Webanwendungen über XML-Spezifikationen

– Verdeutlicht am Beispiel eines holländischen Auktionssystems –

Inaugural-Dissertation zur Erlangung des Grades
eines Doktors der Wirtschaftswissenschaften
der Sozial- und Wirtschaftswissenschaftlichen
Fakultät der Ruprecht-Karls-Universität
Heidelberg

vorgelegt von

Daniela Stoykova
aus Kazanlak, Bulgarien

April, 2005

Abkürzungsverzeichnis

A2B	Administration-to-Business
A2C	Administration-to-Consumer
API	Application Programming Interface
ASP	Active Server Pages
B2B	Business-to-Business
B2C	Business-to-Consumer
B2E	Business-to-Employee
B2M	Business-to-Marketplace
C2C	Consumer-to-Consumer
CGI	Common Gateway Interface
CORBA	Common Object Request Broker Architecture
CSS	Cascading Style Sheets
CSV	Comma Separated Values
DARPA	Defense Advanced Research Projects Agency
DB	Database
DBMS	Database Management System
DCL	Data Control Language
DCOM	Distributed Component Object Model
DIN	Deutsche Industrie Norm
DNS	Domain Name System

DOM	Document Object Model
DSSSL	Document Style Semantics and Specification Language
DTD	Document Type Definition
eCRM	electronic Customer Relationship Management
ER	Entity Relationship
FO	Formatting Objects
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
ISO	International Organization for Standardization
JAR	Java Archive
JDBC	Java Database Connectivity
JSP	Java Server Pages
JVM	Java Virtual Machine
JWS	Java Web Service
P2P	Peer-to-Peer
PDA	Personal Digital Assistant
PDF	Portable Document Format
RFC	Request for Comments
RPC	Remote Procedure Call
S2E	System-to-Employee
S2S	System-to-System
SAX	Simple API for XML Access
SGML	Standard Generalized Markup Language

SHTTP	Secure Hypertext Transfer Protocol (https im Browser)
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
TCP	Transmission Control Protocol
UBR	UDDI od. Universal Business Registry
UDDI	Universal Description, Discovery, and Integration
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WML	Wireless Markup Language
WSDD	Web Service Deployment Descriptor
WSDL	Web Services Description Language
WWW	World Wide Web
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language
XPath	XML Path Language
XSL	Extensible Stylesheet Language
XSL-FO	Extensible Stylesheet Language - Formatting Objects
XSLT	XSL Transformation Language

Danksagung

Die vorliegende Arbeit wurde im Sommersemester 2005 eingereicht. An dieser Stelle möchte ich mich bei allen Personen bedanken, die mich während der Zeit der Promotion unterstützten.

Zuerst möchte ich mich bei Herrn Professor Dr. Roland Fahrion bedanken, der die wissenschaftliche Betreuung der Arbeit übernahm, sich Zeit für fachliche Gespräche genommen hat und damit in vielfacher Hinsicht diese Arbeit beeinflusste. Herrn Professor Dr. Gerhard Reinelt möchte ich herzlich für die Übernahme des Zweitgutachtens danken.

Danken möchte ich Joachim Baier für Korrekturlesen, für inhaltliche und vor allem sprachliche Diskussionen und Hinweise, von denen ich sehr viel profitieren konnte.

Dankbar bin ich Ani Guerdjikova, Corinna Klett und Andreas Grossmann für das Lesen einzelner Teile der Arbeit. Alexander Bub möchte ich für die fachlichen Gespräche in der Anfangszeit der Erstellung dieser Arbeit danken, von denen ich ebenso viel profitiert habe.

Unendlich dankbar bin ich meiner Familie für die moralische Unterstützung während dieser ganzen Zeit. Endlos dankbar bin ich meinem Lebensgefährten Riccardo Amato, der während der Erstellung dieser Arbeit stets an meiner Seite stand und mich in jeder Hinsicht unterstützte.

Daniela Stoykova, im April 2005

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Kapitel 1

Einleitung

1.1 Motivation und Ausgangssituation

Die Technik- und Computerwelt zeigt, von ihrem Ursprung aus betrachtet, einen klaren Trend zur Vernetzung. So war das Internet ein revolutionäres Ereignis, das die Beschleunigung des Austausches von statischen Daten ermöglicht hat. Die heutigen Mittel und Wege zur Datenübertragung wachsen ständig, womit auch die zu übertragende Informationsquantität wächst. Auch die Ansprüche an die Datenverarbeitung haben sich geändert: Daten jeder Art müssen orts- und zeitunabhängig zur Verfügung stehen. Das erfordert beispielsweise die Publikation der Daten in mehreren Medien. Es ist heutzutage selbstverständlich, Produkte oder Dienstleistungen jeder Art auf elektronischem Wege zu kaufen, was wiederum Kommunikationswege zwischen dem Käufer und Verkäufer erfordert.

Die Möglichkeit, diese Art von Kommunikation auf technischem Wege zu realisieren, wurde besonders schnell durch die Industriewelt erkannt. Die Entstehung von Technologien wie CGI[1] (Common Gateway Interface) haben den Datenaustausch in beiden Richtungen, vom Client[2] zum Server[3] und vom Server zum Client, ermöglicht. Darauf folgend kamen weitere Technologien wie JSP[4] (Java Server Pages) zum Einsatz. Sie ermöglichten den Austausch dyna-

[1] „CGI ist der Teil des Web-Servers, der mit anderen Programmen, die auf dem Rechner laufen, kommunizieren kann. Mit CGI kann der Web-Server ein Programm aufrufen und dabei benutzerspezifische Daten ... übergeben. Das Programm verarbeitet dann die Daten, und der Server übergibt die Antwort des Programms an den Web-Browser.“ (Gundavaram, 1996, S. 1)

[2] Client: Eine Anwendung, die die zur Verfügung gestellten Dienste eines Servers in Anspruch nimmt.

[3] Server: Ein Rechner, der an einem Netzwerk angeschlossen ist und allen angeschlossenen Rechnern Daten zur Verfügung stellt.

[4] „JavaServer Pages (JSP) technology provides a simplified, fast way to create dynamic web content. JSP technology enables rapid development of web-based applications that are server- and platform-independent.“ (Sun, 2004)

mischer Daten zwischen dem Server und mehreren Clients. Die Realisierung von Webanwendungen mit herkömmlichen Werkzeugen brachte nach Grüne und Kneuper (2002, S. 269) eine Reihe von Nachteilen:

- Fehlende Trennung der Daten von deren Präsentation, was zu weiteren Wartungsarbeiten führt.

- Die Skalierbarkeit der traditionellen Webanwendungen ist sehr gering und erfordert enorme manuelle Änderungen.

Trotz der zur Verfügung stehenden technischen Möglichkeiten ist der Aufwand für die Verarbeitung der Daten sehr hoch. XML (Extensible Markup Language) ist das Werkzeug, das die Lösung der obengenannten Probleme verspricht. Es ist bereits viel Software entwickelt worden, die mit dem Akronym XML vertrieben wurde. Es muss daher die Unterscheidung zwischen den Software-Produkten, die nur eine XML-Konfigurationsdatei benutzen und diejenigen, bei denen der komplette Datenaustausch in XML abläuft, gemacht werden.

Motivation dieser Arbeit ist die Konzeption und Implementierung einer Auktions-Webanwendung für Holländische Auktionen mit Hilfe von XML-Technologien. Dabei soll eine Trennung von Daten, deren Präsentation und der Anwendungslogik erzielt werden. Damit wird die Plattformunabhängigkeit der Anwendung ermöglicht. Außerdem wird auch die Skalierbarkeit gewährleistet, da einzelne Programmblöcke auch auf eigenen Systemen funktionsfähig sind.

1.2 Aufgaben der Arbeit

Ziel dieser Arbeit ist es, den möglichen Einsatz des Standards XML in Webanwendungen zu untersuchen. Ein wesentlicher Bestandteil der Arbeit ist die Entwicklung eines Prototyps, der die praktische Umsetzung eines Großteils der Technologien veranschaulicht.

Im Rahmen dieser Arbeit werden Einsatzbereiche wie Bearbeitung, Transformation und Präsentation von XML-Dokumenten näher betrachtet. Weiterhin soll der Einsatz von XML zur Anwendungsintegration kurz angegangen werden. Dabei werden die wichtigsten Standards von Web Services-Technologien herausgearbeitet.

Diese Aspekte werden anhand des Beispiels holländischer Auktionen im Pro-

totyp untersucht. Als Repository[5] für die Daten wird eine relationale Datenbank[6] eingesetzt. Der Datenaustausch wird zum größten Teil mit dem XML-Datenaustauschformat realisiert. Es wird eine Öffnung des Systems für andere Umgebungen durch Web Services ermöglicht. Die Integration eines externen Web Services (google.de) soll in der Arbeit ebenso veranschaulicht werden.

1.3 Aufbau der Arbeit

Die vorliegende Arbeit ist in sechs Kapitel untergliedert.

Das zweite Kapitel beschäftigt sich mit dem ökonomisch-theoretischen Teil der Arbeit. Es stellt den Weg der Entwicklung des elektronischen Handels im Web und seine treibenden Faktoren sowie die existierenden Geschäftsmodelle vor. Die Entwicklung des Auktionssystem-Prototyps veranlasst die Behandlung der Auktionsarten, ihrer Funktionen, Vorteile, Nachteile und des Geschäftsmodells der holländischen Auktion. Anschließend werden die Einsatzmöglichkeiten der XML-Technologie in Webanwendungen geschildert. Dabei werden die wichtigsten Standards angesprochen sowie Vor- und Nachteile von XML und seine Einsatzbereiche in Unternehmen.

Kapitel drei stellt die Architektur der holländischen Auktionssystem-Anwendung dar. Zunächst werden der allgemeine Aufbau und angewendete Technologien von Webanwendungen erläutert. Es folgen allgemeinen Anforderungen an Webanwendungen sowie konkreten Anforderungen an die zu entwickelnde Auktionssystem-Webanwendung. Anschließend wird Struktur und Aufbau des Prototyps ausführlich dargestellt. Dabei werden der Aufbau der Datenbasis und die Geschäftsprozesse der Anwendung näher betrachtet.

Kapitel vier ist den Einsatzmöglichkeiten der XML-Familie und ihrer Implementierung in die Webanwendung gewidmet. Dabei werden die wichtigsten Merkmale der Spezifikationen beschrieben und mit Beispielen aus der Webanwendung veranschaulicht.

[5] Repository: Eine stukturierte Ansammlung von Daten, auf welche Selektionsabfragen durchgeführt werden können.

[6] Eine relationale Datenbank ist eine Datenbank, die auf dem relationalen Datenmodell basiert. Die Daten werden dabei in Form von zweidimensionalen Tabellen verwaltet, die über Schlüssel (Primärschlüssel, Fremdschlüssel) miteinander verknüpft werden können. Vgl. (Wikipedia, 2004c).

Der Einsatz des Standards XML beschränkt sich nicht nur auf eine Anwendung, sondern kann auch anwendungsübergreifend stattfinden. Kapitel 5 geht näher auf den Begriff von Web Services und den Integrationsaspekt der Technologie mit ihren Eigenschaften und Vorteilen ein. Hier werden das Web Services-Modell und seine wichtigsten Standards beschrieben. Der Einsatz der Technologie wird mit Beispielen aus der Anwendung veranschaulicht.

Kapitel 6 gibt eine zusammenfassende Schlussbetrachtung und einen weiterführenden Ausblick.

Ein Supplementum, der dieser Arbeit beigelegt wird, beschreibt die Installation der für die Implementierung der Anwendung benötigten Software. Es werden hier sowohl Quellen als auch Installationseinstellungen genannt. Der Quelltext der in der Arbeit beschriebenen Klassen ist an dieser Stelle ebenso zu finden.

Kapitel 2

Integration von Geschäftsprozessen in webbasierten Anwendungen

> *„Wir befinden uns an der Schwelle zu einem Zeitalter vernetzter Intelligenz – einem Zeitalter, das eine neue Wirtschaft, eine neue Politik und eine neue Gesellschaft hervorbringen wird.“*
> *(Tapscott, 1996, S. 18)*

Dieses Kapitel widmet sich der Rolle der technologischen Entwicklung in der Wirtschaft. Dabei wird zuerst der in den letzten Jahren an Bedeutung ständig zunehmende elektronische Handel mit seinen Eigenschaften, Formen und seinem Einfluss auf die traditionelle Ökonomie und Gesellschaft analysiert. Als zweites wird auf Auktionssysteme allgemein und auf die Geschäftsform eines holländischen Auktionssystems näher eingegangen. Der dritte Abschnitt behandelt schließlich die Rolle der Standards in der Internet-Ökonomie und besonders den Standard XML mit seinen Möglichkeiten und dem Einsatz im Unternehmensumfeld. Somit wird mit diesem Kapitel die Grundlage für spätere Kapitel geschaffen, welche auf die technische Entwicklung des XML-unterstützten Auktionssystems genauer eingehen.

2.1 Der Weg zum elektronischen Handel

In diesem Abschnitt wird die Bedeutung des Internets und sein Einfluss auf die Wirtschaftswelt verdeutlicht. Es wird dabei auf geschichtliche Fakten, Zahlen und Definitionen eingegangen. Weiterhin wird der Schwerpunkt auf den elektronischen Handel im Web und auf die Gründe für Käufer und Verkäufer, Transaktionen über das Internet abzuwickeln, gelegt. Anschließend wird auf die Charakteristika eines Geschäftsmodelles und auf mögliche Geschäftsmodelle elektronischen Handels eingegangen.

2.1.1 Von der Industriellen zur Digitalen Revolution

Ein Rückblick auf die Geschichte der Wirtschaft zeigt die Veränderungen, denen sowohl der Mensch als Humankapital-Faktor und Konsument als auch die gesamte Wirtschaft und deren Unternehmen unterlagen. (Stähler, 2002, S. 21 ff.) Diese Veränderungen führten dementsprechend zum Wandel der Gesellschaft und ebenso der Weltwirtschaft. Der Beweggrund dafür ist der Mensch und sein natürlicher Ehrgeiz: „Innovation und Fortschritt, das Streben nach Neuem, dem Unerreichbaren haben seit Jahrhunderten die Entwicklung der Menschheit bestimmt." (Purschke und Wurdack, 2000, S. 258)

Die erste Industrielle Revolution wurde Ende des 18. Jahrhunderts durch die Erfindung der Dampfmaschine von James Watt ausgelöst. Von England aus verbreitete sich diese nach USA und Europa und anschließend nach Russland und Japan. Die Nutzung der mechanischen Kraft der Dampfenergie führte zum Bau von Maschinen und Fabriken, die auch entfernt von den Ressourcen der Produktion gebaut werden konnten. Diese Periode zeichnete sich durch das Ersetzen der handwerklichen Arbeit durch industrielle Fabrikation aus. Innerhalb der Fabriken ermöglichte die Nutzung der Maschinenkraft eine fortschreitende Veränderung der Handarbeit - in Kombination mit Arbeitsteilung und Rationalisierung maschineller Fertigungsabläufe. Die Industielle Revolution hatte auch eine bedeutende Auswirkung auf die Gesellschaft und die gesellschaftlichen Strukturen. Es entstanden die Arbeiter auf der einen Seite und die Unternehmer auf der anderen Seite.

Das Ende des 19. Jahrhunderts ist als Periode der zweiten Industriellen Revolution bekannt. Sie zeichnet sich durch die Erfindung von Telegraph und Telefon aus. Der darauf folgende schnelle Austausch von Informationen wurde von Entwicklungen ermöglicht, die revolutionär für die damalige Zeit und Vorstellungen waren.

Im 20. Jahrhundert hat der Faktor Wissen und Information in der Wirtschaft gewaltig an Bedeutung zugenommen. „Bereits heute sind beinahe 60 Prozent aller amerikanischen Beschäftigten Geistesarbeiter, und acht von zehn neuen Jobs werden in informationsintensiven Wirtschaftssektoren geschaffen. Die heutige Fabrik unterscheidet sich ebenso von der industriellen Fabrik der alten Wirtschaft, wie sich die alte Fabrik von der handwerklichen Produktion davor abgehoben hat." (Tapscott, 1996, S. 24) Als Pendant zu der Entwicklung der Dampfmaschine, des Telegraphs und der Eisenbahn kann das Entstehen des Internets als ein bewegender Faktor für die dritte, digitale Revolution bezeich-

net werden. Die Innovation Internet konnte daher zu einem hohen Maße die Weltwirtschaft beeinflussen. Sie ermöglichte die Bereitstellung von Informationen, was die Notwendigkeit von Faktenwissen nicht mehr relevant machte. Purschke und Wurdack (2000, S. 259) heben daher nicht nur das Wissen allein hervor, sondern die Bedeutung von „Wissen um Methoden, Ressourcen, Quellen und Konzepte“.

Eine Betrachtung der Marktprognosen hinsichtlich der Entwicklung des elektronischen Handels im Internet allein in den letzten Jahren macht die weiterhin wachsende Relevanz der Informationstechnologien in der Wirtschaftswelt deutlich. Besonders große Bedeutung hat die Entwicklung des elektronischen Handels im Business-to-Business-Bereich, was anhand der Tabelle 2.1 verdeutlicht wird.

Tabelle 2.1: Vergleichende Schätzungen des B2B E-Commerce in Milliarden US-Dollar, Quelle: NRW Medien GmbH (2003, S. 3)

Europa: Vergleichende Schätzungen B2B E-Commerce in Milliarden US-Dollar 2000-2005						
	2000	2001	2002	2003	2004	2005
eMarketer, 2001	26,2	52,4	132,7	334,1	797,3	-
Durlacher Research	76,0	159,0	366,0	766,0	1.272,0	-
Forrester Research	79,7	177,2	385,0	784,6	1.425,6	-
Gartner Group	-	177,0	-	-	-	1.641,0
Goldman Sachs (EU)	103,9	227,7	428,1	673,5	1.024,6	1.516,0
IDC	57,4	139,5	267,8	480,9	858,0	1.465,3
Monitoring Informationswirtschaft, 10/2002						

Die Schätzungen der Marktforschungsunternehmen sind sehr unterschiedlich. So liegt die Zahl von eMarketer für das Jahr 2000 bei 26,2 und von Goldman Sachs (EU) bei 103,9 Milliarden US-Dollar. Eines ist der Tabelle von Zahlen gemeinsam: Sie alle deuten darauf hin, dass der Trend beim elektronischen Handel im Business-to-Business-Bereich nach oben strebt. So zeichnet ein Vergleich der Prognosen von Goldman Sachs (EU) eine über 14-fache Erhöhung des B2B-Handelsvolumens allein in der Zeitspanne von 2000-2005. Die Abbildung 2.1 der NRW Medien GmbH verdeutlicht diesen Trend.

Die Entwicklung der Informations- und Kommunikationstechnologien ermöglicht die Automatisierung von Geschäftsprozessen. Die Zustellung von Infor-

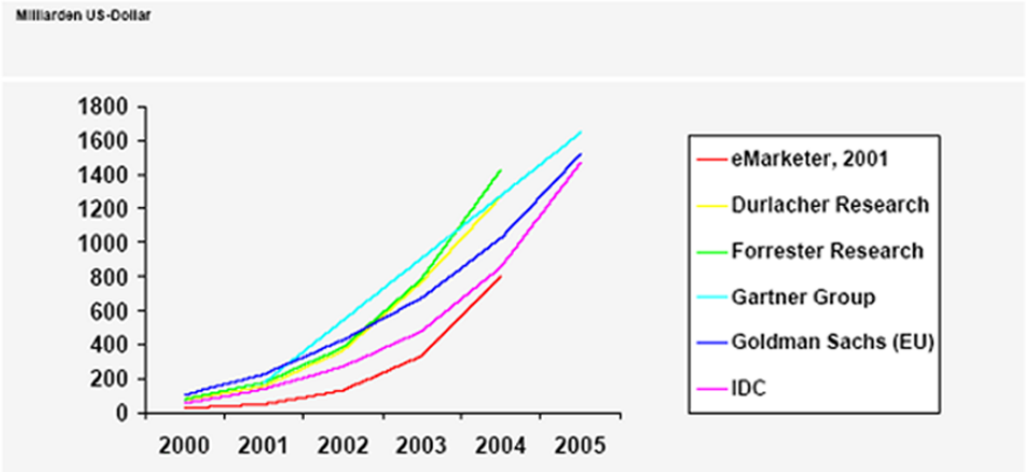

Abbildung 2.1: Europa: Vergleichende Schätzungen B2B E-Commerce, Quelle: NRW Medien GmbH (2003, S. 3)

mationen bereitet kein Problem mehr, da das Internet die Bereitstellung jeglicher Information weltweit ermöglicht. Dieses Phänomen prägt nicht nur das einzelne Unternehmen, sondern auch die ganze Wirtschaft. Es entstehen neue Geschäftsmodelle und neue Vertriebskanäle. Die Unternehmen nutzen das Internet und seine Vorteile sowohl zum Absatz ihrer Produkte als auch für den Beschaffungsprozess beim Lieferanten. Anschließend hat das Medium Auswirkungen auf das Konsumverhalten Endverbraucher. Die Endkunden nutzen heute das Internet als Quelle für Informationen, um sich über alle denkbaren Produkte und Dienstleistungen zu informieren, die verfügbaren Angebote bequem zu vergleichen und per E-Mail Kontakt mit den Anbietern aufzunehmen. Weiterhin haben die Endverbraucher die Möglichkeit, die gewünschten Produkte und Dienstleistungen über das Internet zu kaufen.

Das Medium Internet findet eine hohe Akzeptanz und wird immer intensiver im alltäglichen Leben eingesetzt. Nach Hofmann (2002, S. 29) ist die Zahl der Internet-Nutzer von 407,1 Millionen im November 2000 auf 513,4 Millionen im Oktober 2001 gestiegen. Für 2002 soll die Anzahl der Nutzer nach Nielsen/NetRatings 580 Millionen betragen; Nach dem International Telecommunications Union beträgt ihre Zahl für 2002 sogar 665 Millionen. (Lyman u.a., 2003) Trotz der unterschiedlichen Quellen der Daten deuten die Zahlen auf einem kontinuierlichen Anstieg der Internetnutzer hin. Sie setzen dabei das Medium Internet nicht nur ein, um Informationen über das Internet zu be-

schaffen und auszutauschen, sondern auch, um Güter zu kaufen, zu verkaufen und sogar um persönliche Finanztransaktionen abzuwickeln.

Der Grund für die weite Verbreitung des Internets als Medium in der Wirtschaft ist die Fähigkeit zur Informationsbereitstellung unabhängig von Ort und Zeit. Shapiro und Varian (1999, S. 3) definieren Information als „anything that can be digitalized – encoded as stream of bits". Jedes wirtschaftliche Geschäft basiert auf Informationsaustausch. Sie ist „der Leim, der die Struktur aller Geschäfte zusammenhält" (Evans und Wurster, 2000, S. 44). Evans und Wurster (2000, S. 45 f.) geben eine mathematische Definition für die Wirtschaftlichkeit der Informationen:

$$\textit{Wirtschaftlichkeit der Information} = \frac{\textit{Informationsfülle}}{\textit{Reichweite}}$$

Sie geben folgende Definitionen für die Begriffe Informationsfülle und Reichweite. Die *Reichweite* impliziert die Anzahl der Personen, die Informationen austauschen. Die *Informationsfülle* wird durch folgende drei Aspekte der Information erklärt.

- *Informationsbandbreite*: Bezieht sich auf die Bandbreite der Information, die transportiert werden muss. Je größer die Datenmenge ist, desto größer muss die Bandbreite für den Transport der Daten sein.
- *Individualisierung*: Bezieht sich auf das Niveau der persönlichen Ansprüche der Personen, die durch die Information erreicht werden. Je kleiner die Anzahl der Zielpersonen sind, desto größer ist die Persönlichkeit der Ansprache.
- *Interaktivität*: Bezieht sich auf die Ermöglichung der Interaktion. Je größer die Gruppe ist, desto mehr eignet sich ein Monolog für die Kommunikation.

2.1.2 Der elektronische Handel im Web

Die Vernetzung der Computerwelt ergibt neue Möglichkeiten und Chancen für die Wirtschaft der heutigen Zeit. Eine besondere Bedeutung wird dabei neben dem Internet auch dem Intranet[1] und Extranet[2], aber auch der Entwicklung

[1] Intranet ist die „...Verbindung von unternehmensinternen lokalen Netzen..." (Kauffels, 2001, S. 145).

[2] „Ein Extranet ist eine Menge geschützter privater Verbindungen zwischen Teilen eines Unternehmens oder zwischen einem Unternehmen und anderen Unternehmen (wie z.B. Zulieferern)" (Kauffels, 2001, S. 187).

der mobilen Technologien zugeschrieben.

Der Begriff des Internets hat in der heutigen Welt eine enorme Ausprägung. Es wird als „Datenautobahn“, als „Information Highway“ (Kauffels, 2002, S. 337) oder als „Datenhighway“ (Tapscott, 1996, S. 132) bezeichnet. Etwas technischer betrachtet ist das Internet ein Netz der Netze, „...auf welchem die Datenübertragung paketweise gemäß dem Internet Protokoll (IP) erfolgt...“. (Heinzmann, 2002, S. 43)

Jeder Rechner im Internet hat eine eigene IP-Adresse, welche im gesamten Internet eindeutig ist. Die Kommunikation mit dem jeweiligen Rechner wird über die IP-Adresse realisiert. Um den Zugriff auf die Rechner im Internet zu erleichtern, werden ihnen auch eindeutige Namen vergeben. Die Zuordnung der Namen der Rechner auf deren IP-Adressen wird durch ein DNS (Domain Name System) durchgeführt. Das Suchen der Zielrechner beim Versenden von Datenpaketen im Internet wird durch Router[3] unterstützt. Ein Datenpaket läuft in der Regel von der Quelle bis zum Ziel über mehrere Knoten. Bei jedem Knoten werden die möglichen Wege bis zum Zielrechner neu berechnet. Das Datenpaket wird über den kürzesten Weg versendet.

Eigenschaften des Internets

Das Internet prägt als ein unverzichtbares Medium im heutigen Unternehmen nicht nur seine Wertschöpfungskette, sondern auch das gesamte Wertschöpfungsnetzwerk bei der Entstehung eines Produktes oder einer Dienstleistung. So bringt beispielsweise Intervention in jeder weiteren Transaktion bei der Wertschöpfung Transaktionskostenvorteile. (Skiera und Spann, 2002, S. 691) Diese Vorteile können in den Internet-Eigenschaften Multimedialität, Interaktivitat, Digitalität, gesucht werden. Sie können als Ursache des Internet-Einsatzes bezeichnet werden. (Zwerger und Paulus, 2002, S. 26 ff.)

- *Digitalität*: Die Möglichkeit, Computer in Netzwerken zu verbinden sowie diese über Server an das Internet anzuschließen, ermöglicht die Bildung einer technischen Welt, in welcher keine Medienbrüche beim Austausch von Informationen vorhanden sind. Diese Eigenschaft ermöglicht eine vollständige Abwicklung des Wertschöpfungsprozesses in Echtzeit auf digitaler Ebene mit enormen Transaktionskostenersparnissen.

[3] „Ein Router ist ein Vermittlungsrechner, der am Aufbau einer Verbindung in einem Computernetz mit Paketvermittlung, zum Beispiel dem Internet, beteiligt ist. Solche Rechner leiten ... die Datenpakete anhand der Adresse eines route-fähigen Protokolls wie z.B. TCP/IP zum jeweiligen Zielrechner.“ (net-lexikon.de, 2004c)

- *Interaktivität*: Das Internet zieht theoretisch keine Grenzen auf der Interaktionsebene. So ist z.B. die physische Teilnahme an einem Seminar oder an einer Auktion auf die Größe eines Raumes beschränkt. Das Angebot solcher Veranstaltungen auf einer Internet-Plattform ermöglicht jedem die Teilnahme, der auch Zugang zum Medium hat. Die sich dadurch ergebende Ortsunabhängigkeit der Angebote erbringt Vorteile sowohl für die Anbieter- als auch für die Nachfragerseite. Eine Online-Auktion beispielsweise erfordert keine physische Anwesenheit der Parteien, schließt eine Beschränkung der Teilnehmeranzahl aus und erhöht damit die Zahl der teilnehmenden Bieter. Die Nachfrager auf der anderen Seite genießen die Möglichkeit, ortsunabhängig an der Auktion teilzunehmen, was ihnen zusätzlich Fahrtkosten und Zeit erspart. Die erhöhte Preistransparenz bietet ihnen die Möglichkeit, ihr höchstes Gebot genau zu überlegen. Die Auktionsteilnehmer müssen unter Umständen auch nicht gleichzeitig alle verfügbar sein, da das Internet nicht nur synchrone, sondern auch asynchrone Kommunikation ermöglicht.

- *Multimedialität*: Die vom Internet zur Verfügung gestellten Informationen sind zu jeder Zeit verfügbar und können alle zeitgleich in Medien wie Computer, Handy oder PDA[4] (Personal Digital Assistant) dargestellt werden. Das verstärkt die Möglichkeit, ortsunabhängig auf Informationen zugreifen zu können. So kann der Bieter für eine Auktion beispielsweise per PDA oder Mobiltelefon auf dem Weg zur Arbeit mit öffentlichen Verkehrsmittel, den Stand der Auktion überprüfen und Gebote ausführen.

Faktoren für die Entwicklung von e-Commerce im Internet

Es gilt heute für ein Unternehmen als selbstverständlich, sämtliche Informationen des Unternehmens vom Standort über Produkte bis hin zu Kontaktadressen im Internet auf einer Webseite[5] bereitzustellen. Dies ist eine kostengünstige Art, zeitunabhängig nicht nur Informationen für bereits bestehende Kunden

[4]PDA (Personal Digital Assistant) ist ein persönlicher Digitalassistent mit kleinem Gewicht, was seine bequeme Portabilität erlaubt. Er speichert Daten wie Adressen, Telefonnummern, Kalenderdaten, Notizen und andere persönliche Daten. Die steigende Leistungsfähigkeit dieser Geräte erlaubt deren Nutzung für die mobile Datenverarbeitung.

[5]Eine Webseite, oft auch als Internetseite oder Homepage bezeichnet, stellt eine Ansammlung von Informationen in der Form von HTML-Seiten dar, die auf einem Server mit permanenter Internet-Verbindung abgelegt sind und mit Hilfe eines Browsers geöffnet werden können. Die HTML-Seiten sind mit internen oder externen Hyperlinks miteinander verknüpft.

zur Verfügung zu stellen, sondern auch neue Kunden zu gewinnen. Per Email können weiterhin kostengünstig Informationen zwischen Anbietern und Nachfragern asynchron ausgetauscht werden.

