Ein Framework für Shop-Systeme auf Basis der Java 2 Enterprise Edition

Stefan Eicker

Universität Essen
Lehrstuhl für Wirtschaftsinformatik,
insbes. Betriebliche Kommunikationssysteme
Universitätsstr. 9
D-45141 Essen
Tel.: +49 (201) 183 4031
E-Mail: eicker@wi-inf.uni-essen.de

Holger Schwichtenberg

Universität Essen
Lehrstuhl für Wirtschaftsinformatik,
insbes. Betriebliche Kommunikationssysteme
Universitätsstr. 9
D-45141 Essen
Tel.: +49 (201) 183 4009
E-Mail: schwichtenberg@wi-inf.uni-essen.de

Wolfgang Büning

MarchFIRST
Halbergstr. 28
D-40239 Düsseldorf
Tel.: +49 (211)
E-Mail: wolfgang.buening@marchFirst.com

Ein Framework für Shop-Systeme auf Basis der Java 2 Enterprise Edition

Stefan Eicker, Holger Schwichtenberg
Universität Essen

Wolfgang Büning
MarchFIRST

Zusammenfassung: Insbesondere kleine und mittelständische Unternehmen können oder wollen für ihren Eintritt in die eCommerce-Welt häufig nur sehr begrenzte Mittel bereitstellen. Auf der anderen Seite können statische Web-Anwendungen Unternehmen allenfalls einen geringen Nutzen bringen. Die Entwicklung dynamischer Web-Anwendungen, deren Seiten aus aktuellen Datenbankinhalten generiert werden, erfordert jedoch zumeist einen erheblichen Entwicklungsaufwand. Ein Weg, den Aufwand signifikant zu senken, besteht in der Standardisierung. Damit sich die Auftritte der verschiedenen Unternehmen unterscheiden, darf ein Standardsystem allerdings nur einen Rahmen vorgeben, der die grundlegende Funktionalität umfasst, und beim konkreten Einsatz an die spezifischen Anforderungen angepasst werden kann. In dem Beitrag wird ein solches Standardsystem in Gestalt eines Frameworks für internetbasierte Warenkorbsysteme (Online-Shops) vorgestellt. Es stellt das Ergebnis einer Machbarkeitsstudie dar, in der die Möglichkeit eruiert wurde, auf der Basis einer kostengünstigen Plattform ein Standard-Warenkorbsystem zu entwickeln, das die hohen zu erfüllenden Anforderungen an Skalierbarkeit und Verfügbarkeit erfüllen kann, und das möglichst geringe zusätzliche Kenntnisse der Web-Designer erfordert.

Schlüsselworte: Online-Shop, Warenkorbsystem, Framework, Java 2 Enterprise Edition

1 Einleitung

Das zunehmende Volumen der über das Internet gehandelten Produkte und Dienstleistungen stellt einen starker Anreiz für ein Unternehmen dar, seine eCommerce-Aktivitäten auszuweiten. Eine statische Web-Präsenz eignet sich allerdings i. Allg. nicht als Basis für die Ausweitung, da mit der Menge der zu präsentierenden Informationen der Änderungsaufwand weiter steigt. Außerdem kann durch manuelle Änderungen kaum die notwendige Aktualität der Daten gewährleistet

werden. Schließlich erfordert die Zunahme der über das Web abgewickelten Geschäftsvorfälle eine Anbindung der Web-Anwendungen an die übrigen computergestützten Informationssysteme des Unternehmens.

Aus den genannten Gründen müssen dynamische Web-Techniken für die Entwicklung moderner (dynamischer) Web-Anwendungen eingesetzt werden. Sie erlauben es insbesondere, die zu präsentierenden Daten aktuell aus entsprechenden Datenbanken zu lesen und umgekehrt über das Web gesendete Daten in den Datenbanken zu speichern. Aber die Entwicklung dynamischer Web-Anwendungen ist aufwendig; außerdem treibt der Personalmangel im IT-Bereich die Personalkosten und damit auch die Kosten für Web-Projekte weiter in die Höhe.

Wie bei den klassischen Anwendungssystemen bildet die Standardisierung einen Weg, die Kosten für die Realisierung von Web-Anwendungen zu senken. Allerdings kann ein Web-Standardsystem nur einen Rahmen vorgeben. Denn zum einen verlangen die Unternehmen ein individuelles Design ihrer Web-Präsenz. Zum anderen müssen die Spezifika des Produkt- und Leistungsangebots eines Unternehmens und der Inanspruchnahme des Angebots in der Web-Anwendung abgebildet werden können.

