmierung von Web-Applikationen gelegt wird, beziehen sich die nachfolgenden Ausführungen auf Server, die im Java-Umfeld eingesetzt werden.

Eine weitere Gattung von Servern ist die der Application-Server, die ähnlich wie normale Web-Server HTML-Seiten auf Abruf bereitstellen können. Darüber hinaus allerdings bietet diese Serverart die Möglichkeit einen Pool von Java-Objekten zu verwalten und durch eine reiche Infrastruktur unterschiedlichste Aufgaben zu lösen. Besonders in verteilten Anwendungen werden solche Applikations-Server eingesetzt. Die nachfolgende Einteilung wurde gewählt um zu zeigen, dass je nach Anforderung der Anwendung unterschiedliche Serverarten zur Verfügung stehen und man genau abwägen sollte welcher Servertyp die passendste Lösung für die Anwendung darstellt.

3.7.1 schwergewichtige J2EE-Server

Schwergewichtige J2EE-Server bieten meistens folgende Dienste an:

- *HTTP-Dienste:* Dieser Dienst ist der allgemeingültigste Service und wird von allen Web-Servern angeboten. Ohne ihn wäre die Bearbeitung von Client-Anfragen über das HTTP-Protokoll nicht möglich.
- *Servlet-Ausführung:* Servlets stellen bei der Entwicklung von Web-Applikationen eine Grundanforderung dar und werden von vielen Frameworks als Controller eingesetzt. Ein sehr wichtiger Dienst, der von allen Applikationsservern angeboten

wird.

- *JSP-Ausführung:* JSP-Pages stellen eine Erweiterung zu Servlets dar und werden auch von den meisten Frameworks als View-Komponente benutzt. Auch dieser Dienst ist eine Grundvoraussetzung, der von vielen Web-Frameworks gefordert wird.
- *EJB-Dienste:* Ermöglicht das Erstellen von verteilten Anwendungen mittels Enterprise JavaBeans in Form von Session Beans und Entity Beans.
- *Java Naming and Directory Interface (JNDI):* Dieser Dienst wird von den Enterprise JavaBeans benötigt und muss von dem Applikationsserver auch angeboten werden, falls Enterprise JavaBeans unterstützt werden. Es wird ein Directory-Service angeboten, der die Enterprise JavaBeans in einer Art Telefonbuch für EJB's auflistet und genutzt werden muss, um die EJB's aufzufinden.
- *Java Message Service (JMS):* Ein Nachrichtensystem, welches die Option eröffnet ein asynchrones Nachrichtenmodell umzusetzen. Dabei ist es nicht mehr erforderlich, dass Sender und Empfänger zur gleichen Zeit sende- bzw. empfangsbereit sind, also synchron, sondern die Nachrichtenübermittlung erfolgt losgekoppelt über ein System aus Message-Producer und Message-Subscriber. So können Geschäftsprozesse angestoßen werden, wobei nicht alle Teilnehmer auch gleichzeitig über eine Verbindung miteinander in Beziehung stehen.
- *Java Transaction API (JTA):* Spezifiziert die Implementierung eines Transaction-Managers, welcher JTA unterstützt und die Java-Version des OMG[40] Object Transaction Services (OTS) implementiert.
- *JavaTransaction Services (JTS):* Erlaubt die Verwendung von Transaktionen[41] und sorgt für die nötige Sicherheit.
- *JDBC[42] Pooling:* Da Datenbankresourcen sehr resourcenintensiv sind, bieten viele Applikationsserver die Möglichkeit über einen integrierten Datenbank-Pool Verbindungen zur jeweiligen Datenbank zu verwalten.
- *Security:* Dienste, die für die Sicherheit einer Anwendung sorgen und sowohl

40 OMG: Object Management Group

41 Transaktion: Eine unteilbare Einheit von Aufgaben, um Daten zu modifizieren. Eine Transaktion umfasst eine oder mehrere Programmanweisungen, welche entweder alle ausgeführt oder durch ein Rollback rückgängig gemacht werden. Transaktionen ermöglichen es mehrere Benutzer gemeinsam den gleichen Datenbestand nutzen.

42 JDBC: Java Database Connectivity

deklarativ in der Konfigurationsdatei des J2EE-Servers, als auch programmatisch innerhalb der Anwendung beschrieben werden können.

3.7.2 leichtgewichtige J2EE-Server

Unter leichtgewichtigen J2EE-Server sollen insbesondere Servlet-Engine's verstanden werden, die nicht die volle Anzahl an Diensten bereitstellen wie die schwergewichtigen J2EE-Server. Nachfolgend werden die Dienste aufgeführt, die als Basisdienste angeboten werden. Da diese bereits oben beschrieben wurden, wird auf eine nochmalige Beschreibung verzichtet.

Leichtgewichtige J2EE-Server sollten die folgenden Dienste bereitstellen:

- HTTP-Dienste
- Servlet-Ausführung
- JSP-Ausführung
- JDBC-Pooling
- Security

Für die in Kapitel 3.6 beschriebenen Web-Frameworks mit Ausnahme von ASP.Net, kann ein leichtgewichtiger J2EE-Server benutzt werden. Als Standard hat sich hier die Servlet-Engine Tomcat als sehr zuverlässig erwiesen, weswegen er auch für die Beispielapplikationen zu JavaServer Faces und Struts eingesetzt werden kann.

4. JavaServer Faces

In diesem Kapitel rückt JavaServer Faces als das erste von 2 Web-Frameworks für die Präsentationsschicht in das Blickfeld und wird im Anschluss daran anhand einer Beispielapplikation auf Praxistauglichkeit überprüft.

4.1 Einführung

JavaServer Faces (JSF) ist ein Web-Framework, das die Erstellung von Web-Applikationen vereinfachen und beschleunigen soll. Es handelt sich dabei um den Java Specification Request (JSR) 127[43], welcher JavaServer Faces zugrunde liegt. Von SUN wurde auch eine Referenzimplementierung (RI)[44] vorgelegt, auf die sich die folgenden Ausführungen beziehen.

Auf der Web-Site von SUN wird das Web-Framework wie folgt beschrieben:

„JavaServerTM Faces technology simplifies building user interfaces for JavaServer applications. With the well-defined programming model that JavaServer Faces provides, developers of varying skill levels can quickly and easily build web applications by: assembling reusable UI components in a page, connecting these components to an application data source, and wiring client-generated events to server-side event handlers. With the power of JavaServer Faces technology, these web applications handle all of the complexity of managing the user interface on the server, allowing the application developer to focus on application code.“[45]

Durch Vorgabe eines Architekturrahmens und Stützung auf Technologien, die sich bereits bewährt haben, soll Rapid Application Development (RAD), ein schnelleres Entwickeln von Software, ermöglicht werden. Viele Probleme, auf die man gewöhnlich in der Präsentationsschicht einer Web-Anwendung trifft, sind von JavaServer Faces adressiert worden.

43 http://www.jcp.org/en/jsr/detail?id=127
44 http://java.sun.com/j2ee/javaserverfaces/download.html
45 http://java.sun.com/j2ee/javaserverfaces/download.html

- Trennung von Präsentation und Logik
- Navigation bzw. Seitenfluss
- GUI-Komponenten
- Konvertierung von Eingabedaten
- Validierung von Benutzereingaben
- Event-Handling von Benutzeraktionen
- Internationalisierung

JavaServer Faces steht in Zusammenhang mit folgenden Technologien:

- *JavaServer Pages Specification*[46]: JSP-Seiten können für die View verwendet werden, so auch im Rahmen dieser Arbeit.
- *JavaServlet Specification*[47]: JavaServer Faces benutzt als zentrale Eingangspforte ein Servlet, welches alle Anfragen an die Anwendung weiterleitet.
- *Java 2 Platform*[48]: Dies ist die Basisplattform für alle Java-Anwendungen.
- *JavaBeans Specification*[49]: Die Modellobjekte von JavaServer Faces müssen gemäß JavaBeans-Konvention gestaltet werden.
- *JavaServer Pages Standard Tag Library*[50]: Diese Bibliothek kann zusammen mit JavaServer Faces eingesetzt werden, ist aber mit Vorsicht zu benutzen, da es häufig zu Problemen bei gleichzeitigem Einsatz beider Technologien kommen kann[51].
- *JavaServer Faces Specification*[52]: Die Spezifikation, die das Web-Framework genau beschreibt.

46 http://java.sun.com/products/jsp/reference/api/index.html
47 http://java.sun.com/products/servlet/reference/api/index.html
48 http://java.sun.com/j2se
49 http://java.sun.com/products/javabeans/docs/spec.html
50 http://jcp.org/aboutJava/communityprocess/final/jsr052/index2.html
51 http://www.onjava.com/pub/a/onjava/2004/06/09/jsf.html
52 http://java.sun.com/j2ee/javaserverfaces/download.html

4.2 Architektur von JavaServer Faces

In diesem Kapitel wird ein Überblick über die wichtigsten Funktionalitäten von JavaServer Faces gegeben.

4.2.1 Das UI-Komponenten-Modell

JavaServer Faces stellt vorgefertigte graphische Komponenten zur Verfügung, wie man sie auch von anderen GUI[53]-Bibliotheken kennt, wie beispielsweise Eingabefelder, Listenfelder, Checkboxen, Elemente zur Darstellung von Tabellen und Bildern. Eingebunden werden diese Elemente über Tags auf der JSP-Seite und können dort über entsprechende Attribute in ihrem Look & Feel verändert werden. Eine Komponente besteht aus 2 Teilen, der UI-Komponente[54], die für die Funktionalität zuständig ist und den zugeordneten Render-Klassen, die die Komponente graphisch darstellen. Durch diese Aufgabenteilung kann eine UI-Input-Komponente z.B. auf 4 unterschiedliche Weisen dargestellt werden. Möglich wäre eine Darstellung als einzeiliges Textfeld, mehrzeiliges Textfeld, verstecktes Textfeld oder verschlüsseltes Textfeld für Passworteingaben. In der Beispielapplikation wurden fast alle angebotenen UI-Komponenten benutzt. Daher verweise ich auf Kapitel 4.6.4, wo die UI-Komponenten auf den jeweiligen Screenshots zu sehen sind und dort auf einige genauer eingegangen wird.

4.2.2 Konvertierung von Datentypen

Werden Daten in einem Formular auf Client-Seite gesammelt, müssen diese anschließend über das HTTP-Protokoll zum Server übertragen werden, der dann für die weitere Verarbeitung zuständig ist. Aufgrund der Eigenschaften des HTTP-Protokolls werden die Daten als Zeichenketten zum Server geschickt, wodurch es auf Server-Seite

53 GUI: Graphical User Interface
54 UI: User Interface

notwendig wird die Zeichenketten auf die entsprechenden Datentypen der zugrunde liegenden Modellobjekte abzubilden. Standardmäßig versucht JavaServer Faces automatisch die richtige Konvertierung durchzuführen. Ist dies nicht ausreichend besteht die Möglichkeit eine UI-Komponente mit einem Konverter zu verknüpfen. JavaServer Faces stellt einige Standard-Konverter zur Verfügung, die entweder über eigene JSP-Tags innerhalb der Komponente oder direkt als Attribute der jeweiligen Komponente angegeben werden können.

4.2.3 Validierung

Im Zuge der Formular-Auswertung muss auch geprüft werden, ob die eingegebenen Werte bestimmten Vorgaben Rechnung trägt. Hierfür werden ähnlich wie bei den Konverter-Klassen Standard-Validierer bereitgestellt, die es ermöglichen die Eingaben durch bestimmte Vorgaben zu überprüfen und bei Fehleingaben eine Fehlerausgabe zu erzeugen. Eine Einbindung der Validierer erfolgt durch Verknüpfung mit der entsprechenden Komponente.

4.2.4 Internationalisierung

Eine weitere wichtige Anforderung an moderne Web-Applikationen ist die Internationalisierung oder Lokalisierung einer Anwendung. Beide Begriffe bringen die Anpassung an länderspezifische Details wie Sprache, Datumsangaben, Währungen und Zahlsysteme zum Ausdruck. Im Rahmen dieser Arbeit sollen die beiden Begriffe synonym benutzt werden.

Seit dem JDK1.1 gibt es die Möglichkeit eine Anwendung mithilfe der Klassen Locale, Calendar, TimeZone und ResourceBundle [55] zu lokalisieren. Die JavaServer Faces-Technologie knüpft an dieses Konzept an. Es können einfache Text-Files erstellt werden, die dem Key=Value-Schema folgen.

Als Beispiel wird hier der Inhalt der *commandBundle_de.properties* für deutsche Texte

55 java.util:*; java.text.*

gezeigt:

```
btnSubmit=Abschicken
btnCancel=Abbrechen
btnSave=Speichern
```

Soll auch eine englische Version unterstützt werden, erstellt man die gleiche Datei mit dem Namen *commandBundle_en.properties* und fügt den Werten entsprechende englische Übersetzungen bei.

Im Anschluss daran kann dieses File über ein JSP-Tag geladen werden und steht dann der Seite zur Verfügung. Welches File geladen wird, entscheidet JavaServer Faces automatisch anhand der jeweiligen Benutzereinstellungen des Browsers.

4.2.5 Modell-Komponenten

Modell-Komponenten werden in JavaServer Faces als ManagedBeans bezeichnet. Diese haben ihren Ursprung in den JavaBeans, die im Java Sprachkern enthalten sind, und eine der wichtigsten Möglichkeiten in Java darstellen, wiederverwendbare Objekte zu erzeugen. Sämtliche GUI-Klassen der Swing-Klassen sind z.B. als JavaBeans realisiert, wodurch sich die Entwicklung von herkömmlichen Benutzeroberflächen stark vereinfacht hat und ihren Niederschlag in den WYSWYG-Editoren[56] gefunden hat.

Bislang bestand die Möglichkeit JavaBeans über JSP-Tags der Anwendung zur Verfügung zu stellen. Dies verursacht allerdings einen Verstoß gegen das Prinzip der Trennung von Model und Präsentation, wodurch sich die Les- und Wartbarkeit von JSP-Seiten deutlich verschlechtert. Dieses äußerst wichtige Kriterium bei der Entwicklung von Web-Applikationen wird von JavaServer Faces durch die Managed-Beans gelöst.

Managed-Beans werden in Konfigurationsdatei von JavaServer Faces ‚der faces-config.xml eingetragen und können dort mit Default-Werten versehen werden. Man kann dort auch den Anwendungs-Scope angegeben, der bestimmt, welchen Gültigkeitsbereich die Managed-Bean innerhalb der Anwendung hat. Möglich sind 4

56 what you see is what you get

unterschiedliche Angaben, wobei 3 davon nachfolgend beschrieben werden. Die vierte Möglichkeit besteht darin den Scope mit null anzugeben.

- *application*: Das Managed-Bean ist während der gesamten Laufzeit der Anwendung gültig.
- *session*: Das Managed-Bean ist nur während einer Arbeitssitzung gültig. Schließt der Client beispielsweise den Browser, wird die Session nach einem Timeout beendet und das Bean verliert seine Gültigkeit.
- *request*: Die Gültigkeit des Managed-Beans ist nur während der aktuellen Requestverabeitung gewährleistet.

Der große Nutzen von Managed-Beans besteht darin, dass die Werte von UI-Komponenten mit den Managed-Beans verknüpft werden können. Dadurch wird eine einfache Abbildung von Formulardaten auf Modellobjekte möglich.

4.2.6 Navigation

Das Herz von JavaServer Faces bildet die faces-config.xml, eine Konfigurationsdatei im XML-Format, mit deren Hilfe auch die Navigationssteuerung realisiert wird. Alle Navigationsregeln können hier deklarativ hinterlegt werden und bestimmen den Seitenfluss der Anwendung. Für jede JSP-Seite kann eine Navigationsregel angegeben werden, die je nach Rückgabewert der einzelnen Komponenten auf der Seite, zu einer unterschiedlichen Weiterleitung auf eine andere Seite führt.

4.2.7 Event-Handling

Das Event-Handling von JavaServer Faces erinnert stark an Konzepte, die man auch in Swing wiederfindet und es ist das wesentlichste Merkmal von JavaServer Faces, das es

von anderen Web-Frameworks unterscheidet. Vieles was der Entwickler bei anderen GUI-Frameworks gelernt hat, kann hier in ähnlicher Weise angewandt werden, wodurch für GUI-Programmierer ein sehr leichter Einstieg bietet.

JavaServer Faces bietet 4 Arten von Events:

- *Action-Events:* Sie werden von UI-Komponenten ausgelöst und an registrierte Listener übergeben.Auslöser können z.B. ein Command Button oder Command-Link sein.
- *ValueChange-Events:* Alle Unterklassen von UI-Input erzeugen diese Event-Art. Trägt ein User z.B. einen Wert in ein Textfeld wird dieses Event erzeugt und kann die angeschlossenen Listener benachrichtigen.
- *DataModel-Events:* UI-Data-Komponente erzeugen dieses Event bei Auswahl einer Reihe.
- *Phase-Events:* Diese werden bei Beginn und Ende der Request-Verarbeitung ausgelöst.

4.2.8 Zustandsspeicherung

Durch den Umstand, dass Web-Applikationen über das HTTP-Protokoll kommunizieren, welches zustandslos ist, muss eine Möglichkeit existieren um Zustände zwischen den Requests zu speichern.

Diese Problematik greift JavaServer Faces auf und bietet 2 Lösungsansätze:

- *Client-State-Saving:* Der Zustand wird über ein verborgenes Eingabefeld gespeichert und bei jedem Request an den Server geleitet.
- *Server-State-Saving:* Sämtliche Informationen, die sich auf eine Session beziehen

werden von JavaServer Faces auf dem Server zwischengespeichert. Die zum Browser passende Session wird mit Hilfe von Cookies bewerkstelligt. Sind auf der Client-Seite die Cookies deaktiviert worden, wird automatisch Client-State-Saving benutzt.

Welche Methode zum Einsatz kommen soll, wird im Deployment-Descriptor *web.xml*[57] über *true* bzw. *false* festgelegt. Im unten gezeigten Beispiel wird Client-State-Saving aktiviert.

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

4.2.9 Konfigurationsdateien

Die Konfigurationsdatei von JavaServer Faces, in der alle notwendigen Einstellungen für eine lauffähige Anwendung eingetragen werden, wird als f*aces-config.xml* bezeichnet. Mit ihrer Hilfe lassen sich weite Teile der Anwendung konfigurieren. Die wichtigsten Elementen sind nachfolgend beschrieben.