Nach Zerfaß und Haasis (2002, S. 10 ff.) dienen folgende drei Ansatzpunkte zur Verbesserung der Unternehmensprozesse: Information, Kommunikation und Transaktion. Diese drei Faktoren stellt auch das Internet zur Verfügung.

- *Information*: Das Internet dient als Plattform sowohl zur Bereitstellung als auch zur Beschaffung von Information. Unternehmen können beispielsweise ihre Produktangebote auf einer Webseite veröffentlichen. Diese Möglichkeit ergibt klare Kostenvorteile gegenüber dem Katalogversand auf dem Postweg. Darüber hinaus können den Kunden durch das tägliche Aktualisieren der Daten aktuelle Informationen zur Verfügung gestellt werden. Anschließend ermöglicht das Internet auch eine bessere Darstellung der Informationen als auf zweidimensionalen Katalogseiten wie beispielsweise durch eine 3D Darstellung der Produkte oder die Bereitstellung von Videos. Dadurch steigt die Qualität der Information.

- *Kommunikation*: Das Internet kann auch als Kommunikationsplattform zwischen den beteiligen Akteuren dienen. So können Informationen synchron, beispielsweise über Videokonferenzen, ausgetauscht werden oder asynchron, über Email. So können sich z.B. Mitarbeiter eines Projekts, die in Europa und in den USA arbeiten, bei einer Videokonferenz treffen. Hier liegt erneut ein klarer Kostenvorteil, da dabei Flugkosten, Hotelkosten etc. eingespart werden können. Darüber hinaus wird enorme Flexibilität erzielt. Eine Videokonferenz lässt sich spontaner planen als eine Geschäftsreise. Weiterhin können beispielsweise per Email viele Fragen geklärt werden, ohne dass ein gleichzeitiges Treffen der Akteure stattfindet. Besonders vorteilhaft kann sich das für Mitarbeiter erweisen, die in unterschiedlichen Zeitzonen arbeiten.

- *Transaktion*: In diesem weit über die Information und Kommunikation hinausgehenden Schritt kann die Herstellung und der Vertrieb von Produkten und Dienstleistungen zum großen Teil oder gar vollständig über das Internet stattfinden. Der Kunde kann über das Internet nicht nur Informationen über Produkte bekommen oder sich mit dem Kundenbetreuer in Verbindung setzen und weitere Fragen stellen, sondern auch eine Bestellung abwickeln. Es entstehen erneut klare Kostenvorteile: Nicht nur für den Verkäufer, sondern auch für den Käufer, dem die Zeit und die Fahrtkosten zum Geschäft erspart bleiben. Hinzu kommt noch die

> Möglichkeit, dem Kunde spezielle Wünsche zu erfüllen und diese beizufügen. Bei einem Online-Buchhändler kann beispielsweise ein Buch als Geschenk versandt werden. Dazu kann der Käufer das Papier für die Verpackung wählen und die Adresse für den Empfänger zusammen mit einem Glückwunschtext eingeben. Solche kundenspezifischen Wünsche verschaffen dem Online-Buchhändler einen eindeutigen Wettbewerbsvorteil gegenüber einem herkömmlichen niedergelassenem Buchhändler.

Albers u. a. (2000a, S. 11) nennen diese vom Internet bereitgestellte Kommunikationsmöglichkeit *Kommunikationskanal*. Die vom Internet zur Verfügung gestellte Transaktionsmöglichkeit bezeichnen sie als *Distributionskanal*. Er besteht aus einem Kommunikationskanal und einem Rückkanal. Der Rückkanal ermöglicht eine Erweiterung der einseitigen Kommunikation, was beispielsweise die Bearbeitung von Kundenbestellungen ermöglicht. Dadurch ist die Bearbeitung einer automatischen Produktbestellung und anschließend die Online- oder Offline-Distribution möglich. Eine Online-Distribution kann nach Albers u. a. (2000b, S. 80) bei digitalisierbaren Produkten für Nachfrager, die über ein entsprechendes Ausgabemedium verfügen, gewählt werden.

Warum kaufen die Käufer über das Internet?

Die steigende Akzeptanz des Mediums Internet lässt die Zahlen in jedem Bereich des elektronischen Handels im Internet immens steigen. Dabei lässt sich fast alles im Internet verkaufen. Albers (2000, S. 21 ff.) bringt einen Faktor hervor, welcher für den Verkauf eines Produktes über das Internet entscheidend ist: *Der Kundennutzen*. Er wird durch Faktoren wie

- niedrigere Preise,
- bequemere Distribution,
- Bereitstellung von Information über die Produkte und
- Individualisierung der Produkte

bestimmt.

Der *Preis eines Gutes* als Bestimmungsfaktor des Kundennutzens, spielt eine der wichtigsten Rollen bei der Kaufentscheidung seitens der Käufer. Je niedriger der Preis, desto höher liegt die Wahrscheinlichkeit für die sofortige Kaufentscheidung der Ware. Das Internet stellt den Konsumenten eine Plattform für transparente Preise zur Verfügung. Sie bekommen dadurch günstig

und innerhalb der kürzesten Zeit die notwendigen Preisvergleiche. Dies wird durch die Präsenz von Preisagenten im Internet verschärft. Darüber sind sich die Verkäufer bewusst. Sie müssen ihre Preise entsprechend auch anpassen, da sie sonst Kunden und Umsätze verlieren. Der Preis kann nur dann gesenkt werden, wenn auch die Produktions- und Anschaffungskosten niedrig gehalten werden. Der Preisdruck fällt damit auf die Verkäufer sowie auf das gesamte Wertschöpfungsnetzwerk zurück - im Vorteil der Kunden.

Die *bequeme Lieferung* eines Gutes kann noch ein weiterer Faktor zum Einkaufen im Internet sein. In der Regel fallen für die Lieferung der gekauften Ware auch Lieferungskosten an, was aber auf der Gegenseite auch Vorteile für den Kunden bereitstellt. Sie müssen nicht Zeit und Fahrtkosten für den Einkauf einplanen. Die Lieferung nach Hause kann einige Tage dauern und lässt den Konsumenten die Ware erst sehen, wenn diese geliefert wurde. Daher hängt die Entscheidung für den Kauf über das Internet davon ab, wie gut der Kunde das Produkt, die Marke, den Hersteller und seine Qualität kennt und ob ihm, wie es inzwischen im Versandhandel üblich ist, ein mehrtägiges Rückgaberecht eingeräumt wurde und er Lieferungsfehler und Mängel reklamieren kann.

Das Medium Internet kann für eine umfangreiche *Bereitstellung von Informationen* über die zu verkaufenden Produkte sorgen. So kann ein Käufer von Amazon®[6] beim Erwerb eines Buches außer allgemeinen Informationen auch Bewertungen von anderen Käufern des Buches lesen. Unterstützung bei der Beschaffung von Informationen über Waren können auch Internet-Agenten[7] liefern.

Das Internet lässt weiterhin sehr gut die *Individualisierung von Produkten* realisieren und erlaubt dadurch die bessere Anpassung des Herstellers an die Wünsche seiner Kunden. Der Produkthersteller ist in der Lage, Mass Customization[8] zu betreiben. Er kennt die genauen Wünsche des Kunden und kann

[6] http://www.amazon.de bzw. http://www.amazon.com

[7] Ein Agent ist ein Programm, welches „selbständig eine Such- und Entscheidungsaufgabe über das Internet bearbeitet, nachdem sein Auftraggeber die Rahmenbedingungen (z.B. Preis, Termin, Qualität bzw. Suche, Auswahl, Handlungsentscheidung) festgelegt hat“. (Schildhauer, 2003, S. 5)

[8] „Mass Customization (kundenindividuelle Massenproduktion) ist die Produktion von Gütern und Leistungen für einen (relativ) großen Absatzmarkt, welche die unterschiedlichen Bedürfnisse jedes einzelnen Nachfragers dieser Produkte treffen, zu Kosten, die ungefähr denen einer massenhaften Fertigung vergleichbarer Standardgüter entsprechen. Die Informationen, die im Zuge des Individualisierungsprozesses erhoben werden, dienen dem Aufbau einer dauerhaften, individuellen Beziehung zu jedem Abnehmer.“ (Piller,

damit genau das gewünschte Produkt herstellen. Ein Computerhersteller kann beispielsweise über Web-Formulare die Bestellungen seiner Kunden abwickeln lassen. Im Bestellformular kann der Kunde die gewünschten Komponenten des bestellten PCs individuell eingeben oder über eine Dropdown-Liste auswählen. Der Nutzen des Kunden liegt darin, dass das aufwändige Suchen des von ihm gewünschten Produktes entfällt und er trotzdem ein Produkt bekommt, das vollkommen seinen Vorstellungen entspricht.

Warum wird über das Internet verkauft?

Die bereits am Anfang dieses Abschnitts aufgezählten Eigenschaften für die Entwicklung des e-Commerce im Internet stellen die Grundlage für die immer weiter zunehmenden Angebote im Internet seitens der Verkäufer dar. Durch das Internet lassen sich klare *Wettbewerbsvorteile* gegenüber der Konkurenz erzielen (Albers, 2000, S. 21 ff.): Die Schaffung neuer Absatzkanäle, durchgängige Kostensenkung und die Möglichkeit zum One-to-One-Marketing[9].

Das Internet stellt den Produzenten *weitere Absatzkanäle* zur Verfügung. Die virtuelle Welt des Internets erlaubt die Präsentation und Verkauf der hergestellten Produkte nicht nur regional, sondern auch bundes- und sogar weltweit. Durch diese zusätzlichen Absatzkanäle werden neue Zielgruppen erfasst und höhere Umsätze erzielt. Außerdem sorgt die zusätzliche Möglichkeit, rund um die Uhr über das Internet zu verkaufen, für ein besseres Image des Unternehmens und eine höhere Präsenzdichte.

Ein anderer Vorteil ist die *Kostensenkung*, die das Internet für den Vertrieb bietet. Die Präsenz von Zwischenhändlern ist überflüssig. So können sowohl größere Absätze erzielt als auch die Verkaufspreise gesenkt werden. Für die über das Internet verkauften Produkte müssen keine Flächen für Ladengeschäfte angemietet werden. Es fallen lediglich Lager- und Versandkosten für die verkauften Produkte an, welche mit ersparten Wegezeit für den Kauf und wegfallenden Fahrtkosten kompensiert werden. Diese Kosten lassen sich für sogenannte digitalisierbare Güter sogar gänzlich vermeiden. So können digitale Produkte wie Software oder Musik dem Käufer nach dem Kauf direkt über das Internet zur Verfügung gestellt werden. Dabei werden nicht nur Versandkosten gespart, sondern auch eine sofortige Lieferung des Produktes nach dem Kauf ermöglicht.

1998)

[9] „Unter *One-to-One-Marketing* wird eine einzelkundenorientierte Ausrichtung aller Marktaktivitäten verstanden, deren Ziel die jeweils individuelle Ansprache und individuelle Problemlösung ist“. (Strauß und Schoder, 2000, S. 112)

Das Internet erlaubt das bequeme Umsetzen des *One-to-One-Marketing*. Damit wird eine massenhafte *Differenzierung des Leistungsangebots* erzielt, welche das Internet durch die automatisierte Verarbeitung von Kundenpräferenzen ermöglicht. So können beispielsweise dem Kunden auf seine individuellen Präferenzen zugeschnittene Angebote gemacht werden. Der Kunde kann damit individuell betreut werden. Das sorgt für eine größere Kundenzufriedenheit und damit für eine höhere *Kundenloyalität*. Kundenübergreifend gesehen ermöglicht es die Sammlung von Kundendaten auch deren Auswertung. Die Präsenz dieser Daten ermöglicht deren Auswertung und dadurch die Gewinnung eines größeren Markt-Know-hows. (Strauß und Schoder, 2000, S. 114 f.)

Eigenschaften der Internet-Ökonomie

Die technologische Entwicklung schafft die Voraussetzungen für die Entstehung ökonomischer Transaktionen im Internet. Hier wird eine Wandlung der traditionellen in die neuen internetbasierten Märkte beobachtet. Folgend werden die Auswirkungen auf die traditionelle Ökonomie betrachtet.

Skaleneffekte Die Produktion digitaler Güter ist meistens mit sehr hohen Entwicklungskosten verbunden. Diese stellen die Fixkosten der Produktion dar. Die variablen Kosten (pro Stück) dagegen sind im Vergleich zu den fixen Kosten sehr gering. Nach Stelzer (2000, S. 837 f.) tendiert deren Wert gegen Null. Dadurch entsteht eine sehr hohe Differenz zwischen der Höhe der variablen und der Höhe der fixen Kosten. Wenn Produktion und Absatzmenge steigen, sinken die Gesamtkosten der Produktion. Eine höhere Absatzmenge führt hiermit zu niedrigeren Durchschnittskosten und zu höheren Absatzzahlen. Dies gibt den Herstellern die Möglichkeit, ihre Preise zu senken und dadurch Wettbewerbsvorteile gegenüber ihren Konkurrenten zu erwerben. Die Preissenkung zieht weitere Kunden an und führt zur Erhöhung der Absatzmenge, was als positiver Feedback-Effekt bezeichnet wird.

Netzwerkeffekte Der Netzwerkeffekt besteht darin, dass der Nutzen der einzelnen Nachfrager mit dem Steigen der Anzahl der Nutzer dieses Produktes erhöht wird. So wird beispielsweise das Einrichten von FAQs, Internet Foren oder Chat Räume bei mehreren Nutzern sinnvoller und den Vorteil für allen Teilnehmer größer. Mehrere Nutzer eines Produktes führen damit zum höheren Nutzen der Einzelnen und machen dadurch das Produkt attraktiver und günstiger. Dies zieht weitere Nutzer an und verursacht einen Lawineneffekt. Die höhere Nutzeranzahl erhöht den Nutzen

für den einzelnen Nutzer des Produktes, was erneut zu einem positiven Feedback-Effekt führt. (Stelzer, 2000, S. 838 f.)

Lock-In-Effekte Der Lock-In-Effekt ergibt sich nach Stelzer (2000, S. 840) aus den Wechselkosten eines Produktes. Diese bestehen aus den Anschaffungskosten des Produktes und den Opportunitätskosten bei seinem Einsatz. Die Opportunitätskosten stellen den entgangenen Nutzen beim Einsatz des alten Produktes dar. Der Lock-In-Effekt bedeutet hiermit, dass Nutzer, die sich für ein Produkt entschieden haben, an dieses gebunden sind. Der Wechsel zu einem anderen Produkt ist jedoch möglich und hängt von seinem Preis ab. Ist also der Nutzen des neuen Produktes höher als der des alten und die Anschaffungskosten gering, minimieren sich die Wechselkosten erheblich. Daher versuchen Unternehmen ihre Kunden mit Lock-In-Effekten zu halten und selbst nicht in solchen gefangen zu werden.

Kostensenkung Die Abwicklung von Geschäften über das Internet führt zur überdimensionalen Kostensenkung. Der Hauptgrund dafür sind die gesenkten Informations- und Kommunikationskosten, was durch die Nutzung dieses Mediums ermöglicht wird. (Picot, 2004, S. 1)

Rolle der Intermediäre Die Intermediäre hatten im Pre-Internet Zeitalter die Rolle der Groß- oder Einzelhändler für die hergestellten Produkte. Deren Aufgabe war die Distribution der abgenommenen Produkte. Das Internet bietet heute den Produzenten die Möglichkeit, ihre Produkte unmittelbar an den Endkunden zu vertreiben - und das einfacher und kostengünstiger. (Brandstetter und Fries, 2002, S. 20)

Unbeschränkte Kundenreichweite Aufgrund der Interaktivität des Internets spielt es keine Rolle mehr, welche räumlichen Distanzen zwischen dem Produzenten und dem Kunden liegen. Die drastische Verringerung sowohl der Orts- als auch der Zeitabhängigkeit sorgt für die Erweiterung der Kundenanzahl. (Brandstetter und Fries, 2002, S. 37)

Kundenbindung Das Internet ermöglicht ab bestimmten technischen Voraussetzungen die Identifizierung und damit die individuelle Ansprache eines Kunden und die Ermittlung von Kundenpräferenzen, was für mehr Kundennähe und stärkere Kundenbindung sorgt. Diese Kriterien unterstützen die Umsetzung von One-to-One-Marketing[10] und sorgen für

[10]Siehe auch Fußnote 9.

mehr Kundenzufriedenheit. (Brandstetter und Fries 2002, S. 37; Picot 2004, S. 4)

Der „gläserne Kunde“ Der Kunde hinterlässt „virtuelle Spuren“ auf der Webseite des Unternehmens. (Brandstetter und Fries, 2002, S. 37) Diese sollten gesammelt und bei jedem weiteren Besuch ergänzt werden. Das Halten der Kundendaten vermeidet die Stellung mehrfacher identischer Fragen. Diese Daten ermöglichen weiterhin die Personalisierung der Kunden.

Hoher Wettbewerb Die Möglichkeit des Internets, orts- und zeitunabhängige Informationen bereitstellen zu können, sorgt für eine hohe Preistransparenz. Die Kunden können bequem die vorhandenen Preise vergleichen. Die Anbieter können schnell ihre Preise und weitere Konditionen an den Markt anpassen. (Brandstetter und Fries, 2002, S. 37)

Digitalisierung Das Digitalisierungsausmaß nimmt in der Internet-Ökonomie zu. Einzelne Unternehmensprozesse bis hin zu ganzen Prozessketten werden zunehmend digitalisiert. Digitalisierbare Produkte werden ebenso digitalisiert und sorgen damit für einen schnellen Ablauf von Prozessen und für höhere Kundenzufriedenheit. (Picot, 2004, S. 1)

2.1.3 Geschäftsmodelle des elektronischen Handels

An dieser Stelle wird zunächst der Begriff e-Commerce näher betrachtet. Dabei soll auch eine Abgrenzung der Begriffe e-Commerce und e-Business erfolgen.

E-Commerce und e-Business

Wie aus den folgenden Definitionen ersichtlich, bietet die Literatur leider keine einheitliche Definition hinsichtlich des Begriffs *e-Commerce*. Er wird bezeichnet als

> „...die digitale Anbahnung, Aushandlung und/oder Abwicklung von Transaktionen zwischen Wirtschaftssubjekten“. (Clement u. a., 2001, S. 49)

> „...die Verwendung von elektronischen Medien bei Transaktionen von Gütern, Informationen oder Dienstleistungen zwischen Geschäftspartnern und Kunden“. (Stähler, 2002, S. 54)

Feld und Hoffmann (2000, S. 195) definieren den Begriff e-Commerce aus der Sicht der Geschäftsprozesse als

> „...die Anwendung von Technologien der Informations- und Kommunikationstechnik auf operative Geschäftstransaktionen".

Trotz uneinheitlicher Definitionen lassen sich zwei wichtige Eigenschaften von e-Commerce erkennen:

- Zum Einen finden Transaktionen zwischen Wirtschaftssubjekten statt und
- zum Anderen werden diese Transaktionen über elektronische Wege durchgeführt.

Dabei sind mit Transaktionen sowohl die Transaktionen von Gütern und Dienstleistungen als auch die Finanztransaktionen und das Internet-Marketing wie E-Mail Werbung gemeint.

Mit *e-Business* werden nicht nur die Prozesse in den einzelnen Geschäftsbereichen, sondern die gesamte Prozesskette eines Unternehmens bezeichnet (internet-manual.de, 2004). Die Abgrenzung zwischen e-Commerce und e-Business leitet sich aus den folgenden Definitionen ab:

> „Unter dem Begriff Electronic Business wird die Anbahnung sowie die teilweise respektive vollständige Unterstützung, Abwicklung und Aufrechterhaltung von Leistungsaustauschprozessen mittels elektronischer Netze verstanden." (Wirtz, 2001, S. 34)

> „Electronic Commerce beinhaltet die elektronische Unterstützung von Aktivitäten, die in direktem Zusammenhang mit dem Kauf und Verkauf von Gütern und Dienstleistungen via elektronischer Netze in Verbindung stehen." Wirtz (2001, S. 40)

E-Commerce kann damit als eine Untermenge von e-Business betrachtet werden, was die Abbildung 2.2 verdeutlicht.

Charakteristika eines Geschäftsmodells

In diesem Abschnitt wird der Begriff des Geschäftsmodells erläutert.

Timmers (1998, S. 4) definiert ein Geschäftsmodell wie folgt:

- „An architecture for the product, service and information flows, including a description of the various business actors and their roles; and

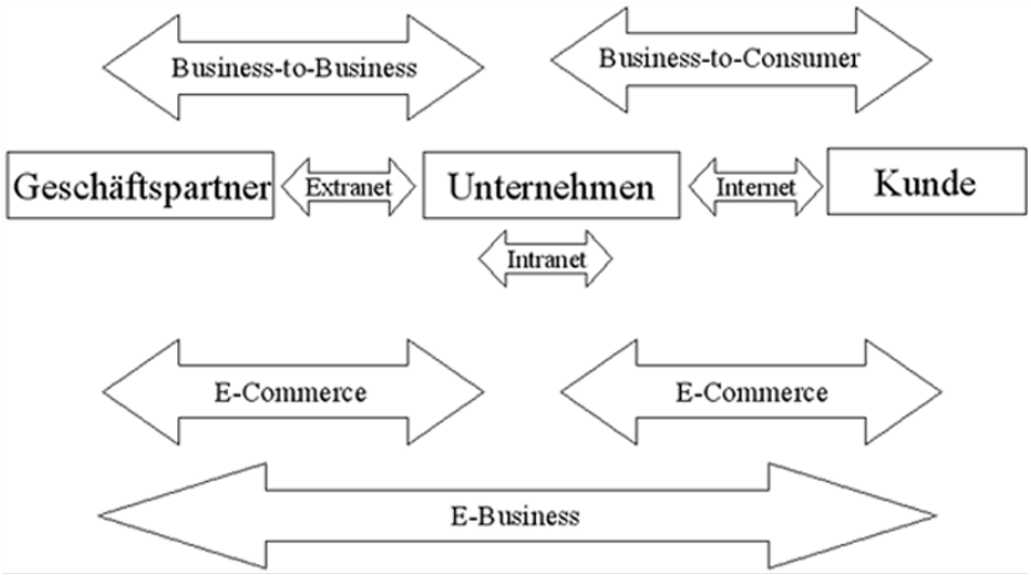

Abbildung 2.2: Beziehung zwischen e-Business und e-Commerce,
Quelle: Stähler (2002, S. 54)

- A description of the potential benefits for the various business actors; and
- A description of the sources of revenues.“

Damit definiert ein Geschäftsmodell das Produkt oder auch die Dienstleistung des Handels und den damit verbundenen Informationsaustausch während des Handelsprozesses mit den Produkten. Weiterhin definiert ein Geschäftsmodell die beteiligten Akteure, zugehörige Aufgaben und den Nutzen, den sie dabei erhalten. Anschließend muss ein Geschäftsprozess die Erlösquellen definieren.

Stähler (2002, S. 41) definiert das Geschäftsmodell als

> „...Geschäftskonzept, das in der Praxis schon angewandt wird“.

Er lässt weiterhin drei Hauptkomponenten des Geschäftsmodells erkennen:

- der Wertbeitrag[11],
- die Architektur der Wertschöpfung und

[11] Wertbeitrag: (engl.) value proposition

- das Ertragsmodell.

Der Wertbeitrag ist nicht in der Definition von Timmers (1998) präsent und enthält den Nutzen, der dem Kunden vom Unternehmen gestiftet wird.

Die Merkmale, die ein Geschäftsmodell charakterisieren, sind die Zielgruppen und deren Interaktion, die Objekte des Handels, das Preisfindungsmodell, die Art der Einkäufe und die Phasen des e-Commerce. (Böhm und Felt, 2001, S. 5) Darauf aufbauend können die unter diesem Abschnitt stehenden Fragen formuliert werden, deren Antworten ein Geschäftsmodell von seinem eigentlichen Aufbau her ausgehend charakterisieren.

Wer handelt? Die Akteure von e-Commerce sind die Unternehmen (*Business*), die Konsumenten (*Consumer*) und die öffentlichen Institutionen (*Administration*) (Peters, 2000, S. 961 f.). Die sich daraus ergebenden Beziehungen werden ab Seite 23 erläutert.

Was wird gehandelt? Die gehandelten Güter im Internet können physische Produkte oder Dienstleistungen sein. Nach Albers (2000, S. 22) ist die Digitalität der Güter nicht der entscheidende Faktor für deren Handel im Internet. Mit Hilfe des Informations- und Distributionskanals, die durch dieses Medium zur Verfügung gestellt werden, werden theoretisch keine Beschränkungen für absetzbare Güter im Internet gesetzt. So nehmen Bücher den ersten Platz in den verkauften Gütern im Internet ein, gefolgt von Computern und Software. Ebenfalls lassen sich Waren wie Kleidung, Schuhe und CDs aber auch Reisen, Kosmetikartikel, Getränke etc. sehr gut verkaufen. (Albers, 2000, S. 23)

Wie wird der Handelspreis festgelegt? Der Preis der Güter und Dienstleistungen kann fest oder variabel sein. Der *Festpreis* ist typisch für das Katalogmodell und ist die unflexibelste Art, die Preise festzulegen. Eine Abweichung davon kann durch Rabatte realisiert werden. Der Preis im Modell kann im Gegenteil zum Festpreismodell auch *variabel* sein. Folgende Möglichkeiten für die Preisfestlegung beinhalten einen flexiblen Preis: Auktionen mit ihren unterschiedlichen Ausprägungsformen[12], Exchanges[13] und Classifieds[14].

[12] Siehe Kapitel 2.2.1 auf Seite 28.

[13] Exchanges – Käufer und Verkäufer haben die Möglichkeit über die Geschäftsbedingungen und Konditionen unabhängig von Fristen zu verhandeln. Es können jederzeit solange Angebote und Gegenangebote gemacht werden, bis ein Konsens über die Transaktion erreicht wird. (Amor, 2000, S. 86)

[14] Classifieds geben die Möglichkeit, Waren und Dienstleistungen anzubieten, ohne eine

Durch wen wird gehandelt? An dieser Stelle kann ein direkter oder indirekter Handel stattfinden. Bei dem direkten Handel handeln die Käufer unmittelbar mit den Verkäufern, bei dem indirekten Handel dagegen werden Intermediäre eingeschaltet. Dies können virtuelle Malls[15] oder Marktplätze[16] sein.

Welcher Art sind die Einkünfte? Die Einkünfte können aus direktem Verkauf von Waren stammen. Dienstleistungen wie Werbung können auch Einkünfte bringen.[17]

Welche Phasen beinhaltet der Transaktionsprozess? Grundsätzlich lassen sich fünf Phasen im Transaktionsprozess[18] herauskristallisieren.

- *Informationsphase*: In dieser Phase suchen die Nachfrager die vorhandenen Anbieter und informieren sich genauer über deren Angebote und Konditionen. Diese Phase ist die Grundlage für die Preisfindungsphase.
- *Preisfindungsphase (Verhandlungsphase)*: Während der Preisfindungsphase wird der Kaufpreis des Gutes festgelegt. Eine verbreitete Möglichkeit für die Preisfindung sind Auktionen. Auf diese wird im Kapitel 2.2 näher eingegangen.
- *Bezahlungsphase*: Diese Phase schließt den Transfer des vereinbarten Preises vom Käufer zum Verkäufer. Der Transfer kann persönlich, per Überweisung oder per Kreditkartenzahlung erfolgen. Die Zahl der Käufer, die solche Transaktionen auch computergestützt wie beispielsweise mit Online-Banking[19] durchführen, nimmt ständig zu.
- *Abwicklungsphase*: Die Abwicklungsphase schließt den Transfer des Gutes vom Verkäufer zum Käufer ein. Es besteht die Möglichkeit,

---

rechtliche Bindung einzugehen. Die Verhandlungen bei Classifieds können im Unterschied zu Exchanges auch offline durchgeführt werden. Sie müssen nicht wie bei Auktionen zeitlich begrenzt sein.

[15] „Eine Mall ist ein Zusammenschluss von elektronischen Läden mit übergeordneter Struktur sowie einheitlichem funktionalen und graphischen Design, das den einzelnen Läden dennoch gewisse Freiräume bietet.“ (Zwißler, 2002, S. 33)

[16] Elektronische Marktplätze sind nach Zwißler (2002, S. 40) Systeme, auf denen Käufer und Verkäufer aus dem Industriellen Bereich zusammentreffen.

[17] Siehe Seiten 26 f.

[18] Vgl. Böhm und Felt (2001, S. 5) und Feld und Hoffmann (2000, S. 206 ff.).

[19] „Online-Banking ist die Abwicklung von Bankgeschäften wie Abbuchungen, Überweisungen, Einrichtung von Daueraufträgen über Computernetze; insbesondere über das Internet.“ (net-lexikon.de, 2004b)

die verkaufte Ware persönlich dem Käufer zu übergeben. Bei gekauften Waren über das Internet ist es üblich, dass diese per Post an den Käufer gesendet werden. Bei digitalen Gütern, die über das Internet gekauft werden, kann auf den Postweg verzichtet werden, um Zeit und Kosten für die Abwicklung zu sparen.

- *After-Sale-Phase*: Diese Phase bezieht sich auf den Kundendienst, der vom Verkäufer nach der Übergabe des Gutes dem Käufer gegenüber geleistet wird.

Geschäftsmodelle nach den beteiligten Akteuren

Aus der Kombination der beteiligten Akteure eines Geschäftsmodells ergeben sich die möglichen Beziehungen, wie aus der Abbildung 2.3 ersichtlich. Die bedeutendsten bilateralen Beziehungen sind B2B, B2C, C2C, A2B und A2C.

- B2B (Business-to-Business): Der B2B-Bereich enthält die elektronische Geschäftsabwicklung zwischen Unternehmen, ihren Lieferanten und Händlern und realisiert derzeit das größte Handelsvolumen.
  Beispiel: COVISINT®[20].

- B2C (Business-to-Consumer): Dieser Bereich enthält die Beziehungen zwischen den Unternehmen und den Endverbrauchern.
  Beispiel: Amazon®[21].

- C2C (Consumer-to-Consumer): Der C2C-Bereich charakterisiert sich dadurch, dass die Konsumenten sowohl als Anbieter als auch als Nachfrager auftreten. Beispiel: eBay®[22].

- A2B (Administration-to-Business): Dieser Bereich wird in den USA von der Administration genutzt, um Ausschreibungen über das Internet durchzuführen.