In den letzten Jahren wurde eine Reihe kommerzieller Shop-Out-of-the-Box-Lösungen für die Realisierung internetbasierter Warenkorbsysteme/Online-Shop-Systeme entwickelt; bekannte Beispiele sind Intershop 4 (http://www. InterShop.de) oder der Microsoft Commerce Server 2000 (http://www.microsoft.com/commerceserver). Der Einsatz der Produkte kostet allerdings fünf- bis sechsstellige Beträge; eine Ein-Prozessor-Lizenz des als sehr günstig geltenden Shop-Systems von Microsoft kostet beispielsweise zur Zeit 12.999 US-Dollar).

Im Folgenden wird ein Framework für die Entwicklung von Warenkorbsystemen vorgestellt; es bildet das Ergebnis einer Machbarkeitsstudie, in deren Rahmen die Möglichkeit untersucht wurde, für kleinere und mittelständische Unternehmen eine einfach nutzbare und kostengünstige Lösung anbieten zu können.

Als Plattform wurde für das Framework die Java 2 Enterprise Edition (J2EE) ausgewählt. Dies begründet sich zum einen darin, dass die von ihr bereitgestellten Techniken insbesondere auf die Realisierung hochverfügbarer, skalierbarer, verteilter Internet-Anwendungen ausgerichtet sind. Zum anderen kann die Plattform unter Linux mit verschiedenen J2EE-Application Servern - z.B. mit JBoss (http://www.jboss.org) oder mit der J2EE-Referenzimplementierung von Sun (http://developer.iplanet.com/tech/java) - *kostenlos* betrieben werden (alternativ stehen auch kommerzielle Application Server wie Sun's Iplanet, der IPortal Server von IONA, der IBM WebSphere Application Server und der BEA WebLogic Server zur Verfügung).

Im folgenden Kapitel wird zunächst kurz die Java 2 Enterprise Edition mit ihren Bestandteilen vorgestellt. Kapitel 3 gibt einen Überblick über die Funktionalität,

die Architektur und die Komponenten des Frameworks und demonstriert an einigen Beispielen die Verwendung der Tags. Ein Fazit schließt den Beitrag ab.

2 Die Java 2 Enterprise Edition

Im Dezember 1999 hat Sun Microsystems die Spezifikation und die Referenzimplementierung der Java 2 Enterprise Edition (J2EE) in der Version 1.2 veröffentlicht. Die Spezifikation der J2EE definiert eine Plattform für die Entwicklung von internetfähigen, verteilten Applikationen. Die Plattform soll die Entwicklung durch die Automatisierung von Basisaufgaben der Programmierung und durch den Einsatz von standardisierten modularen Komponenten vereinfachen; sie beinhaltet dazu die Java Servlets, JavaServer Pages und Enterprise JavaBeans.

Java Servlets sind Java-Programme, die die Funktionalität eines Web-Servers erweitern, indem sie dynamisch und interaktiv Seiteninhalte generieren. Sie haben in den letzten Jahren bei der Realisierung von dynamischen Internetseiten eine große Verbreitung erlangt, da sie die Möglichkeit bieten, server- und plattformunabhängig die Funktionalität eines Servers zu erweitern.

Bei *JavaServer Pages* handelt es sich um eine Erweiterung der Java Servlets, die dynamisch Inhalte in HTML- oder XML-Vorlagen einfügt. Sie wurden entwickelt, um die Präsentation stärker von der Programmierung zu trennen und so den Aufwand für die Erstellung eines Internetangebots zu senken. JavaServer Pages bestehen größtenteils aus HTML- oder XML-Befehlen, in die Java-Befehle eingebettet sind.

Enterprise JavaBeans sind eine Komponentenarchitektur zur Entwicklung und zum Einsatz von objektorientierten, verteilten, skalierbaren Applikationen. Mit Hilfe von Enterprise JavaBeans können verteilte Objekte serverseitig unter Einsatz entsprechender Tools entwickelt werden. Die Enterprise JavaBean-Architektur erleichtert die Erstellung von Applikationen, da der Anwendungsentwickler von grundlegenden Programmieraufgaben, beispielsweise von der Low-Level-Transaktions- und Zustandsverwaltung und vom Multithreading, befreit wird [Mate99, S. 19f.]. Durchgeführt werden die Aufgaben vielmehr – gesteuert über vom Anwendungsentwickler gesetzte Parameter – durch den Application Server. Ein weiterer Vorteil von Enterprise JavaBeans ist darin zu sehen, dass die Beans grundsätzlich auf unterschiedlichen Plattformen ohne Veränderung des Sourcecodes und ohne neue Kompilierung einsetzbar sind [Ayer99, S. 578].