- *application:* Die hier vorgenommenen Einstellungen gelten für die gesamte Web-Applikation und können über die gleich lautende Klasse innerhalb einer Anwendung angesprochen werden, um sämtliche Einstellungen abzufragen, wie z.B. Werte von Managed-Beans. Pro Anwendung existiert diese Klasse genau einmal.
- *component:* Werden eigene Komponenten entwickelt müssen diese hier eingetragen werden.
- *render-kit:* Neue Render-Klassen für neue oder vom Framework vorgegebene Komponenten werden hier angegeben.
- *converter:* Hier werden neue Konverter eingetragen.

57 In der web.xml-Datei werden allgemeine Informationen zur Web-Anwendung hinterlegt und sind der Servlet-Spezifikation definiert.

- *validator:* Neu erstellte Validatoren müssen hier eingetragen werden.
- *managed-bean:* Alle Beans, die als Managed-Beans der Anwendung zur Verfügung stehen sollen, müssen hier eingetragen werden. Zusätzlich lassen sich diese mit Werten vorbelegen.
- *navigation-rule:* Hier werden Navigationsregeln für alle JSP-Seiten hinterlegt.

4.3 Erstellen eigener Komponenten

Die Referenzimplementierung von Sun stellt ein Basispaket an Komponenten, Validierer und Konverter bereit, die bereits viele Problemkreise abdecken können. JavaServer Faces stellt sich nicht nur als ein Basisframework dar, sondern kann auch als Baukastensystem für die Entwicklung eigener Komponenten dienen. Wie auch in Swing eröffnet sich die Möglichkeit Funktionalitäten, die auch von anderen Anwendungen wiederverwendet werden können, als eigenständige Komponenten zu implementieren.

4.4 Nutzung von Komponenten anderer Hersteller

Wie im vorhergehenden Kapitel beschrieben können neue Komponenten, wie UI-Komponents, Validierer und Konverter erstellt werden. Durch diese Möglichkeit sind inzwischen viele neue Komponenten erhältlich, sowohl freie als auch kostenpflichtige.

Oracel ADF-Faces[58]:

ADF-Faces ist im Zusammenhang mit dem ADF-Framework von Oracle zu sehen. Gegenwärtig ist eine Early Acces Version erhältlich, die einen sehr vielversprechenden Eindruck hinterlässt , da eine sehr große Anzahl an JSF-Komponenten integriert sind, die keine Wünsche mehr offen lässt. Diese lassen sich jetzt schon über die Preview Edition des JDeveloper 10.1.3 von Oracle nutzen, müssen allerdings vorher erst installiert werden.

58 http://www.oracle.com/technology/products/jdev/htdocs/partners/addins/exchange/jsf/index.html

Myfaces[59]:

Myfaces bietet eine Open-Source Implementierung von JavaServer Faces unter der Schirmherrschaft von Apache-Software-Foundation. Neben den Komponenten, die in der Referenzimplementierung von SUN enthalten sind, werden darüber komplexere Komponenten, als auch oft vermisste Komponenten angeboten, wie z.B. FileUpload, TabbedPane, Navigationsmenüs, Kalender und viele mehr. Da man auch Zugriff auf den Sourcecode hat, stellt dieses Paket eine interessante Alternative zur Referenzimplementierung von SUN dar.

Smile[60]:

Hierbei handelt es sich um eine Open-Source Implementierung von JavaServer Faces. Das Entwicklerteam von Smile und Myfaces haben im März 2004 beschlossen eine Kooperation einzugehen. Neben erweiterten Komponenten, die in einer eigenen Entwicklungsumgebung nutzbar sein sollen, plant Smile auch die Einführung von Test-Scripten, um die JSF-Seiten zu testen. Da das Projekt noch nicht die Versionsnummer 1.0 kann es noch nicht produktiv eingesetzt werden. Allerdings ein interessantes Projekt, welches im Auge behalten werden sollte.

Otrix[61]:

Es werden 3 UI-Komponenten kostenpflichtig zum Verkauf angeboten.

- WebTree
- WebMenu
- WebGrid

59 http://myfaces.apache.org
60 http://smile.sourceforge.net
61 http://www.otrix.com/

jenia4faces[62]:

Ins Leben gerufen wurde dieses Open-Source-Projekt am 1.Juli 2005 in Italien. Es werden einige interessante Komponenten zum kostenlosen Download angeboten.

- *Popup:* Komponenten, die HTML-Popups rendern.
- *DataTools:* Eine Erweiterung der DataTable-Komponente.
- *Dymamic:* DHMTL Texteffekte.
- *Template:* Ein Templatemanager.

4.5 Entwicklungswerkzeuge und Tools

Da es mittlerweile eine große Anzahl an verfügbaren Entwicklungsumgebungen für JSF gibt, kann im Rahmen dieser Arbeit nur eine kleine Auswahl vorgestellt werden. Die Ausführungen zu Exadel JSF Studio, Faces Console und JSF Form Builder stützen sich auf die Beschreibungen von Sven Haiges[63] und sind nicht in der Praxis getestet worden.

Sun Java Studio Creator[64]:

SUN's eigene Entwicklungsumgebung für JavaServer Faces, die eine komfortable Programmierumgebung für das Entwickeln von JSF-Anwendungen bietet. Sie wird zusammen mit der Datenbank PointBase, dem Tomcat 5 Application Server und dem J2EE 1.4 Application Server ausgeliefert. Alle JSF-Komponenten sind über eine Komponentenleiste verfügbar und lassen eine komfortable Editierung ihrer Eigenschaften über ein separates Fenster zu. Die faces-config.xml kann visualisiert werden und bietet so einen schnellen Überblick über die Navigationsregeln der Anwendung. Da diese Entwicklungsumgebung leider nur auf die Entwicklung von JSF-Applikationen abzielt, stellt sich die Frage über einen möglichen Einsatzbereich, da andere Entwicklungsumgebungen wie beispielsweise der JDeveloper von Oracle

62 http://www.jenia.org/jsp/home.jsf
63 [HAM01] Seite 206 ff
64 http://www.sun.com/software/products/jscreator

darüber hinaus auch andere Technologien unterstützen und somit ein breiteres Spektrum an Möglichkeiten abdecken.

Exadel JSF Studio[65]:

Diese Entwicklungsumgebung bietet neben Support für JSF-Anwendungen auch Unterstützung für Spring, Hibernate und Struts und kann daher flexibler eingesetzt werden als der JavaStudio Creator von SUN. Es stehen unterschiedliche Versionen bereit, die sich durch ihren Funktionsumfang unterscheiden. Als Flaggschiff wird das Exadel Studio 3.0 Pro angeboten, welche Unterstützung für oben genannte Open-Source Frameworks anbietet.

Faces Console[66]:

Die kostenlose Faces Console bietet eine bequeme Möglichkeit die faces-config.xml zu bearbeiten. Sämtliche Einstellungen können über diesen graphischen Editor vorgenommen werden. Ferner besteht die Möglichkeit einer Integration in fast alle gängigen IDE's, falls man dieses Tool nicht Standalone benutzen möchte.

JSF FormBuilder[67]:

Mit Hilfe des JSF FormBuilder, welches kostenlos ist, lassen sich JSF-Formulare erstellen mit einer grafischen Oberfläche erstellen. Das Tool erzeugt dann automatisch die JSP-Seiten mit den JSF-Tags, die ManagedBeans sowie die Elemente der faces-config.xml.

Jdeveloper 10.1.3 (Developer Preview Edition)[68]:

Der JDeveloper von Oracle ist eine Entwicklungsumgebung, die sämtliche Programmieraufgaben in Zusammenhang mit Java unterstützt. Die Preview Edition 10.1.3 beinhaltet Support für JSF-Anwendungen und wurde auch für die

65 http://www.exadel.com/products_products1.htm
66 http://www.jamesholmes.com/JavaServerFaces/console
67 http://piet.jonas.com/FormBuilder/JsfFormBuilder.html
68 http://www.oracle.com/technology/products/jdev/101/index.html

Beispielapplikation eingesetzt. Da dieses Tool seit Juli 2005 kostenlos verfügbar sein soll, ist es meiner Meinung das beste Tool für die Erstellung von JSF-Applikationen, da wirklich sämtliche in Zusammenhang mit JSF stehende Programmieraufgaben unterstützt werden. Angefangen bei der Visualisierung der faces-config.xml, Code Highlighting, lassen sich desweiteren alle JSF-Komponenten komfortabel über eine Leiste anwählen und können anschließend über einen Property-Editor bearbeitet werden. Da es keine Wünsche offen lässt und in Zukunft eine Integration von ADF-Faces beinhaltet ist es ein durchweg empfehlenswertes Hilfsmittel nicht nur für JSF-Applikationen, sondern generell für alle Java-Aufgaben.

4.6 Web-Shop als Beispielapplikation

Der theoretische Teil, in dem alle wichtigen Aspekte von JavaServer Faces beschrieben wurden, soll nun durch eine Beispielapplikation in die Praxis umgesetzt werden. Es wird ein Web-Shop realisiert, der es einer imaginären Musikfirma ermöglicht, Musikinstrumente zum Verkauf anzubieten.

Die Bücher von Andy Bosch[69], Hans Bergsten[70] und Kito Mann[71] bieten viele praxistaugliche Anregungen und waren während der Entwicklung eine gern aufgesuchte Informationsstätte.

4.6.1 Use-Cases

Durch ein Use-Case-Diagramm sollen die Anforderungen an die Benutzerverwaltung grafisch dargestellt werden. Durch Benutzen eines Use-Case-Diagramms lässt sich schnell aufzeigen, welche Funktionalitäten das Programm bieten soll. Es eignet sich hervorragend für die Kommunikation innerhalb des Teams, als auch zur Kommunikation mit möglichen Auftraggebern.

Nachfolgendes Use-Case-Diagramm gibt einen ersten Überblick der Funktionalitäten

69 [BOS01]
70 [BER01]
71 [MAN01]

des Programms.

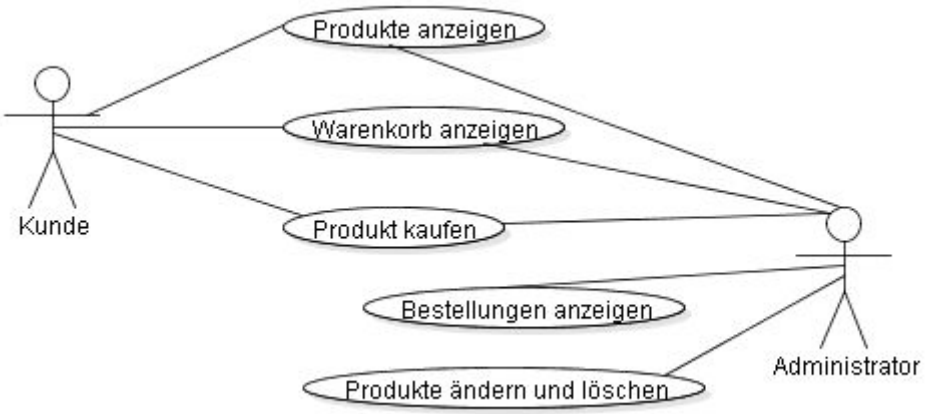


Abbildung 5: *Use Case Web-Shop*

Kunde:

Kunden haben die Möglichkeit sich das Warensortiment anzuschauen, eine Ware auszuwählen und in den Warenkorb zu legen. Sie können sich anschließend den Inhalt des Warenkorbs anzeigen lassen und diesen editieren. Der Kunde hat die Möglichkeit die in den Warenkorb gelegten Artikel zu bestellen. Dazu muss ein Formular ausgefüllt werden, das alle benutzerspezifischen Daten zur Person und zum Versand enthält. Nach erfolgreicher Bestellung bekommt der Kunde eine Bestätigungsmail zugeschickt.

Administratoren:

Administratoren haben zusätzlich die Möglichkeit neue Produkte anzulegen, bzw. vorhandene Produkte zu editieren. Außerdem können sie sich eine Liste aller getätigten Bestellungen anzeigen lassen, die sowohl eine Grobübersicht, als auch eine Detailsicht auf die Bestellungen erlaubt.

4.6.2 Geschäfts- und Persistenzschicht

Wie im Kapitel zur Web-Architektur aufgezeigt, soll auch die Beispielanwendung auf mehrere Schichten aufgeteilt werden. Das Geschäftsmodell besteht aus folgenden Klassen, die nachfolgend als UML-Klassendiagramm dargestellt werden.

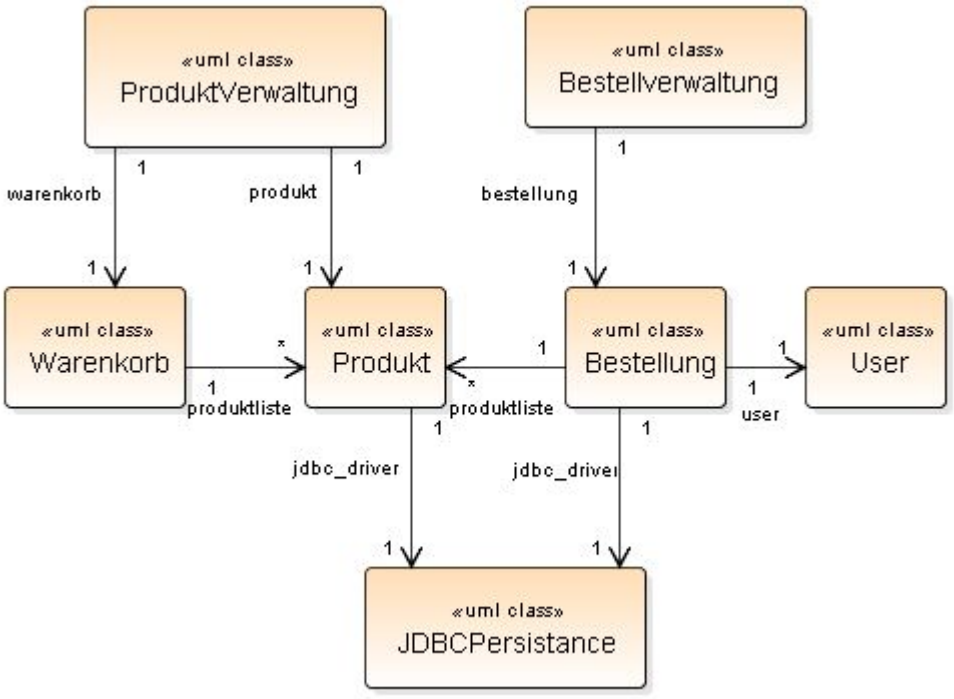


Abbildung 6: UML-Klassendiagramm der Geschäfts- und Persistenzschicht Web-*Shop*

Die Persistenzschicht wird durch JDBCPersistance repräsentiert und regelt sämtlichen Datenbankverkehr.

Das Domain-Model[72], bestehend aus den Klassen Warenkorb, Produkt, Bestellung und User beschreibt die Geschäftsobjekte, die für diesen Web-Shop benötigt werden. Die Klasse Produkt und Bestellung bieten einen Persistenzmechanismus, implementiert durch Datenbankfunktionen, die dann von JDBCPersistance ausgeführt werden.

Der eigentliche Zugriff auf das Domain-Model erfolgt durch die Klassen Produkt- und Bestellverwaltung, die dann als einzige Ansprechpartner für die Präsentationsschicht, in diesem Fall JavaServer Faces zur Verfügung gestellt werden.

72 [EVA01] Seite 81 ff: Eric Evans hat ein sehr schönes Buch zum Thema Domain-Driven Design geschrieben, indem ausführlich das Domain-Model erklärt wird

Durch diesen geschichteten Aufbau wird gewährleistet, dass die einzelnen Schichten jeweils separat nutz- und austauschbar sind. Daraus ergeben sich viele Vorteile, welche bereits im Kapitel zur Web-Architektur angesprochen worden sind.

Klasse JDBC Persistance

Die Klasse JDBCPersistance ist für die Anbindung einer Oracel-Datenbank zuständig und bietet über implementierte Datenbankfunktionen das Absetzen von SQL-Befehlen an die Datenbank.

Jeglicher Datenbankverkehr wird ausschließlich von dieser Klasse bearbeitet und wird als Datenbankservice von den Klassen Produkt und Bestellung benutzt.

Klasse User

Die Klasse User dient zu der Beschreibung eines Kunden durch die Eigenschaften Vorname, Nachname, Kommentar, Strasse, PLZ, Stadt und E-Mail.

Klasse Produkt:

Die Klasse Produkt dient zu der Beschreibung eines Produktes durch die Eigenschaften ID, Hersteller, Name, Preis, Bild und Anzahl. Der Persistenzmechanismus wird durch 4 Datenbankfunktionen realisiert, die die notwendigen Operationen an die Klasse JDBCPersistance weiterreichen. 3 Datenbankfunktionen ermöglichen das Anlegen, Löschen und Editieren eines Produktes. Über eine zusätzliche Datenbankfunktion können sämtliche Produkte abgefragt werden, deren Rückgabewert eine Liste aller Produkte enthält.

Klasse Bestellung

Die Klasse Bestellung dient zur Beschreibung einer Bestellung durch die Eigenschaften ID, Bestelldatum, einem Kunden und einer Liste von Produkten.

Klasse Warenkorb:

Der Einkaufswagen des Web-Shops wird durch die Klasse Warenkorb realisiert. Sie ist dafür zuständig, die in den Warenkorb gelegten Produkte in einer ArrayList zu speichern. Außerdem stehen 2 Funktionen bereit, durch die die Anzahl und Summe der im Warenkorb befindlichen Produkte errechnet werden können.

Klasse ProduktVerwaltung:

Über die Klasse ProduktVerwaltung werden sowohl die Produkte verwaltet, als auch der Warenkorb für den Einkauf eines Kunden bewerkstelligt.

Die Verwaltung aller Produkte ermöglicht das Anlegen neuer Produkte, Löschen von vorhandenen Produkten und das Editieren von vorhanden Produkten. Die eigentliche Ausführung wird an die Klasse Produkt weitergereichtet und dort bearbeitet.

Die Verwaltung des Warenkorbes beinhaltet das Ablegen eines Produktes und das Löschen von Produkten im Warenkorb.

Klasse BestellVerwaltung

Die Klasse BestellVerwaltung ist für den Zugriff auf Funktionen der Klasse Bestellung verantwortlich. Sie ist in der Lage bei einem Bestellvorgang eine neue Bestellung anzulegen, indem der Inhalt des Warenkorbs der Produktverwaltung und ein Userobjekt an eine Bestellung übertragen werden, welche dann für die nötigen Aktualisierungsvorgänge der Datenbank sorgt.