- A2C (Administration-to-Consumer): Die verfügbaren Anwendungen in diesem Bereich sind derzeit nicht gross, nehmen aber mit der Zeit zu. In diesem Rahmen sollen Behörden ihre Leistungen über das Internet zur

[20] COVISINT ist eine von Ford, General Motors und DaimlerChrysler gemeinsam im Jahr 2000 gegründete Einkaufsplattform. Sie unterstützt den elektronischen Einkauf von Waren und Dienstleistungen für die Automobilbranche und sorgt für besseren Informationsaustausch über die gesamte Wertschöpfungskette. Siehe dazu auch http://www.covisint.com

[21] http://www.amazon.de bzw. http://www.amazon.com

[22] http://www.ebay.de bzw. http://www.ebay.com

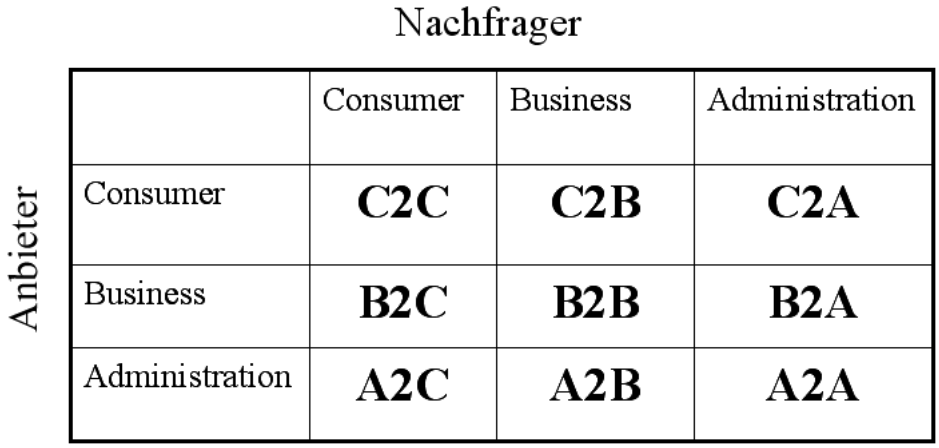

Abbildung 2.3: Mögliche Erscheinungsformen des e-Commerce

Verfügung stellen. Diese können von den Konsumenten bequem genutzt werden.

In neuerer Zeit ist der Akteur Mitarbeiter (*Employee*) hinzugekommen. Dieser bildet eine zusätzliche Beziehung mit dem Business Bereich: B2E (Business-to-Employee). Hiermit werden die Beziehungen zwischen einem Unternehmen und dem Mitarbeiter bezeichnet.

Bei der eigentlichen Erarbeitung des Geschäftsmodells sind

- die Aufzeichnung der Wertschöpfungskette,
- die Bestimmung des Erlösmodells und
- der Kundennutzen, der vom Unternehmen zur Verfügung gestellt wird,

relevant. Diesen Komponenten des Geschäftsmodells wird in den nächsten Abschnitten nachgegangen.

Die Wertschöpfungskette

Die *Wertschöpfungskette* wird von Porter (1999, S. 63 ff.) als ein Instrument eingesetzt, um die Wettbewerbsvorteile eines Unternehmens zu untersuchen und zu beurteilen. Schlüsselrollen haben dabei die wertschöpfenden Tätigkeiten in einem Unternehmen, die immer in einer Wechselwirkung zueinander stehen. Diese nennt Porter *Wertaktivitäten* und definiert die Wertschöpfungskette als

eine Zusammensetzung aus den Wertaktivitäten und der Gewinnspanne. Die Wertaktivitäten teilen sich in *primäre*, welche direkt und *sekundäre*, welche indirekt in die Wertschöpfung eingebunden sind. Die Wertschöpfungskette eines Unternehmens ist nicht isoliert. Sie ist beispielsweise abhängig von den Lieferantenwertketten und von den Abnehmerwertketten. Das Zusammenwirken aller Wertketten wird von Porter als *Wertsystem* bezeichnet.

Aufbauend auf der Wertschöpfungskette und auf dem Wertsystem von Porter definieren Scheer und Loos (2004, S. 28 ff.) das *Wertschöpfungsnetzwerk*, welches sich aus wertschöpfenden Prozessen mehrerer Unternehmen zusammensetzt. Wettbewerbsgründe führen zur Konzentration der Unternehmen auf einzelne Prozesse, die gemeinsam in die Wertschöpfungskette gelegt werden. Scheer u. a. (2000, S. 14) gleichen die Prozessstruktur der Wertschöpfungskette mit den primären Aktivitäten der Wertschöpfungskette von Porter an. Sie verdeutlichen die Möglichkeit der Unterstützung der internen und externen Prozesse eines Unternehmens durch internetbasierte Kommunikationssysteme. Die Wertschöpfungskette besteht nun sowohl aus einer physischen als auch aus einer Informationskomponente. Das ermöglicht die Verbesserung der internen und externen Geschäftsprozesse hinsichtlich Geschwindigkeit, Qualität und Kosten. „Die *Wertschöpfungskette* wird für die Unternehmen zu einem *Wertschöpfungsnetz* auf digitaler Grundlage, mit dessen Hilfe sie ihre Kunden, Lieferanten, verwandte Gruppen oder auch Konkurrenten über die Technologie erreichen können.“ (Tapscott, 1996, S. 116)

Die Informationskomponente der Wertschöpfungskette lässt sich besonders gut von den neuen Entwicklungen der Informations- und Kommunikationstechnologie unterstützen, was zur Entstehung einer neuen Wertschöpfungskette – der virtuellen Wertschöpfungskette – führt. Die virtuelle Wertschöpfungskette sorgt für die Entstehung weiterer Werte und verläuft, wie aus der Abbildung 2.4 ersichtlich, parallel zur physischen Wertschöpfungskette innerhalb des Wertschöpfungsnetzwerkes. Nach Rayport und Sviokla (2000, S. 63 ff.) können Unternehmen in den folgenden drei Schritten den Wertschöpfungsprozess der Informationstechnologien einführen.

- *Sichtbar machen*: Durch diesen ersten Schritt zielt das Management darauf ab, einen besseren Überblick über die Wertschöpfungskette zu erhalten. Die physische Wertschöpfungskette wird dabei mit Hilfe der Informationstechnologien unterstützt. Mit deren Hilfe werden Informationen über die wertschöpfenden Prozesse gesammelt, welche die Verbindung aller Komponenten schafft und den Überblick über die ganze Kette

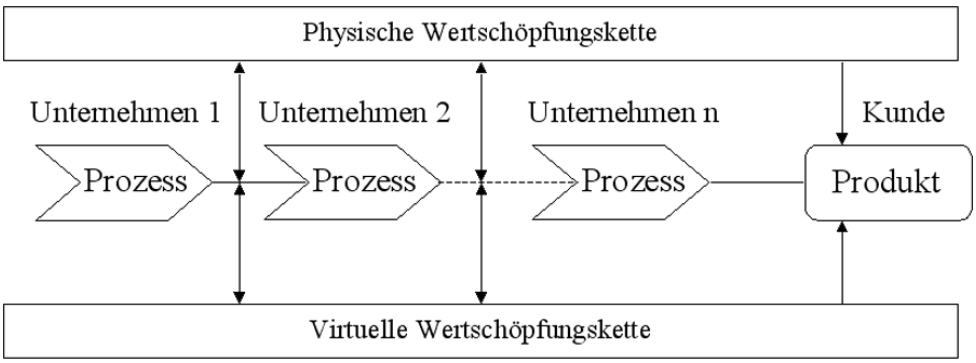

Abbildung 2.4: Die virtuelle und physische Wertschöpfungskette, In Anlehnung an Scheer und Loos (2004, S. 35)

ermöglicht. Das ist in der heutigen Dynamik der Märkte besonders wichtig, da schnelle Entscheidungen gefragt sind.

- *Übertragen von Fähigkeiten*: In diesem Schritt werden nicht nur Informationen über die Wertschöpfungskette geschaffen, sondern auch Prozesse innerhalb der Wertschöpfungskette ersetzt. Nun werden in dieser Phase Prozesse der physischen Wertschöpfungskette mit Hilfe der Informationstechnologie ersetzt. Dadurch entsteht die *virtuelle Wertschöpfungskette*, welche ein grosses Potenzial für enorme Ersparnisse für die Unternehmen und einen größeren Nutzen für die Kunden mit sich bringt.

- *Neue Kundenbeziehungen schaffen*: Anschließend können in der dritten Phase neue Kundenbeziehungen geschaffen werden. Dabei wird auf die Wünsche der einzelnen Kunden mehr Wert gelegt, und damit für sie ein größerer Nutzen gestiftet.

Das Erlösmodell im Internet

Das Erlösmodell hat die Aufgabe, die Einnahmen für die Finanzierung des Systems aufzulisten. Das Bestimmen des Erlösmodells beinhaltet folgende zwei Schritte (Skiera und Lambrecht, 2004, S. 2), als Fragen formuliert:

- Welches sind die Erlösquellen?

- Wie sieht das Preismodell aus?

Die erste Frage erfordert das Aufführen der Einnahmequellen, die zweite listet die konkrete Preisbildung für die jeweilige Erlösquelle auf.

Skiera und Lambrecht (2004, S. 3 ff.) unterscheiden drei Arten von Erlösen nach deren Herkunft:

- Produkte,
- Kontakte und
- Informationen.

Produkte als Erlösquellen können Bücher, CDs, Computer, Haushaltsgeräte aber auch Software sein. Bei dem berechneten Preis vom Verkäufer handelt es sich hier um den Produktpreis. Bei den *Kontakten* als Erlösquelle handelt es sich beispielsweise um Bannerwerbung. Hier wird der Preis für die Freischaltung der Bannerwerbung berechnet; die Konsumenten des Produktes sind erneut die eigenen Kunden. *Informationen* als Erlösquelle können Nutzerprofile oder Paneldaten sein. Daten wie gekaufte oder gesehene aber nicht gekaufte Produkte können z.B. während des Kaufs von Produkten erhoben werden.

Die Erlösquellen können nur Produkte oder nur Kontakte sein. Weiterhin ist es möglich, dass die Erlösquellen eine Kombination aus Produkten und Kontakten sind oder sogar eine Kombination aus Produkten, Kontakten und Informationen.

Nutzenbereitstellung im Internet

Die Nutzenbereitstellung wird von der Preishöhe, von der Möglichkeit zur bequemeren Distribution, von der besseren Produktinformation und Möglichkeit für die Individualisierung der Produkte bestimmt. Dieses Thema wurde ab Seite 13, mit der Antwort auf die Frage, warum Käufer über das Internet kaufen, behandelt.

2.2 Online-Auktionen

Auktionen sind keine neuen Markterfindungen, sondern wurden schon viele Jahre vor Christus für die Preisfindung auf dem Markt eingesetzt:

> „Die heutige Form der Auktion tauchte wohl das erste Mal etwa im Jahre 500 vor Christus im alten Babylon auf. Der Historiker Herodotus beschrieb, daß dort Frauen auf dem jährlichen Heiratsmarkt versteigert wurden. Der Preis richtete sich nach der Schönheit der Frauen: je schöner, desto wertvoller waren sie.“ (Amor, 2000, S. 19)

Heute werden Auktionen massiv in der physischen und besonders in der virtuellen Welt verwendet, um sowohl gängige, als auch exotische Artikel zu versteigern. Das Ziel dabei ist, Produkte zu vertreiben, aber auch indirekt ein besseres Image für das Unternehmen zu erzielen und Internetpräsenz zu schaffen.

Dieser Abschnitt stellt zunächst die möglichen Auktionsarten, Funktionen von Auktionen und deren Vorteile und Nachteile dar. Danach wird das Geschäftsmodell von Auktionen dargestellt.

2.2.1 Auktionsarten und -funktionen

Eine Auktion ist

> „...ein Verfahren, mit dessen Hilfe eine Ressourcenverteilung - also die Zuordnung von Geld und Gütern zwischen Anbietern und Beschaffern - effizient durchgeführt werden kann“. (BMWi, 2001, S. 9)

> „Auctions are formalized trading procedures in which the trading partners' interaction is governed by specific trading rules. In many cases an (electronic) auctioneer is functioning as an intermediary. Electronic auctions are a special case of automated negotiations“. (Klein, 1997, S. 3)

Diese Definitionen weisen darauf hin, dass es bei Auktionen um Verhandlungen geht, bei denen der Preis von Produkten festgelegt wird und anschließend die Produkte und deren Geldwert zwischen dem Käufer und Verkäufer ausgetauscht werden.

Auktionsarten

Die Literatur definiert mehrere Arten von Auktionen. Nach der Offenheit der Gebote werden verdeckte und offene Auktionen unterschieden. Bei *verdeckten* Auktionen werden die Gebote verdeckt abgegeben, ein Bieter kennt daher die Gebote anderer Bieter nicht. Der gebotene Preis bei *offenen* Auktionen ist für alle Auktionsteilnehmer auch während der Laufzeit der Auktion sichtbar.

Nach dem Preisfindungsmechanismus lassen sich drei Auktionsformen wie nachfolgend aufgelistet unterscheiden, die eine weite Verbreitung in der Internetwelt gefunden haben. (Wirtz, 2001, S. 455 ff.)

Englische (Klassische) Auktion Dies ist die gängigste Auktionsform. Sie verwendet das offene Auktionsverfahren, bei dem die Preise während der Auktionszeit allen Parteien bekannt sind. Es wird ein Mindestpreis als Startpreis und der Mindestbetrag, der durch das letzte Gebot vom Verkäufer überboten werden muss, festgelegt. Während der Auktionslaufzeit bieten die potenziellen Käufer (Bieter) einen Preis, den sie zu zahlen bereit sind, welcher über dem aktuellen Preis des Objektes liegt. Den Zuschlag erhält der Bieter, der den höchsten Preis bietet. Bei Online-Auktionen dieser Art ist die Zeit, nicht mehr das räumliche Zusammentreffen, der bestimmende Faktor für die Dauer der Auktion.
Gegründet am 1. Mai 1995 ist eBay[23] heute die weltweit bekannteste und meist besuchte Auktionsplattform.

Holländische Auktion (Rückwärts-, Dutch- oder Top-down-Auktion) Der Verkäufer (Auktionator) setzt einen Höchstpreis und die Auktionsdauer für das zu versteigernde Objekt fest. Weiterhin legt er den Betrag fest, um den der Preis des Auktionsartikels in bestimmten Zeitabständen gesenkt wird. Dieser Abstand wird als "Preisrutsch„ bezeichnet. Beim Ablauf der Auktion wird der Preis gesenkt, bis ein Bieter ein Gebot für den aktuellen Preis abgibt. Dieser bekommt die ersteigerte Ware. Falls mehr als ein Gegenstand versteigert wird, wird die Preissenkung so lange fortgesetzt, bis alle Gegenstände verkauft sind.
Beispiel für solche Auktionen sind die holländische Blumenauktionen von andSTOP[24]. (Peterßen und Janzik, 2004, S. 11)

Höchstpreisauktion (first-price sealed-bid) Jeder Teilnehmer und Interessent bietet dem Verkäufer einen einzigen festen Preis. Dieser ist für alle anderen Auktionsteilnehmer verdeckt, d.h. sie können auf die Gebote anderer Bieter nicht reagieren.
Als Beispiel können die Auktionen des Unternehmens James G. Murphy Co.[25] genannt werden.

In der Literatur wird noch ein weiterer Auktionstyp hervorgehoben:
Die Vickrey-Auktion.

[23] http://www.ebay.com bzw. http://www.ebay.de
Der Bieter mit dem höchsten Gebot zum Endzeitpunkt der Auktion gewinnt diese, er zahlt allerdings den zweithöchsten Preis, wie bei Vickrey-Auktionen, mit einem zusätzlichen minimalen Aufpreis. Der Preis der Auktion zum jeweiligen Zeitpunkt ist bei eBay bekannt, der Höchstbetrag, der von den Bietern abgegeben wurde, bleibt ähnlich wie bei Vickrey-Auktionen unbekannt.
[24] http://www.andstop.de
[25] http://www.murphyauctions.net/

Vickrey-Auktion (second-price sealed-bid) Das Gebot jedes Bieters ist wie bei der Höchstpreisauktion verdeckt und wird ein Mal abgegeben. Die Ware bekommt der Bieter mit dem höchsten Gebotspreis. Er bezahlt aber den zweithöchsten Preis für die gebotene Ware. Der Vorteil dieser Auktionsform gegenüber der Höchstpreisauktion ist, dass die Bieter dazu tendieren werden, höhere Preise zu bieten, da sie wissen, dass sie beim Erhalt des Zuschlags einen niedrigeren Preis als ihr Gebot bezahlen werden.

Beim Aufzählen der vier bedeutendsten Auktionsformen unterscheiden Zwerger und Paulus (2002, S. 49 ff.) nicht nur nach den bereits erwähnten Auktionsformen mit offener und verdeckter Gebotsabgabe, sondern auch nach der Art der Gebote, der Art des Zuschlags und dem Zeitaspekt. Diese Unterscheidung wird auf der Tabelle 2.2 dargestellt. Dabei wurden die Auktionen nach dem Bezahlten vom Bieter Preis gegliedert:

- 1. Preis: Holländische und Höchstpreis Auktionen
- 2. Preis: Englische und Vickrey Auktionen

Tabelle 2.2: Auktionsformen, Quelle: Zwerger und Paulus (2002, S. 50)

Auktionsart	*Gebotsabgabe*	*Art der Gebote*	*Zuschlag*	*Zeitaspekt*
Holländisch	Offen	Fallend	Erstes Gebot	Zeitpunkt
Höchstpreis	Verdeckt	Einmalig	Höchstes Gebot	Zeitraum
Englisch	Offen	Ansteigend	Höchstes Gebot	Zeitpunkt
Vickrey	Verdeckt	Einmalig	Zweithöchstes Gebot	Zeitraum

Nach der Art der Gebote werden ansteigende, fallende und einmalige Gebote unterschieden. Bei den ansteigenden Geboten steigt der Preis mit jedem weiteren Gebot. Bei den fallenden Geboten sinkt der Preis und bei den einmaligen Geboten werden diese einmal und von allen Bietenden gleichzeitig abgegeben. Daher kann bei den Letzteren keine Tendenz der Preisentwicklung während des Auktionsablaufs ermittelt werden.

Des Weiteren unterscheiden sich Auktionen nach dem erhaltenen Zuschlag von den Bietern. Bei der Englischen und Höchstpreis-Auktion erhält der Bieter mit dem höchsten Gebot den Zuschlag. Bei der Vickrey-Auktion zahlt der Bieter

mit dem höchsten Gebot den zweithöchsten Preis. Die Holländische Auktion stellt hier eine Ausnahme dar, indem sie dem Erstbietenden den Zuschlag erteilt.

Zuletzt führen Zwerger und Paulus (2002, S. 50) den Zeitaspekt bei der Definition der Auktionsformen auf. Die Erteilung des Zuschlags kann zu einem Zeitpunkt stattfinden, wie bei der Englischen und bei der Holländischen Auktion. Dagegen fällt bei der Höchstpreis- und der Vickrey-Auktionen die Erteilung des Zuschlags innerhalb eines Zeitraums.

Es gibt eine Reihe weiterer Auktionsformen. Im folgenden sollen nur zwei weitere Formen kurz erwähnt werden:

- *Powerbuying/Group Buying*: Die Nachfragemengen kleinerer Nachfrager werden hier zusammengelegt. Die Käufer erhalten dadurch Mengenrabate.

- *Reverse-Auction/Ausschreibung/Request for Proposal*: Bei dieser Auktionsart gibt es im Unterschied zur Englischen Auktion mehrere Verkäufer und einen Käufer. Der Käufer ist bereit, eine bestimmte Menge einer Ware zu kaufen, wenn der Preis nicht einen gegebenen Betrag überschreitet. Bei dieser Auktion kommt es zum Unterbieten der Verkäufer. Den Zuschlag erhält der Verkäufer, der den niedrigsten Preis anbietet.

Die Regel für den Ablauf einer Auktion haben nach Roth und Ockenfels (2002, S. 2) eine wesentliche Bedeutung für das Endergebnis der Auktion. So unterscheiden sich eBay- von Amazon-Auktionen darin, dass sie eine feste Dauer haben („hard close“). Amazon-Auktionen werden dagegen erst dann beendet, wenn zehn Minuten lang kein Gebot mehr getätigt wurde („soft close“). Diese Unterschiede in den Regeln geben Anlass den Bietern bei eBay-Auktionen ihre Gebote in den letzten Minuten vor dem Schließen der Auktion abzugeben[26]. Bei Amazon-Auktionen wird dieses Phänomen nicht mit sehr hoher Intensität beobachtet.

Eine Studie von Roth und Ockenfels (2002, S. 7 ff.) kam zu diesem Ergebnis. Sie haben dazu 480 Biet-Historien von eBay- und Amazon-Auktionen verglichen, bei welchen Computer- und Antiquität-Artikel gehandelt wurden. So haben 40 % der Computer- und 59 % der Antiquität-Auktionen bei eBay Gebote in den

[26] Die Abgabe eines Gebots in der letzten Minute der Auktion wird noch als „sniping“ bezeichnet.

letzten 5 Minuten aufgewiesen. Bei Amazon sind es nur 3 % der Auktionen für jede der Kategorien die in den letzten 5 Minuten Gebote aufgewiesen haben. Darüber hinaus ergibt die Studie, dass von den 240 eBay-Auktionen 89 Gebote in der letzten Minuten und 29 Gebote in den letzten zehn Sekunden getätigt wurden. Im Gegenteil dazu wurde bei Amazon nur ein Gebot in der letzten Minute abgegeben.

Roth und Ockenfels (2002, S.3 ff.) geben eine Reihe strategische und nicht-strategische Gründe für dieses rationale Verhalten der Bieter wie Antworten auf das Verhalten bei Englischen Auktionen, die Vermeidung von „bidding wars“ und die Nutzung von Suchmaschinenergebnissen für kurzfristig ablaufende Auktionen.

Funktionen von Auktionen

Die Funktionen von Auktionen nach Wirtz (2001, S. 454 f.) und Klein (1997, S. 3) lassen sich wie folgt auflisten.

- *Koordinationsfunktion (co-ordination mechanism)*: Die Nachfrage und das Angebot für ein Gut werden als Bestimmungsfaktor für den endgültigen Preis genutzt.
- *Preisbildungsfunktion (social mechanism to determine a price)*: Bei seltenen Gütern, deren Preis nicht ohne weiteres bestimmt werden kann, können Auktionen als Mechanismus zum Bestimmen des Güterpreises genutzt werden. Der Käufer mit der höchsten Bewertung bekommt dabei den Zuschlag.
- *Revelationsmechanismus*: Nach o.V. (2004) herrscht bei den 2. Preis-Auktionen die Strategie des Bieters, den wahren Wert für den ersteigerten Artikel zu bieten. Das Bieten eines niedrigeren Betrags bringt ihm keinen Nutzen: Die Abgabe eines niedrigeren Gebots verringert seine Gewinnchancen und die Abgabe eines höheren Gebots dagegen kann dazu führen, dass er einen höheren Preis bezahlen muss.
- *Allokationsfunktion (efficient allocation mechanisms)*: Auktionen werden als Allokationsmechanismus für Produkte genutzt, deren Nachfrage nicht so hoch ist. Dabei akzeptieren die Käufer die Nachteile des Produktes dafür, dass sie einen Preis, der unter dem Niveau des normalen Preises liegt, bezahlen müssen. Dies können beispielsweise Produkte sein, die eine begrenzte Haltbarkeit aufweisen, oder welche, die innerhalb eines bestimmten Zeitraums aufgebraucht werden müssen.

- *Distributionsfunktion (high visible distribution mechanism)*: In dieser Funktion haben Auktionen das Ziel, einen neuen Distributionskanal zu öffnen.

Vorteile von Online-Auktionen

An den bereits dargestellten Funktionen von Auktionen lassen sich die wesentlichen Vorteile von Auktionen erkennen. Amor (2000, S. 70) listet folgende Vorteile von Auktionen auf:

- *Vorteilhafte Geschäftsabwicklung*: Alle Auktionsteilnehmer treffen sich auf dem Auktionsmarkt. Die Verkäufer legen am Anfang der Auktion ihre Bedingungen wie Startpreis, Dauer der Auktion und Mindestpreis fest. Durch das offene Medium Internet werden nach Skiera (1998, S. 300) eine hohe Anzahl an Nachfragern für das versteigerte Gut erreicht. Jeder von ihnen hat einen bestimmten Höchstpreis, den er zu zahlen bereit ist. Die hohe Anzahl an Nachfragern, die durch das Internet erreicht werden, und der Maximalpreis der einzelnen Nachfrager bilden den Höchstpreis für den Anbieter. Dieser Preis liegt über dem Minimalpreis, den der Bieter für seine Ware haben möchte. Das Medium Internet sorgt weiterhin für transparente Preise auf dem Markt, was den Nachfragern bessere Preisinformationen zur Verfügung stellt.

- *Geringe Transaktionskosten*: Für die Teilnehmer an einer Auktion entfallen an erster Stelle die Fahrtkosten. Bei Online-Auktionen müssen diese nicht persönlich während des Ablaufs der Auktion anwesend sein. Ferner können die Bieter einer Auktion von jedem Ort aus an dieser teilnehmen und sich über den aktuellen Stand informieren. (Skiera, 1998, S. 300) Weiterhin erlauben Online-Auktionen dank der niedrigen Transaktionskosten die Versteigerung von niedrigpreisigeren Objekten: „One of the big attractions of internet auctions is that buyers do not all have to gather at the same place to participate, so that sellers can use internet auctions to sell even relatively low value items to a potentially wide audience.“ (Ockenfels und Roth, 2002, S. 1)

- *Lagerbestände online versteigern*: Oft sind Restlagerbestände ein Verlustgeschäft für Hersteller und Distributoren. Sie können nach Amor (2000, S. 73) Auktionen nutzen, um kostengünstig ihre Restbestände zu verkaufen und dadurch zusätzlichen Gewinn erzielen.

- *Werkzeug zur Preisfindung*: Der Bietprozess legt den Preis des Gutes fest. Der Verkäufer bekommt dabei den höchsten Preis (nicht bei 2. Preis-Auktionen), der während der Laufzeit der Auktion erzielt werden kann.

Nachteile von Online-Auktionen

Obwohl Online-Auktionen sehr viele Vorteile mit sich bringen, entstehen dabei auch Nachteile gegenüber der physischen Abwicklung des Versteigerungsprozesses, welche bei dieser Art von Auktion in Kauf genommen werden müssen. Turban (1997, S. 8) verweist auf den fehlenden Kontakt mit dem zu ersteigernden Objekt und auf die Möglichkeiten für Betrug als wichtigste Einschränkungen der Online-Auktionen.

Online-Auktionen bieten die Möglichkeit, das zu ersteigernde Objekt kostengünstig online sichtbar zu machen. Leider ist diese Möglichkeit in manchen Fällen nicht ausreichend. Ein oder mehrere Photos auf dem Auktionssystem können nicht die Möglichkeit, physisch das Objekt zu sehen oder anzufassen, ersetzen.

Online-Auktionen bieten Möglichkeiten für Betrüger, auch Angebote in ein Auktionssysstem zu setzen. Es ist daher Pflicht des Auktionssystem-Betreibers, die betrügerischen Angebote herauszufiltern und den potenziellen Verkäufern dieser Waren nachzugehen. Ockenfels (2002) macht in diesem Zusammenhang Überlegungen darüber, welchen Einfluss Reputationsmechanismen auf den Handel in Online-Märkte haben. So bieten Online-Plattformen die Möglichkeit Informationen in der Form eines Feedbacks über Erfahrungen mit Marktteilnehmern abzugeben und dadurch den Handel auf Online-Marktplattformen zu erleichtern.

2.2.2 Geschäftsmodell der holländischen Auktion

Um den Prozess der Auktionsabwicklung innerhalb einer Online-Auktion zu ermöglichen, müssen sich der Angebotsgeber und der Gebotsabgebende zuerst identifiziert haben. Dies erfolgt in der Regel durch eine Anmeldung im Auktionssystem mit Hilfe eines Formulars. Bei der Erstanmeldung im System muss sicher gestellt werden, dass die vom Teilnehmer angegebenen Benutzerdaten gültig sind. Dazu müssen z.B. die ausgefüllten Felder auf leere Werte und abgegebene Werte, wie Postleitzahl oder Straße, auf ihre Gültigkeit geprüft werden.

Weiterhin muss von der technischen Seite her, eine sichere Abwicklung dieses Prozesses ermöglicht werden, welche z.B. das Abhören von Daten auf dem Weg zum Server oder zurück ausschließt. Es wird hier auf die Sicherheitsmechenismen auf der technischen Ebene zurückgegriffen. Die vorhandenen Möglichkeiten in diesem Bereich werden im Kapitel 3.1.3 aufgelistet.

In den nächsten Abschnitten werden die Punkte der einzelnen Prozesse der Auktionsabwicklung eines Holländischen Auktionssystems erläutert. Es wird angenommen, dass die Benutzer bereits eine Registrierung im System vorgenommen haben. Einige dieser Anregungen sind in Anlehnung an Amor (2000, S. 56 ff.) entstanden.

Einloggen der Teilnehmer

Bevor ein Angebot im System veröffentlicht wird oder ein Gebot abgegeben werden kann, muss sich der Systemnutzer identifizieren lassen. Bei der Erstanmeldung werden ein Benutzername und das dazugehörige Passwort vereinbart, die den Benutzer im System identifizieren. Diese werden auch vom System beim Einloggen des Benutzers angefordert und geprüft. Ebenfalls muss dafür gesorgt werden, dass die Abgabe des Passwortes nicht offen und sichtbar für andere Anwesende ist.

Auktionsobjekt online einsetzen

Das richtige Einstellen einer Auktion in das Online-Auktionssystem ist das wichtigste Kriterium, das den Erfolg am Auktionsende zu einem großen Teil ausmacht. Die Eingaben zu dem eingesetzten Artikel setzen sich zusammen aus:

- *Kategorie*: Ähnlich wie in einem Katalog besitzt ein Online-Auktionssystem Kategorien und Unterkategorien. Damit der richtige Käufer den Artikel zu Sicht bekommt, muss dieser auch in der betreffenden Kategorie platziert werden. Eine Veröffentlichung der Auktion in einer ungeeigneten Kategorie kann zum Misserfolg der Auktion führen, auch wenn das Objekt ein schneller „eye-catcher“[27] für die Versteigerer ist.

- *Der Name des Artikels*: Der Name des Artikels ist das Erste, was der potenzielle Bieter und Käufer sieht. Deswegen muss dieser klar formuliert werden. So werden die Gebotsgeber, die sich auch wirklich für den Artikel interessieren, so schnell wie möglich gleich vom Namen aus einer Liste von mehreren Objekten angesprochen werden.