Zu unterscheiden sind verschiedene Typen von Enterprise JavaBeans: *Entity Beans* sind typischerweise Objekte, die Daten aus einer Datenbank repräsentieren. Sie dienen dazu, Daten zu speichern oder gespeicherte Daten zu verändern [Merk00, S. 47]; insbesondere ermöglichen sie mehreren Clients den parallelen

Zugriff auf transaktionsgesicherte Daten. Abgelegt werden die Daten der Entity Beans in einer relationalen Datenbank [Denn00, S. 118f.].

Eine *Session Bean* stellt üblicherweise ein Objekt dar, das die Logik eines Geschäftsprozesses implementiert [Roma99, S. 83 f.]. Um die zugehörige Funktionalität zur Verfügung zu stellen, greift die Session Bean auf andere Enterprise JavaBeans und auf andere Dienste zu, die vom EJB-Container zur Verfügung gestellt werden. Im Unterschied zu den Entity Beans beinhalten die Session Beans keine Daten, die persistent gespeichert werden.

Zwei Ausprägungen der Session Beans sind zu unterscheiden, die Stateless Session Beans und die Stateful Session Beans. Eine Stateless Session Bean ist eine Sammlung zusammengehörender Dienste, wobei ein Dienst jeweils von einer Methode in der Session Bean repräsentiert wird. Im Gegensatz dazu kann in einer Stateful Session Bean ein Dienst von mehreren Methoden repräsentiert werden. Den Stateless Session Beans wird kein Client zugeordnet, d.h., Anfragen eines Clients können von unterschiedlichen Stateless Session Beans bearbeitet werden. Sie behalten keinen Status von einem Methodenaufruf zum nächsten. Stateful Session Beans hingegen werden direkt einem Client zugeordnet und können einen Status zwischen mehreren Aufrufen speichern [Mons00, S. 218ff.]. Als Nachteil ergibt sich daraus eine langsamere Ausführungsgeschwindigkeit, da Stateless Session Beans nur eine Anfrage eines Clients bearbeiten und dann einem anderen Client zur Verfügung stehen.

3 Funktionen, Architektur und Komponenten eines Frameworks für Shop-Systeme

3.1 Funktionsübersicht

Aufgaben eines Shops sind auf Seiten des Anbieters die geeignete Präsentation seiner Produkte, auf Seiten des Konsumenten die Möglichkeit, Produkte auszuwählen, in einen Warenkorb zu legen und schließlich die Produkte des Warenkorbs zu bestellen. Das auf Basis der J2EE entwickelte Framework stellt diese Grundfunktionalität eines Online-Shops zur Verfügung. Dazu existieren vielfältige Möglichkeiten, das Layout eines Shops zu gestalten sowie die Funktionen des Frameworks mit wenig Aufwand anzupassen bzw. zu erweitern.

Den Kern des Shops bildet ein flexibles Katalogsystem, das es erlaubt, beliebig verschachtelte Produktkategorien anzulegen. In jeder Kategorie kann eine unbegrenzte Anzahl von Produkten mit den Produktinformationen gespeichert werden. Abgelegt werden können die Informationen jeweils als Text, als Grafik oder als

Datei. Die Anzahl der Produktinformationen zu einem Produkt ist nicht fest vorgeschrieben, sondern wird beim Einrichten des Shops festgelegt.

Für den Konsumenten beinhaltet das Framework eine einfache Suchfunktion nach Produkten, die erweitert werden kann. Außerdem wird die Warenkorbfunktionalität zur Verfügung gestellt: Eine beliebige Anzahl von Produkten kann in den Warenkorb gelegt, der Inhalt des Warenkorbs und der aktuelle Bestellwert angezeigt und ein Produkt im Warenkorb gelöscht werden. Ist der Warenkorb gefüllt, können die Waren bestellt werden. Beim Bestellvorgang muss ein Dienstleister für die Zahlungsabwicklung eingebunden werden. Da hierfür keine standardisierte Schnittstelle existiert, liefert das Framework keine Implementierung für ein Zahlungssystem; die Integration von Zahlungssystemen wie CyberCash [Cybe00] oder Bibit [Bibi00] kann aber mit geringem Aufwand realisiert werden.

Im Rahmen der Benutzerverwaltung erlaubt das Framework die Registrierung eines neuen Benutzers; zur Anmeldung bereits registrierter Kunden steht ein Authentifizierungsmechanismus zur Verfügung. Funktionen zur Verwaltung der Benutzerinformationen sowie auch von Bestellungen sind ebenfalls in das Framework integriert.