4.6.3 Datenbankmodell

Als Datenspeicher für Produkte und Bestellungen wird eine Oracel-Datenbank angebunden. Das Datenkbankmodell wird durch folgendes ERD[73] veranschaulicht.

Abbildung 7: Datenbankmodell Web-*Shop*

Die Tabelle Bestellung dient für das Ablegen von Bestellungen und wird durch die Attribute BestellungID, Bestelldatum, Vorname, Nachname, Strasse, PLZ und Ort beschrieben.

Die Tabelle Produkt beinhaltet Produkte, beschrieben durch ProduktID, Hersteller, Name, Kategorie und Preis.

Die Tabelle Bestellung_Produkt beinhaltet alle Produkte, die einer Bestellung zugeordnet sind.

Es wurde mit Absicht ein Datenbankmodell gewählt, welches sich lediglich in der 1.Normalform befindet, da es den Zwecken der Beispielapplikation Rechnung trägt. Beispielsweise könnte aus der Tabelle Bestellung eine Tabelle User extrahiert werden. Darauf wurde verzichtet, weil die Beispielapplikation diesbezüglich auch keine Unterstützung bietet. Sinnvoll wäre eine solche Änderung, falls beispielsweise eine Login-Möglichkeit für registrierte Kunden gewährleistet werden soll.

Gleiches gilt für die Tabelle Produkt. Auch hier könnte das Attribut Kategorie extrahiert werden und als einzige Tabelle erstellt werden.

Im Rahmen einer Web-Applikation, die vornehmlich die Funktionsweise eines Web-Frameworks darstellen möchte, wurde deswegen ein möglichst einfaches Datenmodell gewählt.

Alle SQL-Scripte für das Erstellen der Tabellen und das Füllen mit Dummy-Daten

73 ERD: Entity Realationship Diagramm

befinden sich auf beiliegender CD.

4.6.4 JSP-Seiten

Für die View werden JSP-Seiten verwendet, in der fast aller UI-Komponenten von JavaServer Faces zum Einsatz kommen und die Stärke von JavaServer Faces herausstellen sollen. Nachfolgend werden alle JSP-Seiten durch Screenshots dokumentiert und einige UI-Komponenten näher erläutert.

Startseite

Die Startseite stellt den Eintrittspunkt für den Web-Shop dar. Der User kann entweder über den Link Produktliste den Web-Shop betreten, oder sich als Administrator einloggen, um die volle Funktionalität des Programms zu nutzen.

Abbildung 8: Startseite Web-*Shop*

Produktliste

Hier werden alle zum Verkauf angebotenen Produkte angezeigt. Maximal fünf Produkte werden aufgelistet, wobei der User die Möglichkeit hat, sich über die Buttons „vor“ und „zurück“ durch das Warensortiment zu navigieren.

Ein Produkt wird über den Link „buy“ in den Warenkorb gelegt und die Anzahl der

Produkte im Warenkorb wird anschließend aktualisiert und ist in der Navigationsleiste oben rechts grau markiert zu sehen. Der Einkäufer hat hierdurch eine Kontrolle über die Anzahl, der bisher in den Warenkorb gelegten Produkte.

Hat man sich als Administrator angemeldet, werden zusätzliche Links angeboten, die das Anlegen neuer Produkte und das Editieren vorhandener Produkte ermöglicht.

Abbildung 9: Produktübersichtseite (Web-*Shop)*

Abbildung 10: Produktübersichtseite Admin (Web-*Shop)*

Die Produktliste wird durch die JavaServer Faces Komponente HtmlDataTable realisiert, die eine Lösung für das Anzeigen von dynamischen Datenstrukturen bereitstellt. Folgendes Codefragment zeigt diesen Sachverhalt.

```
<h:dataTable value="#{ProduktVerwaltungMB.produktModel}"
                        var="pList" cellpadding="5"
                        headerClass="tableHeader left" rows="5"
                        first="#{ProduktMB.counter}"
                        cellspacing="0"
                        columnClasses="left,left,left,left,right,center"
                        frame="hsides">
        <h:column>
                <h:graphicImage url="#{pList.bild}" height="50"
                        width="35"/>
        </h:column>
```

Über das Attribut *value* wird die Produktliste mit Hilfe der Klasse ProduktVerwaltung geladen. Der zu vergebende Name für diese Datenstruktur, über den später auf Elemente der Datenstruktur angesprochen werden können, kann beliebig gewählt werden Die Attribute rows und first sind dafür zuständig, dass jeweils nur fünf Produkte aus der Liste angezeigt werden, wobei die Startposition des ersten Elementes durch *first* bestimmt wird und dieser Zähler durch die Managed Bean ProduktMB verwaltet wird.

Anschließend werden sämtliche Spalten über jeweils ein c*olumn* definiert, über die auf die einzelnen Elemente der eingetragenen Datenstruktur zugegriffen werden kann.

Anhand des Links *buy* und des Buttons *vor* soll das Event-Handling von JavaServer Faces erläutert werden. Folgendes Codefragment zeigt den Link *buy*.

```
<h:commandLink action="#{ProduktVerwaltungMB.addZumWarenkorb}">
        <h:outputText value="buy"/>
</h:commandLink>
```

Mit Hilfe des Attributes *action* wird der Rückgabewert der Komponente angegeben, der in die faces-config.xml als Navigationsregel eingetragen werden kann und bestimmt, welche Seite als nächstes aufgerufen wird. Der Rückgabewert kann entweder direkt, oder wie in diesem Falle als Rückgabewert einer Funktion angegeben werden. Nach betätigen des Links wird die Funktion addZumWarenkorb der ManagedBean ProduktVerwaltungMB aufgerufen, die den Warenkorb aktualisiert und anschließend einen addedZumWarenkorb zurückgibt. Dieser Wert wird in diesem Fall nicht in die faces-config.xml als Navigationsregel eingetragen, da die Seite nicht verlassen werden

soll.

Der Button *vor* zeigt einen anderen Weg Funktionen über die JavaServer Faces Komponenten anzusprechen. Ausgegangen wird von folgendem Codefragment.

```
<h:commandButton value="vor" action="forward"
        actionListener="#{ProduktMB.forward}"/>
```

Hier wird die Funktion *forward()* der ManagedBean ProduktMB in das Attribut *actionListener* eingetragen. Bei dieser Version wird eine Funktion aufgerufen, die keinen Rückgabewert hat. Der Unterschied der beiden Formen des Eventhandling wird deutlich, wenn beispielsweise sowohl in dem Atrribut action, als auch in dem Attribut actionListener Funktionen eingetragen werden. In diesem Fall wird die Funktion des actionListener vor dem der action aufgerufen. Dies ermöglicht eine Hirarchie von Funktionsaufrufen.

Anlegen oder Editieren eines Produktes

Diese Seite erlaubt das Anlegen oder Editieren eines Produktes. Für beide Fälle wird die gleiche JSP-Seite benutzt, jeweils angepasst an die Anforderungen durch einen vorherigen Funktionsaufruf, der bei Anklicken der jeweiligen Links auf der Produktübersichtseite ausgeführt wird. Wird ein neues Produkt angelegt, sind die Eingabefelder nur zum Teil mit Werten vorbelegt. Falls ein vorhandenes Produkt editiert wird, werden alle Eingabefelder mit den entsprechenden Werden gefüllt. Bis auf die Eingabefelder Bild und Kommentar müssen die Eingabefelder ausgefüllt werden, ansonsten werden Fehlermeldungen ausgegeben. Durch den Button hinzufügen beziehungsweise ändern, werden die entsprechenden Aktualisierungen ausgelöst und man wird wieder auf die Produktübersichseite gelenkt.

Abbildung 11+12: Seite für das Anlegen oder Editieren eines Produktes (Web-*Shop)*

Dieses Eingabeformulare nutzt die Validierungsmöglichkeiten von JavaServer Faces. Anhand dieses Eingabeformulares wird einerseits die Verdrahtung einer ManagedBean mit einer JSP-Seite und die Validierungsmöglichkeiten von JavaServer Faces aufgezeigt. Als Beispiel soll das Eingabefeld für den Preis genommen werden. Folgendes Codefragment bietet die Grundlage der Ausführungen.

```
<h:inputText value="#{ProduktMB.preis}" required="true"
        id="preis">
        <f:validateLongRange minimum="1"/>
</h:inputText>
```

Zuerst wird das Eingabefeld deklariert und über das Attribut *value* der Wert bestimmt, welcher angezeigt werden soll. In JavaServer Faces können alle Eigenschaften der ManagedBeans, auf die später noch eingegangen wird, über die HTML-Komponenten von JavaServer Faces angesprochen werden. So wurde auch hier durch Angabe von *ProduktMB.preis* das Eingabefeld mit der Eigenschaft *preis* der ManagedBean *ProduktMB* fest verdrahtet. Wird nun ein Text eingegeben, so ist die jeweilige Eigenschaft der ManagedBean mit dem entsprechenden Wert belegt und kann anschließend weiterverarbeitet werden. Diese Anwendung führt nach Abschluss der Eingaben eine Aktualisierung der angebundenen Datenbank durch.

Über das Attribut *required* kann festgelegt werden, ob es sich um ein Pflichteingabefeld handelt durch Angabe von *true*. Die zweite Validierungstechnik, die eingesetzt wurde, ist der Standardvalidierer *validateLongRange*, durch den der Zahlenbereich eingeschränkt werden kann. In diesem Falle muss der Wert größer als eins sein.

Anzeigen des Warenkorbes

Der User kann sich jederzeit den Inhalt des Warenkorbes anschauen, indem er auf den Link oben in der Navigationsleiste klickt. Anschließend wird eine Liste aller im Warenkorb liegenden Produkte angezeigt, einschließlich der Summe dieser Produkte.

Hier kann der User entscheiden, welche Produkte im Warenkorb bleiben sollen, indem durch den Link „remove“ die Möglichkeit besteht ein bestimmtes Produkt aus dem Warenkorb zu entfernen.

Der User kann sich hier entscheiden, ob er weiter einkaufen möchte, dann wird er durch den Button „zurück zum Shop“ auf die Produktlistenseite geleitet, oder er kann durch Betätigen des Buttons „bestellen“ den Bestellvorgang auslösen und wird dann auf die Bestellungsseite geführt.

Abbildung 13: Warenkorbseite (Web-*Shop)*

Für das Anzeigen der Warenkorbliste wurde die HTML-Komponente DataTable benutzt. Diese wurde bereits vorher beschrieben und wird hier in ähnlicher Weise benutzt.

Bestellung:

Hier muss der User alle notwendigen Daten über das bereitgestellte Formular eintragen, die für den Bestellvorgang notwendig sind. Die Eingabefelder sind alle mit der ManagedBean User verknüpft und die Pflichtfelder mit Validierern ausgestattet.

Wurden alle Felder ausgefüllt, kann durch Betätigen des Buttons „Abschicken“ die Bestellung abgeschickt werden. Der User wird dann auf die Seite für die erfolgreiche Bestellung weitergeleitet. Hier sei darauf hingewiesen, dass diese Funktion zu Testzwecken durch den Button add umgangen werden kann.

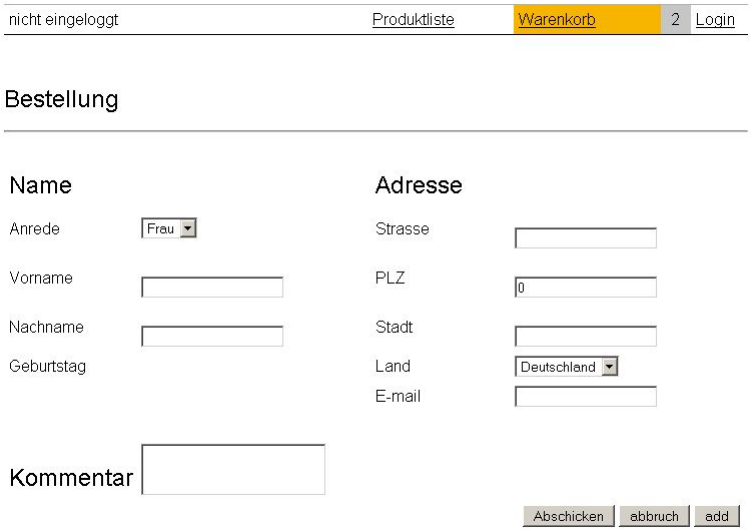


Abbildung 14: Bestellseite (Web-*Shop)*

Abbildung 15: Ausgabe bei Eingabefehlern (Web-*Shop)*

Bestellbestätigung

Hier bekommt der User ein Feedback über die erfolgreiche Bestellung durch einen Text, der anzeigt, dass die Bestellung erfolgreich aufgenommen wurde und er in Kürze eine Bestätigungsmail zugeschickt bekommt.

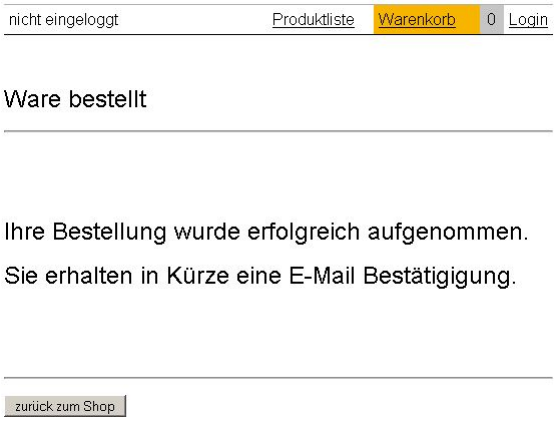


Abbildung 16: Bestellbestätigungsseite (Web-*Shop)*

Login

Die Navigationsleiste bietet einen Link an, über den sich der Web-Shop-Besitzer einloggen kann, um sämtliche Funktionen der Anwendung freizuschalten. Er muss anschließend auf der Login-Seite die Eingabefelder ausfüllen. Werden Eingabefelder ausgefüllt, erfolgt eine Weiterleitung an die Produktübersichtseite. Die Überprüfung auf einen bestimmten Namen und Passwort wurde mit Absicht weggelassen, könnten aber im Falle einer Erweiterung hinzugefügt werden. Somit können sämtliche Funktionen durch einen beliebigen Namen und Passwort freigeschaltet werden.

Abbildung 17: Loginseite (Web-*Shop)*

Übersicht aller Bestellungen

Diese Seite ist über einen Link in der Navigationsliste erreichbar. Links wird eine

Übersicht aller getätigten Bestellungen angezeigt, geordnet nach zeitlicher Reihenfolge. Rechts werden die Details der jeweiligen Bestellung angezeigt, die durch einen Link Details für jede Bestellung abgerufen werden können. Diese beinhaltet eine Auflistung der Produkte, die in der jeweiligen Bestellung enthalten sind.

Abbildung 18: Übersichtseite aller Bestellungen (Web-*Shop)*

4.6.5 Managed-Beans

Für die Anwendung werden 5 Managed Beans benötigt, die nachfolgend näher beschrieben werden. Als Dokumentationsgrundlage werden UML-Klassendiagramme benutzt, mit deren Hilfe die Feinheiten der Managed Beans erklärt werden. Die Managed Beans folgen der Code-Konvention der JavaBeans, wodurch jede Eigenschaft durch entsprechende getter- und setter-Methoden ausgestattet ist. Bei der Darstellung als UML-Klassendiagramm möchte ich darauf hinweisen, dass zum Teil absichtlich nur die Eigenschaften und zusätzliche Funktionenen im UML-Klassendiagramm zu sehen sind, um die Grafiken möglichst schmal zu halten. Hier sei noch einmal betont, dass jede Eigenschaft natürlich durch die entsprechenden getter- und setter-Methoden repräsentiert werden. Diese wurden lediglich für die Dokumentation aus dem UML-Diagramm gestrichen.

UML-Klassendiagramme für ProduktMB und ProduktVerwaltungMB

Abbildung 19: ProduktMB (Web-*Shop)*

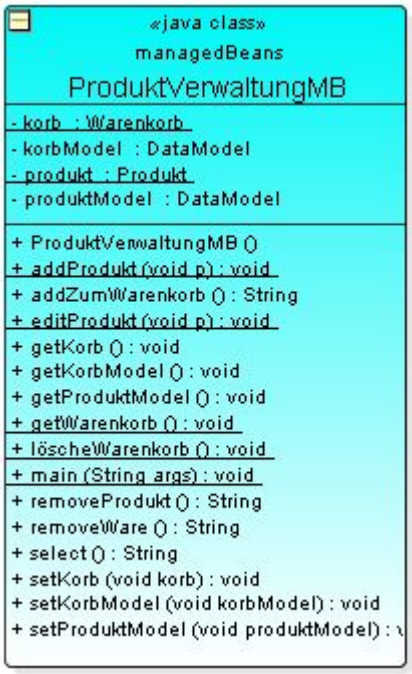


Abbildung 20: ProduktVerwaltungMB (Web-*Shop)*

ProduktMB

Die ManagedBean Produkt MB gleicht der Produktklasse aus dem Domain-Model. Es wäre denkbar gewesen die Produktklasse als ManagedBean zu implementieren, erweitert um zusätzliche Funktionen. Darauf wurde aber verzichtet, um eine strenge Aufgabenteilung zu erreichen. Deswegen ist die ProduktMB nur mit Aufgaben betraut, die die Präsentationsschicht betreffen.

Das Formular für das Anlegen oder Editieren eines neuen Produktes wird durch diese ManagedBean gestützt. Alle dabei ausgefüllten Formularfelder werden auf diese ManagedBean abgebildet und anschließend durch die Funktionen addProdukt() und editProdukt() an die ManagedBean ProduktVerwaltung weitergereicht, die die nötigen Verwaltungsarbeiten ausführt.

Das Navigieren durch die Produktliste auf der Produktübersichtseite wird durch die Methoden backward() und forward() durch aktualisieren einer Zählervariable erreicht.

Ferner ermöglicht sie die Vorbereitung des Formulares für das Produkt durch

entsprechendes Vorbelegen mit Werten der Eingabefelder.

ProduktVerwaltungMB

Die Klasse ProduktVerwaltung ist Bestandteil der Geschäftsschicht und wurde in Kapitel 4.6.2 besprochen. Diese Klasse wird nun der Präsentationsschicht zur Verfügung gestellt, indem man sie als ManagedBean in die faces-config.xml einträgt.