- *Ein oder mehrere Bilder*: Die heutigen Technologien ermöglichen eine kostengünstige Erstellung von Bildern und auf diese sollte nicht verzichtet werden. Die Veröffentlichung von mehreren Bildern mit Vorder-, Seiten-

[27] eye-catcher: (engl.) Blickfang

und Detailansichten eines Artikels kann seine Beschreibung deutlich erleichtern und sogar verbessern. Dies ist vorteilhaft für die Käufer, die eine bessere Vorstellung von ihrer gesuchten Ware bekommen und für die Verkäufer, da durch die verbesserte Beschreibung des Objektes die Chancen, mehr Versteigerer anzuziehen, steigen. Das lässt den Preis des Artikels steigen.

- *Beschreibung*: Der beschreibende Text muss genau sein und sowohl die überzeugenden Eigenschaften als auch mögliche Mängel des Artikels angeben. Eine Formatierung der Beschreibung wird zur besseren Gestaltung und Lesbarkeit des Textes beitragen.

- *Startpreis*: Das Auktionsobjekt kann einen Startpreis besitzen. Der Startpreis bei der holländischen Auktion wird vom Verkäufer festgelegt. Er repräsentiert den Maximalpreis des Objektes, der in bestimmten Zeitabständen gesenkt wird.

- *Dauer der Auktion*: Die Dauer der holländischen Auktion wird ebenfalls vom Verkäufer festgelegt und bezeichnet die Laufzeit der Auktion von der Eröffnung bis zu ihrem Ende. Sie ist besonders wichtig für verderbliche Güter oder für solche, die in einer beschränkten Laufzeit verbraucht werden müssen.

- *Schlusspreis*: Der Schlusspreis ist der niedrigste Preis, für den der Verkäufer bereit ist, sein Produkt abzugeben.

Versteigerung

Im Gegensatz zum Veröffentlichen der Auktion, wird der Versteigerungsprozess von den Käufern ausgeführt. Der Verkäufer kann Auskunft über das Objekt geben und Fragen der Käufer während des Versteigerungsprozesses beantworten. Dies hängt davon ab, wie lange die Auktion dauert. Eine Dauer von zehn Tagen wie auch 10 Minuten ist möglich. Wesentlich dabei ist, dass die potenziellen Käufer die Möglichkeit haben müssen, vom Stattfinden der Auktion zu erfahren. Bei Auktionen von kurzer Dauer muss daher eine Vorankündigung über das Objekt, den Startpreis und die Dauer der Auktion stattfinden. Außerdem muss bei Kurzauktionen die Verbindung zwischen Server und Client während der Auktion permanent aufrecht erhalten werden. Der Versteigerungsprozess wird auf der Webseite des Auktionators mit den nötigen vereinbarten Informationen entsprechend angekündigt.

Auswertung der Gebote

Die Bewertung der Gebote hängt von der Auktionsart ab. Bei der holländischen Online-Auktion ist das erste Gebot auch das, welches den Zuschlag bekommt. Bei einer Mehrzahl von Objekten, die versteigert werden, wird die Auktion weiter fortgesetzt, bis alle vorhandenen Objekte verkauft sind.

Abschluss des Online-Auktionsgeschäftes

Die Auktion wird beendet, wenn die vorhandene Anzahl eines Artikels bereits versteigert wurde oder wenn die Auktionszeit abgelaufen ist. Bei Online-Auktionen werden die Gewinner in einer Auktion mit den dazugehörigen Informationen auf der Webseite bekanntgegeben. Meistens findet auch eine Benachrichtigung per Email statt. In diesem letzten Prozess der Kette müssen die Ware und der durch die Auktion vereinbarte Preis zwischen Käufer und Verkäufer ausgetauscht werden. Dazu kann es nötig sein, dass Käufer und Verkäufer Informationen wie Bankverbindung und Addressdaten austauschen.

Wertschöpfungsnetzwerk des Online-Auktionssystems

Analog zu der Wertschöpfungskette[28] nach Porter (1999, S. 63 ff.) und Scheer und Loos (2004, S. 28 ff.) ist das Wertschöpfungsnetzwerk eines Auktionssystems in der Abbildung 2.5 mit der virtuellen und physischen Kette dargestellt. Die Akteure der Geschäftsabwicklung sind die Verkäufer des zu ersteigernden Objektes, das Auktionssystem und der Käufer.

Das Auktionssystem spielt die Rolle eines Intermediärs im Geschäftsprozess. Dadurch ergeben sich die Beziehung Business-to-Business oder Consumer-to-Business zwischen dem Verkäufer und dem Auktionssystem. Zwischen dem Auktionssystem und dem Käufer kann die Beziehung Business-to-Business oder Business-to-Consumer annehmen. Die Art der Beziehungen im Online-Auktionssystem hängen davon ab, ob es sich beim Käufer und Verkäufer um eine Privatperson oder ein Unternehmen handelt.

Die Erstellung der Leistung vom Verkäufer läuft nach der eventuellen physischen Beschaffung und Lagerung der Waren ausschließlich über die virtuelle Wertschöpfungskette ab. Der Prozess der Versteigerung läuft bei Online-Auktionen ebenfalls über die virtuelle Kette ab. Nach dem Abschluss der Auktion kommt die physische Kette zum Einsatz. Der Austausch der Ware gegen

[28] Siehe Abschnitt auf Seite 24.

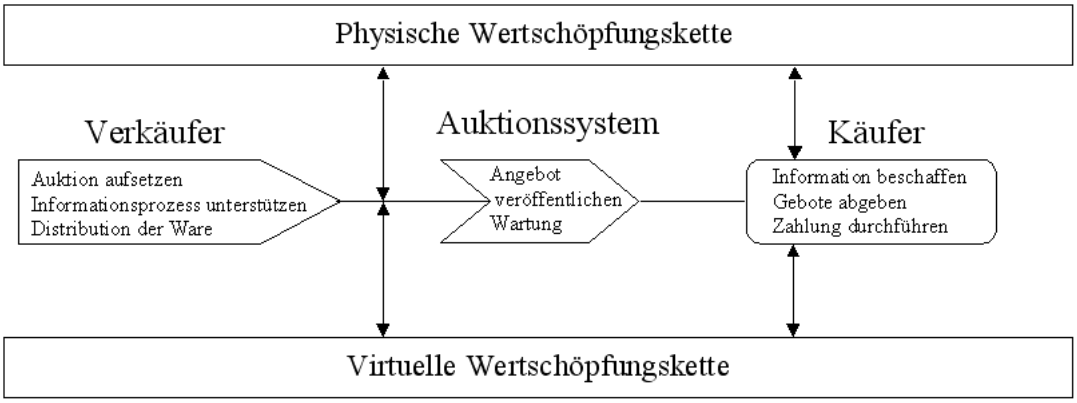

Abbildung 2.5: Wertschöpfungsnetzwerk eines Auktionssystems

seinen Geldwert läuft über die physische Kette ab und kann von der virtuellen Kette unterstützt werden.

Erlösmodell des Online-Auktionssystems

Ausgehend von den auf Seite 26 aufgelisteten Erlösquellen kann an dieser Stelle das Modell in der Tabelle 2.3 für das Auktionssystem aufgestellt werden. Die Haupteinnahmen für die Betreiber des Auktionssystems sind die Einnahmen pro eingesetztem Artikel. Des Weiteren können Betreiber Bannerwerbung einschalten, welche ihnen zusätzliche Einnahmen bereitstellt. Sie können weiterhin Nutzerprofile ihrer Kunden erstellen. Deren Weitergabe gegen Einnahmen muss anhand der Datenschutzrichtlinien der Gesetzgebung geprüft werden, da sie sensible persönliche Daten enthalten können.

Tabelle 2.3: Erlösquellen des Auktionssystems

Produkte	Einnahmen pro eingesetztem Artikel
Kontakte	Werbungseinnahmen
Information	Nutzerprofile erstellen

Nutzenbereitstellung des Online-Auktionssystems

Das Online-Auktionssystem verspricht sowohl den Verkäufern als auch den Käufern, die im System miteinander agieren, zusätzlichen Nutzen.

Für die *Verkäufer* stellt das Auktionssystem einen zusätzlichen Absatzkanal dar. Sie können dadurch beispielsweise ihren Restpostenbestand über das System verkaufen. Ihre Präsenz in der virtuellen Welt erhöht ihr Image und beschafft ihnen Kunden, deren Anzahl durch die Ortsunabhängigkeit der Teilnahme an der Auktion steigt. Eine höhere Anzahl von potenziellen Kunden sorgt für höhere Preise der abgesetzten Artikel und eine Erhöhung des Absatzes wie auch der Umsätze.

Die *Käufer* auf der anderen Seite können sich über die von ihnen gewünschten Produkten ortsunabhängig informieren. Ebenso können sie diese auch bequem von jedem Ort aus zu ihrem gewünschten Preis ersteigern. Beim Erwerb eines Artikels kann dieser beim persönlichen Zusammentreffen abgeholt werden oder bequem per Post dem Käufer zugesandt werden. Digitalisierbare Güter können gleich nach dem Ende der Auktion vom Käufer erhalten werden, was zusätzlich Zeit und Versandkosten erspart.

Die *Betreiber* des Auktionssystems konzentrieren sich auf die technische Umsetzung des Systems und erzielen ihren Nutzen durch die Einnahmen, die sie für jeden eingesetzten Artikel berechnen, durch mögliche Werbeeinnahmen als Erlös für die Freischaltung von Bannerwerbung oder Pop-Up-Fenstern.

2.3 Einsatzmöglichkeiten von XML

Bisher wurde der Datenaustausch hauptsächlich über das World Wide Web, das eigentlich für die Präsentation von Informationen zuständig ist, realisiert. Diese Informationen werden von Bitzer (2003, S. 224) hier als „*maschinenlesbar*“, aber nicht als „*maschinenverständlich*“ bezeichnet. Maschinenlesbar, da der Browser in der Lage ist, die Daten zu lesen und darzustellen, weil er HTML (Hypertext Markup Language) versteht. Maschinenunverständlich, da das HTML-Format ausschließlich zur Darstellung der Daten beiträgt. Demzufolge können die Daten nicht sinnvoll zur Weiterverarbeitung genutzt werden.

Um eine Weiterverarbeitung zu ermöglichen, müssen die eigentlichen Daten beschrieben werden. XML eignet sich besonders dafür, denn es ist in der Lage,

die eigentlichen Daten mit Metadaten[29] zu versehen. Damit spielen die XML-Tags[30] die Rolle der Daten über die Daten, und beschreiben demzufolge diese.

So sind im Semantic Web alle Daten mit Metadaten versehen, was die bessere Verarbeitung der Nutzdaten ermöglicht. Ein sinnvolles Beispiel für den Einsatz des Semantic Webs ist daher die Möglichkeit der gezielten Abfragen von Daten. Diese Abfragen liefern die qualitativ besseren Ergebnisse im Vergleich zum herkömmlichen Web.

2.3.1 Standards und Standard-Organisationen

Ein großer Teil der Akronyme in der IT-Welt besitzen eigene Standards. Sie definieren den Einsatz bestimmter Technologien, um die Kompatibilität entwickelter Systeme sicherzustellen. Die größte Standards-Organisation mit 147 Teilnahmestaaten ist die ISO (International Organization for Standardization)[31]. ISO besitzt weltweit Organisationen wie DIN (Deutsche Industrie Norm).

(Bryden, 2003, S. 3) definiert Standards als

> „...documented agreements containing technical specifications or other precise criteria to be used consistently as rules, guidelines, or definitions of characteristics to ensure that materials, products, processes and services are fit for their purpose.“

Eine weitere Organisation mit einer besonderen Rolle in der Welt des World Wide Webs ist das W3C[32] (World Wide Web Consortium). Es wurde von Tim Berners-Lee, dem Erfinder von HTML, im Oktober 1994 gegründet. Es ist organisatorisch betrachtet ein Projekt von:

- Das Massachusetts Institute of Technology[33]/Laboratory for Computer Science[34] in Zusammenarbeit mit CERN[35] (Centre Européen pour la Recherche Nucléaire),

[29] Metadaten sind Daten, die Informationen über andere Daten enthalten. Sie werden häufig als „Daten über Daten“ bezeichnet.
[30] XML-Tag: (engl.) Marke innerhalb eines XML-Dokuments. Siehe auch *Elemente und Attribute* auf Seite 134.
[31] http://www.iso.org
[32] http://www.w3c.org
[33] http://web.mit.edu
[34] http://www.lcs.mit.edu
[35] http://www.cern.ch

- Keio University Japan[36],
- ERCIM[37](European Research Consortium for Informatics and Mathematics)

und wird durch DARPA[38] (Defense Advanced Research Projects Agency) und durch die Europäische Kommission[39] unterstützt. W3C[40] unterscheidet sich von ISO grundsätzlich darin, dass es keine auf rechtlichem Wege einklagbaren Normen definiert. Daher erscheint der höchste Grad eines Dokuments, der zum Standardisierungsprozess gehört, in der Form einer *recommendation*. Das W3C veröffentlicht die fünf folgenden Dokumentenarten:

- *Working Draft*: Arbeitspapier mit konkreten Vorschlägen zu einem Thema, welches weiter bearbeitet wird.
- *Last Call Working Draft*: Die letzte Stufe eines Working Drafts, welches die Ziele bis zu einem gewissen Grad erreicht hat.
- *Candidate Recommendation*: Die Erreichung der definierten Anforderungen werden durch den Direktor des W3C bestätigt.
- *Proposed Recommendation*: Voraussichtlicher Standard.
- *Recommendation*: Die Proposed Recommendation wird durch den Direktor des W3C zur Recommendation erklärt und hiermit zu einem W3C-Standard.

Die Aufgabe des W3C in Bezug auf XML ist die Herausgabe von XML-Standards. OASIS[41] (Organization for the Advancement of Structured Information Standards) dagegen beschäftigt sich mit den Problemen der XML-Technologien und deren Implementierungen. Sie besteht aus mehr als 400 Teilnehmern. Diese hat auch unter ihrer Schirmherrschaft das XML.org[42]-Clearinghouse.

Erwähnenswert ist an dieser Stelle noch IETF[43] (Internet Engineering Task Force), welche Standards für Internet Protokolle wie TCP/IP (Transmission Control Protocol/Internet Protocol) entwickelt.

[36]http://www.keio.ac.jp
[37]http://www.ercim.org
[38]http://www.w3.org/Consortium/Prospectus/DARPA
[39]http://europa.eu.int
[40]http://www.w3.org/Consortium/
[41]http://www.oasis-open.org
[42]http://xml.org
[43]http://www.ietf.org

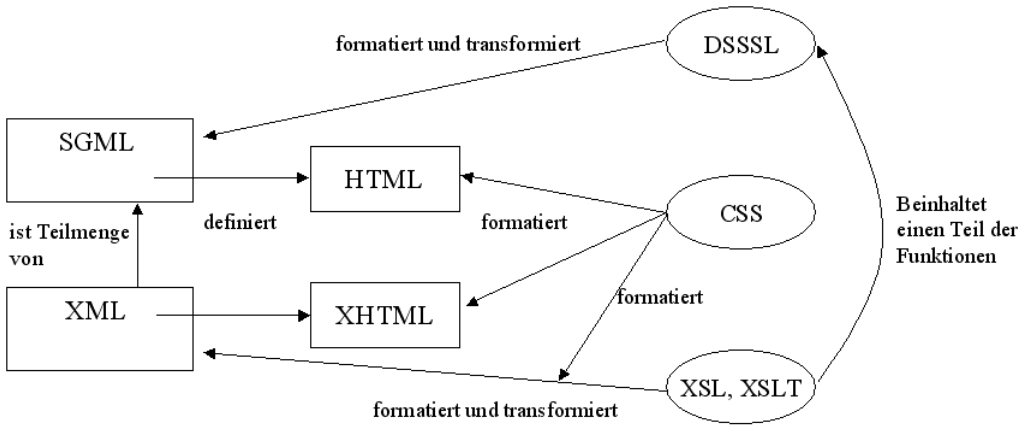

Abbildung 2.6: Zusammenhang zwischen SGML, HTML, XML, XHTML und deren Formatierungssprachen, Quelle: Mintert (2002, S. 27)

2.3.2 Kurze Geschichte der wichtigsten Auszeichnungssprachen

Die „Muttersprache“ von XML ist die Sprache SGML (Standard Generalized Markup Language), welche im Jahr 1986 als offizieller Standard mit dem ISO-Standard 8879 verabschiedet wurde. SGML legt die Form für die Inhalte elektronischer Dokumente fest. Die Formatierung des Inhalts sollte mit Hilfe der Sprache DSSSL (Document Style Semantics and Specification Language) ermöglicht werden. SGML ist allerdings eine sehr umfangreiche Sprache, was ihren Einsatz aufwendig macht. HTML ist wie SGML auch eine Auszeichnungssprache, welche im Jahr 1989 von Tim Berners-Lee zu Forschungszwecken entwickelt wurde. HTML kann als eine Anwendung der Sprache SGML betrachtet werden. Sie ist unter dem Einfluss des WWW (World Wide Web) zur meist verbreiteten Auszeichnungssprache geworden.

Im Dezember 1996 wurde die erste CSS (Cascading Style Sheet)-Recommendation veröffentlicht. Diese sollte die Idee von SGML, den Inhalt der Daten von der Präsentation der Daten zu trennen, im Umfeld von HTML ermöglichen.

Schnell wurden allerdings die Nachteile von HTML erkannt, was zu der Entwicklung von XML führte. Die Arbeit an XML begann im Jahr 1996. Seit dem 10. Februar 1998 ist XML ein offizieller Standard. Die Veröffentlichung erforderte eine Redefinition des Standards HTML nach den Gesetzen von XML. Daraus entstand im Januar 2000 der neue Standard von W3C — die Aus-

zeichnungssprache XHTML (Extensible Hypertext Markup Language).

2.3.3 Vorteile und Nachteile beim Einsatz von XML

Der Einsatz vom XML Standard bringt eine Reihe positiver Eigenschaften mit sich.

- XML ist ein einfacher Standard, welcher zum Erstellen weiterer Sprachen dienen soll.
- XML ermöglicht die semantische Repräsentation und Verarbeitung strukturierter Daten.
- XML ermöglicht die Kommunikation von Maschine zu Maschine und die Trennung vom Inhalt und seiner Darstellung.
- Der wesentliche oder sogar der wesentlichste Vorteil von XML mit ihren breiten Einsatzmöglichkeiten ist ihre Plattformunabhängigkeit. Daher wird in XML-Anwendungen immer häufiger XML mit der plattformunabhängigen Programmiersprache Java eingesetzt.
- XML bietet die Möglichkeit zur Darstellung von strukturierten Informationen und damit Möglichkeit zur Abbildung von Datenbankinformationen in Form von XML-Dokumenten.

Allerdings ist der Einsatz von XML auch mit Nachteilen verbunden, welche hier kurz erläutert werden sollen.

Ressourcen-Anspruchsvolle Sprache Die Einsatzmöglichkeiten von XML sind sehr vielversprechend bei der Lösung diverser Probleme. Dies ist allerdings nicht so problemlos, wie es auf den ersten Blick aussieht. XML ist zuerst *keine ressourcenschonende Sprache*, was sich negativ auf die Performance der XML-Anwendung auswirkt. So führen die sich wiederholenden XML-Tags zu großen XML-Dateien (*nicht redundanzfrei*). Dieser Effekt verstärkt sich mit zunehmender Größe des Elementnamens und mit zunehmender Anzahl der Elemente im XML-Dokument. Um weiterhin problemlos die Sprache XML einzusetzen, sind schnelle Prozessoren und viel Speicher notwendig. Analog sind beim Transport der Dateien über Netzwerke große Bandbreiten notwendig.

Seemann (2004, S. 30 ff.) gibt folgende Vorschläge für die Lösung der zuletzt genannten Probleme:

- *Komprimierter*: Der Vorschlag lehnt sich an den Transport von Daten in komprimierter Form an. Sie werden mit einem Verfahren wie z.B. gzip[44] komprimiert und in dieser Form transportiert. Der Empfänger muss nun die Daten entpacken, um diese zu lesen. Damit werden kleinere Datenmengen transportiert und kleinere Bandbreiten benötigt, um das gewünschte Ergebnis zu erzielen.
- *Kompakter*: Hier werden zwei Formen unterschieden:
  - Verarbeitung der XML-Dokumente in binärer Form als ein von Menschen unlesbarer Datenstrom und
  - Kodierung der zu ersetzenden Zeichenketten.

  Bei dem Letzteren werden die Daten aus der DTD (Document Type Definition) oder dem XML Schema gewonnen, um die Gewinnung der Daten für die Kodierung vorzunehmen. Diese Vorgehensweise wird von der Xqueese Markup Language (xqML)[45] angewendet.
- *Lesbarer*: XML-Syntax ist bei kleineren Dokumenten gut lesbar. Dieser Zustand verschlechtert sich allerdings mit einer wachsenden Größe und Strukturtiefe der XML-Dokumente. Projekte planen und setzen daher auch eine nicht-XML-Syntax voraus, um dieses Problem zu umgehen. Beispiele dafür sind die Schemasprache RelaxNG[46] und die Abfragesprache XQuery[47].
- *Transformierbarer*: In diesem Bereich werden Ansätze gesucht, die eine schnellere Transformation der XML-Dokumente in anderen Formate erlauben.

Offenheit der Sprache XML ist eine Sprache, die von Mensch und Maschine gut lesbar ist. Diese Möglichkeit bringt den Nachteil mit sich, dass nicht öffentliche Daten zu „unerwünschter Transparenz“ führen können. (Bitzer, 2003, S. 41) Um die Datenschutz-Verletzung zu vermeiden, müssen vertrauliche Daten verschlüsselt werden. Aus diesem Grund hat das W3C eine XML-Encryption-Arbeitsgruppe[48] gegründet, die einen Standard für die Verschlüsselung der XML-Daten entwickelt. Der XML-Encryption-Standard liegt bereits in der Form einer W3C Recommendation vor. Imamura u. a. (2002) Auf diese Standards wird im Kapitel 4.6 genauer eingegangen.

[44] http://www.gzip.org/
[45] http://xqueeze.sourceforge.net/
[46] http://www.relaxng.org/
[47] http://www.w3.org/XML/Query
[48] http://www.w3.org/Encryption/2001/

Trotz der Nachteile von XML nimmt ihr Einsatz rasant zu. XML hat zwei wesentliche Einsatzgebiete. Diese Sprache wird für den plattformunabhängigen Austausch von Daten zwischen Systemen, Programmen, Datenbanken und für eine flexible Darstellung in unterschiedlichen Formaten für unterschiedliche Medien eingesetzt. Das nächste Kapitel beschäftigt sich mit den Einsatzbereichen von XML.

2.3.4 Einsatzbereiche von XML

Bitzer (2003, S. 147 ff.) differenziert zwischen einem Einsatz in den Bereichen Publikation, Interaktion mit dem Client, Integration von Systemen, Kombination (Datenbankanbindung), Transaktion (e-Commerce-Datenaustausch) und positioniert diese in den Bereichen des e-Commerce. Die daraus entstehende Matrix ist aus der Abbildung 2.7 ersichtlich.

Auf der horizontalen Achse sind folgende Bereiche zu unterscheiden:

- S2S (System-to-System):
  In diesen Bereich fällt die Integration von Anwendungen.
- S2E (System-to-Employee):
  Dieser Bereich ermöglicht die Kommunikation zwischen den Unternehmen und den Mitarbeitern auf elektronischem Wege.
- B2B (Business-to-Business):
  Der Kommunikationsbereich auf Unternehmensebene wurde bereits im Kapitel 2.1.3 angesprochen.
- B2M (Business-to-Marketplace):
  Dieser Bereich ist häufig durch einzelne Großunternehmen initiiert und deckt die Beziehungen zwischen Kunden und Lieferanten.
- B2C (Business-to-Consumer):
  In diesem Bereich sind die Beziehungen zu den Endkunden präsent.

Jedem dieser Bereiche werden Aufgaben zugeordnet, die auf einen oder mehreren Ebenen des Unternehmens Implementierung finden. Diese Aufgaben werden nachfolgend näher betrachtet.

Publikation

An dieser Stelle geht es um den Einsatz von XML bei der Vorbereitung von Daten zur Publikation sowohl in elektronischer als auch in gedruckter Form.

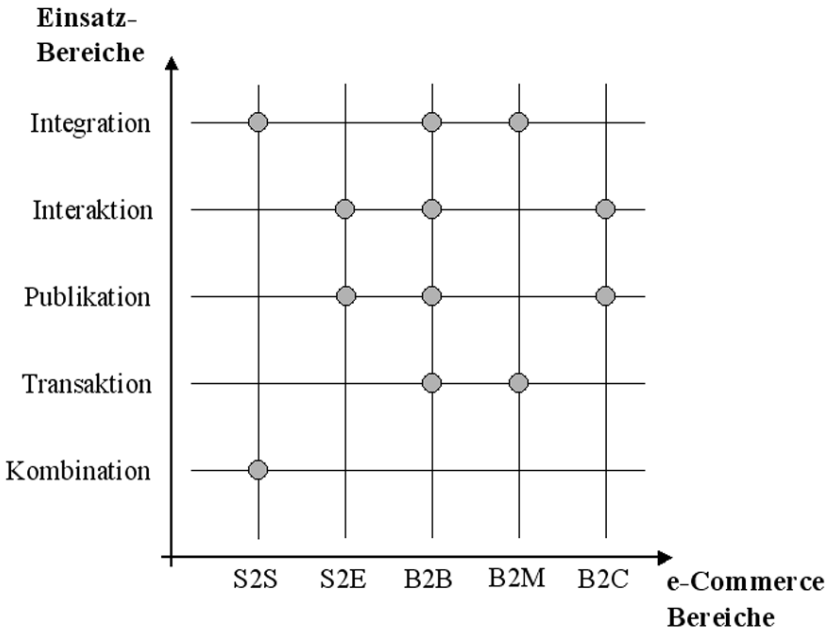

Abbildung 2.7: Einsatzfelder von XML, Quelle: Bitzer (2003, S. 148)

Im Folgenden sind einige Gründe für die optimierte Publikation mit XML aufgelistet:

- Das Format XML ist plattformunabhängig.
- Es ermöglicht einfache Publikation verschiedener Formate durch Trennung von Inhalt und Präsentation.
- Die automatische Bearbeitung von Publikationen ist möglich.

Ein beim Einsatz von XML zur Publikation betroffener Bereich ist das Content Management. Der Bereich *Content Management* unterstützt die Verwaltung von Inhalten und die Eigenschaft des XML-Standards zur Trennung des Inhalts der Daten von deren Darstellung und trägt zum erfolgversprechenden Einsatz des Standards in diesem Bereich bei.

Eine mögliche Kernkomponente des Content Managements ist eine XML-fähige Datenbank wie Tamino[49] oder Xindice[50], die als Medium zum Halten der Daten

[49] http://www2.softwareag.com/Corporate/products/tamino/default.asp
[50] http://xml.apache.org/xindice/

eingesetzt wird. Die benötigten Daten können mit Hilfe von XML-Stylesheetsprachen[51] in das gewünschte Format wie HTML, PDF (Portable Document Format) und XML zur Publikation transformiert werden. Dazu ist die XML-Datei mit den Nutzdaten und eine XSL-Datei zur Transformation[52] notwendig. Das Nutzen separater Dateien – eine für das Halten der Daten und eine für die Transformation – ermöglicht die strikte Trennung von Daten und deren Präsentation innerhalb eines Formats. Wenn es notwendig wird, die Daten zu aktualisieren, genügt allein das Editieren der XML-Datei.

Auf die Transformation zu HTML wird in dieser Arbeit mittels Beispielen im Kapitel 4.4.1 hingewiesen. Die Transformation nach PDF wird beim Erstellen eines PDF-Katalogs im Kapitel 4.4.2 veranschaulicht.

Interaktion

Bei der Interaktion geht es im speziellen Fall der Webanwendungen um die Kommunikation zwischen Mensch und Maschine. Sie findet heute häufig innerhalb von Webanwendungen über das Internet statt. Als Schnittstelle zwischen dem Benutzer und dem Anwendungssystem hat sich der Browser etabliert.

Der mögliche Einsatz von XML in Bezug auf die Interaktion ist im Bereich *eLearning* zu finden. Bruns und Dunkel (2003, S. 86 ff.) beschreiben die Entwicklung eines eLearning-Portals unter Einsatz von XML-Technologien. XML wird hier beispielsweise für die Erzeugung der dynamischen Inhalte der Präsentationsschicht der Anwendung genutzt. Es werden HTML-Inhalte erstellt, die in einem Browser aufgerufen werden können. Im Einsatz wird auch WML[53] (Wireless Markup Language) gebracht, um die Inhalte auf einem mobilen Endgerät zur Verfügung zu stellen. Durch den Einsatz der XML-Technologie wird eine flexible Trennung von Daten einerseits, Layout-Generierung andererseits und der Logik der Anwendung erzielt.

Zum Unterstützen des Interaktionsprozesses wurde beispielsweise das *Cocoon-*

[51] Stylesheetsprachen sind Sprachen, die für Formatierungszwecke eingesetzt werden. Beispiele für solche Sprachen sind CSS, XSL und DSSSL. Siehe dazu auch Abbildung 2.6 auf Seite 42.

[52] Sie dazu Abschnitt 4.4 auf Seite 164.

[53] Die WML-Spezifikation ist vom WAP-Forum, einen Zusammenschluss der Unternehmen Nokia, Phone.com, Motorola und Ericsson, entwickelt worden und dort weiterhin betreut. Siehe dazu http://www.wapforum.org/

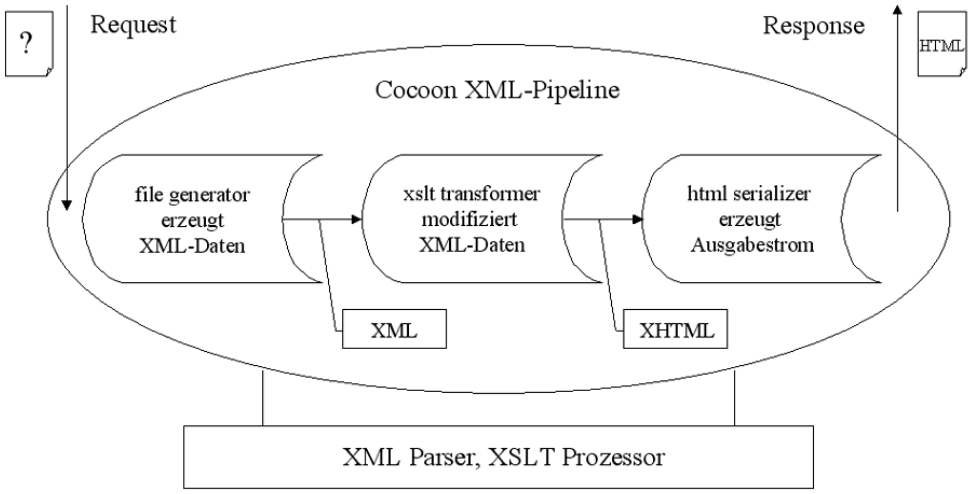

Abbildung 2.8: Die XML-Pipeline von *Cocoon*, Vgl. Langham und Ziegeler (2002, S. 43)

Softwarepaket[54] von Apache[55] entworfen. *Cocoon* ist ein integriertes Softwarepaket, welches auf Standardkomponenten wie XML-Parser[56] und XSLT-Prozessor[57] basiert. *Cocoon* kann in der Java Runtime Umgebung Apache Tomcat[58] eingesetzt werden. Das Erzeugen eines Dokuments findet in einer XML-Pipeline[59] statt.