Für die Administration der Shop-Informationen existiert ein Administrationstool, mit dem die verschiedenen Informationen wie zum Beispiel Sprach- und Produktinformationen verwaltet werden können.

Schließlich erfüllt das Framework auch die Anforderungen eines internationalen Einsatzes. Es bietet zum einen Funktionen für die Mehrsprachigkeit, zum anderen zur automatischen Umrechnung von Währungsbeträgen.

3.2 Architektur des Framework

Da das Framework auf der J2EE Technologie basiert, kann es als Grundlage für Shops auf verschiedenen Application Servern dienen. Der Application Server kann in Abhängigkeit von dem erwünschten Leistungsumfang des Shops, von der Anzahl der erwarteten Benutzerzugriffe und von weiteren Spezifika in der entsprechenden Leistungsklasse ausgewählt werden. Sollten sich die Bedürfnisse des Shopbetreibers ändern, kann ein anderer Application Server genutzt werden. Darüber hinaus bietet die J2EE-Technologie den Vorteil, dass der Online-Shop unabhängig von einer speziellen Datenbank und von einem bestimmten Webserver eingesetzt werden kann.

Im Framework werden Enterprise JavaBeans, JavaServer Pages und ein Java Servlet eingesetzt:

1. Enterprise JavaBeans

Mehrere funktional zusammenhängende Enterprise JavaBeans bilden die Komponenten des Shop-Systems. Von den Komponenten werden die persistente Daten-

haltung und die grundlegende Logik zur Verfügung gestellt. Die Komponenten sind leicht durch das Hinzufügen von weiteren Enterprise JavaBeans oder durch das Ergänzen zusätzlicher Funktionen zu den bestehenden Enterprise JavaBeans zu erweitern. Auf diese Weise kann mit geringem programmiertechnischen Aufwand eine kundenspezifische Logik in die Komponenten integriert werden. Eine Beschreibung der Komponenten erfolgt im Kapitel 3.3.

2. JavaServer Pages

Die in dem Framework eingesetzten JavaServer Pages dienen der Visualisierung der Informationen, die in den Komponenten abgelegt sind. Um von den JavaServer Pages auf die Komponenten zugreifen zu können, wurden benutzerdefinierte Tags definiert, durch die die Methoden der Komponenten aufgerufen werden; Bespiele der benutzerdefinierten Tags werden in Kapitel 3.4 vorgestellt.

Bei den benutzerdefinierten Tags wird der Grundsatz verfolgt, möglichst kein HTML, sondern unformatierten Text als Rückgabe zu liefern, damit der Ersteller des Shops große Freiheiten beim Formatieren der Seiten hat. Bei einigen Tags ist die Rückgabe von HTML jedoch aus Gründen des Komforts für den Seitenersteller unvermeidbar.

Die Funktionen der benutzerdefinierten Tags können durch Hinzufügen von entsprechendem Java-Code innerhalb der JavaServer Pages ergänzt werden.

3. Java Servlet

In dem Framework wird ein Java Servlet eingesetzt, das alle Seitenanfragen bearbeitet. Damit wurde dem *Command Pattern* (vgl. Abbildung 1 sowie [Fiel00, S. 227f.]) gefolgt, um die Struktur des Frameworks möglichst einfach und übersichtlich zu gestalten. Das Java Servlet erkennt durch die in der Anfrage übergebenen Parameter, welche Aktion der Benutzer ausführen will, beziehungsweise welche Seite er dargestellt haben möchte. Als Parameter wird immer ein Command übergeben. Das Java Servlet überprüft, welches Command angegeben wurde, und ruft dann die Klasse auf, die mit dem Command verknüpft ist; sie übernimmt dann die Abarbeitung der Seitenanfrage. Die Zuordnung der Commands zu Java-Klassen wird in einer XML-Datei festgelegt. Dadurch lässt sich die Java-Klasse, die bei einem bestimmten Command aufgerufen werden soll, sehr einfach austauschen.

Ein weiterer Vorteil der Nutzung des Command Pattern besteht darin, dass das Java Servlet bei jeder Seitenanfrage bestimmte Überprüfungen vornehmen kann. Zum Beispiel wird kontrolliert, ob der Benutzer eingeloggt ist, und ob er die Rechte für den Seitenaufruf besitzt. Ist eine der beiden Bedingungen nicht erfüllt, wird eine Seite mit einer entsprechenden Fehlermeldung angezeigt. Die im Framework realisierte einfache Rechteüberprüfung kann beliebig verfeinert werden.