Als ManagedBean führt sie einige JavaServer Faces spezifische Funktionen aus, die benötigt werden, um die DataTable für die Warenkorbseite und die Produktübersichtseite mit Werten zu versorgen und bei Auswahl einer Zeile die Daten einer Zeile zu ermitteln, die durch die DataTable dynamisch erzeugt werden.

UML-Klassendiagramme für UserMB und LoginMB

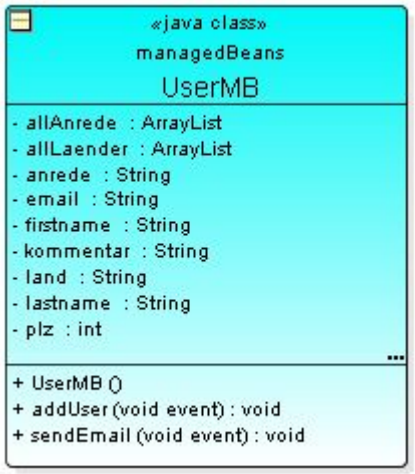


Abbildung 21: UserMB (Web-*Shop)*

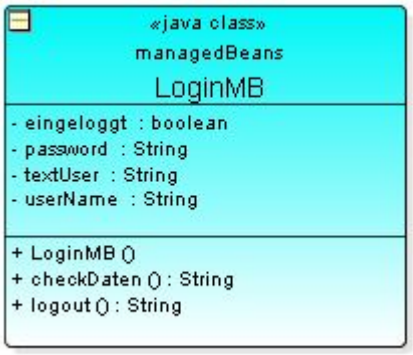


Abbildung 22: LoginMB (Web-*Shop)*

UserMB

Die Klasse UserMB wird ebenfalls wie die Produktklasse als eigenständige ManagedBean erstellt und ist nur auf die Erfordernisse der Präsentationsschicht abgestimmt. Alle Eingabefelder der Bestellseite, auf der Angaben zur Person gemacht werden müssen, sind mit den Eigenschaften dieser ManagedBean verknüpft und werden nach Eingabe aller Eingabefelder durch Übergabe eines normalen User-Objektes an die ManagedBean Bestellverwaltung weitergereicht, die den Bestellvorgang verwaltet.

Zusätzlich dazu wird durch die Funktion sendEmail(...) durch Übergabe der E-Mail Addresse an die Klasse E-Mail das Abschicken einer E-Mail eingeleitet.

LoginMB

Die ManagedBean LoginMB dient lediglich zur Unterstützung der Login-Seite. Sie überprüft die Daten der Eingabefelder auf Richtigkeit und weist den Benutzer auf eventuelle Fehler hin, die den Login-Vorgang vereitelt haben.

BestellVerwaltungMB

Die Klasse BestellVerwaltung aus dem Domain-Model, welche bereits in Kapitel 4.6.2 beschrieben wurde, wird unverändert als ManagedBean in die faces-config.xml eingetragen, um sie der Präsentationsschicht zur Verfügung zu stellen.

UML-Klassendiagramm für BestellVerwaltungMB

Abbildung 23: BestellVerwaltungMB (Web-*Shop)*

4.6.6 web.xml und faces-config.xml

Alle ManagedBeans müssen in die faces-config.xml eingetragen werden, um sie der Anwendung zur Verfügung zu stellen. Anhand der LoginMB wird stellvertretend für alle ManagedBeans gezeigt, wie der Eintrag dazu aussieht. Folgendes Codefragment zeigt die Deklaration der ManagedBean ProduktMB.

```
<managed-bean>
        <managed-bean-name>LoginMB</managed-bean-name>
        <managed-bean-class>managedBeans.LoginMB</managed-bean-class>
        <managed-bean-scope>session</managed-bean-scope>
        <managed-property>
                <property-name>textUser</property-name>
                <value>nicht eingeloggt</value>
        </managed-property>
        <managed-property>
                <property-name>eingeloggt</property-name>
                <value>false</value>
        </managed-property>
</managed-bean>
```

Über den Namen LoginMB kann die ManagedBean von der Anwendung angesprochen werden. Die Bezeichnung der genauen Klassenangabe managedBeans.LoginMB erfolgt in der nächsten Zeile. Anschließend wird Gültigkeitsdauer der ManagedBean als Session angegeben, wodurch für jede Session ein ManagedBean zur Verfügung steht.

Zusätzlich zu dieser Standarddeklaration werden die Eigenschaften *textUser* und *eingeloggt* mit Standardwerten versehen.

Nachdem alle ManagedBeans deklrariert worden sind, müssen alle notwendigen Navigationsregeln aufgeführt werden, die notwendig sind, um den Benutzer an die jeweils nächste JSP-Seite zu lenken. Es werden 5 globale Navigationsregeln festgelegt, die unabhängig einer JSP-Seite in Aktion treten können und die Navigationsleiste der Anwendung widerspiegeln. Die restlichen Navigationsregeln beziehen sich auf bestimmte JSP-Seiten und werden nachfolgend anhand der Navigationsregel zur Warenkorbseite exemplarisch gezeigt. Folgendes Codefragment zeigt die Deklaration dazu.

```
<navigation-rule>
        <from-view-id>/warenkorb.jsp</from-view-id>
        <navigation-case>
                <from-outcome>removedWare</from-outcome>
                <to-view-id>/warenkorb.jsp</to-view-id>
        </navigation-case>
        <navigation-case>
                <from-outcome>go_bestellung</from-outcome>
                <to-view-id>/userdaten.jsp</to-view-id>
        </navigation-case>
</navigation-rule>
```

Zuerst muss die JSP-Seite angegeben werden, für die die Navigationsregel gilt. Anschließend werden 2 Navigationsfälle bestimmt, die jeweils bestimmen, welche Seite aufgerufen werden soll, falls der richtige Rückgabewert vorliegt. Auf der Warenkorbseite gibt es die Möglichkeit, Produkte aus dem Warenkorb zu nehmen oder die Bestellung einzuleiten. Dieser Sachverhalt spiegelt sich hier wieder und die dazugehörigen Links liefern die entsprechenden Rückgabewerte *removedWare* und *go_bestellung*.

Bis auf die globalen Navigationsregeln sind alle anderen in der nachfolgenden Grafik visualisiert und zeigen den kompletten Navigationsfluss der Anwendung.

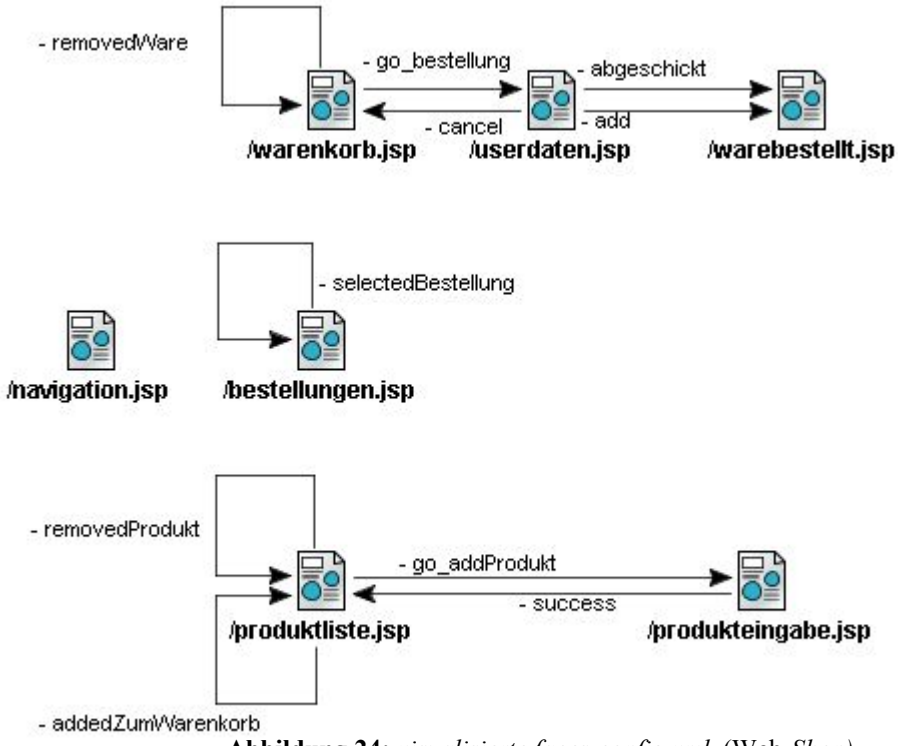


Abbildung 24: *visualisierte faces-config.xml* (Web-*Shop)*

4.6.7 Deployment als WAR-File

Um die erstellte Anwendung über einen Servlet-Container verfügbar zu machen, muss eine bestimmte Struktur der Anwendungsdateien eingehalten werden. Die Struktur wird durch folgende Grafik gezeigt.

Abbildung 25: *Verzeichnisstruktur* (Web-*Shop)*

Im css-Ordner befinden sich die CSS-Dateien, die für die Formatierung der HTML-Seiten benötigt werden. Im images-Ordner Bilder der Anwendung. Das Root-Verzeichnis enthält auch alle JSP-Seiten. Der WEB-INF-Ordner bildet das wichtigste Verzeichnis, da es sowohl alle notwendigen Klassen, als auch die für die Strtus-Anwendung benötigten Libraries enthält. Nachfolgende Grafik zeigt die genaue Struktur des WEB-INF-Ordners.

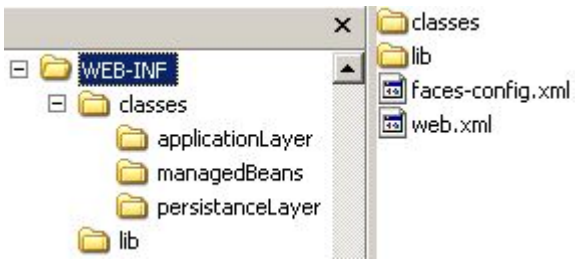


Abbildung 26: *WEB-INF-Inhalt* (Web-*Shop)*

Wie zu sehen ist, müssen sich außerdem die Konfigurationsdateien faces-config.xml und web.xml in dem WEB-INF-Ordner befinden.

Anschließend kann aus dieser festgelegten Verzeichnisstruktur, die alle notwendigen Dateien enthält, eine WAR-Datei erzeugt werden. Das Web Applikation Resource (WAR)-Format ist speziell für Web-Applikationen gedacht und setzt die oben beschriebene Verzeichnisstruktur voraus. Das WAR-File enthält wie ein JAR-File die gewünschten Dateien in komprimierter Form und kann so in jedem Servlet-Container in das richtige Verzeichnis kopiert werden und ist dann als Web-Applikation durch Eingabe der ihr zugeordneten URL erreichbar, die in der web.xml eingetragen wird.

Folgende Codefragmente der web.xml erläutent diesen Aspekt.

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

Hier wird der Name des Servlet angegeben und mit dem Servlet von JavaServer Faces verknüpft.

```
<servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>/faces/*</url-pattern>
</servlet-mapping>
```

Anschliessend wird noch eine URL angegeben werden, unter der die Anwendung erreichbar ist.

4.8 Fazit

JavaServer Faces ist ein aufstrebendes Web-Framework, welches durch die vielen bereitgestellten Funktionalitäten für viele Probleme, die während der Entwicklung einer Web-Applikation beachtet werden müssen, Unterstützung bietet.

Da es erst seit 2 Jahren verfügbar ist, ist es noch nicht ganz frei von Bugs. Dennoch ist davon auszugehen, dass dies in nächster Zeit behoben wird. Allerdings lassen sich damit

auch jetzt schon professionelle Anwendungen erstellen, die die Anforderungen an moderne Web-Applikationen erfüllen.

Da mittlerweile auch viele Implementierungen und Entwicklungstools für JavaServer Faces verfügbar sind, kann eine Web-Applikation in kurzer Zeit entwickelt werden. Durch das Angebot an vorgefertigten Komponenten vieler Hersteller ist man in der Lage das Grundangebot der Referenzimplementierung von SUN zu erweitern und kann eine Web-Applikation erstellt werden, die fast einer normalen Standalone-Anwendung gleicht.

Bei JavaServer Faces handelt es sich um ein professionelles Web-Framework, welches sich in den nächsten Jahren behaupten wird und bei der Entwicklung von Web-Applikationen als Web-Framework in Betracht gezogen werden sollte.

5. Struts

In diesem Kapitel rückt Struts als das zweite von 2 Web-Frameworks für die Präsentationsschicht in das Blickfeld und wird im Anschluss daran anhand einer Beispielapplikation auf Praxistauglichkeit überprüft.

5.1 Einleitung

Struts ist momentan sicherlich eines der beliebtesten Web-Frameworks, das wie auch JavaServer Faces auf dem MVC-Prinzip basiert. Da es als OpenSource-Software vertrieben wird, konnte sich das Framework schnell unter den Enwicklern ausbreiten und kann derzeit als „De facto-Standard“ für Java-Web-Anwendungen angesehen werden[74].

5.2 Architektur von Struts

Struts ist in der Präsentationsschicht einer Web-Anwendung angesiedelt, und bietet durch sein einfaches und schlankes Design, 3 wesentliche Komponenten, die den Kern einer Struts-Anwendung ausmachen.

- *Controller:* Ein Front-Controller, realisiert durch das *ActionServlet,* leitet alle Anfragen weiter zu den Views, welche beispielsweise durch JSP-Seiten oder Velocity-Pages umgesetzt werden können.
- *FormBeans:* Über Formulare eingegebene Daten werden hier zwischengespeichert und anschließend an die Aktionen weitergeleitet.

74 [WES01] Seite:16; Der Autor dieses Buches teilt diese Auffassung und spricht auch von einem momentanen „De Facto Standard“ im Bereich der Web-Frameworks.

- *Aktionen:* Diese bilden den Dreh- und Angelpunkt einer Struts-Anwendung, da sie bei jedem eintreffenden Request angesteuert werden und verantwortlich für die Folgeverarbeitung und Weiterleitung auf andere Seiten sind.

5.2.1 ActionServlet als Controller

Das Herzstück einer jeden Struts-Applikation stellt das ActionServlet dar. Alle HTTP-Anfragen eines Browsers, die von Struts verarbeitet werden sollen, durchlaufen das ActionServlet, das durch Angaben in der Konfigurationsdatei die jeweiligen weiteren Aktionen ausführt. Anhand eines Sequenzdiagrammes[75] soll der Ablauf eines Requests veranschaulicht werden.

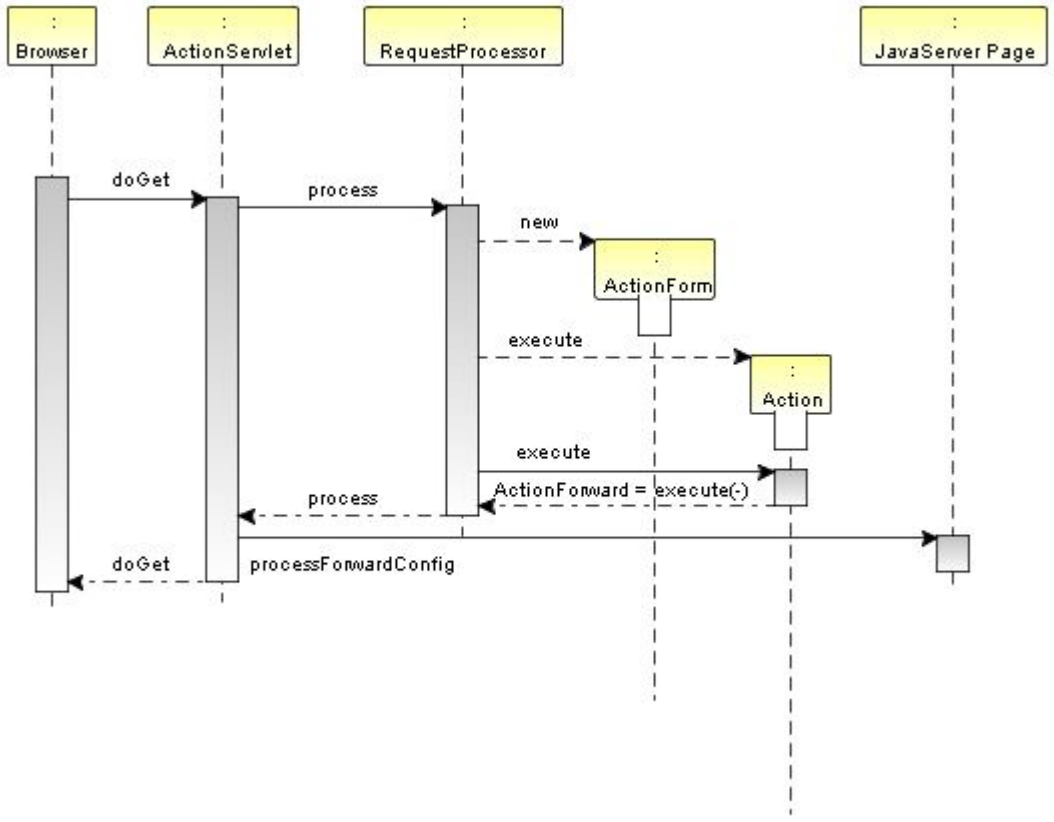


Abbildung 27: *Ablauf eines Requests* [76]

75 Sequenzdiagramme zeigen den Informationsaustausch zwischen Kommunikationspartnern und erlauben die Modellierung von festen Reihenfolgen, zeitlichen und logischen Ablaufbedingungen.

76 [WES01] Seite 54

Nachdem ein Client-Request an das ActionServlet gerichtet worden ist, wird die Methode process() aufgerufen. Die Methode process() beschafft sich ein Objekt der Klasse RequestProcessor und ruft dessen process()-Methode auf. Der RequestProcessor erstellt eine ActionForm und bildet die eingegebenen Formulardaten darauf ab. Dann wird die Methode process() der Klasse Action aufgerufen, in der der Entwickler die Formulardaten auswerten kann und diese gegebenenfalls an die Geschäftsschicht der Anwendung weiterreichen kann, in der sich die Geschäftslogik der Applikation befindet. Nachdem all diese Schritte vollzogen sind, wird die nachfolgende JSP-Seite angezeigt. Der RequestProcessor erhält alle Informationen, die er für den Workflow braucht, aus der Konfigurationsdatei, in der z.B. die Pfade für die jeweiligen FormBeans und Actions hinterlegt werden müssen.