Wie aus der Abbildung 2.8 ersichtlich, funktioniert die XML-Pipeline nach dem Request-Response-Prinzip. Dabei sendet der Client eine Anfrage an *Cocoon* auf

[54] http://cocoon.apache.org/

[55] http://www.apache.org/

[56] Ein Parser (engl. to parse: analysieren) „ist ein Programm, das entscheidet, ob eine Eingabe zur Sprache einer bestimmten Grammatik gehört. ... Während des Parsens erfolgt eine syntaktische Überprüfung der Eingangsdaten. Bei der Überprüfung wird in der Regel aus den Daten ein Parse-Baum aufgebaut, um die Daten anschließend weiterverarbeiten zu können“. (Wikipedia, 2004a)

[57] Ein Prozessor (lat. procedere: voranschreiten) „bezeichnet man in der elektronischen Datenverarbeitung (EDV) eine Verarbeitungseinrichtung“. (Wikipedia, 2004b) In dem vorliegenden Fall bezieht sich der Begriff auf Software, der XSLT-Stylesheets verarbeiten kann.

[58] http://jakarta.apache.org/tomcat/

[59] „Eine XML-Pipeline ist eine funktionale Einheit, die aus einer Kombination von verschiedenen Komponenten zusammengesetzt wird.“ (Langham und Ziegeler, 2002, S. 42) Die *Cocoon* XML-Pipeline besteht aus den Komponenten *file generator*, *xslt transformer* und *html rerializer* (siehe Abbildung 2.8).

dem Server. Dieser verarbeitet die Anfrage mit der *Cocoon* XML-Pipeline und generiert eine Antwort, welche dann an dem Client gesendet wird. Bei der Verarbeitung der Anfrage an das System wird zunächst das XML-Dokument von einem *file generator* geliefert. Die Quelle der Daten für den Generator kann die Festplatte oder eine Datenbank, aber auch eine externe Datenzustellung, über eine HTTP-Verbindung (beispielsweise aus dem Extra-, Intra- oder Internet) sein. Eine weitere Komponente der Pipeline ist der *xslt transformer*. Er kann die bereits bestehende XML-Datei beispielsweise in ein Format wie beispielsweise XHTML überführen. Dabei kann die vom Generator erzeugte XML-Datei modifiziert werden, indem bestehende Elemente geändert oder entfernt werden oder neue Elemente eingefügt werden. Anschließend enthält die Pipeline einen *html serializer*[60], der den eigentlichen Ausgabe-Datenstrom erstellt und als Antwort auf die Anfrage an den Client weitergibt. Alternativ zum HTML-Format kann die Ausgabe in PDF erfolgen.

Integration

Die Integration deckt den Bereich der Kommunikation von Maschine zu Maschine und betrifft den Austausch von Daten zwischen homogenen und heterogenen Anwendungen diverser Betriebssysteme. Der zunehmende Einsatz elektronischer Systeme trägt zur raschen Entstehung von e-Business-Lösungen bei. Das sind Anwendungen, an deren Entwicklung, Bedienbarkeit und funktioneller Umsetzung immer höhere Anforderungen gestellt werden.

Als Standard-Struktur für heutige Webanwendungen gilt die Client-Server-Architektur. Die hohen Ansprüche an Webanwendungen führen zu ihrer steigenden Komplexität. So wird ihre Ausführung in drei Schichten aufgeteilt: Datenschicht, Datenverarbeitungsschicht und Präsentationsschicht. Die Funktionalität der Schichten wird auf mehreren Rechnern verteilt, um ihre Leistung den Ansprüchen gerecht zu halten. Auf die Grundsätze der Struktur von Webanwendungen wird im Kapitel 3.1.1 eingegangen.

Die Aufteilung der Webanwendung in mehreren Schichten und die steigende Komplexität besonders in der Datenverarbeitungsschicht erfordert das sinnvolle Zusammenwirken der einzelnen Komponenten. Das kann von XML unterstützt werden, indem beispielsweise die vom Client zum Server gesendeten Daten in XML umgeformt und mittels einer DTD oder einem Schema statt mit einem speziell erstellten Skript geprüft werden. (Bitzer, 2003, S. 179) In

[60] Ein Serializer erzeugt einen Binär- oder Textdatenstrom für den Client.

der Datenverarbeitungsschicht selbst können auch Daten im XML-Format ausgetauscht werden. Eine mit XML und Java implementierte Anwendung trägt damit zur Plattformunabhängigkeit und Flexibilität der Software bei. (Bitzer, 2003, S. 184)

XML lässt sich nicht nur innerhalb von Webanwendungen implementieren. XML kann als Format für den Datenaustausch auch zwischen Webanwendungen eingesetzt werden. Da XML-Daten reine Text-Daten sind, können sie über TCP/IP und damit über das Internet transportiert werden. An dieser Stelle spielen Web Services eine bedeutende Rolle. Sie werden im Kapitel 5 näher beschrieben.

Transaktion

Hier geht es um den Austausch von Daten zwischen Unternehmen, aber auch innerhalb der Unternehmensbereiche, womit Bereiche wie ERP[61] (Enterprise Resource Planning), SCM[62] (Supply Chain Management), CRM[63] (Customer Relationship Management) betroffen sind.

Enterprise Resource Planning ist nach net-lexikon.de (2004a) ein Begriff,

> „...der sich auf betriebliche Planung, Buchführung und das Management von Unternehmensressourcen wie Lagerhaltung, Einkauf, Absatz, usw. bezieht“.

Das Ziel des ERP ist, die zur Verfügung stehenden Ressourcen möglichst effizient einzusetzen. Eine der wichtigsten Einsatzmöglichkeiten im Rahmen des Unternehmensbereiches ERP ist die Möglichkeit zur Integration von Systemen diverser Bereiche eines Unternehmens. So können beispielsweise Vertrieb und Finanzwesen mittels XML-Schnittstellen sinnvoll miteinander gekoppelt werden.

Pickruhn (2003, S. 188) unterscheidet zwei folgende unternehmensinterne Schnittstellen:

- *Schnittstellen im Bereich eigenentwickelter Software*: Diese Verbindung kann einen transaktionsorientierten Datenaustausch realisieren, welcher

[61] ERP (Enterprise Resource Planning): (engl.) die Warenwirtschaft
[62] SC (Supply Chain): (engl.) die Beschaffungskette, die Lieferkette
[63] CRM (Customer Relationship Management): (engl.) Verwaltung von Kundenbeziehungen

> durch Datentransaktionen von kleinen Größen und durch hohe Leistungsfähigkeit und Stabilität charakterisiert wird; eine zweite Möglichkeit ist der batchorienterte[64] Datentransfer, welcher sich durch großes Datenvolumen und Verarbeitung der Daten in bestimmten Zeitspannen auszeichnet.

- *Schnittstellen im Bereich zugekaufter Software*: Diese Schnittstellen verknüpfen eigenentwickelte mit zugekaufter Software.

Die XML-Technologie kann weiterhin im Bereich des *Supply Chain Managements* eingesetzt werden. Das Supply Chain Management ist auf die gesamte Wertschöpfungskette verteilt und hat die Aufgabe, für ein Gesamtoptimum zu sorgen.

„Supply Chain Management

- ist ein prozessorientierter, integrierter Ansatz für die Beschaffung, Herstellung und Auslieferung von Produkten, Systemen und Services an Kunden.

- umfasst Auftragnehmer, Unterauftragnehmer und Lieferanten, unternehmensinterne Funktionen und Prozesse, Handelspartner, Logistikdienstleister, Einzel- und Großhändler sowie Endkonsumenten.

- bezieht sich auf die Planung, Optimierung und das Management von Material-, Informations- und Zahlungsströmen.“ (Schmidt, 2002, S. 514)

XML trägt seinen Teil dazu bei, indem dieser Standard die flexible Anbindung der Prozessketten der einzelnen logistisch hintereinander liegenden Unternehmen ermöglicht. (Schinzer und Thome, 1999, S. 214)

Unternehmen nutzen den elektronischen Datenaustausch auch im Bereich des Customer Relationship Managements. Schomakers (2001, S. 145) definiert *Customer Relationship Management* als

> „...ein strategischer Ansatz, mit dem Firmen ihre Profitabilität durch die Erhöhung der Kundenzufriedenheit steigern“.

Nicht mehr so sehr die Produktorientierung, sondern vielmehr die Kundenorientierung steht dabei im Mittelpunkt. Der „gläserne Kunde“ und bessere

[64] batch: (engl.) Stapel, stapelweise verarbeiten

Kenntnise über das örtliche und zeitliche Konsumverhalten, die örtlichen Einzugsgebiete, die Vorlieben und die Umsatzhöhen des Kunden sind Ziel des CRM. Diese Informationen erlauben eine umfangreichere und bessere Konzentration beim Verfolgen des obersten Ziels der Kundenzufriedenheit und dienen damit dem Unternehmenserfolg.

Die so genannte „Multi-Channel-Fähigkeit“ als Eigenschaft des CRM gibt den Kunden mehrere Möglichkeiten, mit dem Unternehmen in Verbindung zu treten, wie über das Telefon, Fax, Post oder Internet. (Schomakers, 2001, S. 150) Bei dem letzten Kanal werden die Vorteile des Internets bei der Interaktion mit Kunden genutzt. Es handelt sich dabei um das eCRM[65] (electronic Customer Relationship Management).

Brandstetter und Fries (2002, S. 38 f.) unterscheiden zwischen analytischem und operativem eCRM. Beim *analytischen eCRM* geht es um die Beobachtung und Auswertung des Kundenverhaltens. Das *operative eCRM* hat die Aufgabe der Anwendung von eCRM-Methoden. Das analytische und operative eCRM müssen in enger Verbindung und Zusammenarbeit stehen, um das Ziel der Kundenzufriedenheit besser zu erreichen. Die Funktion dieser zwei Bereiche kann mit Hilfe des XML-Standards unterstützt werden. So können mit XML Daten aus unterschiedlichen Datensystemen in ein einheitliches Format gebracht und dem eCRM zur Verfügung gestellt werden.

Für die optimale Prozessgestaltung und Zielerreichung im eCRM sollte dieser nicht als alleinstehender Bereich betrachtet werden. Supply Chain Management[66], Business Intelligence[67] (BI), Enterprise Resource Management[68] (ERM) sind Bereiche, mit welchen CRM sich eng abstimmen sollte. Da die Vernetzungen innerhalb des Unternehmens eine wachsende Rolle spielen, ist es um so wichtiger, die Beziehungen zwischen eCRM, SCM, BI und ERM auf technischer Ebene zu implementieren. Hier muss das Problem der Integration von Systemen, beispielsweise mit Hilfe von Web Services und deren XML-basierten Standards, gelöst werden. Somit können aktuelle Daten für den analytischen eCRM-Bereich „on the fly“ gewonnen werden. Das operative eCRM kann die aufbereiteten Daten durch die Multimedialität des Internets auf unterschiedlichen Endgeräten ausgeben. Der effiziente Ansatz für die Realisierung dieser

[65] Siehe auch Fußnote 63.
[66] Siehe Fußnote 62.
[67] Business Intelligence ist der Unternehmensbereich, der angemessene wirtschaftliche Handlungen anhand von Unternehmensdaten bestimmt.
[68] Siehe Fußnote 61.

Aufgabe ist die Trennung der Daten von deren Darstellung, welcher mit XML umgesetzt werden kann. Es handelt sich dabei um die gleichen Informationen, die bei der Interaktion mit den Kunden eingesetzt werden sollen. Mit Hilfe von XML können die zu vermittelnden Informationen aufbereitet und in geeigneten Formaten zur Darstellung, sowohl auf dem Bildschirm des Kunden als auch auf seinem Mobiltelefon oder PDA[69] aufbereitet werden.

Kombination

In diesem Bereich geht es um den Export und Import von Daten zwischen XML-Formaten und Datenbanken.

Der Export von Daten aus relationalen Datenbanken[70] leistet nach Hohenstein (2003, S. 52) folgende Vorteile:

- XML wird für den Datenaustausch zwischen Rechnern eingesetzt.
- XML unterstützt die Präsentation der Daten. Mit Hilfe von Stylesheets kann kunden- und endgerätespezifisch eine dynamische Selektion und Darstellung der Daten aus relationalen Datenbanken stattfinden.

So nutzt die Auktionssystem-Anwendung Daten aus der Datenbank, um XML-Daten zu generieren und diese anschließend weiter zu verarbeiten. Eine relationale Datenbank enthält sowohl die Nutzdaten der Anwendung als auch die Daten über die zu erstellenden XML-Daten (die Metadaten). Das Erstellen der Letzteren aus den Nutzdaten der Datenbank findet mit Hilfe von Mappings[71] über die vorhandenen und die gewünschten Strukturen statt.

Auf die eigentliche Vorgehensweise beim Erstellen der XML-Daten wird im Kapitel 4.2.2 eingegangen.

[69] Siehe Fußnote 4.
[70] Siehe Fußnote 6 und Kapitel 3.3.2 auf Seite 71.
[71] mapping: (engl.) Zuordnung

Kapitel 3

Architektur des Auktionssystems

> *„Wir können jetzt Technologien nutzen, die noch vor einem Vierteljahrhundert in den kühnsten Träumen der Futuristen nicht vorkamen.“*
> *(Kauffels, 2001, S. 23)*

Die stetig wachsende Komplexität der Struktur von Webanwendungen wird von der Komplexität und den Anforderungen des Electronic Commerce beeinflusst. Die resultierende Vielschichtigkeit von Anwendungen erfordert weiterhin eine fehlerfreie Funktion und ein störungsfreies Zusammenwirken der einzelnen Bausteine von Webanwendungen und stellt hohe Anforderungen an das System als Ganzes.

Dieses Kapitel beschäftigt sich zuerst mit der Struktur von Webanwendungen und den verfügbaren für ihren Aufbau Technologien und Standards. Des Weiteren werden auf die allgemeinen Anforderungen an Webanwendungen und auf die Ziele der zu implementierenden Auktionssystem-Anwendung eingegangen. Anschließend werden die Datenbasis und die Business Logik der Anwendung beschrieben. Diese drei Themen bilden die Grundlage für den weiterführenden praktischen Teil der Arbeit, welcher im Kapitel vier vertieft wird.

3.1 Überblick über Webanwendungen

In diesem Abschnitt wird auf die grundsätzlichen Eigenschaften von Webanwendungen eingegangen. Eine Anwendung ist nach dem net-lexikon.de (2003) eine Software, die auf einem Rechner oder Rechnersystem läuft und Benutzern zur Erledigung von bestimmten Aufgaben dient. Kappel u. a. (2003a, S. 2) definieren eine Webanwendung als

> „...ein Softwaresystem, das aus Spezifikationen des World Wide Web Consortiums (W3C) beruht und Web-spezifische Ressourcen wie Inhalte und Dienste bereitstellt, die über eine Benutzerschnittstelle, den Web-Browser, verwendet werden.“

3.1.1 Allgemeine Struktur

Ein bereits etabliertes Systemarchitektur-Modell für Webanwendungen ist das Client/Server-Modell.

> „Ein System wird als Client/Server-System bezeichnet, wenn es aus mehreren Komponenten besteht, diese miteinander kommunizieren und eine Aufgabenverteilung zwischen ihnen stattfindet in der Form, daß eine Komponente (Client) Dienste von einer anderen Komponente (Server) anfordert, die diese Dienste erbringt.“ (von Thienen, 1995, S. 5)

In diesem Modell interagieren die Komponenten Client und Server miteinander. Der Server stellt dabei dem Client Dienste zur Verfügung. Der Client besteht aus einem Browser, mit dessen Hilfe die Webanwendung angesprochen werden kann. Der Server stellt in der Regel den Datenbestand des Systems in Form einer Datenbank zur Verfügung (Datenbank-Server) und beantwortet die vom Client gestellten Anfragen. (Hoppe u. a., 2002, S. 1427)

Demnach können bei Webanwendungen, die nach diesem Modell aufgebaut sind, die aufgelisteten drei Schichten abgegrenzt werden (Winzerling 2002, S. 13[1]; Weitz 2002, S. 210):

- Die *Datenschicht* (1) beschäftigt sich mit der Speicherung der Daten. Sie ist in der Regel ein Datenbankprogramm, welches auf einem Computer ausgeführt wird (1. Schicht des Systems).
- Die *Datenverarbeitungsschicht* (2) beschäftigt sich mit der Aufbereitung (Verarbeitung) der Daten und sorgt für den logischen Aufbau der Webseiten. Die Verarbeitung der Daten erfolgt im Allgemeinen auf dem Server. Dieser verarbeitet sie selbst oder fragt einen anderen Computer danach (2. Schicht des Systems).
- Die *Präsentationsschicht* (3) ist verantwortlich für die Darstellung der zu vermittelnden Inhalte. Sie ist mittels Browser an den Client gebunden (3. Schicht des Systems).

Die Abbildung 3.5 auf Seite 73 bildet die Struktur des Auktionssystem-Prototyps hinsichtlich der drei Schichten ab, welche im Rahmen dieser Arbeit entwickelt wurde. Die Technologien, die für die Realisierung von Webanwendungen verwendet werden, werden im Kapitel 3.1.2 erläutert. Auf die eigentliche

[1] Winzerling (2002) spricht von den drei Funktionen einer Client-Server-Anwendung.

Realisierung der Anwendung wird in den späteren Kapiteln näher eingegangen.

Abhängig davon, auf welchen Geräten die einzelnen Schichten laufen, wird folgende Unterscheidung der Anwendungen getroffen. (Langner, 2002, S. 14 ff.)

- *Single-tired* Anwendung: Die drei Schichten der Anwendung befinden sich auf demselben Computer.

- *Two-tired* Anwendung: Schicht zwei und drei agieren auf dem selben Computer. Die Datenbank befindet sich in diesem Fall auf einem anderen Computer.

- *Three-tired* Anwendung: Alle drei Komponenten sind auf drei verschiedene Computer verteilt.

- *N-tired* Anwendung: In der Praxis existiert bei größeren Anwendungen eine Aufteilung der zweiten Ebene in weiteren Schichten.

Die Aufteilung der Dienste auf mehrere Server hat den Vorteil, dass die Aufgaben und damit die Belastung für den einzelnen Computer nicht zu gross werden, da beispielsweise die Größe von Datenbanken ständig wachsen. Das ist besonders relevant, wenn die Anzahl der Benutzer und damit das Transaktionsvolumen des Systems steigt. Die Systemkonzeption sollte also von Anafang an Erweiterungsmöglichkeiten beinhalten.

3.1.2 Technologien

Unumstritten sind die zugrundeliegenden, meist verbreiteten Technologien zur Entwicklung von webbasierten Anwendungen: Das sind die Protokolle TCP/IP, HTTP[2] und SHTTP (Secure Hypertext Transfer Protocol), serverseitige Technologien[3] wie HTML, CSS, XML, Cookies, Programmiersprachen wie Java, Perl, PHP und Webserver wie z.B. Apache HTTP- und Apache Tomcat Server. Als clientseitige Technologien[4] können der Web-Browser, Java-Applets, Java-Skripts und Plugins bezeichnet werden.

[2]Siehe W3C (2004b).
[3]Siehe Seite 60 dieses Abschnitts.
[4]Siehe Seite 59 dieses Abschnitts.

Internetprotokolle

Das TCP/IP-Schichtenmodell ist die Grundlage für die Datenübertragung im Internet. Es wird in der Literatur oft auch als Vier-Schichten-Modell bezeichnet, welches die verschiedenen Protokolle unterteilt. (Heinzmann, 2002, S. 46) Die Spezifikation des TCP/IP-Protokolls ist im RFC 793[5] enthalten. (RFC793, 2004)

Nachfolgend werden die Schichten des TCP/IP-Modells und die relevanten für jede Schicht Protokolle anhand der Abbildung 3.1 auf Seite 58 kurz erläutert.

Auf der untersten Ebene des Modells befinden sich ein oder mehrere Hardware-Geräte mit Netzwerkkarten, mit deren Hilfe die vorbereiteten Daten verschickt werden können. Abhängig von den zu bewältigenden Aufgaben, die die Rechner ausführen müssen, müssen entsprechende Systemvoraussetzungen gegeben sein wie beispielsweise eine Mindestgeschwindigkeit, eine bestimmte Festplatten- und Arbeitsspeicher-Größe sowie entsprechende andere technische Voraussetzungen. Protokolle aus der Hardwareschicht sind nicht innerhalb von TCP/IP definiert.

- *Verbindungsschicht* (1): Hier spielen Netzwerk-Topologien und Zugriffsverfahren wie *Fiber Distributed Data Interface* (FDDI), *Address Resolution Protocol* (ARP) und *Reverse Address Resolution Protocol* (RARP) eine große Rolle. Weitere eingesetzte Protokolle sind X.25, *Serial Line Interface Protocol* (SLIP).

- *Internetschicht* (2): Auf der Internetschicht des TCP/IP-Modells befindet sich das *Internet Protocol* (IP). Wichtiger Bestandteil dieses Protokolls ist die IP-Adresse. In der Version 4 ist die IPv4 eine Adresse aus 32 Bits (= 4 Bytes). Die Notierung besteht daher aus vier Zahlen, die mit einem Punkt getrennt werden. Die Übertragung der von der Transportschicht vorbereiteten Pakete findet mit Hilfe des IP-Protokolls statt.

- *Transportschicht* (3): Die Transportschicht ist unter der Anwendungsschicht ansässig. Protokolle dieser Schicht sind das *Transmission Control Protocol* (TCP) und das *User Datagram Protocol* (UDP). Während TCP ein verbindungsorientiertes Protokoll ist, ist UDP verbindungslos. In dieser Schicht wird eine Nachricht in mehrere Datenpakete zerlegt und quittiert. TCP beachtet bei der Verarbeitung der Daten die Reihenfolge und Nummerierung der Datenpakete. UDP vernachlässigt im Gegensatz

[5]Siehe dazu ftp://ftp.nic.de/pub/rfc/rfc793.txt.

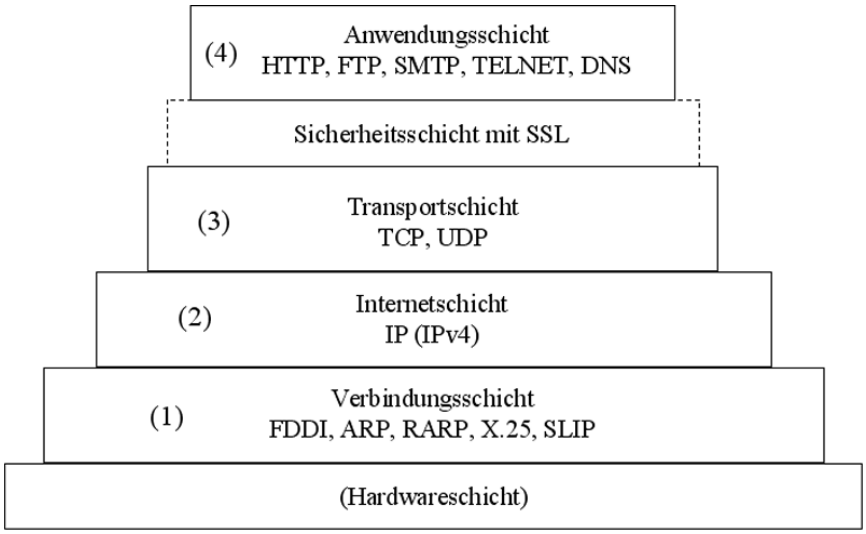

Abbildung 3.1: TCP/IP-Schichtenmodell

dazu die Reihenfolge der Pakete, die doppelten oder fehlenden Pakete. Daher ist UDP auch das schnellere Protokoll.

- *Anwendungsschicht* (4): Die Anwendungsschicht stellt die oberste Schicht dar und dient, wie vom Namen her ersichtlich, als Schnittstelle für die Anwendungsebene. In dieser Schicht befinden sich die Dienstprotokolle des Internets. Die bekanntesten Dienste des Internets sind der WWW-Dienst, welcher über das *Hypertext Transfer Protocol* (HTTP) ermöglicht wird und der Email-Dienst, welcher über das *Simple Mail Transfer Protocol* (SMTP) ermöglicht wird. Das *File Transfer Protocol* (FTP) erlaubt die Übertragung von Daten zwischen Rechnern. Weitere Protokolle wie z.B. TELNET[6] ermöglichen den direkten Zugriff auf eine Maschine, die sich an einem vom Benutzer entfernten Ort befindet. Das *Domain Name System* (DNS) ermöglicht die Zuordnung von Rechnernamen zu deren IP-Adressen.

Das bedeutendste Protokoll für Webanwendungen aus den bereits aufgezählten ist das HTTP. Das Protokoll HTTP ermöglicht den WWW-Dienst auf der Anwendungsschicht der TCP/IP-Protokollfamilie. Es sorgt für die Datenübertragung zwischen dem Client und dem Web-Server. Die Spezifikation von HTT-

[6]TELNET bietet den Benutzern den Zugang zum Internet über die Kommandozeile.

P/1.0 ist in RFC 1945[7] und von HTTP/1.1 in RFC 2616[8] beschrieben. HTTP in seiner Version 1.1 kann eine dauerhafte Verbindung aufbauen. Dies bedeutet, dass bei Webseiten mit mehreren Links nur eine Verbindung aufgemacht werden muss. Bei HTTP in der Version 1.0 muss dagegen für jeden Link eine neue Verbindung zur Verfügung gestellt werden. (Schmeh, 2001, S. 412) Der Client sendet mit Hilfe einer URL seine Anfrage an dem Server. Es wird eine TCP/IP-Verbindung aufgebaut, die in der Regel über den Port[9] 80 ausgeführt wird. Der Server sendet über die aufgebaute Verbindung die angeforderten Daten als Antwort zurück. (Kauffels, 2002, S. 388)

Eine Schwachstelle von HTTP ist die unsichere Verbindung, die aufgebaut wird. Eine Nachbesserung dieses Nachteils liefern die Protokolle SHTTP und SSL (Secure Socket Layer). SSL ist eine eigene Schicht innerhalb von TCP/IP und befindet sich zwischen der Transport- und der Anwendungsschicht (siehe Abbildung 3.1). Sie hat die Aufgabe, Daten vor dem Absenden zu verschlüsseln und Daten nach dem Empfang zu entschlüsseln. SHTTP[10] ist ein Protokoll, das durch die Verschlüsselung der Daten für die Sicherheit von HTTP sorgt. Weiteres zum Thema TCP/IP vergleiche Pidde (2004) und Hansen und Neumann (2001, S. 1180 ff.).

Clientseitige Technologien

Webbasierte Anwendungen werden von einem Client mittels eines Browsers angesprochen. Ein *Browser* ist eine Software, welche auf dem Client läuft und zum Abruf von WWW-Inhalten genutzt wird. Die benötigten Daten werden über die so genannte URL (Uniform Resource Locator) aufgerufen. Diese stellt den Verweis auf die benötigten Inhalte und hat die Form *Protokoll://Server/Verzeichnis/Datei*. (Kappel u. a., 2003b, S. 101)

Auf den HTML-Seiten, die von einem Browser aufgerufen werden, können *Java-Script*-Programme ausgeführt werden. Das sind kleine Programme, welche im Quelltext der HTML-Seite eingebettet sind und auf dem Client ausgeführt werden.

[7] Siehe RFC1945 2004.
[8] Siehe RFC2616 2004.
[9] Ein Port wird durch eine Nummer dargestellt und repräsentiert den Dienst, der auf dem Server aufgerufen wird. Die Standardports einiger wichtiger Dienste sind WWW - 80; DNS - 53; FTP - 20, 21; Email - 110(POP), 25(SMTP); Telnet - 23. (Janowicz, 2002, S. 15)
[10] Siehe RFC2660 2004.

Seit der Vorstellung der Programmiersprache Java von Sun® Microsystems[11] wurden *Java-Applets* eine wichtige clientseitige Technologie. Java-Applets sind kleine Java-Programme, die im Web-Browser integriert und anschließend im Client geladen und ausgeführt werden. Zum Ausführen des Applets, welches im Bytecode vom Server zur Verfügung gestellt wird, ist eine JVM (Java Virtual Machine) notwendig. Die JVM interpretiert den plattformunabhängigen Bytecode in einen ausführbaren plattformabhängigen Programmcode zur Laufzeit der Anwendung. Die verfügbare Installation des JVMs kann durch Eingabe des Kommandos *jview* im DOS-Fenster geprüft werden. Die JVM kann auch nachträglich von Sun® Microsystems heruntergeladen und installiert werden. Besonders beliebt und damit von bedeutender Relevanz ist der Zugriff von Java-Applets auf Datenbanksysteme. (Kappel u. a., 2003b, S. 107)

Eine weitere Technologie, die auch mit Hilfe des Web-Browsers eingesetzt wird sind *Plugins*. Ziel dieser Programme ist es, die Funktionalität von Browsern zu erweitern. Ein verbreitetes Beispiel für ein Browser-Plugin ist das Macromedia® Shockwave™-Plugin[12]. Es wird zum Abspielen von Animationen innerhalb des Browsers genutzt. Ein weiteres Beispiel ist der RealPlayer®[13] von RealNetworks®[14], welcher zum Abspielen von Audio- und Videodateien eingesetzt wird.

Serverseitige Technologien

Die wichtigste serverseitige webanwendungsunterstützende Technologie ist HTML. Das ist die Sprache, die zur Zeit am häufigsten zur Beschreibung von Webseiten genutzt wird. Die Seiten werden anschließend bei der Anfrage über HTTP dem Client zur Verfügung gestellt. Ein einfaches HTML-Dokument wird auf Listing 3.1 dargestellt.