Abbildung 1: Implementierung des Command Patterns

Schließlich protokolliert das Java Servlet auch alle Aktivitäten eines Benutzers, damit eine Benutzeranalyse des Shops durchgeführt werden kann. Die Protokollierung erfolgt über ein Session-Objekt, welches das Servlet einem Benutzer zuordnet.

Die Interaktionen des Java Servlets/der JavaServer Pages, der Enterprise JavaBeans und der Datenbank stellen sich folgendermaßen dar (vgl. auch Abbildung 2): Beim Aufruf einer Seite des Frameworks durch einen Client wird zuerst das Java Servlet angesprochen. Das Java Servlet ruft eine Command-Klasse auf und gibt danach die Anfrage an eine JavaServer Page weiter. Die Command-Klasse und die in der JavaServer Page enthaltenen benutzerdefinierten Tags rufen zur Bearbeitung der Anfrage die entsprechenden Enterprise JavaBeans auf. Die Enterprise JavaBeans greifen, um die Daten zu erhalten oder abzulegen, auf die Datenbank zu. Anschließend geben die Enterprise JavaBeans die Daten aus der Datenbank an die aufrufende Command-Klasse bzw. an den aufrufenden benutzerdefinierten Tag zurück. Von der Command-Klasse können die Informationen für die weitere Verarbeitung verwendet werden. Die benutzerdefinierten Tags nutzen diese Informationen, um sie anzuzeigen.

Für die Zwischenspeicherung von Daten mit einer langer Lebensdauer wird ein Cache eingesetzt. Der Grund hierfür liegt darin, dass der Zugriff auf eine Datenbank über die JDBC-Schnittstelle schneller ist als der Zugriff über Enterprise JavaBeans; der Cache ermöglicht es, bei der Nutzung von Enterprise JavaBeans für datenintensive Aufgaben akzeptable Antwortzeiten zu erreichen.

Der Aufbau einer HTML-Seite im Framework benötigt jeweils eine Reihe von Daten und es müssen wiederholt Enterprise JavaBeans aufgerufen werden, die dann auf die Datenbank zugreifen. Diese Aufrufe sind relativ zeitintensiv, und häufig werden Informationen abgefragt, die wenige Sekunden zuvor bereits abgefragt wurden. Da das Framework ein Shop-System realisiert, kommt hinzu, dass im Normalfall langlebige Informationen wie die Produktpalette, die Kategorien und eingegebene Übersetzungen existieren.

Abbildung 2: Architekturübersicht

Der Cache für das Framework liegt in der Java Virtual Machine, in der das Java Servlet und die JavaServer Pages ablaufen. Er ist so konstruiert, dass die Informationen über die Produkte, Kategorien und Übersetzungen nur einmal von den Komponenten angefordert werden. Bei jeder Anfrage einer im Cache verwaltbaren Information wird überprüft, ob diese im Cache liegt. Ist das der Fall, wird die Information aus dem Cache zurückgegeben. Anderenfalls wird die Information von den Komponenten angefordert, im Cache gespeichert und zurückgegeben. Werden Informationen geändert, die im Cache liegen, wird die veraltete Information im Cache gelöscht.

Der einzige Nachteil des Einsatzes des Caches besteht darin, dass die Ergebnisse der ersten Seitenaufrufe nach einem Neustart des Systems wesentlich langsamer zurückgeliefert werden als die nachfolgenden Aufrufe.

3.3 Komponenten des Frameworks

Das Framework setzt sich aus sechs Komponenten zusammen, die sich in zwei Gruppen einteilen lassen: Die Komponenten Sequence, Mail, Translation und Currency können unabhängig vom Shop-System auch bei anderen Entwicklungen eingesetzt werden. Die Komponenten Product und User dagegen stellen spezielle Funktionen für das Shop-System zur Verfügung. Abbildung 3 gibt einen Überblick über die Komponenten und über den funktionalen Zusammenhang zwischen ihnen. Für alle Komponenten gilt, dass sie aus einer oder mehreren Enterprise JavaBeans bestehen.

Die Struktur der Komponenten basiert auf dem *Façade Pattern* [Gran98, S. 205ff.]. Dieses sieht vor, dass eine Komponente mindestens eine Session Bean enthält, über die die Bestandteile der Komponente angesprochen werden. Durch diese vorgelagerte Session Bean werden logische Zusammenhänge zwischen den Entity Beans einer Komponente koordiniert. Ein direkter Zugriff auf die Entity Beans der Komponenten soll verhindert werden; dies erfolgt technisch über die Sicherheitseinstellungen beim Deployen der Enterprise JavaBeans.