Da der RequestProcessor die Hauptarbeit während der Laufzeit einer Struts-Anwendung ausübt, soll er nun im Detail anhand eines Sequenzdiagrammes vorgestellt werden, welcher auf nächster Seite abgebildet ist.

Zunächst wird ein ActionForm-Objekt erzeugt und die Methoden reset() und validate() dieses Objektes aufgerufen, in der Standardwerte für die ActionForm vergeben werden können und eine Validierung der Werte der ActionForm erfolgt.

Im Anschluss daran wird die zugehörige Action erstellt und dessen execute()-Methode aufgerufen, die für die Folgeverarbeitung verantwortlich ist.

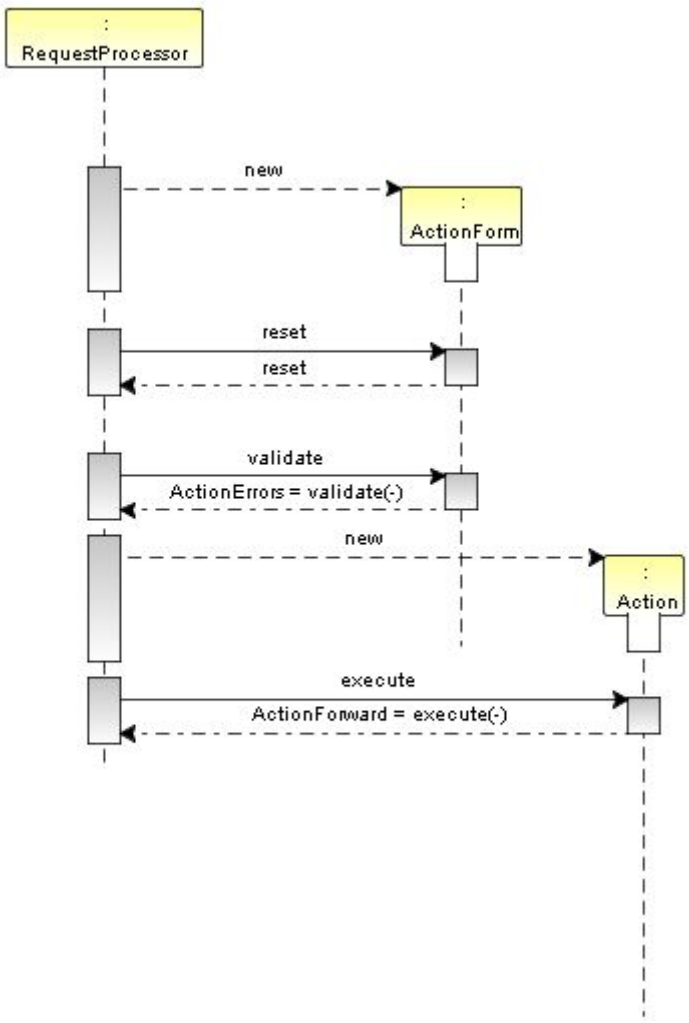


Abbildung 28: *Ablauf eines Requests im Detail* [77]

5.2.2 FormBeans

Die FormBeans einer Struts-Anwendung können als Helfer der View angesehen werden, da dort eingegebene Daten auf die FormBean gemappt werden. Jede FormBean wird von der abstrakten Klasse org.apache.Struts.action.ActionForm abgeleitet und bietet 2 Standardmethoden, die der Validierung und Initialisierung der FormBean-Eigenschaften dienen. Ist die Verarbeitung einer FormBean abgeschlossen, wird diese an die zugehörige Aktions-Klasse übergeben.

77 [WES01] Seite 57

2 Standardmethoden stehen zur Verfügung:

- *reset():* diese Methode der ActionForm dient der Belegung von Standardwerten der Eigenschaften
- *validate():* diese Methode dient der Überprüfung von Benutzereingaben

5.2.3 DynaActionForms

Da FormBeans hauptsächlich als Datenspeicher für Formulareingaben genutzt werden und keine echte Geschäftslogik beinhalten, besteht die Möglichkeit , FormBeans als sogenannte DynaActionForms direkt in die struts-config.xml einzutragen.

Dies eröffnet die Möglichkeit einfache Datenspeicherobjekte zu erstellen, ohne eigene Klassen zu erstellen und sie können als abgespeckte Version einer ActionForm angesehen werden.

Beispiel:

```
<form-bean name="userForm" type="org.apache.struts.actin.DynaActionForm">
        <form-property name="vorname" type="java.lang.String"/>
        <form-property name="nachname" type="java.lang.String"/>
        <form-property name="land" type="java.lnag.String" initial="Deutschland"/>
</form-bean>
```

5.2.4 Aktionen

Die Aktionsklassen bilden das Herz einer jeden Struts-Anwendung. Hier kann auf Eigenschaften der FormBeans und auf HTTP-Parameter zugegriffen werden, um diese im Anschluss an die Geschäftsschicht weiterzuleiten, die z.B. eingegebene Daten in eine

Datenbank einträgt. Die Verknüpfung mit der Eingabe-JSP-Seite, der dazugehörigen FormBean und die bei Erfolg anzusteuernde Ausgabe-JSP-Seite wird für jede Action in der struts-config.xml eingetragen.

5.2.5 Fehlerbehandlung

Zusätzlich zu der standardmäßigen Fehlerbehandlung von Java bietet Struts ein integriertes Fehlerbehandlungsmanagement an.

Der Ablauf stellt sich wie folgt dar:

1. Über die Klasse *ActionError* wird ein neuer Fehler erstellt, der eine einzelne Fehlernachricht beinhaltet. Diese besteht aus bis zu 5 keys, die im *appliaction.resource.bundle* eingetragen sein müssen.
2. Hinzufügen der Klasse ActionError zur Collection-Klasse *ActionErrors.*
3. Falls Fehler aufgetreten sind, können diese nun z.B. in der View über das Tag *<html:errors/>* angezeigt werden.

5.2.6 Validierung

Das Validierungssystem von Struts verdient besondere Beachtung, da es standardmäßig Routineüberprüfungen, wie z.B. die Formatprüfung von E-Mail-Adressen und Kreditkartennummern, durchführen kann. Es wurde eine eigene FormBean, die ValidatorBean, entwickelt, in der nun im Vergleich zur ActionForm die validate()-Methode nicht mehr überschrieben werden sollte, da die Validierung extern durch die Dateien *validation.xml* und *validator-rules.xml* vorgenommen wird.

In der validator-rules.xml befinden sich alle vorgefertigten Regeln, die viele wichtige Bereiche abdecken und der Struts-Anwendung automatisch zur Verfügung stehen. Werden eigene Validatoren entwickelt, müssen diese in dieser Datei eingetragen werden.

Für den Entwickler von größerer Bedeutung ist die validation.xml. Diese muss angelegt werden und beinhaltet alle Validierungsregeln, die benutzt werden sollen.

Um das Validierungsframework von Struts benutzen zu können, muss es über ein Plugin in die struts-config.xml eingebunden werden.

5.2.7 Internationalisierung

Da das Prinzip der Internationalisierung von Struts dem von JavaServer Faces gleicht, soll hier auf eine ausführlichere Darstellung verzichtet werden.

Um eine Web-Anwendung zu lokalisieren sind folgende Schritte notwendig:

1. Erstellen der landesspezifischen Resource Bundles.
2. Eintrag der Resource Bundles in die struts-config.xml.
3. Zugriff über JSP-Tags in der View.

5.2.8 Struts-Taglibs

Sicherlich eines der Kernstücke des Struts-Frameworks sind die zahlreichen JSP-Tags, die auf der JSP-Seite eingebunden werden können, um unterschiedliche Fragestellungen zu lösen. Die Tags lassen sich in 4 Pakete gliedern, welche im Folgenden beschrieben werden.

Html-Taglib:

Alle notwendigen HTML-Tags für die Formularerstellung finden sich hier wieder in der Form spezifischer HTML-Struts-Tags, die optimal auf die FormBeans abgestimmt sind.

Neben den herkömmlichen Elemente, die auch zum normalen HTML-Befehlssatz gehören, bietet Struts einige zusätzliche nützliche HTML-Tags an, von denen nachfolgend 2 beschrieben werden.

- **<html:file property="datei"/>:** nützlich für den File-Upload auf den entfernten Server
- **<html:errors/>:** Treten Fehler im Rahmen der Validierung auf, lassen sich diese hiermit anzeigen.

Bean-Taglib:

Diese Tags erleichtern den Umgang mit JavaBeans, einzubindenden Nachrichten und HTTP-Anfragen. Im Gegensatz zu den normalen JSP-Tags ,wie z.B. *<jsp:useBean>,* ist bei Gebrauch von BeanTags nur ein lesender Zugriff möglich.

Um Web-Anwendungen mehrsprachig zu gestalten, können Texte aus der Internationalisierungsdatei gelesen werden und über folgendes Tag eingebunden werden: *<bean:message key="name des resource bundles.schlüssel der Zeile">*

Logic-Taglib:

Durch die Logic-Taglib von Struts werden der View Objekte zugänglich gemacht, die es ermöglichen, relationale Operatoren, logische Operatoren zu nutzen. Auch der Verarbeitung von Datenstrukturen wurden einige Tags gewidmet.

- *Überprüfungen:* Über reguläre Ausdrucke kann der Inhalt beispielsweise von HTTP-Requests oder Cookies auf bestimmte Zeichenketten untersucht werden.
- *relationale Operatoren:* Ein Satz von Vergleichsoperatoren, wie sie in jeder Programmiersprache vorkommen, werden auch durch Struts für die View

bereitgestellt.

- Verarbeitung von Datenstrukturen: Struts bietet eine kleine Anzahl an Tags an, mit deren Hilfe Datenstrukturen in Form von Listen, Arrays und ähnliches durchlaufen werden können.
-

5.2.9 Tiles

Ein sehr nützliches Paket, das ab der Struts-Version 1.1 eingeführt wurde, um auf das Layout der JSP-Seiten Einlfuss zu nehmen. In herkömmlichen HTML-Seiten gibt es die Möglichkeit mithilfe von Frames mehrere Seiten zu einer Seite zusammenzufügen und so für ein einheitliches Layout aller darzustellenden Seiten zu sorgen. Dadurch, dass Suchmaschinen einzelne Seiten indexieren, trifft man auf das Problem, dass die Seite dann einzeln angezeigt wird, wodurch wichtige Teile der Seite fehlen können.

Probleme dieser Art können durch Tiles gelöst werden, indem ein Layout für die Seite als sogenanntes Seitenschema auf jeder anzuzeigenden Seite eingebunden werden kann und dadurch eine korrekte Darstellung aller eingebundenen Seiten erfolgen kann.

5.2.10 Konfigurationsdatei

Die Konfigurationsdatei von Struts wird als struts-config.xml bezeichnet und lässt eine Konfigurierung weiter Teile der Anwendung zu, die über XML-Tags in die Datei eingetragen werden.

Folgende Konfigurationseinstellungen müssen mindestens angegeben werden:

- *Konfiguration von FormBeans:* alle verwendeten FormBeans müssen eingetragen werden.
- *Aktionsklassen:* Alle Actions müssen eingetragen werden. Hier erfolgt auch die

Zuordnung zu einer FormBean.

- *Weiterleitungen:* über sogenannte Global Forwards können Navigationspfade angegeben werden, die für die ganze Anwendung Gültigkeit besitzen.

5.3 Erweiterungen über PlugIns

Um das Rahmenwerk zu erweitern gibt es die Möglichkeit durch PlugIns und deren Eintrag in die struts-config.xml eine Anwendung durch neue Funktionalitäten zu bereichern. Beispielsweise können ab der Struts-Version 1.1 die Template-Engine Tiles und auch das Validierungsrahmenwerk über PlugIns eingebunden werden. Will man ein eigenes PlugIn erstellen, muss in der erstellten Klasse die Schnittstelle PlugIn implementiert werden und ein Eintrag in die struts-config.xml erfolgen. Danach steht die Funktionalität der Klasse zur Verfügung.

5.4 Nebenprodukte im Zusammenhang von Struts

Die nachstehenden Ausführungen sind auf Grundlage des Buches von Harshad Oak[78] entstanden, der die einzelnen Bibliotheken von Jakarta Commons anschaulich beschreibt und diese in vielen Praxisbeispielen erläutert.

5.4.1 Jakarta Commons

Viele Komponenten, die ursprünglich nur für Struts entwickelt worden sind, wurden im Laufe der Zeit ausgelagert und bei den Jakarta Commons[79] abgelegt. Die Komponenten, die man hier findet, sind für allgemeinere Zwecke bestimmt und müssen nicht nur im Zusammenhang mit Struts benutzt werden. Die Softwarebibliothek, die dabei entstanden ist, ist ein beeindruckendes Zeugnis der Struts-Entwicklergemeinde. Der Zweck dieser

78 [OAK01]
79 http://jakarta.apache.org/commons

Bibliothek besteht darin, bestimmte Schwächen der Java-Standard-Implementierungen auszugleichen und bestimmte API's auf höherer Ebene zu kapseln.

Diese Bibliothek ist bei Entwicklern sehr beliebt und eine Unterstützung durch Tool-Hersteller bereits bemerkbar. Viele Frameworks, wie z.B. Hibernate, JavaServer Faces oder MyFaces sind auf Basis dieser Bibliotheken entstanden bzw. haben diese integriert.

Die wichtigsten Pakete sind:

- *Lang Component:* Viele nützliche Klassen, beispielsweise für die Verarbeitung von Zeichenketten, für den Umgang mit der Java Input/Output oder für die Konvertierung zwischen Datentypen, werden hier zusammengefasst.
- *Logging Component:* Diese Komponente ermöglicht einen allgemeinen Zugriff auf andere Logging-API's, wie Log4j, Avalon LogKit und JDK Logging. Sollte man diese Komponente benutzen, sollte beachtet werden, dass dann wirklich nur die Mechanismen benutzt werden können, die allen Log-API's gemeinsam ist.
- *Validator Component:* Diese Komponente kann in allen Applikationen benutzt werden, in der Validierung erforderlich ist, wodurch auch andere Applikationen in den Genuss von komfortabler Validierung kommen können.
- *BeanUtils Component:* Die enthaltenen Klassen vereinfachen den Umgang mit JavaBeans und sind sicherlich eine der nützlichsten Klassen des Commons-Packets.
- *Digester Component:* Der Umgagng mit XML stellt ein nicht leichtes Unterfangen dar, wenn man z.B. API's wie DOM und SAX benutzt. Abhilfe schafft die Digester Komponente, mit der mühelos XML-Dateien geparst und auf Klassen abgebildet werden können. Die Komplexitätsverringerung mit einhergehender Zeitersparnis machen diese Komponente zum „Pflichtpaket“ für jeden Java-Entwickler.
- *Collection Component:* Das Java Collection Framework, welches seit der JDKVersion 1.2 existiert, wird durch diese Komponente um weitere Datenstrukturen und Zugriffsmethoden erweitert.
- *FileUpload:* Wie der Name schon andeutet kann damit der File-Upload und das damit zusammenhängende Management zu entfernten Server vereinfacht werden.

5.5 Entwicklungswerkzeuge und Tools

Da Struts ein sehr etablierte Web-Framework zur Erstellung von Web-Applikationen, gibt es eine umfangreiche Menge an Tools, die das Arbeiten mit Struts vereinfachen. Hier soll eine kleine Auswahl an Tools vorgestellt werden, unterteilt in kostenlose und kostenpflichtige Tools. Am Ende des Kapitels folgt eine kritische Würdigung der vorgestellten Tools im Hinblick auf einen vernünftigen Einsatz und die Integration in eine bestehende Programmierumgebung.

Die Beschreibungen der Open Source Tools sind in Anlehnung an einen Artikel der Zeitschrift Javamagazin[80] entstanden.

5.5.1 Open Source- und kostenlose Tools

Struts Console:

Ähnlich wie die JavaServer Console, kann die struts-config.xml, als auch die Konfigurationsdateien für die Validatoren und für die Tiles, editiert werden. Besonders hilfreich ist die Funktion, HTML-Dateien in JSP-Files umzuwandeln.Wie auch bei der JavaServer Console lässt sich dieses Tool in fast jede IDE integrieren.

Struts Builder:

Es kann nur die struts-config.xml bearbeitet werden, allerdings bietet dieses Tool zusätzlich die Möglichkeit einige Code-Generierungs-Funktionen abzurufen, wie z.B. das Erstellen von ActionForms und Actions.

Easy Struts:

Hierbei handelt es sich um ein weiteres OpenSource-Projekt, welches sich sowohl in Eclipse als auch im JBuilder von Borland integrieren lässt. Es werden nützliche Schritt für Schritt Helfer für die Erstellung von Actions und ActionForms angeboten.

80 [JAV01]

5.5.2 kommerzielle Tools

Allen kommerziellen Tools ist gemeinsam, dass neben einer umfangreiche visuellen Entwicklungsumgebung unterschiedlichste Wizards bereitgestellt werden, um den Entwicklungsprozess von Struts-Anwendungen zu beschleunigen. Features, wie das Visualisieren der struts-config.xml, Code-Highlighting und das Einbinden der Struts-Tags über Drag&Drop gehören fast immer zur Grundausstattung. In Bezug auf Software-Ergonomie und Qualtität heben sich diese Tools deutlich von den kostenlosen- bzw. Open Source-Tools ab.

MyEclipse:

Das kostenpflichtige Plugin für Eclipse, bietet unter anderem auch Unterstützung für Struts der Versionen 1.0 – 1.2. Die struts-config.xml wird graphisch aufbereitet und kann visuell bearbeitet werden. MyEclipse bietet eine hervorragende Unterstützung beim Deployment der erstellten Anwendung für alle gängigen Web-Container.

Struts Studio:

Eine umfassende visuelle Entwicklungsumgebung aus dem Hause Exadel Inc., welches alle wichtigen Features mitbringt, die für die schnelle Entwicklung von Struts-Applikationen notwendig sind. Neben der Visualisierung der struts-config.xml, der Integration von Wizards für die Erstellung der ActionForms und Actions, lassen sich sämtliche Struts-Tags per Drag&Drop auf die JSP-Seite ziehen. Es steht sowohl eine Community, als auch eine Professional Version zum Download bereit. Die Professional Version bietet neben der Grundausstattung noch zusätzlich Unterstützung für den Gebrauch von Tiles und dem Validator-Framework von Struts.