Listing 3.1: Quelltext einer HTML-Seite

```
1 <html>
2    <head>
3       <title>Titel der Seite</title>
4    </head>
5    <body>
6       <h1>&Uuml;berschrift1</h1>
```

[11] http://java.sun.com
[12] http://www.macromedia.com/software/shockwaveplayer/
[13] http://www.real.com/player/index.html
[14] http://www.realnetworks.com/

```
7         <h2>&Uuml;berschrift2</h2>
8         <p>Inhalt der HTML-Seite</p>
9     </body>
10 </html>
```

Auf der HTML-Seite sind sowohl die Daten als auch die Formatierung der Daten enthalten. HTML ist eine Anwendung von SGML (Standard Generalized Markup Language). SGML ist als eine Definitionssprache eigentlich für Nutzdaten gedacht, welche mit Hilfe der Sprache DSSSL (Document Style Semantics and Specification Language) formatiert werden. Nun hat sich HTML nicht mehr an diese Regel gehalten und orientierte sich mehr an dem Layout der Daten als an deren Strukturierung. Um die Daten in HTML von deren Formatierung zu trennen, wurde die Sprache CSS (Cascading Style Sheet) entwickelt.[15]

Eine weitere serverseitige Technologie sind *Cookies*. Sie stellen den Servern die Möglichkeit zu Verfügung, clientspezifische Daten abzuspeichern und wieder aufzurufen. Cookies sind kleine Textdateien, die auf dem Client abgelegt werden und nur vom Server, welcher diese erstellt und abgelegt hat, abgerufen werden dürfen. Cookies lösen das Problem der Zustandslosigkeit[16] bei der Verbindung mit Hilfe des HTTP-Protokolls. Damit kann die Sitzung des Clients weiterverarbeitet werden, ohne eine neue Verbindung aufbauen zu müssen. (Kappel u. a., 2003a, S. 110) Cookies können allerdings vom Client gelöscht oder durch spezielle Browservoreinstellungen nicht zugelassen werden, was ein Problem dieses Ansatzes darstellen könnte.

Für die eigentliche Business Logik der Webanwendung können unterschiedliche Technologien eingesetzt werden. Eine der bekanntesten sind *CGI-Skripte*. Das sind Programme, die auf dem Server abgelegt und dort beim Aufruf vom Client ausgeführt werden können. Als solche Skriptsprachen können Perl, PHP[17] und C eingesetzt werden.

Eine weitere Möglichkeit zur Realisierung der Business Logik einer Anwendung sind *SSIs* (Server Side Includes). Sie stellen Quelltext dar, welcher innerhalb der HTML-Seite eingebettet wird. Er sorgt, meistens mit Hilfe von

[15] Vgl. Abschnitt 2.3.2 auf Seite 42.

[16] Das Problem der Zustandslosigkeit besteht darin, dass jeder Aufruf für sich allein besteht und keinen Bezug auf bereits ausgeführte oder noch auszuführende Befehle hat. (Meyer, 2003)

[17] PHP (usprünglich *Personal Home Page* Tools, jetzt rekursives Akronym für *Hypertext Preprozessor*) ist Open Source und dient zur Erstellung dynamischer Webseiten.

SQL-Abfragen auf Datenbanken, für die Erstellung des dynamischen Inhalts der Seite und wird auf der Serverseite ausgeführt. SSIs sind leichter als CGI-Skripte zu programmieren, haben allerdings den Nachteil, dass Verbindungen mit der Datenbank nur während der Ausführung der Datei bestehen. (Kappel u. a., 2003a, S. 112 ff.)

Kappel u. a. (2003a, S. 114 ff.) beschreiben auch weitere Formen von SSIs wie z.B. ASP (Active Server Pages) und JSP (Java Server Pages). ASPs stellen Visual Basic- oder Java Skript-Quelltexte dar, welche innerhalb von HTML-Seiten eingebettet werden. Sitzungsinformationen werden über Cookies oder Parameter in der URL ausgetauscht. ASP ist eine Technologie, die eng an Microsoft-Produkte gebunden und damit plattformabhängig ist, was einen wesentlichen Nachteil für den Einsatz von ASPs darstellt.

Herstellerunabhängig ist dagegen die Technologie der Java Server Pages. Sie stellen die Möglichkeit dar, dynamische Inhalte innerhalb des statischen HTML-Codes einzubetten. Der Anfang und das Ende des JSP-Codes innerhalb einer Seite werden mit Hilfe des JSP-Tags[18] <% und %> bekanntgegeben. Auf Listing 3.2 ist ein einfaches Beispiel für eine JSP-Seite dargestellt.

Listing 3.2: Quelltext einer JSP-Seite

```
1 <html>
2    <head>
3       <title>Titel der JSP-Seite</title>
4    </head>
5    <body>
6       <h1>Eine einfache JSP-Seite</h1>
7       <% out.println(request.getParameter("productName")); %>
8    </body>
9 </html>
```

Für die Verarbeitung von den Inhalten einer JSP-Seite stehen serverseitig *Java Beans*[19] zur Verfügung, welche die Verarbeitung des dynamischen Inhalts vollständig unterstützen. Kappel u. a. (2003a, S. 117) sehen den wesentlichen Vorteil von JSPs bei der Trennung der Logik der Seiten von deren Präsentation. Als Nachteil bezeichnen sie die Notwendigkeit zur Installation einer JSP-

[18] JSP-Tags sind Elemente, die JSP- bzw. Java-spezifischen Code in eine HTML-Seite einbetten.

[19] Java Beans sind Java-Klassen, die wiederverwendbaren Code für JSPs zur Verfügung stellen und die Vermeidung von langem Quelltext in die JSP-Seiten ermöglicht.

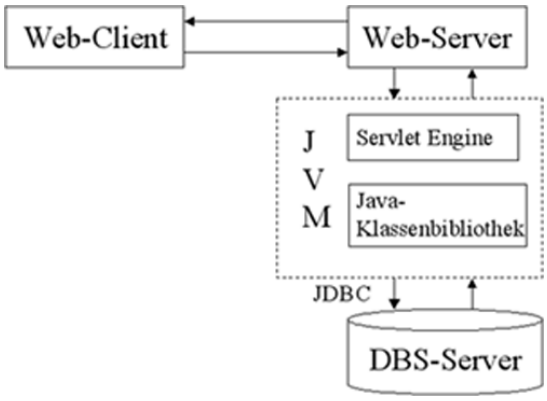

Abbildung 3.2: Web-DB-Anbindung über Java Servlets,
Quelle: Kappel u. a. (2003a, S. 115)

Engine[20]. Als weiterer Vorteil von JSPs kann auch ihre Portabilität bezeichnet werden. Der dynamische Inhalt wird in der plattformunabhängigen Programmiersprache Java erstellt, was die Ausführung auf jeder Plattform ermöglicht.

Es besteht ein enger Zusammenhang zwischen JSPs und *Java Servlets*. JSPs sind Java Servlets, weil der JSP-Code durch einen JSP-Compiler[21] in Servlet-spezifischen Java-Code übersetzt wird und weiterhin von einer Servlet Engine[22] als normales Servlet verarbeitet wird. Ein Servlet ist ein Java-Programm, das dazu dient, Anforderungen des Web-Servers in Form von dynamisch generierten HTML-Seiten zu beantworten. Voraussetzung für den Einsatz von Java Servlets ist die Installation von einer JVM und einer Servlet Engine auf dem jeweiligen Server (siehe Abbildung 3.2.). Dies ist der Nachteil, der auch von Kappel u. a. (2003a, S. 115) erkannt wird. Sie nennen allerdings die berechtigten Vorteile von Java Servlets: Die Plattform- und Herstellerunabhängigkeit und die Möglichkeit zum Einsatz von objektorientierten Konzepten.

3.1.3 Anforderungen

Die nachfolgend aufgelisteten Ziele werden allgemein bei der Entwicklung von Webanwendungen verfolgt (Joung 2004, S. 14 f.; Kappel u. a. 2003b, S. 104 ff.):

[20] Beispiel für eine solche Engine ist der Tomcat Server von Apache.
[21] Compiler: Ein Programm, das den Quelltext einer Programmirsprache in einem für die Sprache spezifischen Code (bzw. ausführbares Programm) übersetzt.
[22] Siehe Fußnote 20.

- Zuverlässigkeit/Konsistenz und Datenintegrität (*Reliability*)
- Vorhandensein (*Availability*)
- Skalierbarkeit (*Scalability*)
- Sicherheit (*Security*)
- Geschwindigkeit (*Performance*)
- Interoperabilität/Interaktivität (*Interoperability*)

Zuverlässigkeit

Die Zuverlässigkeit impliziert die Fähigkeit des Systems, unter verschiedenen Umständen seine Aufgaben korrekt auszuführen. Eine Webanwendung sollte beispielsweise von Clients mit verschiedenen Browsern aufrufbar sein, um ihre Funktionalität richtig durchführen zu können. Die Anwendung sollte keine fehlerhaften Hyperlinks enthalten – weder interne noch externe.

Vorhandensein

Ein möglicher Ausfall des Systems soll im Voraus verhindert werden. So muss die Funktionalität des Servers ununterbrochen geprüft und die fehlerhafte Ausführung von Aufgaben sofort gemeldet werden. Ein unerwarteter Stromausfall kann durch den Einsatz eines Notstromaggregats verhindert werden. Das System muss des Weiteren regelmäßig gewartet werden, um seine gewünschte Funktionalität sicherzustellen.

Skalierbarkeit

Die Anforderung Skalierbarkeit erfordert die Flexibilität des Systems bezüglich mögliche Änderungen. Das System sollte an eine steigende Anzahl von Zugriffen und eine anwachsende Datenmenge und -größe anpassbar sein. So kann die erforderliche hohe Leistung des Systems durch eine Verteilung der Aufgaben auf mehrere Rechner erzielt werden.

Sicherheit

In Webanwendungen der heutigen Zeit spielen die Sicherheitsaspekte eine bedeutende Rolle. Festzulegen ist bei der Entwicklung einer Webanwendung das Sicherheitsniveau, welches eine Anwendung braucht. Öffentliche Daten sind in

der Regel kein Grund zur Sorge. Im e-Commerce-Bereich werden allerdings vertrauliche Daten, wie beispielsweise Kreditkartennummer, Kontonummer und andere, durch das Netz abgesendet und sollten nicht in die Hände eines Dritten mit böswilligen Absichten gelangen. Diese Aufgabe wird durch den Einsatz der Kryptographie[23] ermöglicht.

Seemann (2001, S. 87) unterscheidet folgende vier Nutzungsbereiche der Kryptographie:

- „Geheimhaltung: Wie wird der Zugriff auf vertrauliche Daten verhindert?
- Authentifikation: Wie wird festgestellt, wer der Kommunikationspartner ist?
- Nichtanerkennung: Wie kann sichergestellt werden, dass der Inhalt eines Dokuments später nicht abgestritten werden kann?
- Integrität: Der Absender muss sicher sein, dass seine Daten nicht verändert werden können.“

Für die Sicherheit der im Internet laufenden Daten und die Erfüllung der oben genannten Anforderungen sorgen Mechanismen wie die Verschlüsselung, das digitale Signieren von Daten und Protokollen wie SSL[24].

Die Verschlüsselung von Daten stellt sicher, dass die von zwei Parteien ausgetauschten Daten nicht von Dritten gelesen werden können und damit die Vertraulichkeit der Daten nicht verletzt wird. Bei der Verschlüsselung werden grundsätzlich die symmetrische und asymmetrische Verschlüsselung unterschieden. Das *symmetrische Verfahren* nutzt einen einzigen Schlüssel. Er sollte über einen sicheren Kanal zwischen den beiden Parteien ausgetauscht werden und dient sowohl zum Verschlüsseln als auch zum Entschlüsseln der geheimen Nachrichten. Das Problem bei den symmetrischen Verfahren ist der Austausch des geheimen Schlüssels. Über große Entfernungen zwischen den Partnern ist es schwer, den Austausch über einen sicheren Kanal durchzuführen. Die Lösung für dieses Problem stellt die asymmetrische Verschlüsselung zur Verfügung. (Schmeh 2001, S. 49 ff.; Zwißler 2002, S. 104 ff.)

[23] „Kryptographie ... ist die (mathematische) Wissenschaft, die sich mit Methoden der Ver- und Entschüsselung von Daten - ursprünglich nur zum Zweck der Geheimhaltung, inzwischen aber weit vielfältiger eingesetzt - beschäftigt.“ (SECUDE GmbH, 1999, S. 5)

[24] Siehe Seite 59.

Die *asymmetrische Verschlüsselung* benutzt zwei Schlüssel. Beide Schlüssel sind mathematisch voneinander abhängig, aber keiner der Schlüssel ist vom anderen ableitbar. Der Besitzer des privaten Schlüssels behält diesen und stellt seinen öffentlichen Schlüssel, z.B. in einem öffentlichen Verzeichnis, zur Verfügung. Beim Absenden einer signierten Nachricht verschlüsselt er zuerst die Daten mit seinem privaten Schlüssel. Der Empfänger kann die Signatur nach dem Erhalt der Nachricht mit dem bereits bekanntgegebenen öffentlichen Schlüssel verifizieren. Die beiden Schlüssel funktionieren auch in die umgekehrte Richtung. Eine vertrauliche Nachricht kann mit dem öffentlichen Schlüssel des Empfängers verschlüsselt und gesendet werden. Nach dem Erhalt der Nachricht kann diese nur mit dem privaten Schlüssel entschlüsselt werden. (Schmeh 2001, S. 93 ff.; Zwißler 2002, S. 104 ff.)

Ein Großteil von Daten, die für die Öffentlichkeit vorgesehen sind, müssen nicht verschlüsselt werden. Andererseits ist es beim Austausch von Daten oft notwendig, die Integrität und Authentizität derselben sicherzustellen. Die *Integrität* der Daten beweist, dass die übertragene Nachricht nicht auf dem Weg zum Empfänger geändert wurde, die *Authentizität* dagegen beweist, dass die Nachricht von der Person stammt, die tatsächlich auch als Absender angegeben ist. Wenn die Signatur ungültig ist, ist entweder die Authentizität oder die Integrität der Daten verletzt.

Geschwindigkeit

Das Durchführen der Aufgaben soll möglichst schnelle Ergebnisse liefern. Das hängt von der Umgebung ab, in der die Webanwendung läuft. Eine Anwendung, bei der sich die Business Logik-Schicht und die Datenbank-Schicht auf einem Rechner befinden, weist bei einer großen Anzahl von Zugriffen eine schlechtere Leistungsfähigkeit auf. Je größer die Anzahl der Transaktionen einer Anwendung, desto schlechter wird die Leistung, was zu einem Datenbankabsturz führen kann. In solchen Fällen werden die Schichten einer Webanwendung auf mehreren leistungsfähigen Rechnern verteilt.

Interoperabilität/Interaktivität

Interoperabilität ist die Fähigkeit des Systems mit anderen Systemen zusammen zu arbeiten. Besonders relevant in diesem Bereich sind die bereits existierenden Web Services-Standards. Mit deren Hilfe soll die plattformunabhängige Anbindung an verschiedene Systeme ermöglicht werden. Im Kapitel 5 wird auf dieses Thema näher eingegangen.

3.2 Anforderungen an die Anwendung

Die zu entwickelnde Anwendung soll den grundsätzlichen Ablauf von holländischen Auktionen ermöglichen. Dabei sollen Funktionalitäten wie die Verwaltung von Benutzerdaten und Auktionsdaten sowie Gebotsabgaben für Auktionsartikel zur Verfügung gestellt werden.

Die Präsentationsebene der Anwendung hat die Aufgabe, die Interaktion zwischen dem Benutzer und der Anwendung zu realisieren. Diese Ebene stellt Schnittstellen zur Verfügung, welche das Durchführen der unten aufgezählten Aufgaben ermöglichen.

- *Neuanmeldung/Registrierung eines Benutzers an dem System*: Für die aktive Teilnahme an den laufenden Auktionen muss eine einmalige Anmeldung des Benutzers vorgenommen werden. Die Anmeldung wird durch das Ausfüllen und Absenden eines Formulars ausgeführt. Jeder Benutzer erhält bei der Anmeldung einen Benutzernamen und ein Passwort, welche ihm die Authentifizierung und damit den Zugang zum System ermöglichen.

- *Anmelden und Abmelden eines Benutzers*: Bei jeder Anmeldung am System wird innerhalb der Anwendung eine Sitzung (engl. *session*) für den jeweiligen Benutzer gestartet. Sie hält benutzerspezifische Daten so lange, bis der Benutzer sich vom System abmeldet. Nach der Abmeldung wird die Session für diesen Benutzer annulliert.

- *Anmelden des Verkaufs eines Produktes*: Jeder Benutzer, der einen Benutzernamen besitzt, kann Gegenstände zum Versteigern nach dem holländischen Auktionsprinzip anmelden. Die Anmeldung einer neuen Auktion wird nach einer erfolgreichen Authentifizierung und mit dem Ausfüllen eines Formulars mit den notwendigen Daten zum Starten der Auktion ermöglicht.

- *Ersteigern/Kauf eines Artikels*: Das Ersteigern eines Artikels im Auktionssystem wird nur Anwendern ermöglicht, die sich am Auktionssystem angemeldet haben. Das Bieten im System wird nach den Regeln der holländischen Auktion abgewickelt.

- *Katalog des Auktionssystems*: Die Artikel des Systems werden in Form eines Katalogs geführt. Der Zugang zu diesem erfordert keine Authentifizierung der Benutzer. Um die Übersichtlichkeit der Artikel zu erhalten, wird der Katalog in Kategorien unterteilt. Insgesamt werden drei

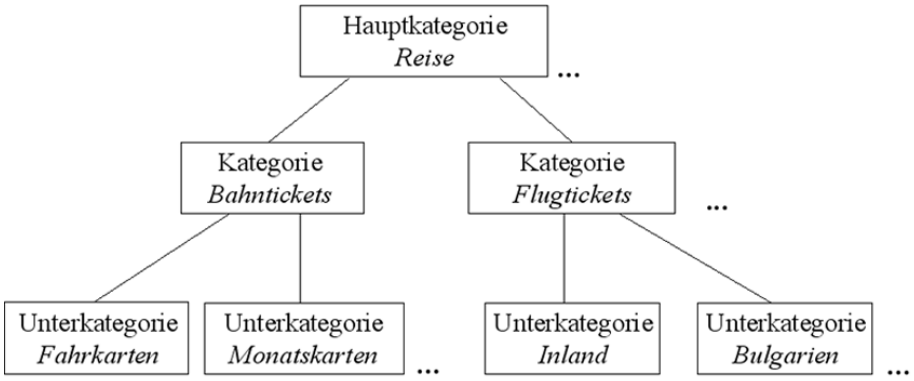

Abbildung 3.3: Katalogunterteilung im Auktionssystem

Arten von Kategorien zur Verfügung gestellt: *Hauptkategorie*, *Kategorie* und *Unterkategorie*. Jeder der zu ersteigernden Artikel ist jeweils einer Unterkategorie zugeordnet. Jede Unterkategorie ist einer Kategorie zugeordnet und jede Kategorie wiederum einer Hauptkategorie zugeordnet. Die Hauptkategorie *Reise* in der Anwendung enthält beispielsweise die Kategorie *Bahntickets*, welche wiederum die Unterkategorien *Fahrkarten* und *Monatskarten* besitzt. Der Unterkategorien *Fahrkarten* und *Monatskarten* können die Auktionsartikel zugeordnet werden. Die Abbildung 3.3 zeigt diese Unterteilung grafisch.

Das ermöglicht eine relativ breite Unterteilung der Artikel, was das Suchen für die Auktionsteilnehmer vereinfacht.

3.3 Struktur und Aufbau der Anwendung

Die Datenhaltung, die sowohl Informationen über die Artikel und ihre Auktionen als auch Systemdaten enthält, wird mit Hilfe eines Datenbanksystems zur Verfügung gestellt. Die Präsentation der Daten wird mit JSP-Seiten realisiert, da sie eine Trennung der Präsentation der Daten von deren Inhalt und weiterhin die Wiederverwendbarkeit vom Quelltext, der in Java Beans ausgelagert ist, ermöglichen. Auf den Aufbau der JSP-Seiten wird im Abschnitt 3.3.3 genauer eingegangen.

Im folgenden Abschnitt werden zunächst grundlegende Prozesse bei der Entwicklung einer Webanwendung angesprochen.

Des Weiteren wird die Entwicklung der Datenbasis für die Haltung der Daten beschrieben. Dabei wird auf das Entity-Relationship-Modell[25] und seine Überführung in das relationale Datenbank-Modell[26] anhand der Auktionssystem-Daten eingegangen. Weitere eingesetzte Mittel für die Flexibilität der Datenbasis wie Domänen[27], Generatoren[28] und Prozeduren[29] werden jeweils aufgelistet und anhand von Beispielen erläutert.

Anschließend wird die Business Logik der Anwendung genau dargestellt. An dieser Stelle wird auf die Geschäftsprozesse der Anwendung eingegangen und eine mögliche Implementierung anhand von Ablaufskizzen und Quelltext-Ausschnitten erklärt.

Der Schwerpunkt bei der Entwicklung von der Anwendung liegt in der Nutzung der XML-Technologie mit der Programmiersprache Java. Das gewährleistet die Plattformunabhängigkeit des Systems sowie Trennung der Daten von deren Präsentation. Einen Großteil der Daten zwischen der Datenbank und der Business Logik-Schicht wird in XML übertragen. Die Anwendung stellt dafür Klassen zur Verfügung, welche im Zusammenwirken mit einer relationalen Datenbank das Erstellen beliebiger XML-Dokumente ermöglicht. Die Flexibilität dieser Logik ermöglicht die Anwendung dieses Einsatzes auch auf anderen Systemen. Auf die konkrete Vorgehensweise und Implementierungsansätze wird im Kapitel 4.2.2 näher eingegangen.

3.3.1 Prozess der Anwendungsentwicklung

Die Realisierung der Geschäftsprozesse in der Form von einer Webanwendung ist in der Regel ein Prozess, der mit wesentlichem Ressourcenverbrauch verbunden ist. Eine solche Anwendung muss viele Anforderungen realisieren: Verfügbarkeit für mehrere Benutzer, Sicherheitsanforderungen, hohe Leistungsfähigkeit. Die einzelnen Abläufe müssen gut überlegt sein und korrekt vom System durchgeführt werden können. Solche Aufgaben sind zeitintensiv. Alle Prozesse müssen mehrmals getestet werden, hinzu kommen noch weitere

[25] Das ER-Modell wurde erstmal von Peter Chen im Jahr 1976 vorgestellt und ermöglicht die Abbildung der realen Welt mit Objekten und die bestehenden zwischen diesen Beziehungen. Siehe dazu Seite 75.
[26] Siehe Seite 79.
[27] Siehe Seite 83.
[28] Siehe Seite 87.
[29] Siehe Seite 88.

gewünschte Anforderungen, welche auch nach dem Fertigstellen des Systems zusätzlich realisiert werden müssen. Um einen komplett neuen Entwurf der Anwendung zu vermeiden, sollte die Entwicklung gleich von Anfang an mittels eines Projektplans[30] genau konzipiert sein. Darin sollen die genauen Anforderungen an das System, die einzelnen Arbeitsschritte bei der Realisierung, die verfügbaren Ressourcen und deren Zuordnung zu den Aufgaben festgelegt sein. Damit ist die Anwendungsentwicklung ein Prozess, der aus mehreren Phasen besteht. Matthiessen und Unterstein (2003, S. 88 ff.) stellen ein Phasenmodell vor, welches die einzelnen Ereignisse der Systementwicklung berücksichtigt. Dieses Modell ist auf Abbildung 3.4 auf Seite 72 dargestellt und unterscheidet folgende Ereignisse bei der Entwicklung einer Anwendung:

- *Systemanalyse*: Es ist undenkbar, mit der Implementierung eines Systems zu beginnen, bevor die klare Definition der Anforderungen an die Anwendung nicht festgelegt worden sind. Die Phase der Systemanalyse zielt auf die genaue Definition der Systemanforderungen. Wichtig dabei ist die Interaktion zwischen Auftraggebern, Projekt- oder Produktmanagern, den Entwicklern und den zukünftigen Nutzern des Systems.

- *Datenbankentwurf*: Ein stabiles und die Anforderungen erfüllendes System benötigt eine Datenbasis. Die Datenbasis, in der Regel in Form eines Datenbanksystems, enthält die Daten, mit welchen die Anwendung operieren wird und stellt damit die Grundlage für ihre korrekte Funktionalität dar. Dies erfordert eine genaue Konzeption des Datenmodells. Für den Datenbankentwurf des holländischen Auktionssystems wurde das Entity-Relationship-Modell[31] eingesetzt.

- *Datenbankrealisierung*: Die Phase der Datenbankrealisierung beinhaltet die Umsetzung der Datenbankkonzeption in einem funktionierenden Datenbanksystem. Dazu dient die Sprache DDL (Data Definition Language), mit welcher das System von Domänen, Tabellen, Prozeduren, Sichten der Datenbank definiert wird. (Ebner, 1999, S. 40) Damit steht das Grundgerüst für die Datenbasis der Anwendung fest. Dieses soll mit den Daten gefüllt und weitergepflegt werden. Für das Nutzen und das Weiterpflegen der Daten steht die Sprache DML (Data Manipulation Language) zur Verfügung. (Ebner, 1999, S. 40) Weiterhin können mit der Sprache DCL (Data Control Language) beispielsweise Benutzer der Datenbank und deren Zugriffsrechte definiert oder entfernt werden. (computerlexi-

[30] Für Weiteres dazu vgl. Buhl (2004).
[31] Siehe dazu Seite 75.

con.com, 2004) Auf Datenbankentwurf und Realisierung der Anwendung wird im Kapitel 3.3.2 näher angegangen.

- *Anwendungsentwurf*: Das Programmieren der Webanwendung erfordert eine genaue Definition der einzelnen Prozesse im System, was das Ziel des Anwendungsentwurfs ist. Auf der Basis der bereits durchgeführten Systemanalyse wird der Ablauf der einzelnen Prozesse in der Anwendung und ihre gesamte Struktur bestimmt.

- *Anwendungsimplementierung*: Nachdem der Anwendungsentwurf und das zugrunde liegende Datenbanksystem feststehen, muss die Implementierung vorgenommen werden. In der Webanwendung wird die Sprache HTML eingesetzt, um den clientseitigen Inhalt der Anwendung zu erstellen. Weiterhin sind die serverseitigen Programme zu realisieren, welche als Schnittstelle zwischen dem Client und der Datenbank dienen. Das Kapitel 3.3.3 ist dem Entwurf und der Implementierung der Anwendung gewidmet.

- *Test und Korrektur*: Die bereits entworfene Anwendung muss, bevor sie zur Nutzung freigegeben wird, durch die Testphase kommen. Dies ist ein sehr wichtiger Bestandteil der Anwendungsentwicklung. Hier werden alle Prozesse der Anwendung auf ihren korrekten Ablauf geprüft. Mögliche Fehler werden beseitigt bevor die Webanwendung dem Nutzern zur Verfügung gestellt wird.

- *Nutzung*: In dieser letzten Phase wird das fertiggestellte System eingesetzt. Mögliche kleine Änderungen oder gewünschte Erweiterungen werden veranlasst.

Wie im Kapitel 3.1.1 bereits beschrieben, basieren Webanwendungen in der Regel auf drei Ebenen. Sie sind auch in der Anwendung des Auktionssystems erkennbar. Die *Datenebene* der Auktionssystem-Anwendung ist eine SQL-fähige Datenbank. Diese ermöglicht die redundanzfreie Haltung der Daten sowie das bequeme Abfragen der Daten und die Möglichkeit mehrere Benutzer mit spezifischen Rechten für den Zugriff auf die Daten einzurichten. Die *Datenverarbeitungsebene* wurde mit der Programmiersprache Java realisiert und stellt die Verbindung zwischen der Datenebene und die Präsentationsebene dar. Die Plattformunabhängigkeit der Sprache ermöglicht den universalen Einsatz der Anwendung auf unterschiedlichen Plattformsysteme. Die Objektorientiertheit der Sprache gewährleistet weiterhin die Wiederverwendung von Quelltext. Die *Präsentationsebene* ist für die Darstellung der Daten zuständig und ermöglicht

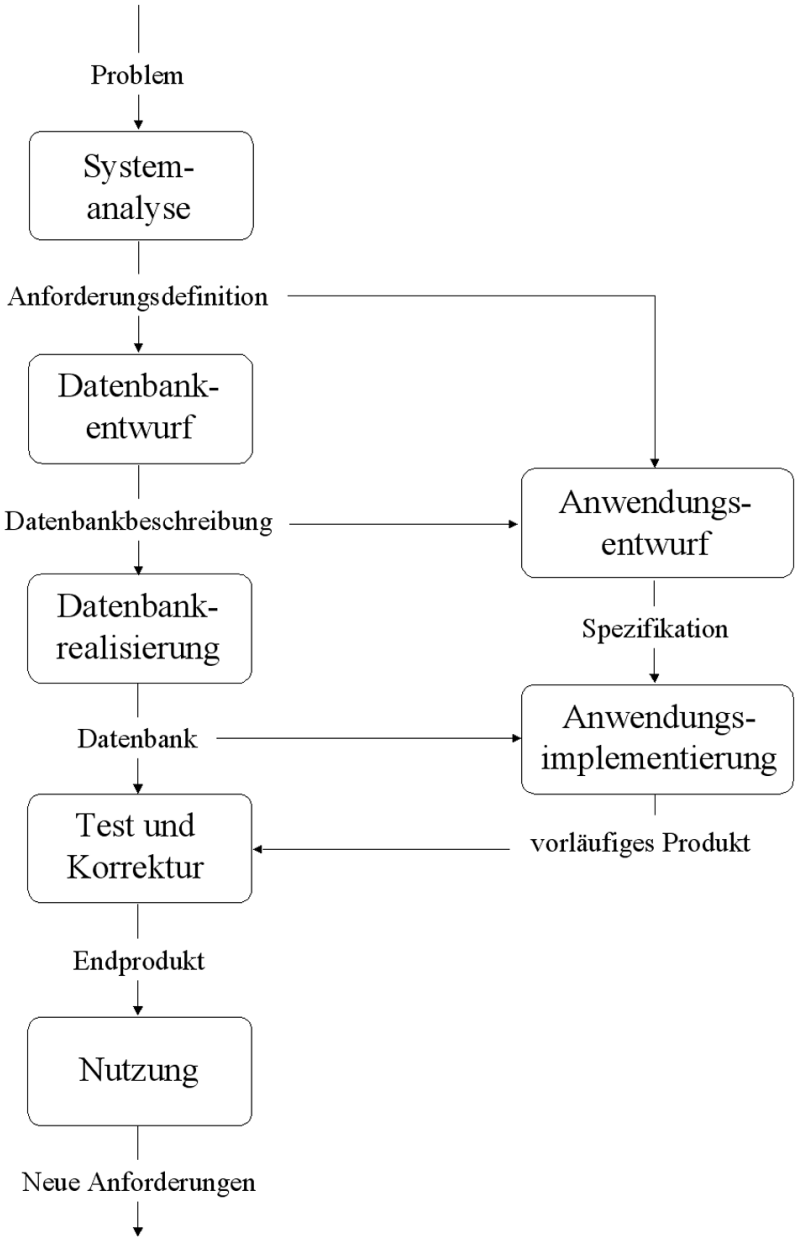

Abbildung 3.4: Phasenmodell der Anwendungsentwicklung, Quelle: Matthiessen und Unterstein (2003, S. 89)

die Kommunikation zwischen dem Auktionssystem und dem Anwender. Der Einsatz von XML und JSP trennt die Darstellung der Daten und deren Präsentation, was sowohl die Verarbeitung der Daten von Programmierer und Layout-Designer erlaubt als auch die Darstellung der gleichen Daten in unterschiedlichen Formate und auf unterschiedliche Geräte ermöglicht und ihre Weitergabe an andere Systeme erlaubt. Die Systemarchitektur ist aus der Abbildung 3.5 auf Seite 73 zu entnehmen.