Abbildung 3: Komponenten des Frameworks

Durch die *Sequence*-Komponente wird für jeden übergebenen Text eine eindeutige Zahl generiert, die bei jeder Übergabe dieses Textes um eins erhöht wird. Diese Funktionalität dient hauptsächlich zur Generierung von eindeutigen Primärschlüsseln für Tabellen der relationalen Datenbanken. Die entsprechende Funktion, die relationale Datenbanksysteme i. Allg. anbieten, kann beim Anlegen von Datensätzen über Container Managed Persistence Entity Beans nicht genutzt werden; denn Entity Beans benötigen den Primärschlüssel *vor* dem Anlegen eines Datensatzes. Daher wird der Schlüssel für die Entity Bean auf Anfrage von der Sequence-Komponente generiert.

Die Aufgabe der *Mail*-Komponente ist es, E-Mails zu verschicken. Sie besteht aus einer Stateful Session Bean, die die Funktionen des JavaMail APIs zur Versendung von Emails nutzt.

Die *Translation*-Komponente erlaubt es, das Framework mehrsprachig einzusetzen. Dies wird dadurch ermöglicht, dass jede sprachabhängige Information durch eine eindeutige Integerzahl repräsentiert wird. In der Translation-Komponente werden die jeweiligen sprachabhängigen Informationen zu dieser Integerzahl abgelegt. Jede Integerzahl kann bei Angabe einer konkreten Sprache über die Komponente in einen sprachabhängigen Text umgewandelt werden. Außerdem ist

die Translation-Komponente auch in der Lage, sprachabhängige Kategorien zu speichern und verwalten. Ein Beispiel für eine Kategorie, die in der Translation-Komponente gespeichert werden kann, sind Anreden in den jeweiligen Sprachen. Dadurch ist es möglich, alle gespeicherten Anreden in einer speziellen Sprache gesammelt abzufragen.

Die *Currency*-Komponente hat die Aufgabe, Wechselkurse zu speichern und Währungen umzurechnen. Basiswährung in dem Framework ist der Euro; für alle anderen Währungen wird der Umrechnungskurs zum Euro gespeichert. Betragswerte in den verschiedenen Währungen werden dynamisch aus dem gespeicherten Euro-Wert berechnet.

Die Aufgabe der *Product*-Komponente ist die Verwaltung des Produktkatalogs. Mit Hilfe der Komponente können Produktkategorien eingerichtet und in den Produktkategorien Produkte mit den Produktinformationen angelegt werden.

Von der *User*-Komponente werden die Funktionen angeboten, die in Verbindung mit den Nutzern des Shop-Systems stehen. Dazu zählen das Anlegen neuer Benutzer, die Verwaltung von Benutzergruppen und von Bestellungen.

3.4 Benutzerdefinierte Tags

Für das Framework sind benutzerdefinierte Tags entwickelt worden, die in den JavaServer Pages eingesetzt werden. Sie ermöglichen den direkten und einfachen Zugriff auf die Komponenten, und geben die unformatierten Daten von den Komponenten zurück, diese Daten werden in den folgenden Beispielen mit Hilfe von HTML-Befehlen formatiert.

Um die benutzerdefinierten Tags in einer JavaServer Page verwenden zu können, muss die Seite folgende Befehlszeilen enthalten:

```
<%@ page language="java" %>
<%@ taglib uri="/SHOP/TAGLIB/taglib.tld" prefix="shop" %>
```

Innerhalb der Seite können durch die Verwendung dieser Befehle die benutzerdefinierten Tags mit dem Präfix „shop“ angesprochen werden.

Im Folgenden werden, stellvertretend für die insgesamt 35 entwickelten benutzerdefinierten Tags, der Tag zur Behandlung von Übersetzungen und die Tags zur Anzeige von Produktkategorien und Produkten erläutert.

1. Der Tag <translate>

Der <translate>-Tag wird für mehrsprachige JavaServer Pages verwendet. Mit dem <translate>-Tag können Texte in einer JavaServer Page dynamisch an eine vom Benutzer gewählte Sprache angepasst werden. Der Tag nutzt die Funktionen der Translate-Komponente. Über das Administrationstool des Frameworks können sprachunabhängige Textbausteine angelegt werden. Ein Textbaustein bekommt

einen Namen und der Text des Textbausteins wird in den Sprachen angegeben, die dargestellt werden sollen. Für das unten gezeigte Beispiel ist ein Textbaustein mit dem Namen „sampletext“ erforderlich. Um die angezeigten Ergebnisse zu erhalten, muss als deutscher Text „Beispiel“ und als englischer Text „sample“ eingegeben werden.