JDeveloper(10.1.3 Developer Preview Edition):

Der JDeveloper bietet eine perfekte Tool-Unterstützung für das Arbeiten mit Struts. Man kann ein Struts-Projekt erstellen, in dem alle notwendigen Struts-Bibliotheken importiert werden. Gleichzeitig wird die web.xml und die struts-config.xml angelegt.

Über einen visuellen Editor, der die struts-config.xml als Grafik darstellt, können die notwendigen Struts-Komponenten eingefügt werden und bieten durch integrierte Wizzards eine gute Unterstützung. Für das Editieren der JSP-Seiten stehen alle Struts-Tag-Libs zur Verfügung und können über die Toolleiste auf die Seite gezogen werden und anschliessend im Eigenschaftsfenster bearbeitet werden.

5.6 Benutzerverwaltung als Beispielapplikation

Der theoretische Teil, in dem alle wichtigen Eigenschaften des Struts-Frameworks beschrieben wurden, soll nun durch ein praktisches Beispiel veranschaulicht werden. Es wird eine Benutzerverwaltung als Web-Applikation erstellt, die die wesentlichen Komponenten von Struts benutzt und das Zusammenspiel dieser einzelnen Teile zeigen soll.

Auf eine Darstellung sämtlicher Codelistings wurde verzichtet. Sämtliche Codelistings, wie die von dem Web-Shop befinden sich aber auf der beigelegten CD und sind dort vollständig bis auf die JSP-Seiten kommentiert. Ferner wird im Anhang beschrieben, wie sich beide Beispiele in den JDeveloper laden lassen und wie sie dann auf den Tomcat-Server installiert werden können. Hier werden nur Codeauszüge herangezogen, um bestimmte implementierte Funktionen zu zeigen. Für die Dokumentation und Beschreibung der Beispielapplikation wird auf UML-Diagramme zurückgegriffen, um einzelne Klassen zu dokumentieren.

Die Bücher von Matthias Weßendorf[81], Bill Siggelkow[82], Ted Husted[83] und James Goodwill[84] bieten viele praxistaugliche Anregungen und waren während der Entwicklung der Beispielapplikation eine gern aufgesuchte Informationsstätte.

81 [WES01]
82 [SIG01]
83 [HUS01]
84 [GOO01]

Folgende Funktionalitäten werden durch die Beispielapplikation verwirklicht:

- Es werden JSP-Seiten für die View benutzt.
- ActionForms unterstützen die JSP-Seiten und dienen ihr als Formulardatenspeicher
- Actions ermöglichen die Auswertung der Formulare und stellen eine Verbindung zur Geschäfts- und Datenbankschicht her.
- Die Navigation wird in der struts-config.xml festgelegt.
- Das Fehlerbehandlungsmodell von Struts wird zur Überprüfung der Formulareingaben genutzt.
- Internationalisierung der Anwendung. Die Anwendung wird je nach Spracheinstellungen des Users auf Deutsch oder Englisch angezeigt.
- Ein Zugriff auf die Anwendung ohne Login wird durch den Verweis auf die Login-Seite quittiert.
- Als Datenspeicher wird eine Datenbank angebunden.

Nachfolgend werden die einzelnen Programmteile beschrieben, visualisiert durch UML-Diagramme. Als Entwicklungsumgebung wurde der JDeveloper von Oracle benutzt, mit dem sowohl alle notwendigen Programmteile, alle UML-Diagramme , als auch die Datenbankanbindung realisiert wurde. Als Datenbank wird Oracle 10g und als Servlet-Container Tomacat eingesetzt, wobei sich das Programm auch intern über den JDeveloper starten lässt.

5.6.1 Use-Cases

Durch ein Use-Case-Diagramm sollen die Anforderungen an die Benutzerverwaltung grafisch dargestellt werden. Durch Benutzen eines Use-Case-Diagramms lässt sich schnell aufzeigen, welche Funktionaliäten das Programm bieten soll. Es eignet sich hervorragend für die Kommunikation innerhalb des Teams, als auch zur

Kommunikation mit möglichen Auftraggebern.

Abbildung 29: *Use Case (Benutzerverwaltung)*

Die Benutzerverwaltung wird so konzipiert, dass nach erfolgreichem Login eine Liste aller angelegten Benutzer angezeigt wird, die sich dann editieren lässt. Es können neue Benutzer angelegt, vorhandene Benutzerdaten verändert und schon vorhandene Benutzer gelöscht werden. Um die Daten permanent zu speichern wird eine Datenbank angebunden.

5.6.2 Geschäfts- und Persistenzschicht

Wie im Kapitel zur Web-Architektur aufgezeigt, soll auch die Beispielanwendung auf mehrere Schichten aufgeteilt werden. Das Geschäftsmodell und die Datenbankanbindung werden durch 3 Klassen realisiert, welche im folgenden UML-Klassendiagramm visualisiert werden.

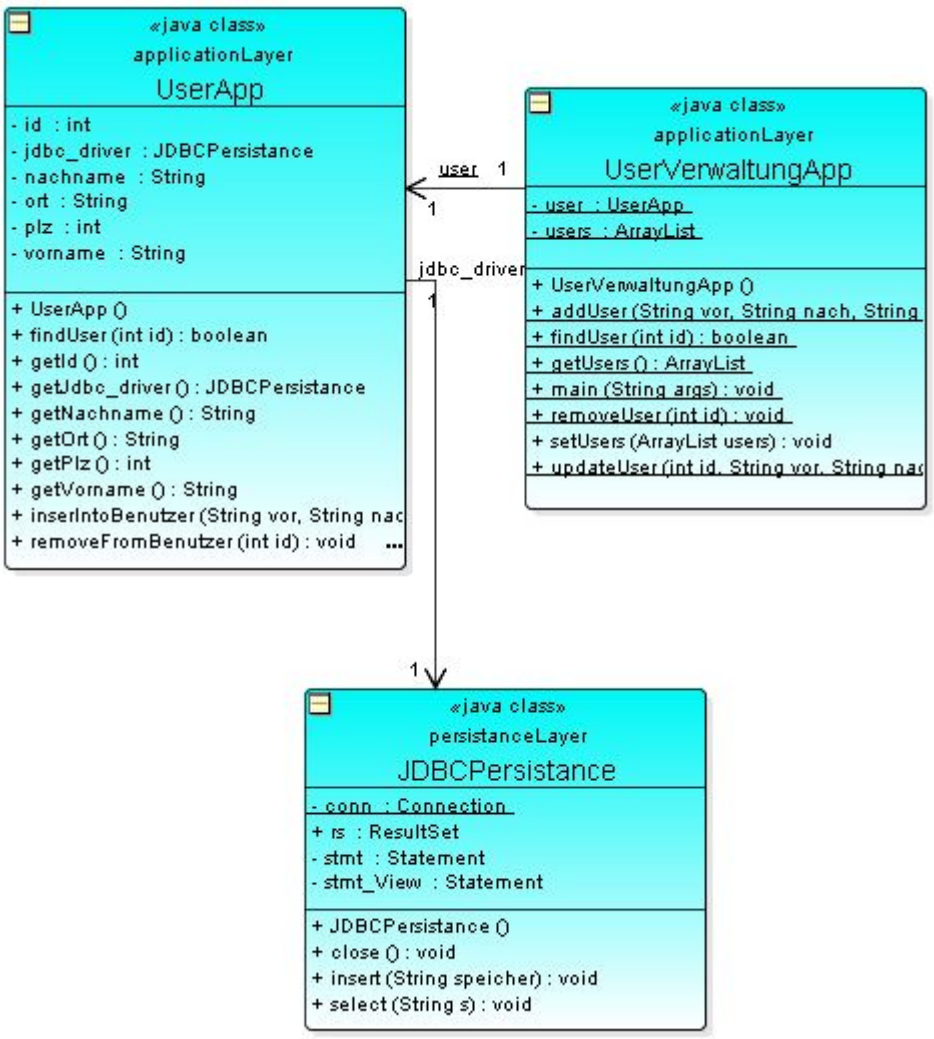


Abbildung 30: *Geschäfts- und Persistenzschicht (Benutzerverwaltung)*

Wie das Klassendiagramm veranschaulicht, befindet sich auf unterster Ebene, der Datenbankschicht, die Klasse JDBCPersistance, mit der die Datenbank angesprochen werden kann. Eine Ebene höher befindet sich die Klasse UserApp, welche dazu dient einen Benutzer mit seinen Eigenschaften zu beschreiben und über eine Referenz der Klasse JDBCPersistance die für den Benutzer nötigen Datenbankfunktionen auszuführen. Über die Klasse UserVerwaltungApp werden die Benutzer verwaltet und sie dient als einziger Ansprechpartner für die Präsentationsschicht. Durch diese Aufteilung wurde zusätzlich ein einseitiger Kommunikationsfluss festgelegt, durch den gewährleistet wird, dass nur eine Klasse höherer Ebene auf eine Klasse niedrigerer

Ebene zugreifen darf.

Durch diese Schichtung ist eine problemlose Erweiterung denkbar. Auf der Ebene der Klasse UserApp können weitere Klassen untergebracht werden, die wie die Klasse UserApp die Klasse JDBCPersistance für den Zugriff auf die Datenbank benutzt. Werden weitere Klassen eingefügt, kann auch die Klasse UserVerwaltungApp wiederverwendet werden, ergänzt durch Zugriffmethoden auf die neu erstellten Klassen.

Die einzelnen Klassen werden jetzt genauer vorgestellt.

JDBCPersistance:

Die Datenbankanbindung erfolgt durch die Klasse JDBCPersistance, die in der Lage ist verschiedene benötigte Datenbankfunktionen auszuführen. Neben der Anmeldung bei einer Datenbank wurden folgende Methoden implementiert, die allerdings nicht alle für die Anwendung gebraucht werden:

- *JDBCPersistance():* innerhalb des Konstruktors wird eine Verbindung zur Datenbank hergestellt.
- *close():* Schliessen der Datenbankverbindung.
- *insert():* Absetzen von SQL-Befehlen.
- *select():* Abfragen an die Datenbank.

UserApp:

In der Klasse UserApp wird ein einzelner Benutzer beschrieben. Als Eigenschaften sind ID, Name, Vorname, Ort, Strasse und PLZ vorgesehen. Die Klasse JDBCPersistance wird von der UserApp-Klasse benutzt um folgende Use-Cases umzusetzen:

- *insertIntoBenutzer():* neue Benutzer können angelegt werden und vorhandene Benutzer nach Änderung upgedatet werden. Dies erfolgt durch Aufruf der Funktion

mit den passenden Parametern (vorname,nachname,ort,plz), aus denen dann der jeweils passende DML-Befehls zusammengesetzt wird und an die Klasse JDBCPersistance weitergereicht wird, die letztlich den Befehl in der Datenbank absetzt.

- *remove():* vorhandene Benutzer können gelöscht werden. Der Aufruf erfolgt mit dem Parameter Benutzer-ID, mit dem der passende DML-Befehl erzeugt wird, um anschliessend die Funktion select(DML-Befehl) der Klasse JDBCPersistance aufzurufen, die den Benutzer aus der Datenbank löscht.
- *updateUser():* vorhandene Benutzer können editiert werden, indem diese Funktion mit der Benutzer-ID aufgerufen werden, die dann ähnlich zu remove() und insertIntoBenutzer() den passenden DML-Befehl generiert und diesen an die Klasse JDBCPersistance weiterreichtet, die das Update in der Datenbank durchführt.

UserVerwaltungApp:

Die Klasse UserVerwaltungApp bildet die einzige Schnittstelle für Struts, um auf die Geschäfts- und Datenbankschicht zuzugreifen. Über diese Klasse werden alle notwendigen Funktionen zur Verwaltung des Geschätsmodells und der Datenbank gekapselt. Dadurch wird erreicht, dass die Präsentationsschicht komplett durch ein anderes Framework ersetzt werden kann. Ausserdem können so die Geschäfts- und Datenbankschicht unabhängig von der Präsentationsschicht getestet werden, was zu einer robusteren und besser wartbaren Anwendung führt.

Datenmodell der Datenbank:

Die Tabelle, die die Klasse UserApp in der Datenbank repräsentiert, wird durch folgendes Entity Relation Modell gezeigt.

Abbildung 31: *Datenbankmodell (Benutzerverwaltung)*

Es wurde bei Erstellung der Tabelle zusätzlich festgelegt, dass Vorname, Nachname, Ort und PLZ nicht Null sein dürfen. Das dazu passende SQL-Script zur Erzeugung der Tabelle ist auf beiligender CD enthalten.

5.6.3 JSP-Seiten

Da wie im vorherigen Kapitel beschrieben das Geschäftsmodell erstellt wurde, kann jetzt die Präsentationsschicht entworfen werden. Die Benutzerverwaltung besteht aus mehreren JSP-Seiten, die mithilfe nachfolgender Screenshots erläutert werden. Um nachfolgende JSP-Seiten durch unberechtigten Zugriff zu schützen, wird auf jeder JSP-Seite geprüft, ob der User sich bereits angemeldet hat. Ist dies nicht der Fall, erfolgt die Zurückleitung auf die Login-Page.

Login-Seite

Die Startseite wird als Login-Seite dargestellt, die den Einstiegspunkt in die Anwendung ergibt. Nur Benutzer, die sich mit dem richtigen Namen und dem dazugehörigen Passwort anmelden, bekommen Zugriff auf die Anwendung. Benutzer, die fehlende Eingaben tätigen, werden auf die Login-Seite zurückverwiesen und durch entsprechende Fehlermeldungen aufgeklärt warum der Login-Vorgang nicht erfolgreich war.

Abbildung 32+33+34: *Loginseiten (Benutzerverwaltung)*

Benutzerlistenseite

Wurden die richtigen Anmeldedaten eingegeben wird der User auf die nächste Seite geleitet, in der eine Auflistung aller angelegten Benutzer zu sehen ist. Auf dieser Seite werden alle Funktionalitäten bereitgestellt, die durch die Use-Cases vorgegeben wurden.

Die Funktionen, neuen Benutzer anlegen, vorhandene Benutzer löschen, vorhandene Benutzer editieren, sind durch Hyperlinks erreichbar. Die einzelnen Hyperlinks sind durch entsprechende Verknüpfungen mit den Actions von Struts verbunden. Auf dieser und den folgenden JSP-Seiten wird in der Navigationsleiste angezeigt, unter welchem Namen man sich eingeloggt hat und kann über den Link „Logout“ die Benutzersitzung beenden, worauf man auf die Login-Seite zurückgeleitet wird.

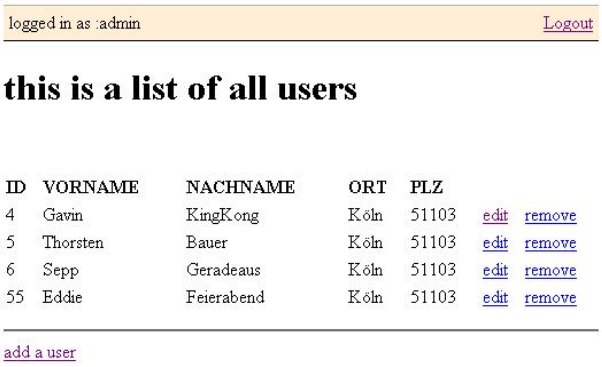


Abbildung 35: *Benutzerlisteseite (Benutzerverwaltung)*

Vor Anzeige dieser JSP-Seite wurde dem Servlet-Context ein Objekt zugefügt, welches alle Benutzer enthält. Diese Aufgabe wird durch eine Action übernommen, die in dem

Kapitel zu den Actions näher erläutert wird.

Initialisierung der Datenstruktur „users“

```
<logic:iterate id="user" name="users">
```

Das Objekt „users“ repräsentiert eine ArrayList, in der alle Benutzer abgelegt sind. Da dieses dem Servlet-Context zugefügt wurde, kann darauf über das Struts-Bean-Tag „logic:iterate“ zugegriffen werden. Dazu muss der entsprechende Name angegeben werden und die ID, unter der das Objekt dann angesprochen werden kann. Anschliessend kann die ganze Datenstruktur durchlaufen werden.

Ausgabe der einzelnen Benutzer

```
<bean:write name="user" property="id"/>
```

Mit dem Struts-Bean-Tag „bean:write“ wird auf die einzelnen Eigenschaften eines Elementes eines „users“ unter Angabe der property zugegriffen. In diesem Fall die ID eines Benutzers. In der Anwendung wird daher für jede Eigenschaft eine „bean:write“ Anweisung erzeugt.

Hyperlinks edit und remove

```
<a href="CheckEditUser.do?username=<bean:write name="user" property="id"/> ">
        <bean:message key="app.linkedit"/>
</a>
```

Hierbei handelt es sich um einen normalen Hyperlink, der den jeweiligen Namen und die zugehörige id als Dateianhang mitübergibt, anhand dessen dann in diesem Falle der Benutzer geändert werden kann.

Der Hyperlink remove funktioniert in gleicher Weise.

Seite um einen User zu erstellen oder einen vorhandenen User zu editieren

Wurde die Funktion neuen User anlegen ausgewählt, wird man auf folgende Seite gelenkt, die es ermöglicht, die Daten für den neuen Benutzer einzugeben. Die Eingaben werden durch den Submit-Button bestätigt und man wird auf Benutzerlisten-Seite

zurückgeleitet.

Abbildung 36: *Anlegen eines Benutzers (Benutzerverwaltung)*

vorhanden Benutzer editieren

Möchte man einen schon angelegten Benutzer editieren, wird man auf folgende Seite gelenkt, die den gleichen Screen verwendet, der auch benutzt wird, wenn ein neuer Benutzer angelegt wird. Die Felder des Formulars werden jetzt allerdings mit den Daten des angelegten Benutzers gefüllt und können hier verändert werden. Wurden die Daten aktualisiert wird man nach Drücken des Submit-Buttons auf die Benutzerlisten-Seite zurückgeleitet und kann die aktualisierte Version betrachten.

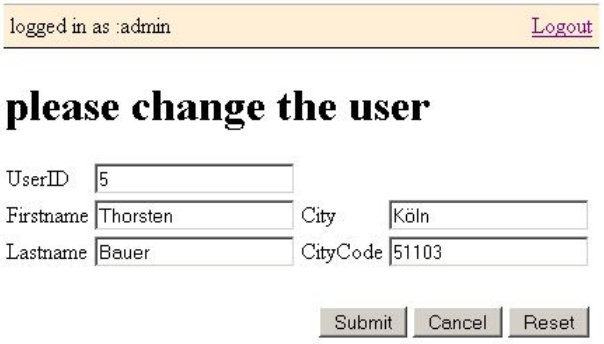


Abbildung 37: *Ändern eines Benutzers (Benutzerverwaltung)*

5.6.4 ActionForms

Jedem Formular wird in Struts ein Datenspeicher zugeordnet, der durch ActionForms realisiert wird und nachfolgend jeweils durch Klassendiagramme veranschaulicht wird.