3.3.2 Datenbasis

Um eine möglichst hohe Flexibilität in der Datenbasis-Verwaltung der Anwendung zu ermöglichen, wurde eine Datenbank als Haltungsform für die Daten gewählt, die auf dem Serversystem installiert wird.

> „Eine Datenbank (engl. *data base*) ist ein integrierter, persistenter Datenbestand einschließlich aller relevanten Informationen über die dargestellten Information (sog. Metainformation, d.h. Integritätsbedingungen und Regeln), der einer Gruppe von Benutzern in nur einem Exemplar zur Verfügung steht und durch ein DBMS verwaltetet wird.“ (Jeckle, 2004)

Ein DBMS (Datenbank Management System) ist dabei eine Software zur Manipulation der Datenbank. (Jeckle, 2004)

Die Verwaltung der Daten in einer Datenbank, welche in nur einem Exemplar vorliegt, bringt nach Matthiessen und Unterstein (2003, S. 20) die nachfolgend aufgelisteten Vorteile:

- *Redundanzfreiheit*: Diese Eigenschaft verhindert, dass Datensätze mehrfach in der Datenbank auftreten. Bei möglichen Änderungen der Daten müssen sie nur an einer Stelle aktualisert werden.
- *Konsistenz*: Konsistente Daten schließen widersprüchliche Informationen in der Datenbank aus. Dazu gehört beispielsweise das Löschen der Produktkategorie eines Katalogs, in der sich immer noch vorhandene Produkte befinden.
- *Auswertungen nach neuen Gesichtspunkten möglich*: Die Datenbankdaten in einer Datenbank sind klar strukturiert und räumen die Möglichkeit ein, einfache Auswertungen vorzunehmen. Eine Datenbank ermöglicht die Auswertung der Daten nach vielen Gesichtspunkten. So können sowohl Abfragen von externen Benutzern als auch interne Abfragen für die Erstellung von Reports genutzt werden.

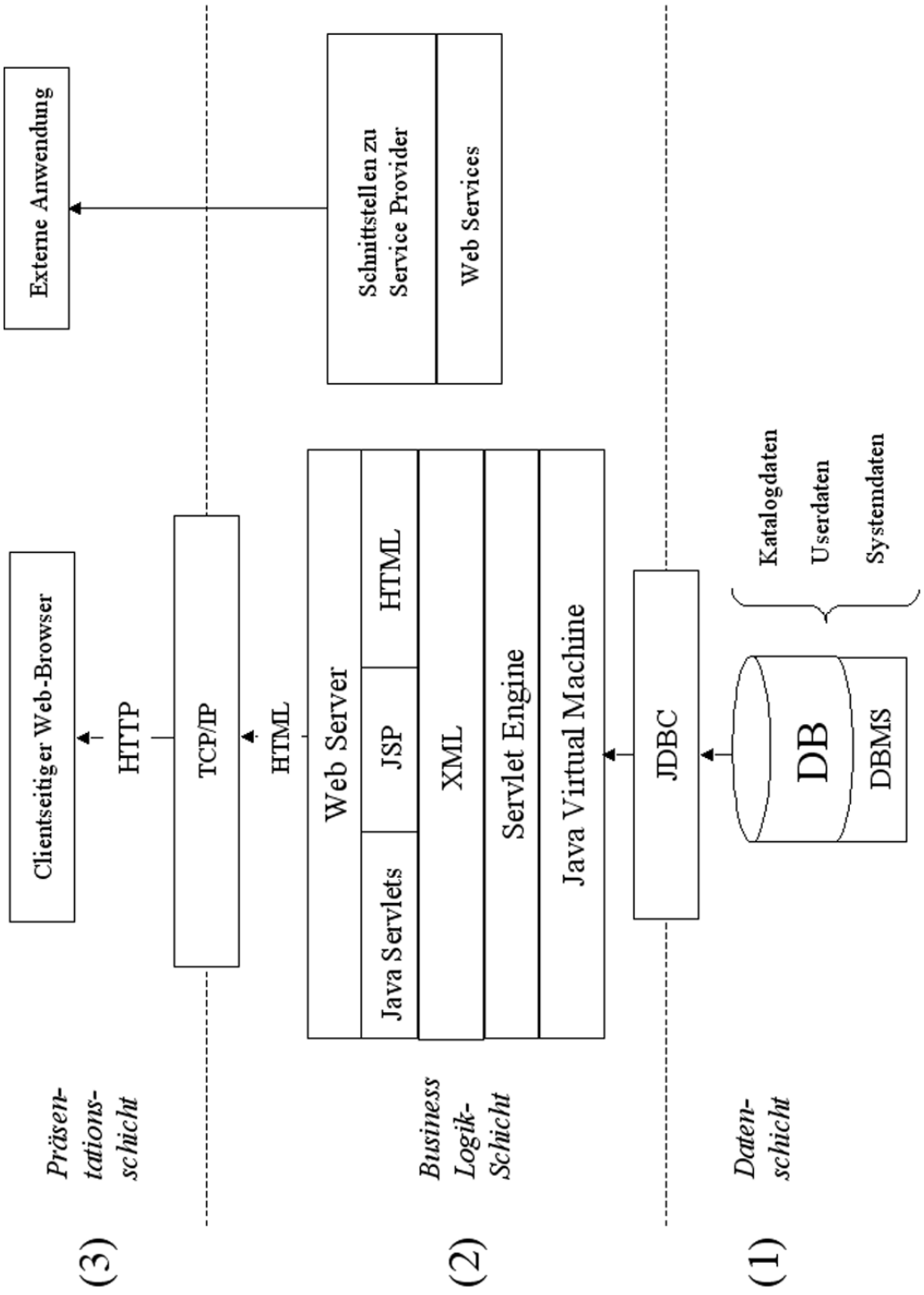

Abbildung 3.5: Schichten der Webanwendung

- *Datenunabhängigkeit*: Hiermit ist die Unabhängigkeit der Daten von den anderen Bestandteilen der Anwendung gewährleistet. So ist es beispielsweise möglich, die Präsentation der eigentlichen Daten im Browser mittels Programmiersprachen so zu realisieren, dass Änderungen der eigentlichen Daten keine weiteren Änderungen in der Business Logik der Anwendung erfordern. Dies rationalisiert notwendige Daten-Updates.

Für die Datenhaltung des Auktionssystems wurde das relationale Datenbanksystem *Borland® Interbase® 6.0*[32] gewählt. Dieses enthält die Daten der Präsentationsschicht wie Katalogdaten und Daten der laufenden oder bereits abgeschlossenen Auktionen. Des Weiteren werden in der Datenbank auch Daten gehalten, die ausschließlich von der Business Logik der Anwendung eingesetzt werden. Das sind Daten, die zum Aufbau der XML- oder HTML-Dateien genutzt werden.[33]

Das Entity-Relationship-Modell

Für den Entwurf der Datenbank wurde das Entity-Relationship-Modell[34] eingesetzt. Mit Hilfe dieses Modells wird die zu beschreibende reale Welt in Objekte (Entitäten) und deren Attribute und Beziehungen (Relationships) zerlegt. Meier (2004, S. 16) definiert eine Entität als

> „...ein bestimmtes, d.h. von anderen wohlunterscheidbares Objekt der realen Welt oder unserer Vorstellung."

Die Attribute jeder Entität stellen seine Eigenschaften dar. Sie sind atomar (nicht zerlegbar) und besitzen einen Wertebereich (Domäne). Zwischen den Entitäten des Modells bestehen Abhängigkeiten, welche die Beziehungen (Relationen) zwischen den Entitäten des Modells darstellen. Mehrere Entitäten und Beziehungen des selben Typs bilden entsprechend eine Entitätsmenge und eine Beziehungmenge. Die Datenbank enthält Informationen über folgende Objekte des Auktionssystems:

- Die Hauptkategorien, die Kategorien und Unterkategorien des Katalogs.
- Die Artikel, die den Unterkategorien zugeordnet sind.
- Die Benutzer des Auktionssystems.

[32] http://www.borland.com/interbase/
[33] Siehe Abschnitt auf Seite 77.
[34] Siehe Fußnote 25.

- Die Auktionen, die innerhalb des Systems laufen oder bereits abgeschlossen sind.
- Weitere Objekte aus den Systemdaten der Anwendung. Solche Objekte sind beispielsweise *Page*, *XML-Dokument* und *XML-Parameter*. Diese Daten werden, im Unterschied zu allen anderen, ausschließlich von der Business Logik-Schicht zum Zusammensetzen der Seiten der Anwendung oder deren dynamischen Inhalt genutzt.

Aus den bereits aufgelisteten Objekten ergeben sich die Entitäten der Datenbank. Die Abhängigkeiten zwischen den Entitäten stellen ihre Beziehungen dar. Nach Meier (2004, S. 18) bestehen Beziehungen aus zwei Assoziationen: „Unter *Assoziation* (engl. *association*) einer Entitätsmenge EM_1 nach einer zweiten Entitätsmenge EM_2 versteht man die Bedeutung der Beziehung in dieser Richtung“. Meier (2004, S. 19) unterscheidet vier Assoziationstypen:

- *Einfache Assoziation* (Typ 1): Jeder Entität aus einer Entitätsmenge ist genau *eine* Entität einer zweiten Entitätsmenge zugeordnet.
- *Konditionelle Assoziation* (Typ c): Jeder Entität aus einer Entitätsmenge ist *keiner oder genau einer* Entität aus einer zweiten Entitätsmenge zugeordnet.
- *Mehrfache Assoziation* (Typ m): Jeder Entität aus einer Entitätsmenge ist *einer oder mehreren* Entitäten aus einer zweiten Entitätsmenge zugeordnet.
- *Mehrfach-konditionelle Assoziation* (Typ mc): Jeder Entität aus einer Entitätsmenge ist *keine, eine oder mehrere* Entitäten aus einer zweiten Entitätsmenge zugeordnet.

Datenbankdaten der Präsentationsschicht und ihre Beziehungen

Alle Artikel, die im System zum Ersteigern angeboten werden, werden in einem Katalog gehalten. Der Katalog des Systems enthält Hauptkategorien, welche sich in weitere Kategorien unterteilen. Daraus ergibt sich eine Tiefe mit drei Ebenen und den Entitäten *Prodtype*[35], *Prodsubtype*[36] und *Prodsubsubtype*[37]. *Product* ist eine weitere Entität, welche einem *Prodsubsubtype* zugeordnet ist und die Namen der zu ersteigernden Artikel enthält.

[35]Enthält die Hauptkategorien des Katalogs.
[36]Enthält die Kategorien des Katalogs.
[37]Enthält die Unterkategorien des Katalogs.

Bevor eine Person einen Artikel ersteigern kann, muss zuerst eine Anmeldung abgeschlossen werden. Dabei werden die persönlichen Daten des Benutzers erfragt und im System zur späteren Verwendung abgelegt. Diese Daten bilden die Entität *Customer*. Benutzer besitzen weiterhin ein Geschlecht (Entität *Gender*) und werden einem Land zugeordnet (Entität *Country*) (siehe dazu Abbildung 3.6).

Ein wichtiger Bestandteil des Systems sind die Auktionsdaten. Wenn ein Benutzer einen Artikel mit Hilfe des Auktionssystems verkaufen möchte, muss dieser dem System folgende Daten zur Verfügung stellen:

- Die Hauptkategorie, die Kategorie und die Unterkategorie, in der der Artikel platziert werden soll,
- der Artikelname,
- die Menge des Artikels,
- die Mengeneinheit,
- der Startpreis (Höchstpreis) des Artikels pro Mengeneinheit,
- der niedrigste Preis,
- der Preisrutsch - dies ist die Preisspanne, um die der Preis sinkt und damit innerhalb einer bestimmten Zeitspanne eine bestimmte Minimalpreishöhe erreicht wird,
- die Auktionsdauer - hier ist eine Eingabe der Genauigkeit in Minuten möglich.

Daraus ergeben sich weitere Entitäten wie *Auktionsstatus, Mengeneinheit* und *Auktion*. Die Entität *Auktionsstatus* enthält die möglichen Zustände, welche eine Auktion annehmen kann. Die Entität *Mengeneinheit* bildet die möglichen Mengen ab, welche bei der Veröffentlichung einer Auktion angegeben werden können. Die Entität *Auktion* enthält die oben aufgezählten Attribute der Benutzer. Des Weiteren enthält sie Attribute wie die Zeit, in der der Preis variiert wird, die Anzahl der Preissenkungen, die Zeit zwischen den Preissenkungen, die verbleibende Zeit für die Auktion, der aktuelle Preis, Anzahl bereits getätigter Gebote, das Startdatum und das Enddatum der Auktion. Diese Attribute werden nach der Veröffentlichung der Auktion berechnet.

In der Abbildung 3.6 wird das ER-Modell der in der Datenbank enthaltenen Daten des Systems grafisch abgebildet.

ER-Modell der Business Logik

Im Datenbanksystem werden weiterhin Daten abgelegt, die ausschließlich von der Business Logik-Schicht der Anwendung in Anspruch genommen werden. Sie werden in Abbildung 3.7 auf Seite 80 grafisch dargestellt.

Eine solche Entität in der Datenbank ist *XmlDoc*. Sie enthält die XML-Dokumente, die innerhalb des Systems erstellt werden können. Jedes XML-Dokument kann ein oder mehrere Parameter erfordern. Diese Parameter sind in der Entität *XmlPrm* definiert. Ein Dokument kann aus einem oder aus mehreren Dokumenten bestehen, welche in der Entität *XmlData* definiert sind. Jedes Dokument bzw. Teildokument von *XmlData* wird mit Hilfe der Entität *XmlSql* erstellt. Unter anderem enthält *XmlSql* auch das Attribut mit dem eigentlichen SQL-Befehl für die Datenbankabfrage. Die Abfrage in der *XmlSql*-Entität kann wiederum eigene Parameter erfordern, welche in der Entität *XmlPrm* definiert sind.

Ein XML-Dokument kann außer untergeordneten Dokumenten auch die Attribute *XmlDocAttr* enthalten. Die Definition der Attribute kann Parameter erfordern, welche erneut eine Definition in der Entität *XmlPrm* finden. Die Entität *XmlData* hat als Ziel, den Abruf der Nutzdaten aus der Datenbank zu definieren. Dabei werden die Tags[38] in der XML-Datei die zugehörigen Namen der Tabellenattribute übernehmen. Für die Flexibilität der Tag-Namen in den XML-Dateien sorgt die Entität *XmlElMap*. Von den Attributnamen abweichende Tag-Namen müssen daher in dieser Entität definiert werden.

Für die bequeme Gestaltung der dynamischen HTML-Seiten der Anwendung wurden die weiteren Relationen *Page* und *PagePrm* erstellt. Jede *Page* kann einen *PagePrm* erfordern, welche in der Relation *XmlPrm* definiert ist. Eine Seite der Anwendung kann mit oder ohne Parameter dynamische Daten in einer XML-Datei *XmlDoc* bekommen. Das ER-Modell der Systemdaten für die Verarbeitung der XML- und HTML-Dokumente ist in der Abbildung 3.7 dargestellt.

[38] Siehe Fußnote 30.

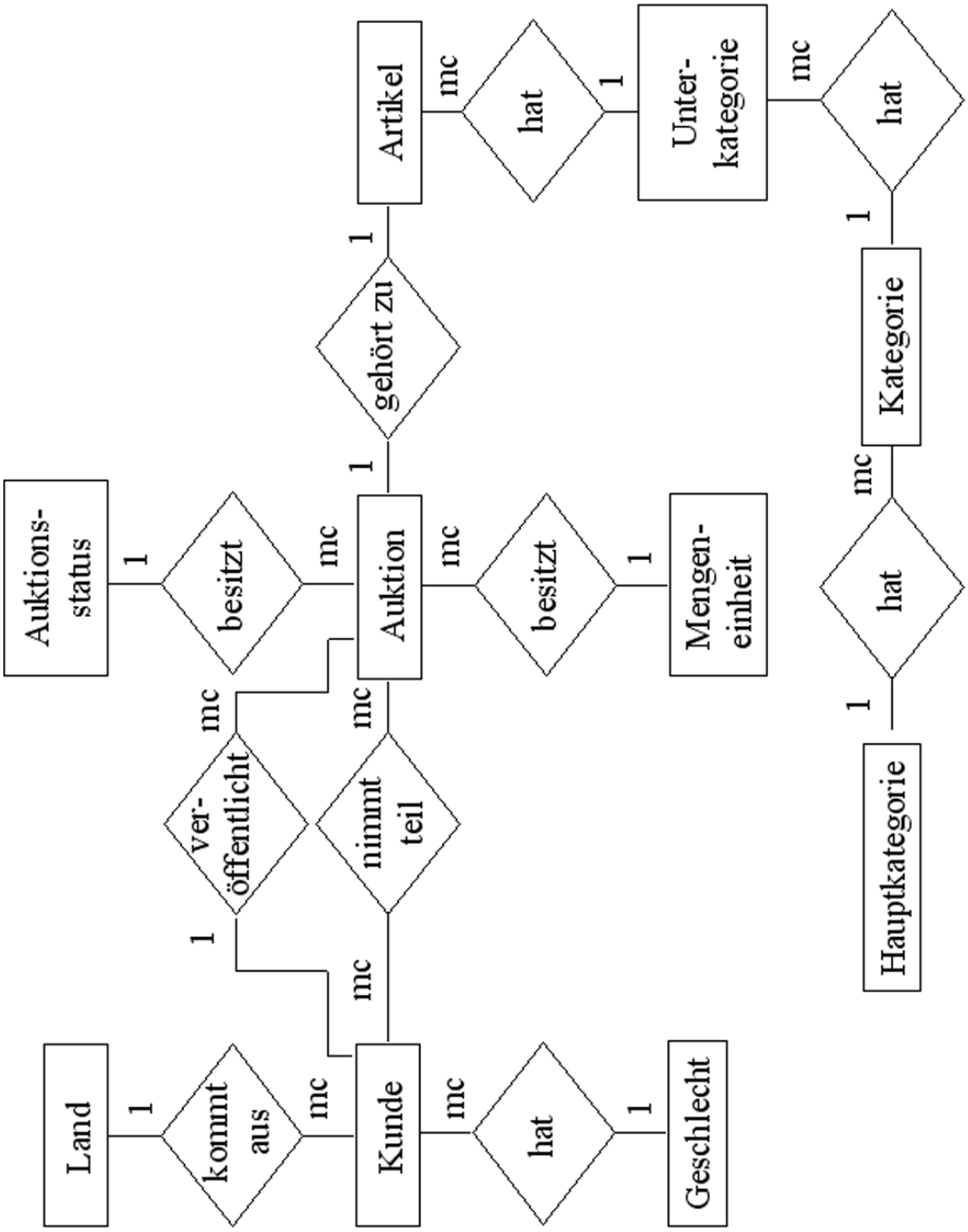

Abbildung 3.6: ER-Modell

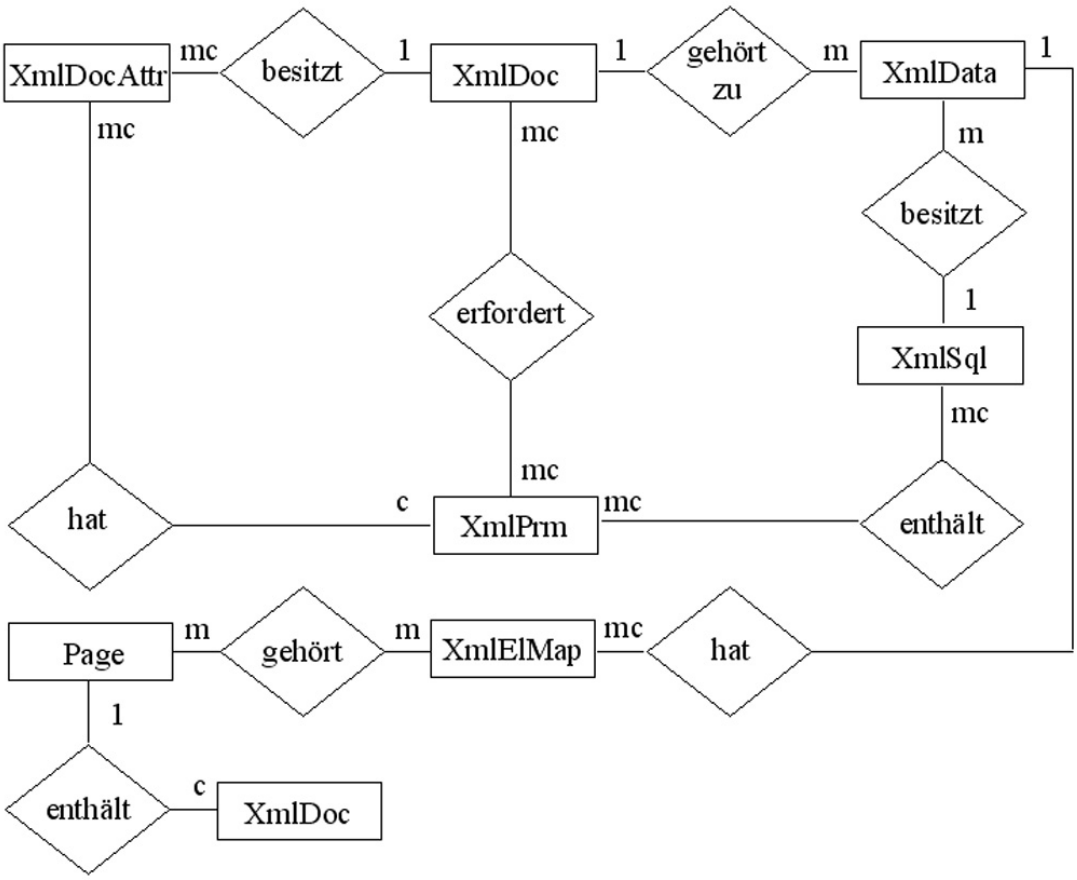

Abbildung 3.7: ER-Modell der Business Logik

Überführung des ER-Modells in das relationale Modell

Das vorgestellte ER-Modell ist die Grundlage für das relationale Datenbankmodell. Wie aus dem Namen bereits ersichtlich, besteht dieses Modell aus Relationen, welche in Form von Tabellen abgebildet werden. Tabellen sind die wichtigsten Bestandteile dieses Modells - eine Tabelle wird von Sauer (2002, S. 33) als „eine logisch zusammenhängende Einheit von Informationen“ definiert. Eine Relation besitzt *Attribute*, die innerhalb der Tabelle durch Spalten abgebildet werden. Jedes Attribut hat seinen eigenen Wertebereich. Die Tabelle enthält die eigentlichen Daten in Form von Zeilen oder Datensätzen, welche im relationalen Modell *Tupeln* genannt werden.

Mit dem *Ausdehnungsgrad* (Degree) der Tabelle wird die Anzahl der Spalten bezeichnet und mit der *Kardinalität* die Anzahl der Zeilen (Tupel). Eine Tabelle muss mindestens eine Spalte beinhalten. Eine Spalte mit einem Attribut wird als *unär* und solche mit zwei Attributen als *binär* bezeichnet. Während der Ausdehnungsgrad einer Tabelle mindestens *eins* sein muss, kann die Kardinalität dieser auch null sein. Die letztere Konstellation entspricht eine Tabelle ohne Daten. (Sauer, 2002, S. 33)

Die Überführung in Tabellen

Das Entity-Relationship-Modell der Daten besitzt einige mehrfache Beziehungen. Bei der Überführung in das relationale Datenbankmodell müssen diese durch eigenständige Tabellen ersetzt werden. So können beispielsweise zu einer *Page*-Entität mehrere Parameter *XmlPrm* gehören. Ein *XmlPrm* kann auch zu mehreren *Page*-Tupeln gehören. Die *Page-XmlPrm*-Zugehörigkeit wird durch die eigenständige Tabelle TCAT_PAGEPARAM ausgedrückt.

Folgende weitere Relationen durch mehrfache Beziehungen entstehen bei der Überführung des ER-Modells in das relationale Modell[39]:

- *Gebote*: Die Tabelle TDAT_BID[40] entsteht zwischen den Relationen *Auktion* und *Kunde* und speichert die Daten der einzelnen Gebote, die jeder Kunde für die zu ersteigernden Artikel abgegeben hat.
- *XmlSqlParameter*: Die Tabelle TDAT_XMLSQLPARAM verbindet die Relationen *XmlSql* und *XmlPrm*. Sie enthält die Zuordnung der Parameter zu jeder SQL-Abfrage, die von ihr in Anspruch genommen werden.

[39] Siehe dazu die Abbildungen 3.8 auf Seite 85 und 3.9 auf Seite 86.
[40] bid: (engl.) Gebot

- *XmlDocParameter*: Die Tabelle TDAT_XMLDOCPARAM befindet sich zwischen den Relationen *XmlDoc* und *XmlPrm*. In ihr wird die Zuordnung der Parameter zu den möglichen erstellbaren XML-Dokumenten innerhalb der Anwendung abgelegt.
- *XmlAttrParameter*: Die Tabelle TDAT_XMLATTRIBUTE_PARAM entsteht zwischen den Relationen *XmlDocAttr* und *XmlParam* und ordnet die Parameter den Attributen zu.
- *PageParameter*: Die Tabelle TCAT_PAGEPARAM verbindet die Relationen *XmlPrm* und *Page* und ordnet die Parameter jeder Seite der Anwendung zu, welche für ihre Erstellung benötigt werden.

Die Tabellen und deren Abhängigkeiten

Die Tabellen im relationalen Datenbankmodell werden mit Schlüsseln miteinander verknüpft. Sie können folgende Arten von Schlüsseln enthalten:

- Primary Key (Primärschlüssel),
- Foreign Key (Fremdschlüssel),
- Candidate Key (eindeutiger Schlüssel) und
- Alternate Key (Zweitschlüssel oder Alternativschlüssel).

Jedes Tupel einer Tabelle muss eindeutig identifizierbar sein, d.h. zwei Tupel können nicht miteinander übereinstimmen. Das wird mit Hilfe eines Primärschüssels erreicht. Der Wert des Primärschlüssels darf nicht *null*[41] sein, um die eindeutige Identifizierung des Tupels zu gewährleisten. Zwei Tabellen in der Datenbank werden mit Hilfe eines Fremdschlüssels verknüpft. Die verweisende Tabelle enthält ein Attribut mit dem Fremdschlüssel. Dieser enthält den Wert des Primärschlüssels der Tabelle, auf die verwiesen wird.

Der Primärschlüssel wird durch eine oder mehrere Spalten bestimmt, die eindeutige Werte für die Tupel enthalten. Dazu müssen zuerst die möglichen *Candidate Keys* gefunden werden.

> „Eine Attributmenge wird Candidate-Key genannt, wenn alle Werte dieser Attributmenge unique (eindeutig) sind.“[42] (Sauer, 2002, S. 34)

[41] In der Welt der relationalen Datenbanken steht der Wert *null* für nicht zutreffende oder unbekannte Werte. (Ebner, 1999, S. 22)

[42] Fettformatierung im Zitat wurde vom Autor entfernt.

Anschließend kann ein weiterer Schlüssel, der Zweitschlüssel (*Alternate Key*), definiert werden, der neben dem Primärschlüssel mit dem Ziel eingebaut wird, die Leistungsfähigkeit zu steigern.

Bei der Realisierung des relationalen Datenbankmodells muss die wichtigste Regel, die *Referenzielle Integrität*, eingehalten werden. Diese Aufgabe übernimmt das DBMS[43] der Datenbank. Sauer (2002, S. 37) definiert die Referenzielle Integrität wie folgt:

> „Eine Relation R2 besitze einen Foreign-Key, der auf einen Primary-Key in einer Relation R1 verweist.“[44]

Das bedeutet, dass der Wert eines Fremdschlüssels in der Tabelle R2 als Primärschlüssel in der Tabelle R1 existieren muss. Wenn das nicht der Fall ist, muss das Einfügen des Datensatzes in die Tabelle R2 verhindert werden.[45]

In der Abbildungen 3.8 auf Seite 85 und 3.9 auf Seite 86 sind die Tabellen des relationalen Datenbankmodells der Anwendung dargestellt.

Domänen der Tabellen

Vor der Erstellung der Datenbank-Tabellen wurden zuerst mehrere Domänen definiert. Domänen definieren Wertebereiche, welche die einzelnen Tabellenspalten annehmen können. (Ebner, 1999, S. 95) Je mehr Tabellen die Datenbank enthält, desto sinnvoller ist es, Domänen zu erstellen, welche die Arbeit mit der Datenbank unterstützen und bei größeren Datenbanken die Arbeit sogar deutlich vereinfachen. Tabelle 3.1 listet alle Domänen der Datenbank mit ihren entsprechenden Wertebereichen auf.

Tabelle 3.1: Domänen der Auktionssystem-Datenbank

Name der Domäne	*Definierter Wertebereich*
D_ADDRESS	VARCHAR(30)
D_BLOB_TEXT	BLOB(80)
D_CITY	VARCHAR(25)
D_COUNTRY	VARCHAR(15)

Fortsetzung auf der nächsten Seite

[43] Siehe Seite 74.
[44] Kursive Schrift wurde vom Autor entfernt.
[45] Weiteres dazu siehe Ebner (1999, S. 117 f.).