Der <translate>-Tag kann auf zwei Arten genutzt werden: Entweder wird der Name des Textbausteins zwischen den Start- und den Ende-Tag geschrieben oder über den Parameter „translate“ festgelegt. Optional kann über den Parameter „language“ ein Sprachcode für die Zielsprache angegeben werden. Fehlt der Parameter, wird zuerst versucht, den Text in einer vom Benutzer ausgewählten Sprache anzuzeigen. Hat der Benutzer noch keine Sprachauswahl getroffen, wird versucht, den Text in einer bevorzugten Sprache des Browsers anzuzeigen. Ist dies auch nicht möglich, wird die bei der Konfiguration des Frameworks bestimmte Standardsprache für den auszugebenden Text genutzt.

Das folgende Beispiel verdeutlicht die Funktionsweise des Tags:

```
Browsersprache: <shop:translate>sampletext</shop:translate><br>
Englisch: <shop:translate language="EN">sampletext</shop:translate><br>
Deutsch: <shop:translate language="DE">sampletext</shop:translate><br>
alternativ:<br>
Browsersprache: <shop:translate translate="sampletext"/><br>
Englisch: <shop:translate translate="sampletext" language="EN"/><br>
Deutsch: <shop:translate translate="sampletext" language="DE"/>
```

Das Ergebnis beim Aufruf durch einen Browser mit der vom Benutzer eingestellten bevorzugten Sprache Deutsch ist:

Browsersprache: Beispiel
Englisch: sample
Deutsch: Beispiel
alternativ:
Browsersprache: Beispiel
Englisch: sample
Deutsch: Beispiel

2. Die Tags <category>, <categoryinternalnr>, <categoryname>

Mit Hilfe des <category>-Tag werden alle direkten Unterkategorien einer festgelegten Kategorie angezeigt. Der Tag wird für jede in einer Kategorie enthaltene Unterkategorie einmal durchlaufen.

Innerhalb des <category>-Tags kann mit den Tags <categoryinternalnr> und <categoryname> auf die interne Nummern der Unterkategorien beziehungsweise auf die Unterkategorienamen zugegriffen werden. Der Kategoriename wird entweder in der bevorzugten Sprache des Browsers angezeigt oder - wenn diese Sprache nicht vorhanden ist - in der Standardsprache des Frameworks.

Beispielsweise liefert

```
<shop:category>
<shop:categoryinternalnr/> <shop:categoryname/><br>
</shop:category>
```

das Ergebnis:

176 Sonderangebote
178 Bücher
181 Haushaltswaren

3. Die Tags <productcategory>, <product>, <producttext>, <productpicture>, <productprice>, <productinternalnr>

Der <productcategory>-Tag wird für jedes Produkt einer Produktkategorie einmal durchlaufen.

Innerhalb des <productcategory>-Tags kann mit dem <product>-Tag auf das Produkt des aktuellen Durchlaufs zugegriffen werden. Mit dem Tag <producttext> wird über den Parameter „textnr“ die Nummer des Textes ausgewählt, der ausgegeben werden soll. Hierbei wird versucht, den Text in einer der bevorzugten Sprachen, die der Benutzer im Browser eingestellt hat, anzuzeigen. Ist dies nicht möglich, wird die Standardsprache des Shop-Systems genutzt.

Um Bilder, die zu einem Produkt gehören, zu erhalten, wird der <productpicture>-Tag verwendet. Der Tag erhält als Parameter „picturenr“ die Nummer des Bildes übergeben, das angezeigt werden soll. Der Tag liefert einen Link auf das Bild zurück; der Seitenersteller kann sich dann mit Hilfe von HTML-Befehlen das Bild anzeigen lassen.

Zur Ausgabe des Preises eines Produkts dient der <productprice>-Tag. Er versucht, über die Browsereinstellungen die Sprache und die Währung des Benutzers zu ermitteln und rechnet den gespeicherten Produktpreis in die Währung um. Kann die Währung nicht ermittelt werden, wird der Preis in Euro, der Standardwährung des Shop-Systems, angezeigt.

Die interne Nummer eines Produkts erhält man über den Tag <productinternalnr>.

Der <product>-Tag kann auch eigenständig ohne den <productcategory>-Tag eingesetzt werden. Dazu muss dem Tag mitgeteilt werden, auf welche Produktnummer er sich beziehen soll. Dazu dient der Parameter „productid“, dem die interne Produktnummer zuzuweisen ist.