LoginForm:

Das Formular für die Login-Seite wird durch die ActionForm *LoginForm* gestützt. Alle Eingabefelder werden durch entsprechende Variablen in der LoginForm abgebildet.

Die Eingaben, die gemacht werden, werden dann automatisch in die *LoginForm* geschrieben und sind anschliessend zugänglich für eine Auswertung, die durch die LoginAction vorgenommen wird.

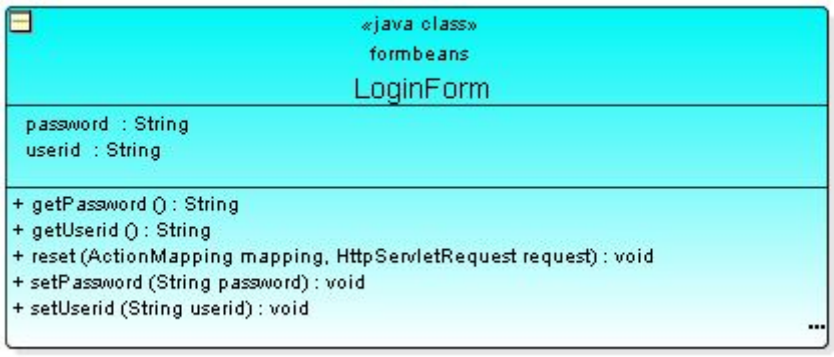


Abbildung 38: *LoginForm (Benutzerverwaltung)*

AddUserForm:

Das gleiche gilt für das Formular, das für das Anlegen eines Benutzers und das Editieren vorhandener Benutzer zuständig ist. Hier werden auch alle Eingabefelder als Variablen in der ActionForm *AddUserForm* deklariert.

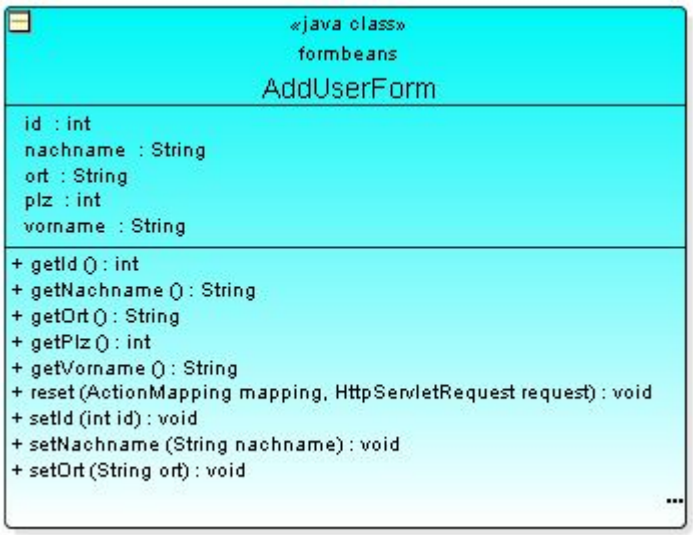


Abbildung 39: *AddUserForm (Benutzerverwaltung)*

5.6.5 Actions

Die Actions der Benutzerverwaltung bilden den „Klebstoff“, der alle bis jetzt beschriebenen Einheiten zusammenfügt und für die Verbindung zur Geschäfts- und Datenbankschicht sorgt. Insgesamt wurden 9 Actions erstellt, die nachfolgend beschrieben werden.

LoginAction:

Nachdem die Login-Seite aufgerufen wurde und die Felder für Name und Passwort ausgefüllt wurden, wird die LoginAction aufgerufen, die für die Validierung der eingegebenen Daten zuständig ist. Sind die Eingaben korrekt, wird man auf die Benutzerlistenseite gelenkt. Werden Felder unvollständig oder falsch ausgefüllt, landet man wieder auf der Login-Seite, in der die Fehler angezeigt werden und man kann

erneut die Formularfelder ausfüllen.

Überprüfung der Benutzerdaten

Über die der Action zugeordnete ActionForm „LoginForm“ werden die Formularfelder ausgelesen und es wird geprüft, ob man sich mit dem richtigen Namen und Passwort angemeldet hat. Sind die Eingaben korrekt wird dem Servlet-Context der Anmeldename zugefügt, der auf nachfolgenden JSP-Seiten angezeigt wird und gleichzeitig für die Sicherheit der JSP-Seiten eine Rolle spielt. Dieser Sachverhalt wird im Kapitel zur Sicherheit genauer erklärt. Anschliessend erfolgt eine Weiterleitung an die JSP-Seite „list“, die alle Benutzer anzeigt.

Dies wird durch folgendes Code-Fragment illustriert:

```
LoginForm loginForm = (LoginForm)form;
String userid = loginForm.getUserid();
String password = loginForm.getPassword();
if (userid.equals("admin") && password.equals("admin")){
        HttpSession session = request.getSession();
        session.setAttribute ("USER", userid);
        return mapping.findForward("list");
        }
```

Fehlerhafte Benutzerdaten

Sind die eingegebenen Benutzerdaten inkorrekt, wird über die Fehlerbehandlungs-mechanismen von Struts ein neuer ActionError erzeugt und es erfolgt eine Weiterleitung an die Login-Seite.

```
else    {
        ActionErrors errors = new ActionErrors();
        errors.add(ActionErrors.GLOBAL_ERROR,
        new ActionError("error.login.unknown", userid));
        saveErrors(request, errors);
        return mapping.findForward("login");
        }
```

UserList:

War der Login-Vorgang erfolgreich wird man zu der UserList-Action weitergeleitet. Diese sorgt für für ein Laden aller Benutzerdaten aus der Datenbank in eine Instanzvariable, die dann durch das Setzen der Servlet-Session-Variable der Anwendung während einer Benutzersitzung zur Verfügung gestellt wird.

Diese Action wird auch aufgerufen, wenn ein Benutzer gelöscht , ein neuer Benutzer angelegt oder ein vorhandener Benutzer editiert wurde.

```
ArrayList users = (ArrayList)session.getAttribute("users");
users = UserVerwaltungApp.getUsers();
session.setAttribute("users",users);
return mapping.findForward("showlist");
```

CheckAction:

Häufig trifft man auf das Problem, dass durch Betätigen des Aktualisierungsbuttons im Browser sämtliche Daten noch einmal an den Server geschickt werden. Wurde beispielsweise ein neuer Benutzer angelegt und man betätigt dann den Aktualisierungsbutton des Web-Browser während man auf der Userlist-Seite ist, würde der gleiche Benutzer noch einmal in die Datenbank geschrieben.

Um dieses Problem zu umgehen wird die CheckAction vor dem Anlegen eines neues Benutzers aufgerufen und durch Erstellen eines Tokens oben genanntes Problem gelöst wird. Diese Funktionalität ist bereits in Struts enthalten und kann daher leicht implementiert werden.

```
saveToken(request);
return mapping.findForward("checked");
```

AddAction:

Diese Action wird aufgerufen nachdem ein neuer Benutzer angelegt wurde. Handelt es sich um ein korrektes Token werden die eingegebenen Benutzerdaten ausgelesen und über die Klasse UserVerwaltungApp in die Datenbank geschrieben. Anschliessend wird das zuvor erzeugte Token gelöscht und die Benutzerlistenseite wird aufgerufen.

```
if (isTokenValid(request)){
        AddUserForm addUserForm = (AddUserForm)form;
        String vorname = addUserForm.getVorname();
        String nachname = addUserForm.getNachname();
        String ort = addUserForm.getOrt();
        int plz = addUserForm.getPlz();
        UserVerwaltungApp.addUser(vorname,nachname, ort, plz);
        resetToken(request);
        }
return mapping.findForward("list");
```

CheckEditUserAction:

Bevor ein Benutzer editiert werden kann, wird die CheckEditUserAction aufgerufen, die über einen Zugriff auf den Servlet-Context den passenden Benutzer auswählt, der durch Übergabe der ID bei dem Aufruf des Links „ändern“ auf der Benutzerlistenseite an den Servlet-Context übergeben wurde und durch ein Durchlaufen aller ID's identifiziert wird. Anschliessend werden die Daten des Benutzers in die AddUserForm geschrieben. Die so präparierte ActionForm dient dann als Grundlage, um das Formular für das Ändern der Benutzerdaten mit den Daten des zuvor ausgewählten Benutzers zu füllen.

Nach erfolgreichem Aufruf dieser Action erfolgt eine Weiterleitung an das Formular, in dem dann die Benutzerdaten editiert werden können.

EditUserAction:

Wurden die Daten eines vorhandenen Benutzers geändert, kommt es zu einem Aufruf der EditUserAction. Diese sorgt für einen Aktualisierungsvorgang, der ein Update in der Datenbank durchführt.

Nach erfolgreichem Aufruf dieser Action erfolgt eine Weiterleitung an die Benutzerlistenseite.

Da der Code dieser Action der AddAction ähnelt, wird an dieser Stelle auf ein Code-Fragment verzichtet.

RemoveAction:

Wird der Button remove auf der Benutzerlistenseite betätigt wird die RemoveAction aufgerufen, die den jeweiligen Benutzer aus der Datenbank löscht und den Servlet-Context aktualisiert.

```
Integer id = new Integer((String) request.getParameter("username"));
UserVerwaltungApp.removeUser(id.intValue());
return mapping.findForward("list");
```

LogoutAction:

Auf jeder JSP-Seite hat der Benutzer die Möglichkeit sich auszuloggen. Wird der Link „Logout“ ausgeführt, wird diese Action aufgerufen, die für eine Weiterleitung an die Login-Seite sorgt und die Benutzersession löscht.

```
HttpSession session = request.getSession();
session.invalidate();
return mapping.findForward("logout");
```

AccessDeniedAction:

Diese Action wird immer dann aufgerufen, falls ein nicht eingeloggter User versucht eine der JSP-Seiten oder Actions der Anwendung aufzurufen. Es erfolgt eine Fehlerbehandlung und eine Weiterleitung an die Login-Seite.

```
ActionErrors errors = new ActionErrors();
errors.add(ActionErrors.GLOBAL_ERROR,new ActionError("error.login.required"));
saveErrors(request, errors);
return mapping.findForward("login");
```

5.6.6 Internationalisierung

Die Anwendung ist grundsätzlich so konzipiert, dass je nach Spracheinstellungen des Browsers die Anwendung auf deutsch oder auf englisch angezeigt werden kann. Dies

wird durch Anlegen zweier ApplicationResource-Dateien ermöglicht, in denen alle wichtigen Bezeichnungen der Anwendung jeweils auf deutsch und auf englisch hinterlegt werden.

Ausschnitt aus der ApplicationResource_de.properties:

error.city.required=
Stadt fehlt
app.titleedit=*Benutzer ändern*
app.lastname=*Nachname*

Auschnitt aus der ApplicationResource.properties:

error.city.required=
city is missing
app.titleedit=*edit User*
app.lastname=*Lastname*

Beispiel für einen Zugriff auf ein Element der ApplicationResource-Datei

<bean:message key="app.lastname"/>

Je nach Spracheinstellung wird die richtige ApplicationResource-Datei benutzt und das jeweileige Element angezeigt.

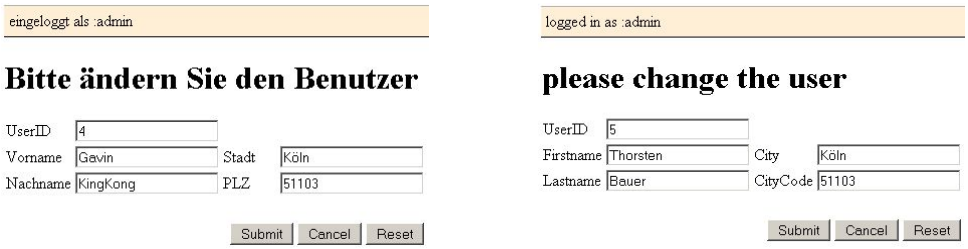


Abbildung 40+41: *Internationalisierte Seiten (Benutzerverwaltung)*

5.6.7 Sicherheit

Um zu verhindern, dass ein unberechtigter User Zugriff auf die Anwendung bekommt, indem der Login-Vorgang durch Aufrufen nachfolgender JSP-Seiten oder Actions übergangen wird, sind die JSP-Seiten und Actions geringfügig geändert werden.

Auf jeder JSP-Seite wird geprüft, ob sich der User ordnungsgemäss angemeldet hat.

Dies wird erreicht durch Überprüfung des Servlet-Contextes auf den Name „USER“, der im Rahmen der Anmeldung angelegt wird. Falls dieser vorhanden ist, wird die JSP-Seite normal angezeigt, andernfalls kommt es zum Aufruf der Action AccessDeniedAction, die für die notwendige Fehlerbehandlung und Weiterleitung zur Login-Seite verantwortlich ist.

```
<logic:notPresent name="USER">
<logic:forward name="failure"/>
</logic:notPresent>
```

Jede Action, die aufgerufen wird, überprüft in ähnlicher Weise, sich ob ein „USER“ im Servlet-Context befindet. Ist alles in Ordnung wird die Action ausgeführt, ansonsten wird durch Aufruf der Action AccessDeniedAction die Fehlerbehandlung und Weiterleitung wie bei den JSP-Seiten durchgeführt.

```
HttpSession session = request.getSession();
if ( session.getAttribute ("USER") == null){
        return mapping.findForward("failure");
        }
```

5.6.6 web.xml und struts-config.xml

In der struts-config.xml müssen alle notwendigen Angaben für die Struts-Anwendung gemacht werden.

Die Formbeans müssen deklariert werden. Folgendes Codefragment illustriert die Deklaration der 2 Formbeans LoginForm und AddUserForm.

```
<form-beans>
        <form-bean name="LoginForm" type="formbeans.LoginForm"/>
        <form-bean name="AddUserForm" type="formbeans.AddUserForm"/>
</form-beans>
```

Eine globale Weiterleitung wird festgelegt für den Fall, dass der Benutzer nicht

eingeloggt ist. Dann wird bei einem Rückgabewert von „failure“ die Adresse AccessDenied.do aufgerufen, die auf die Loginseite führt.

Folgendes Codefragment illustriert dies.

```
<global-forwards>
        <forward name="failure" path="/AccessDenied.do"/>
</global-forwards>
```

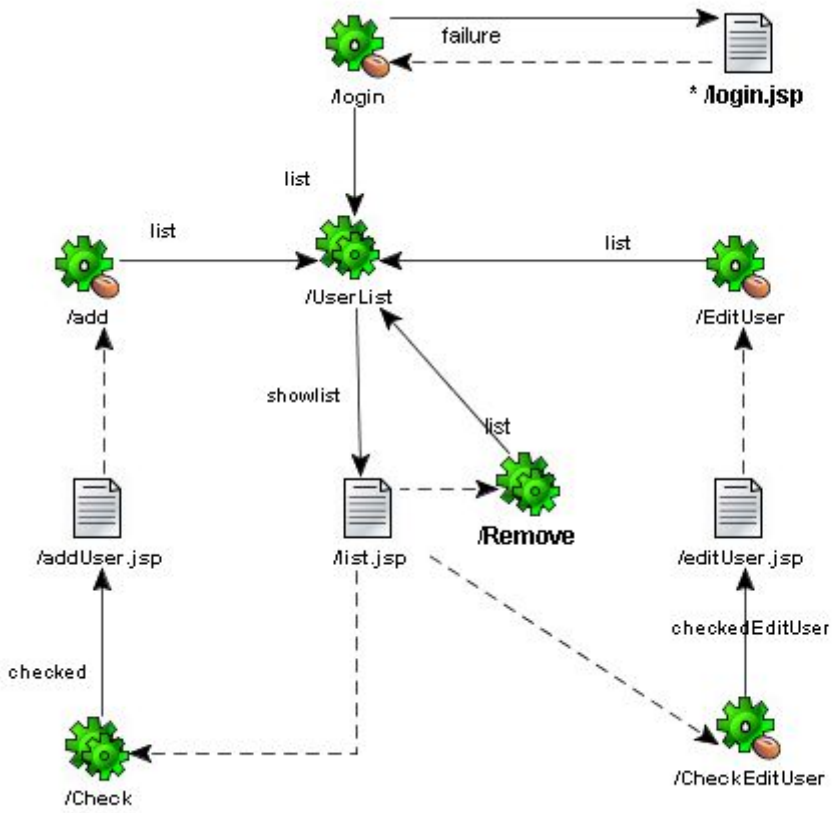
Jede Action muss in die struts-config.xm eingetragen werden. Anhand der LoginAction zeigt das Codefragment wie der Eintrag zu gestalten ist.

```
<action-mappings>
        <action name="LoginForm" path="/login" input="/login.jsp" scope="request"
                type="actions.LoginAction" validate="true">
                <forward name="list" path="/UserList.do"/>
                <forward name="login" path="/login.jsp"/>
        </action>
</action-mappings>
```

Abschließend wird noch der Name des Resource-Bundles für die Internationalisierungsdatei angegeben durch folgende Zeile.

```
<message-resources parameter="properties.ApplicationResources"/>
```

Die nachfolgend visualisierte struts-config.xml zeigt wie die Actions mit den JSP-Seiten in Zusammenhang stehen und macht gleichzeitig den Ablauf innerhalb der Anwendung deutlich.



Legende	Zahnräder:	Action
	weisse Dokumente:	JSP-Seite
	durchgezogene Linie:	Forward
	gestrichelte Linie:	Links

Abbildung 42: *visualisierte struts-config.xml (Benutzerverwaltung)*

5.6.7 Deployment als WAR-File

Um die erstellte Anwendung über einen Servlet-Container verfügbar zu machen, muss eine bestimmte Struktur der Anwendungsdateien eingehalten werden. Die Struktur wird durch folgende Grafik gezeigt.

Abbildung 43: *Verzeichnisstruktur (Benutzerverwaltung)*

Der WEB-INF-Ordner bildet das wichtigste Verzeichnis, da es sowohl alle notwendigen Klassen, als auch die für die Strtus-Anwendung benötigten Libraries enthält. Nachfolgende Grafik zeigt die genaue Struktur des WEB-INF-Ordners.