Die Tags

```
<shop:productcategory categoryid="178">
<shop:product>
<hr>
<h1><shop:producttext textnr="1"/></h1>
```

```
<shop:producttext textnr="2"/>
<br>
<image src="<shop:productpicture picturenr="1"/>" height="100" width="100">
<image src="<shop:productpicture picturenr="2"/>" height="75" width="75">
<shop:productprice/>
</shop:product>
</shop:productcategory>
```

können beispielsweise ausgeben:

4 Fazit

Die Entwicklung des hier präsentierten Frameworks hat gezeigt, dass auf der Basis der J2EE Plattform, die u.a. über kostenlose Produkte realisiert werden kann, eine kostengünstige und einfach zu nutzende Alternative zu den kommerziellen Shop-Out-of-the-Box-Lösungen bereitgestellt werden kann. Neben den vergleichsweise geringen Kosten liegt ein weiterer wesentlicher Vorteil des Frameworks in der Trennung von den Aspekten der Anwendungsentwicklung auf der einen und den Aspekten der Gestaltung eines Shop-Systems auf der anderen Seite: Aufbauend auf den durch das Framework definierten Tags können Web-Designer ohne Programmierkenntnisse dynamische Shop-Systeme erstellen. Die Designer müssen lediglich zusätzliche Tag erlernen, die Sie bei der Codierung der HTML-Seiten zur Realisierung des Shops nutzen können. D.h., die Tags bilden eine voll-definierte und einfache Schnittstelle zwischen den Anwendungsentwicklern und den Web-Designern.

Das entwickelte Framework bietet – zumindest in der aktuellen Version – noch nicht die Funktionalität der verfügbaren kommerziellen Shop-Out-of-the-Box-Lösungen. Es stellt jedoch die grundlegende Funktionalität eines Shops zur Verfügung; individuelle Shops können schnell eingerichtet werden.

Sicherlich bestehen noch grundsätzliche Probleme bei der Nutzung der J2EE-Plattform; insbesondere lässt die Spezifikation Interpretationsspielräume, die dazu führen, dass J2EE-Anwendungen, auch wenn sie spezifikationskonform programmiert wurden, trotzdem u.U. an den eingesetzten J2EE Application Server angepasst werden müssen. Die im Rahmen der Studie getesteten Server erwiesen sich darüber hinaus teilweise als nicht vollkommen spezifikationskonform. Die Komponenten des Frameworks, die für den WebSphere Application Server 3.0 unter strikter Berücksichtigung der Spezifikationen entwickelt wurden, konnten jedoch problemlos auf den WebLogic Server der Version 5.1 portiert werden, der ein J2EE-Kompatibilitätszertifikat von Sun Microsystems besitzt [Java00, S. 6].

Auch die Performance-Schwächen der Plattform konnten bei der Entwicklung des Frameworks weitestgehend kompensiert werden: Durch den Einsatz des Cache

Systems erzielt ein auf der Basis des Framework entwickeltes Shop-System sehr gute Antwortzeiten.

Literatur

[Ayer99] Ayers, D.: Professional Java Server Programming, Birmingham, 1999.

[Bibi00] Firma Bibit Internet Payments Inc.: www.bibit.com, Aufruf am 2001-03-01.

[Cybe00] Firma Cybercash GmbH: www.cybercash.de, Abruf am 2001-03-01.

[Denn00] Denninger, S.; Peters, I.: Enterprise JavaBeansTM, München 2000.

[Fiel00] Fields, D.K.; Kolb, M.A.: Web Development with JavaServer Pages, Greenwich 2000.

[Gran98] Grand, M.: Patterns in JavaTM, Volume 1, New York 1998.

[Mate99] Matena, V.; Hapner, M.: Enterprise JavaBeansTM Specification, v1.1 vom 17.12.1999, Palo Alto 1999.

[Merk00] Merkle, B.: Die Geister, die ich rief – Tutorium zur Programmierung von Enterprise-JavaBeans. In: Java Spektrum (2000) Mai/Juni, S. 47-52.

[Mons00] Monson-Haefel, R.: Enterprise JavaBeansTM Second Edition, Sebastopol, 2000.

[Moss98] Moss, K.: Java Servlets, New York 1998.

[Java00] O.V.: WebLogic besteht J2EE-Test. In: Java Magazin (2000) September, S. 6.

[Roma99] Roman, E.: Mastering Enterprise JavaBeansTM and the JavaTM 2 Platform Enterprise Edition, New York 1999.