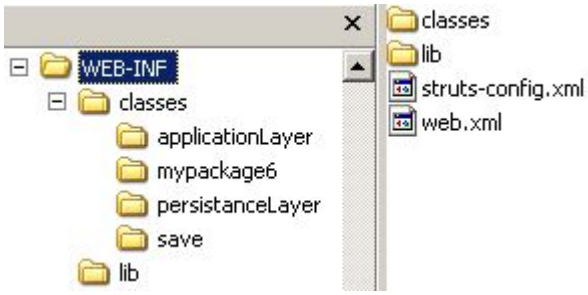


Abbildung 44: *WEB-INF-Inhalt (Benutzerverwaltung)*

Wie zu sehen ist, müssen sich ausserdem die Konfigurationsdateien struts-config.xml und web.xml in dem WEB-INF-Ordner befinden.

Anschliessend kann aus dieser festgelegten Verzeichnisstruktur, die alle notwendigen Dateien enthält, als WAR-File gepackt werden. Das Web Applikation Resource (WAR)-Format ist speziell für Web-Applikationen gedacht und setzt die oben beschriebene Verzeichnisstruktur voraus. Das WAR-File enthält wie ein JAR-File die gewünschten Dateien in komprimierter Form und kann so in jedem Servlet-Container in das richtige Verzeichnis kopiert werden und ist dann als Web-Applikation durch Eingabe der ihr zugeordneten URL erreichbar, die in der web.xml eingetragen wird.

Folgende Codefragmente der web.xml erläutent diesen Aspekt.

```
<servlet>
        <servlet-name>action</servlet-name>
        <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
        <init-param>
        <param-name>config</param-name>
        <param-value>/WEB-INF/struts-config.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
</servlet>
```

Hier wird der Name des Servlet angegeben und mit dem Servlet von Struts verknüpft.

```
<servlet-mapping>
        <servlet-name>action</servlet-name>
        <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

Anschliessend werden sämtliche Actions mit dem oben angegebenen Servlet verknüpft, falls die URL die Endung *.do enthält.

5.7 Fazit

Struts ist ein ausgereiftes Web-Framework, welches durch die vielen bereitgestellten Funktionalitäten für viele Probleme, die während der Entwicklung einer Web-Applikation beachtet werden müssen, Unterstützung bietet.

Da es mittlerweile seit einigen Jahren verfügbar ist, kann auf eine breite Literaturbasis zurückgegriffen werden, die sowohl für Einsteiger, als auch für Struts-Experten passende Unterstützung bietet.

Struts hat seit seiner Einführung viele Anhänger gefunden und sich als professionelles Web-Framework durchgesetzt, was durch eine Vielzahl von auf Struts basierten Web-Projekten belegt wird, die auf der Struts-Website[85] nachgeschlagen werden können.

Zukünftige Entwicklungspläne[86] der Struts-Entwickler sehen weitere Verbesserungen

85 http://wiki.apache.org/struts/StrutsConslutants
86 http://struts.apache.org/roadmap.html: angedachte Neuerungen für zukünftige Versionen von Struts

des Web-Frameworks vor. An der Version Struts 2.0 wird bereits gearbeitet. Diese soll neben der Unterstützung der Servlet 2.4/JSP 2.0 Plattform auch Unterstützung für die Java 1.5 Plattform bieten. Als View-Komponenten werden die Struts-Taglibs wahrscheinlich von JavaServer Faces und der JSTL abgelöst. Auch über eine Unterstützung von Web-Services und Portlets macht man sich bereits Gedanken.

Als professionelles Web-Framework für Web-Applikationen kann Struts empfohlen werden, wobei sich allerdings gerade im Hinblick auf JavaServer Faces noch zeigen muss, inwieweit es sich weiterhin als „de facto Standard“ behaupten kann.

6. JavaServer Faces im Vergleich mit Struts

Ein Vergleich soll die Gemeinsamkeiten und die Unterschiede der beiden Web-Frameworks beleuchten und aufzeigen in welchen Situationen sich das eine oder andere Web-Framework besser eignet.

6.1 Gegenüberstellung der beiden Web-Frameworks

Anhand unterschiedlicher Aspekte werden die jeweiligen Web-Frameworks miteinander verglichen. Der Schwerpunkt wird auf folgende Themen beschränkt :

- Reifegrad der Web-Frameworks
- Einarbeitungszeit und Entwicklungsaufwand
- Viewkomonenten
- Modellkomponenten
- Fehlerbehandlung und Validierung
- Internationalisierung
- Navigation
- Erweiterbarkeit
- Integrationsmöglichkeiten

Reifegrad der Web-Frameworks

Da Struts seit längerer Zeit am Markt verfügbar ist und dadurch eine grössere Entwicklergemeinde für sich gewinnen konnte, ist Struts weitgehend Bugfrei und befindet sich in der Konsolidierungsphase. JavaServer Faces dagegen ist erst seit 2 Jahren am Markt erhältlich, was sich darin niederschlägt, dass JavaServer Faces noch einige Bugs enthält, welche aber in den nächsten Versionen verschwinden werden.

Einarbeitungszeit und Entwicklungsaufwand

Je nach Kenntnisstand der Programmierer ist der Einarbeitungsaufwand nicht unbeträchtlich. Beide Web-Frameworks setzen solide Java-Kenntnisse voraus und die Funktionsweise der Beiden muss vor dem Erstellen einer Web-Applikation verstanden werden.

Ist man mit der GUI-Programmierung unter Java vertraut, ist die Einarbeitungszeit für JavaServer Faces kleiner als bei Struts, da ähnliche Konzepte der Event-Verarbeitung eingesetzt werden. Struts baut auf einer Request getriebenen Verarbeitung auf, die anfänglich etwas gewöhnungsbedürftig ist und weniger Spielraum bei der Verarbeitung von Benutzereingaben bietet.

Viewkomponenten

Hier biete JavaServer Faces ein reichhaltiges Angebot an vorgefertigten Komponenten, das dem von Struts weit überlegen ist. Zusätzlich dazu werden bereits von unterschiedlichen Herstellern Komponenten für unterschiedliche Einsatzbereiche angeboten. Struts dagegen bietet nur die notwendigsten Komponenten an, wobei sich allerdings die Möglichkeit eröffnet JavaServer Faces Komponenten für die View einzusetzen.

Für beide Web-Frameworks gilt, dass die View-Komponenten durch eine andere Technologie ersetzt werden kann.

Modellkomponenten

Die ManagedBeans von JavaServer Faces sind als normale JavaBeans konzipiert, wodurch es möglich wird, diese separat und unabhängig von JavaServer Faces zu testen. Die Formbeans von Struts dagegen sind fest mit dem Web-Framework verknpüft und können nur im Rahmen von Struts als komplette WebApplikation getestet werden. Allerdings werden durch Struts unterschiedliche FormBeans bereitgestellt, beispielsweise die ValidatorForm, welche für das Validieren von Benutzereingaben vorgesehgen wurde.

Fehlerbehandlung und Validierung

Die mitgelieferten Fehlerbehandlungs- und Validierungskonzepte von Struts und JavaServer Faces bieten eine solide Grundlage für die Überprüfung von Benutzereingaben. Allerdings ist Struts hier deutlich fortschrittlicher, da es von Hause aus ein ausgereifteres Validierungskonzept beinhaltet, das standardmässig eine grössere Anzahl an vorgefertigten Validierungsmöglichkeiten bietet.

Internationalisierung

Das Konzept der Internationalisierung ist bei beiden Web-Frameworks fast identisch umgesetzt worden, wodurch weder Struts noch JavaServer Faces Vorteile für sich verbuchen kann.

Navigationskonzepte

Das Navigationskonzept von JavaServer Faces ist auf die jeweiligen JSP-Seiten ausgerichtet und bietet durch Rückgabewerte der HTML-Komponenten flexible Weiterleitungsmöglichkeiten.

Bei Struts erfolgt die Ausrichtung auf die Actions, bei denen dann die jeweilige Weiterleitung auf die nächste Seite angegeben werden kann.

Da JavaServer Faces ein Event-basiertes Navigationskonzept bietet ist es deutlich flexibler als die Request-orientierte Variante von Struts.

Erweiterbarkeit

Beide Web-Frameworks bieten die Möglichkeit die Basis an vorgefertigten Komponenten zu erweitern. Da sowohl JavaServer Faces, als auch Struts flexibel erweiterbar sind, kann keines der beiden Web-Frameworks Vor- oder Nachteile für sich verbuchen.

Integrationsmöglichkeiten

Da JavaServer Faces und Struts als Programmiersprache Java einsetzen sind sie beide

beliebig erweiterbar durch andere vorhandene Technologien, die auf Java aufbauen. Da für beide Web-Frameworks lediglich eine Servlet-Engine benötigt wird, können Anwendungen erstellt werden, die auf nahezu jeder Plattform gehostet werden können,

7. Schlussbetrachtung

Die Entwicklung von modernen Web-Applikationen, die allen an sie gestellten Anforderungen genügen sollen, stellt ein Unterfangen dar, welches mit Bedacht sorgfältig geplant und ausgeführt werden sollte.

Da mittlerweile zahllose Technologien für die Entwicklung von Web-Applikationen am Markt existieren, ist es schwierig den Überblick zu behalten und die jeweils richtige Technologie für das Projekt auszuwählen. Häufig werden ähnliche Anforderungen an eine Web-Applikation gestellt, wodurch ähnliche Lösungen bereitgestellt werden müssen.

Dafür bietet sich die Möglichkeit an, ein Web-Framework zu benutzen, welches genau diese Probleme adressiert und durch eine vorgegebene Vorgehensweise einen Rahmen bietet, in dem Web-Applikationen deutlich effizienter und effektiver erstellt werden können. Der Neuanfang auf der grünen Wiese sollte nur in den Ausnahmefällen begangen werden, falls dies unbedingt erforderlich ist und für den adressierten Problemkomplex kein geeignetes Web-Framework existiert.

JavaServer Faces ist ein aufstrebendes Web-Framework, welches mittlerweile durch viele Entwicklungswerkzeuge unterstützt wird und die Erstellung von robusten und zuverlässigen Web-Anwendungen ermöglicht. Durch sein Event getriebenes Architekturmodell ist es flexibel für viele Probleme einsetzbar und kann gewinnbringend für ein Web-Projekt eingesetzt werden.

Struts ist ein Web-Framework, welches sich am Markt etabliert hat und bereits für viele Web-Projekte erfolgreich eingesetzt wurde.

Beide Web-Frameworks eignen sich in hervorragender Weise für die Erstellung von Web-Applikationen und bieten durch ihr Baukastensystem ein flexibles Zusammenbauen einer Web-Anwendung.

Literaturverzeichnis

BAL01: Balzert, Helmut; Lehrbuch der Software-Technik I;
Spektrum Wissenschaftlicher Verlag; 2000

BER01: Bergsten, Hans; JavaServer Faces;
O'Reilly; 2004

BOS01: Bosch, Andy; JavaServer Faces;
Addison-Wesley; 2004

EVA01: Evans, Eric; Domain-Driven Design;
Addison Wesley; 2004

FOW01: Fowler, Martin; Patterns of Enterprise Application Architecture;
Addison Wesley; 2003

GOO01: Goodwill, James; Mastering Jakarta Struts;
Wiley; 2002

HAM01: Haiges, Sven; May Marcel; JavaServer Faces;
Software & Support; 2004

HUS01: Husted, Ted; Dumoulin, Cedric; Franciscus, George; Winterfeldt, David;
Struts in Action; Manning 2003

JAV01: Javamagazin; Ausgabe 10/2003;
Software & Support,

JOH01: Johnson, Rod; J2EE Design and Development;
Wrox; 2003

MAN01: Mann, Kito; JavaServer Faces in Action;
Manning; 2005

NAS01: Nash, Michael; Java Frameworks and Components;
Cambridge University Press; 2003

OAK01: Oak, Harshad; Pro Jakarta Commons;
Apress; 2004

SHI01: Ship, Howard M. Lewis; Tapestry in Action;
Manning; 2004

SHR01: Shklar, Leon; Rosen, Richard; Web Application Architecture;
Wiley; 2003

SIG01: Siggelkow, Bill; Jakarta Struts Cookbook;
O'Reilly; 2005

WES01: Wessendorf, Matthias; Struts;
W3L GmbH; 2005

Anhang A

Inhalt der CD

Auf der CD finden sich beide Beispielapplikation verteilt auf die Ordner JavaServer Faces und Struts. Beide Beispielapplikationen sind als komplette JDeveloper-Projekte beigelegt. Sämtliche Sourcecodes sind dort zu finden.

Installationsanleitung für den Web-Shop

- *als JDeveloper-Projekt-Import:*

1.

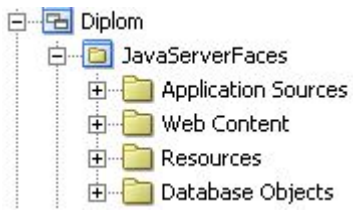
Um das Projekt im JDeveloper zu testen wird eine funktionsfähige Version der Preview Edition 10.1.3 vorausgesetzt und eine Oracle Datenbank.

2.

Es muss ein neues leeres Projekt mit dem Namen JavaServerFaces erstellt werden. Anschließend können alle Dateien des Unterordners JDeveloper Projekt des Ordners JavaServer Faces der CD in das soeben erstellte Verzeichnis kopiert werden.

3.

Man erhält folgende Verzeichnisstruktur:



Im Ordner Resources befinden sich die Datenbankscripte zum Erstellen der Datenbanktabellen und zum Füllen mit Dummy-Daten. Diese müssen ausgeführt

werden unter einem Schema Ihrer Wahl.

4.

Sind die Tabellen angelegt muss die Datenbankklasse JDBCPersistance angepasst werden. Diese befindet sich unter ApplicationSources im Paket persistanceLayer.

Folgende Zeile muss gemäß Ihrer Oracle Konfiguration angepasst werden:

```
conn = DriverManager.getConnection("jdbc:oracle:thin:@hadoko:1521:orcl",
"astro", "astro");
```

5.

Soll die E-Mail-Funktionalität genutzt werden, muss die Klasse EmailKunde angepasst werden.

Folgende Codezeilen bedürfen der Anpassung

email.setHostName("mail.gmx.de");//ein gültiger SMTP-Server

und

email.setFrom("schmid.karl@gmx.net"); //Absenderadresse
email.setAuthentication("schmid.karl@gmx.net","Ihr Passwort");//Ihre Account-Daten

6.

Kompilieren Sie das komplette Projekt und starten Sie die navigation.jsp

- *als Tomcat-Projekt*

1.

Um das Projekt über Tomcat verfügbar zu machen wird die Version 5.5 und das aktuelle Tiger Release 5 von Java vorausgesetzt, da nur unter diesen beiden Voraussetzungen die Anwendung getestet wurde.

2.

Nachdem wie oben beschrieben wurde, das Projekt erfolgreich in den JDeveloper importiert wurde, kann nach oben geschilderten Anpassungen eine WAR-Datei erstellt werden. Unter Resources finden Sie das ein fertig angelegtes webshop.deploy File, mit dem sich die WAR-Datei erzeugen läßt.

3.

Die soeben erstellte WAR-Datei befindet sich nun in Ihrem Projektverzeichnis unter deploy und kann nun in das webapps-Verzeichnis von Tomcat kopiert werden.

4.

Starten Sie Tomcat.

Über folgenden Link der allerdings noch angepasst werden muss, ist die Anwendung ausführbar:

http://localhost:8095/webshop/faces/navigation.jsp

Falls Sie die Standardkonfiguration von Tomcat benutzen, müssen Sie als Port 8080 angeben.

Installationsanleitung für die Benutzerverwaltung

- *als JDeveloper-Projekt-Import:*

1.

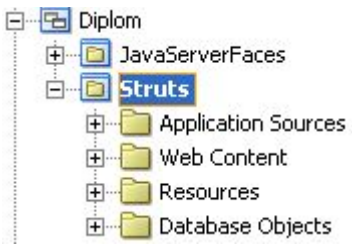
Um das Projekt im JDeveloper zu testen wird eine funktionsfähige Version der Preview Edition 10.1.3 vorausgesetzt und eine Oracle Datenbank.

2.

Es muss ein neues leeres Projekt mit dem Namen Struts erstellt werden. Anschließend können alle Dateien des Unterordners JDeveloper Projekt unter dem Ordner Struts der CD in das soeben erstellte Verzeichnis kopiert werden.

3.

Man erhält folgende Verzeichnisstruktur:



Im Ordner Resources befinden sich die Datenbankscripte zum Erstellen der Datenbanktabellen. Diese müssen ausgeführt werden unter einem Schema Ihrer Wahl.

4.

Sind die Tabellen angelegt muss die Datenbankklasse JDBCPersistance angepasst werden. Diese befindet sich im gleichen Verzeichnis wie bei der JavaServer Faces Applikation und es müssen die gleichen Änderungen vorgenommen werden, da die gleiche Klasse für beide Anwendungen benutzt wurde.

5.

Kompilieren Sie das komplette Projekt und starten sie die login.jsp.

- *als Tomcat-Projekt*

1.

Um das Projekt über Tomcat verfügbar zu machen wird die Version 5.5 und das aktuelle Tiger Release 5 von Java vorausgesetzt, da nur unter diesen beiden Voraussetzungen die Anwendung getestet wurde.

2.

Nachdem wie oben beschrieben wurde, das Projekt erfolgreich in den JDeveloper importiert wurde, kann nach oben geschilderten Anpassungen eine WAR-Datei erstellt werden. Unter Resources finden Sie ein fertig angelegtes deployUserVerwaltung.deploy File, mit dem sich die WAR-Datei erzeugen läßt. Der Vorgang ist identisch mit dem von oben.

3.

Die soeben erstellte WAR-Datei befindet sich nun in Ihrem Projektverzeichnis unter deploy und kann nun in das webapps-Verzeichnis von Tomcat kopiert werden.

4.

Starten Sie Tomcat.

Über folgende URL läßt sich das Programm starten:

http://localhost:8095/deployUserVerwaltung/

Der Port muss entsprechend Ihrer Tomcatkonfiguration angegeben werden.

Anhang B

Erklärung

Ich erkläre, die von mir vorgelegte Arbeit selbstständig verfasst zu haben. Alle Seiten, die wörtlich oder sinngemäß aus veröffentlichten oder nichtveröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfmittel, die ich für die Arbeit genutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.