Fortsetzung aus der vorherigen Seite

Name der Domäne	*Definierter Wertebereich*
D_DATE	DATE
D_EMAIL	VARCHAR(40)
D_FIRSTNAME	VARCHAR(15)
D_GENDER	VARCHAR(5)
D_INT	INTEGER
D_LASTNAME	VARCHAR(20)
D_LONGNAME	VARCHAR(80)
D_MIDNAME	VARCHAR(30)
D_PASS	VARCHAR(10)
D_PHONENUMBER	VARCHAR(20)
D_PK	INTEGER
D_POCODE	CHAR(8)
D_PRICE	NUMERIC(15,2)
D_PRODUCTNAME	VARCHAR(80)
D_PRODUCTTYPENAME	VARCHAR(25)
D_QUANTITY	NUMERIC(15,2)
D_SHORTNAME	VARCHAR(15)
D_SQL	VARCHAR(100)
D_STATUS	CHAR(6)
D_UNIT	CHAR(5)
D_USERID	VARCHAR(10)

Wie aus der obigen Tabelle ersichtlich, der Wertebereich INTEGER wird den Domänen D_INT und D_PK zugewiesen. Die Domäne D_PK soll zusätzlich keine *null*-Werte enthalten. Die SQL-Anweisung für ihre Definition lautet damit

```
CREATE DOMAIN D_PK AS INTEGER NOT NULL
```

Die Werte der Tabellenspalten, welche Zeichendaten enthalten, werden als CHAR oder VARCHAR definiert. Die Werte der Domänen D_POCODE, D_UNIT und D_STATUS sind als CHAR-Werte definiert - alle anderen Domänen, die die Attribute einer Tabelle festlegen, sind als VARCHAR definiert. Im unterschied zu CHAR wird bei VARCHAR der Speicherplatz der ungenutzten belegten Speicher freigegeben, was bei grossen Wertebereichen, die ungenutzt bleiben, sinnvoll ist.

Des Weiteren werden die Domänen D_QUANTITY und D_PRICE definiert.

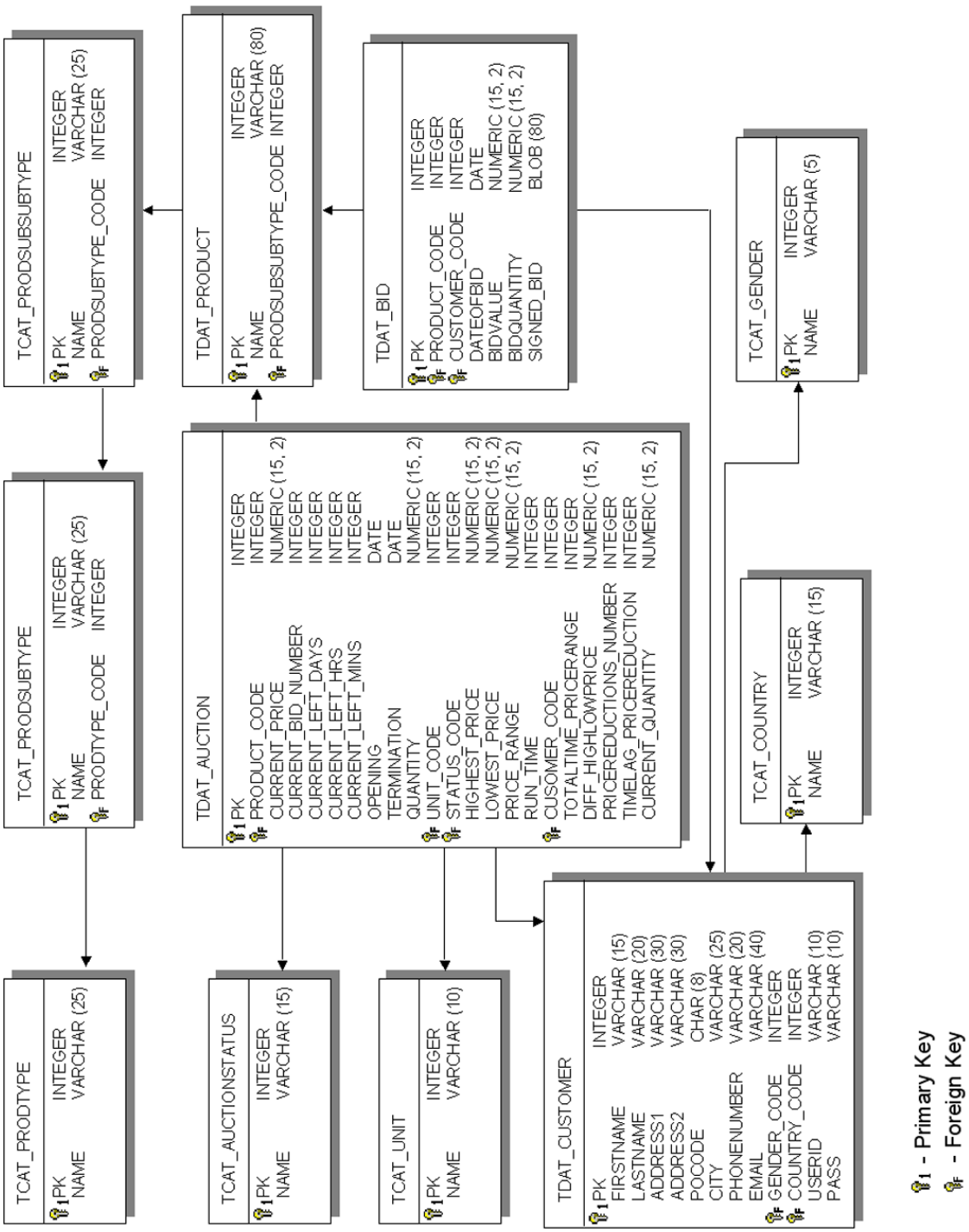

Abbildung 3.8: Auktionssystem: Datenbankmodell

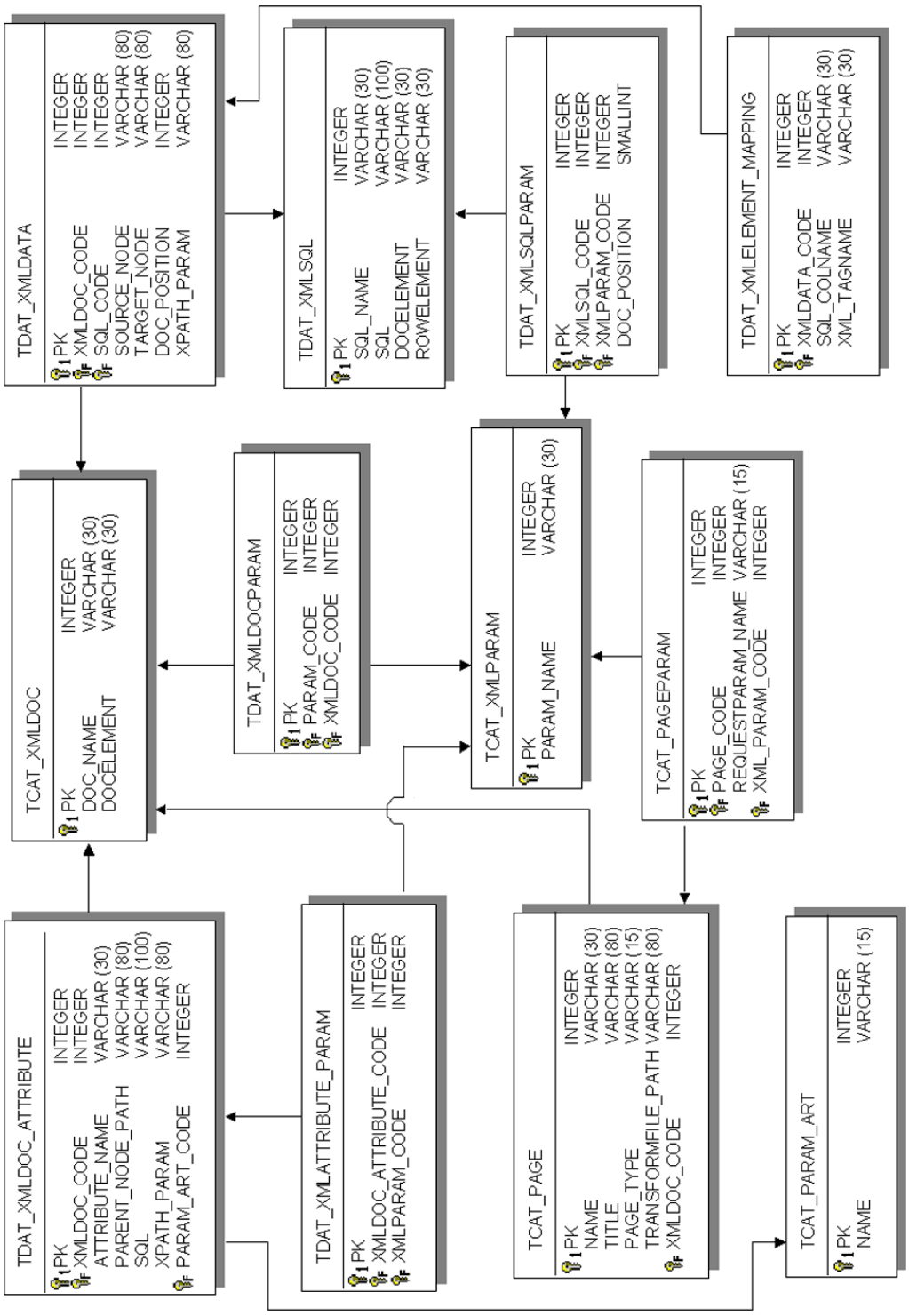

Abbildung 3.9: Auktionssystem: Datenbankmodell der Systemdaten

Sie besitzen den Wertebereich NUMERIC(15,2). Diese Definition des Datentyps erlaubt Zahlenwerte, die bis zu 15 Zahlen vor dem Komma und zwei Zahlen nach dem Komma enthalten können.

Anschließend wird die Domäne D_DATE definiert. Die Domäne D_DATE besitzt den Typ DATE und ist für das Speichern von Datumsdaten vordefiniert. Die Domäne D_BLOB_TEXT ist für Text-BLOB-Daten vordefiniert und besitzt den Datentyp BLOB[46].

Beim Einfügen der Daten aus der Anwendung in das Datenbanksystem müssen die Werte der abgegebenen Daten bereits vor dem eigentlichen INSERT-Befehl überprüft werden. So werden Formularfelder im System auf die Größe der einzugebenden Daten beschränkt. Beim Absenden der Daten vom Anwender an das System können diese weiterhin auf ungültige Zeichen, Größe etc. geprüft werden. Der Anwender muss über die Ungültigkeit der eingegebenen Daten informiert werden und ihm muss die Möglichkeit zur Korrektur der Daten gegeben werden.[47]

Generatoren von den Tabellen und ihre zugehörigen Trigger[48]

Die Erzeugung der Primärschlüsselwerte in den Tabellen wird in der Datenbank durch Trigger und Generatoren unterstützt. Für jede Tabelle wurde ein Generator definiert, der den aktuellen Wert des Primärschlüssels enthält. Die SQL-Anweisung für das Erstellen des Generators GEN_AUCTION der Tabelle TDAT_AUCTION lautet:

```
CREATE GENERATOR GEN_AUCTION;
SET GENERATOR GEN_AUCTION TO 0;
```

Für jede Tabelle wird weiterhin auch ein Trigger erstellt, der vor dem Einfügen eines neuen Tupels in die Tabelle den Wert des Generators um eins erhöht. Der zugehörige Trigger SET_AUCTION_NO für den Generator GEN_AUCTION wird mit der SQL-Anweisung

[46] BLOB (binary large object): Der Datentyp BLOB wird häufig für Binärdaten wie Beispielsweise Bilder oder Audiodaten eingesetzt. In der Datenbank des Auktionssystems wird der BLOB-Typ als Text definiert, was für das Abspeichern von großen XML-Dokumenten eingesetzt werden kann.

[47] Empfehlenswert ist es, einen größeren Wertebereich für die Datenbank-Daten anzugeben, um die Wahrscheinlichkeit für das Auftreten von Fehlern zu minimieren.

[48] Ein Trigger ist ein Programm in der Datenbank, welches das Einfügen, Ändern oder Löschen von Daten unterstützt.

```
CREATE TRIGGER SET_AUCTION_NO FOR TDAT_AUCTION
ACTIVE BEFORE INSERT POSITION 0
AS begin
  new.pk = gen_id(gen_auction, 1);
end
```

erstellt.

Die Liste der Generatoren und ihre dazugehörigen Trigger in der Datenbank sind aus der Tabelle 3.2 zu entnehmen.

Tabelle 3.2: Generatoren und Trigger bzw. Prozeduren, die diese in Anspruch nehmen

Name des Generators	*Name des Triggers / der Prozedur*
GEN_AUCTION	SET_AUCTION_NO
GEN_AUCTIONSTATUS	SET_AUCTIONSTATUS_NO
GEN_BID	SET_BID_NO
GEN_COUNTRY	SET_COUNTRY_NO
GEN_CUSTOMER	SET_CUSTOMER_NO
GEN_GENDER	SET_GENDER_NO
GEN_PAGE	SET_PAGE
GEN_PAGEPARAM	SET_PAGEPARAM_NO
GEN_PARAM_ART	SET_PARAM_ART_NO
GEN_PRODSUBSUBTYPE	SET_PRODSUBSUBTYPE_NO
GEN_PRODSUBTYPE	SET_PRODSUBTYPE_NO
GEN_PRODTYPE	SET_PRODTYPE_NO
GEN_PRODUCT	P_INSERT_NEWAUCTION
GEN_UNIT	SET_UNIT_N0
GEN_XMLATTRIBUTE_PARAM	SET_XMLATTRIBUTE_PARAM_NO
GEN_XMLDATA	SET_XMLDATA_NO
GEN_XMLDOC	SET_XMLDOC_NO
GEN_XMLDOCPARAM	SET_XMLDOCPARAM_NO
GEN_XMLDOC_ATTRIBUTE	SET_XMLDOC_ATTRIBUTE_NO
GEN_XMLELEMENTMAPPING	SET_XMLELEMENTMAPPING_NO
GEN_XMLPARAM	SET_XMLPARAM_NO
GEN_XMLSQL	SET_XMLSQL_NO
GEN_XMLSQLPARAM	SET_XMLSQLPARAM_NO

Die Arbeit des Generators GEN_PRODUCT wird im Unterschied zu allen anderen Generatoren des Datenbanksystems nicht durch einen Trigger unterstützt. Es ist die Prozedur P_INSERT_NEWAUCTION, die den Wert des Generators erhöht.

Prozeduren im Datenbanksystem

Um die Arbeit der Datenverarbeitungsschicht mit der Datenbank zu erleichtern wurden innerhalb der Datenbank mehrere Prozeduren erstellt. Ebner (1999, S. 160) definiert Prozeduren als Unterprogramme, die Parameter erwarten und Ergebnisse zurückliefern. Sie haben in der Auktionssystem-Anwendung die Aufgabe, eine Schnittstelle mit der Verarbeitungsebene zu bilden und bestimmte Aufgaben für sie zu erledigen.

Die *productlist.jsp*-Seite, auf deren Aufbau und Funktionalität ab Seite 94 eingegangen wird, benutzt die Prozedur P_XMLDATA_PRODLIST. Listing 3.3 stellt den Quelltext für ihre Erstellung dar. In der ersten Zeile der Prozedur wird der erwartete Eingabeparameter PARAM1 vom Typ SMALLINT übergeben. Im konkreten Fall stellt dieser das PK der Unterkategorie, für die die Produktliste geliefert werden soll. Die Zeilen 4-11 enthalten die Ausgabeparameter, die von der Prozedur als Ergebnis geliefert werden. Ab Zeile 14 folgt eine SELECT-Anweisung, mit der Daten aus den Tabellen TDAT_PRODUCT und TDAT_AUCTION selektiert werden. Die WHERE-Bedingung für die Auswahl der Daten folgt in den Zeilen 24 und 25. Es werden damit die Produkttupel ausgewählt, die zu der Unterkategorie gehören, deren Schlüsselwert als Eingabeparameter der Prozedur übergeben wurde. Das Schlüsselwort AND in der Zeile 25 verknüpft diese Bedingung mit der weiteren Voraussetzung, dass die Auktion zu diesem Produkt den Status *1* für eine laufende Auktion besitzt. Die ausgewählten Werte in den Zeilen 14-21 werden den Ausgabeparametern in den Zeilen 27-34 zugeordnet. Die Anweisung DO SUSPEND in der Zeile 35 ermöglicht die Auswahl von allen Ergebniszeilen.

Listing 3.3: Die Datenbank-Prozedur P_XMLDATA_PRODLIST

```
1 CREATE PROCEDURE P_XMLDATA_PRODLIST (
2     PARAM1 SMALLINT)
3 RETURNS (
4     PK SMALLINT,
5     NAME VARCHAR(80),
6     SUBSUBTYPE_PK SMALLINT,
7     CURRENT_LEFT_DAYS INTEGER,
8     CURRENT_LEFT_HRS INTEGER,
```

```
9       CURRENT_LEFT_MINS INTEGER,
10      CURRENT_BID_NUMBER INTEGER,
11      CURRENT_PRICE INTEGER)
12 AS begin
13   for select
14      p.PK,
15      p.name,
16      p.prodsubsubtype_code,
17      a.current_price,
18      a.current_bid_number,
19      a.current_left_days,
20      a.current_left_hrs,
21      a.current_left_mins
22 from tdat_product p join tdat_auction a
23                              on (p.pk = a.product_code)
24 where (p.prodsubsubtype_code = :PARAM1)
25                              and (a.status_code = 1)
26  INTO
27      :PK,
28      :NAME,
29      :SUBSUBTYPE_PK,
30      :current_price,
31      :current_bid_number,
32      :current_left_days,
33      :current_left_hrs,
34      :current_left_mins
35 DO SUSPEND;
36 END
```

Das Datenbanksystem besitzt insgesamt 27 Prozeduren, welche die Kommunikation der Verarbeitungsebene mit dem Datenbanksystem unterstützen. Als nächstes werden die Prozeduren aufgelistet und ihre zugehörige Aufgaben erläutert:

- P_AUCTIONER_WINNER: Selektiert das zuletzt abgegebene Gebot und den Gewinner eines bestimmten Artikels.
- P_BIDNUMBER: Zählt die abgegebenen Gebote für einen Artikel.
- P_COMPLETE_BID: Trägt ein abgegebenes Gebot in das System ein.
- P_INSERT_NEWAUCTION: Erstellt einen Eintrag im System für eine Auktion, d.h. eröffnet eine neue Auktion.

- P_PAGEDATA: Liefert die notwendigen Daten zum Aufbau einer JSP-Seite.
- P_PRODSUBTYPE: Liefert eine Liste der im System verfügbaren Kategorien zu einer gegebenen Hauptkategorie des Katalogs.
- P_PRODSUBTYPE_ALL: Liefert eine Liste aller im System verfügbaren Kategorien.
- P_PRODSUBSUBTYPE: Liefert eine Liste der verfügbaren Unterkategorien zu einer gegebenen Kategorie des Katalogs.
- P_PRODUCTNUMBER: Zählt die laufenden Auktionen zu einer Unterkategorie.
- P_PSSTNAME_FROM_PCODE: Liefert den Namen der Unterkategorie zu einem gegebenen PK (Primärschlüssel) eines Artikels.
- P_PSTNAME_FROM_PCODE: Liefert den Namen der Kategorie zu einem gegebenen PK des Artikels.
- P_PSTNAME_FROM_PSSTCODE: Liefert zum PK einer Unterkategorie den Namen seiner zugehörigen Kategorie.
- P_PTNAME_FROM_PCODE: Liefert zu einem vorgegebenen PK eines Artikels den Namen der Kategorie, dem dieser zugeordnet ist.
- P_PTNAME_FROM_PSSTCODE: Liefert den Namen einer Hauptkategorie zu einer vorgegebenen PK einer Unterkategorie, die der Hauptkategorie zugeordnet ist.
- P_PTNAME_FROM_PSTCODE: Liefert den Namen einer Hauptkategorie zum PK einer Kategorie, welche der Hauptkategorie zugeordnet ist.
- P_XMLDATA_PRODBID: Bereitet die Daten Artikel-PK, Name, Unterkategorie-PK, Anfang und Ende der Auktion, Preis, Menge und Käufernummer eines Gebots vor.
- P_XMLDATA_PRODBIDS: Liefert eine Liste der bereits abgegebenen Gebote zu einem Artikel.
- P_XMLDATA_PRODLIST: Liefert eine Liste der Artikel zu einer Unterkategorie, die einer laufenden Auktion zugeordnet sind.

- P_XMLDATA_PRODSUBSUBTYPE_ALL: Liefert eine Liste mit allen verfügbaren Kategorien und ihren zugehörigen Unterkategorien.
- P_XMLDATA_PRODSUBTYPE: Liefert eine Liste der Unterkategorien zu einer Kategorie und die Anzahl ihrer jeweiligen laufenden Auktionen.
- P_XMLDATA_PRODTYPEWITHSUBTYPE: Liefert eine Liste der Kategorien zu einer im System verfügbaren Hauptkategorie.
- P_XMLDATA_PRODUCT: Liefert aktuelle Daten zu einer Auktion anhand des PKs eines Produktes, welches versteigert wird. Das sind Daten wie der Produktname, der Anfang und das Ende der Auktion und die Anzahl bereits getätigter Gebote.
- P_XMLDATA_SUBTYPEPRODLIST: Liefert anhand der PK einer Kategorie eine Liste der laufenden Auktionen, die zu dieser Kategorie angehören.
- P_XMLDOC_ATTRIBUTE: Liefert alle Attribute eines XML-Dokuments anhand seinem PK sowie ihre zugehörigen Werte, die für ihre Erstellung notwendig sind.
- P_XMLDOC_PARAM: Nach Übergabe des Namens vom XML-Dokument werden die notwendigen Parameter für seine Erstellung geliefert. Diese müssen mit Werten versehen werden, um das Zusammenstellen des XML-Dokuments zu ermöglichen.
- P_XMLDOC_SQLDATA: Liefert die notwendigen Daten zum Erstellen eines XML-Dokuments. Als Eingabeparameter wird der Name des zu erstellenden Dokuments erwartet. Die Prozedur nutzt dabei Daten aus den Tabellen TDAT_XMLDATA, TCAT_XMLDOC und TDAT_XMLSQL. Sie liefert für jedes zu erstellende Teildokument des XML-Dokuments ein Tupel. Es enthält Daten wie die SQL-Abfrage auf die Datenbank, Elementnamen des Teildokuments und Zielknoten innerhalb des XML-Dokuments, an das die Teildokumente angehängt werden.
- P_XMLDOC_SQLPARAM: Liefert die System-Parameter, die zum Erstellen einer im System eingetragenen SQL-Abfrage notwendig sind.

3.3.3 Business Logik

Die Datenverarbeitungsebene ist die unentbehrliche Ebene für eine Anwendung. Sie ist für die korrekte Verarbeitung der Daten verantwortlich und stellt

„das Herz“ der Anwendung dar. Das Ausführen der einzelnen Prozesse in der Auktionssystem-Anwendung wird von den JSP-Seiten ausgelöst und von Java Beans unterstützt. Das Erstellen der dynamischen Inhalte wird mit der XML-Technologie realisiert.

Eine Übersicht über die wesentlichen JSP-Seiten der Prozessabläufe in der Verarbeitungsebene stellt die Abbildung 3.10 auf Seite 93 dar. Nachfolgend sind die Aufgaben und die auslösenden JSP-Seiten aufgelistet, auf die über die Seite *index.jsp* zugegriffen wird:

- Registrierung eines Benutzers an dem System (*register.jsp*),
- Anmelden und Abmelden eines Benutzers (*login.jsp* und *logout.jsp*),
- Ersteigern bzw. Kauf eines Produktes (*buy.jsp*),
- Anmelden des Verkaufs eines Produktes (*sell.jsp*[49]),
- Katalog des Auktionssystems (*prodtype.jsp*[50]).

Der Datenaustausch findet im XML-Format statt. Die einzelnen JSP-Seiten mit dynamischen Daten lösen jeweils das Erstellen eines XML-Dokuments mit den Datenbankdaten aus. Das XML-Dokument wird mit Hilfe einer XSL-Datei transformiert und an die JSP-Seite in dem benutzerfreundlichen HTML-Format weitergeleitet. Das Erstellen der XML-Dokumente wird im Kapitel 4.2.2 und deren Transformation im Kapitel 4.4.1 beschrieben.

Die Realisierung der JSP-Seiten in der Anwendung

Der Inhalt der JSP-Seiten innerhalb der Auktionssystem-Anwendung ist auf verschiedene Seiten aufgeteilt. Die auf Listing 3.4 gezeigte JSP-Seite *productlist.jsp* soll als Beispiel erläutert werden. Sie stellt eine Liste der vorhandenen Auktionen zu einer Unterkategorie dar.

Listing 3.4: Die JSP-Seite *productlist.jsp*

```
1 <%@ page contentType="text/html" %>
2 <%@ page errorPage="error.jsp" %>
3 <%@ page import="org.w3c.dom.Document" %>
4 <%@ page import="myapplication.xml.*" %>
5 <% out.clear(); %>
```

[49] Siehe Supplementum, Listing B.4 auf Seite 13.
[50] Siehe Supplementum, Listing B.2 auf Seite 12.

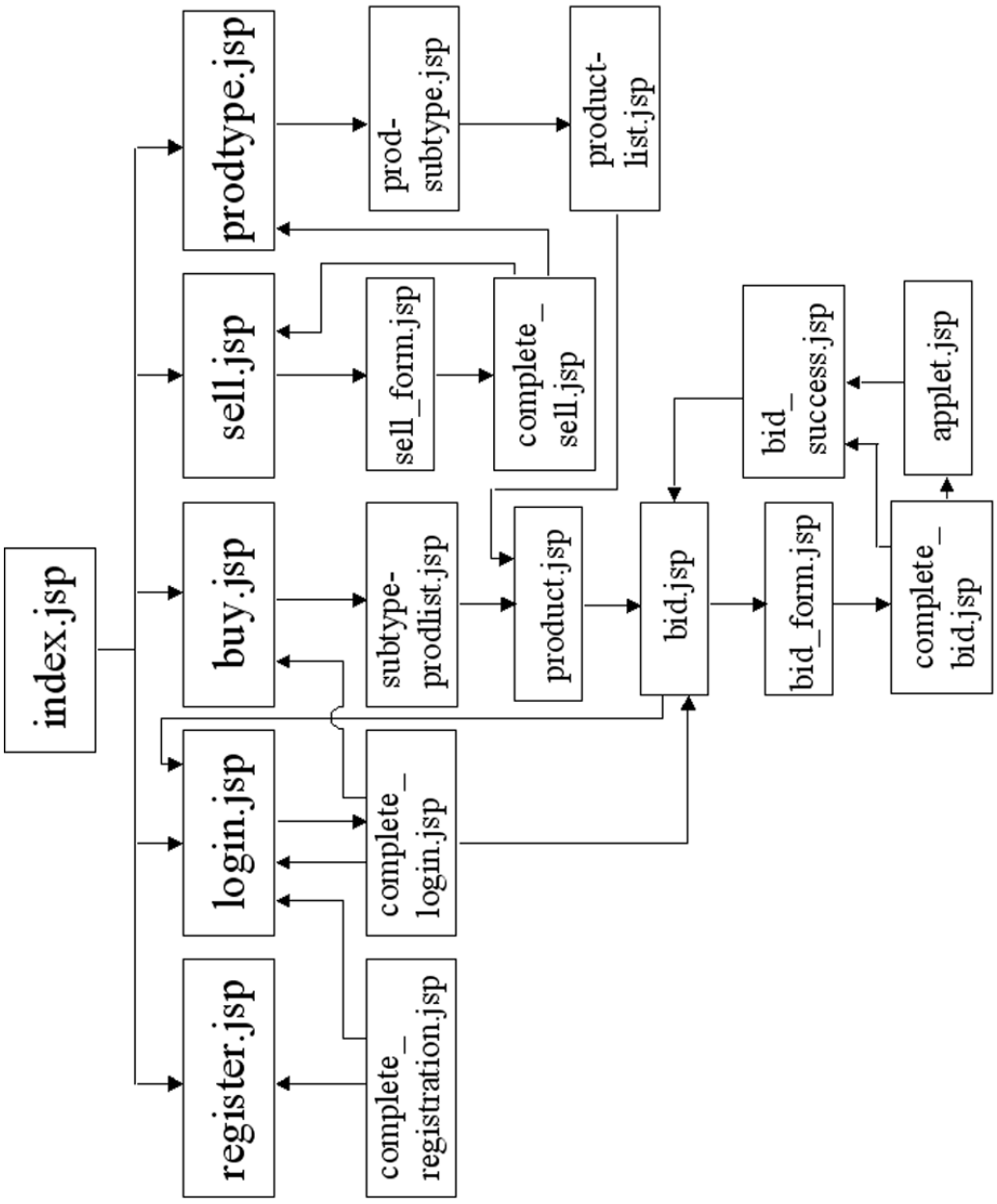

Abbildung 3.10: JSP-Seiten der Anwendung

```
6 <jsp:useBean id="auction"
7               class="myapplication.bean.Auction"
8               scope="page" />
9 <jsp:useBean id="currentpage"
10              class="myapplication.bean.Page"
11              scope="session" />
12 <%
13     currentpage.setPageNr(12);
14 %>
15 <%@ include file="include_files/head.html" %>
16 <%@ include file="include_files/body_begin.html" %>
17 <%@ include file="menue.jsp" %>
18 <!-- ********** BEGIN CONTENT ********** -->
19
20 <%@ include file="include_files/xmldoc_withprms.jsp" %>
21
22 <!-- ********** END CONTENT ********** -->
23 <%@ include file="copyright.html" %>
24 <%@ include file="include_files/body_end.html" %>
```

In der ersten Zeile der JSP-Seite wird bekanntgegeben, dass der Inhalt das HTML-Format besitzt. Die zweite Zeile gibt die Fehler-Seite bekannt. Sie wird aufgerufen und angezeigt, falls Fehler bei der Darstellung des Inhalts der *productlist.jsp* auftreten. Zeilen 3 und 4 importieren von der JSP-Seite benötigte Klassen. Der Quelltext *out.clear()* löscht sämtliche Ausgaben, die im *Output Buffer* bislang geschrieben sein könnten. In den Zeilen 6 bis 11 folgen die JSP-Tags *useBean*. Mit diesen Tags werden der Ort mit dem Attribut *class* und ihr Kontext mit dem Attribut *scope*[51] der benötigten *Bean*-Klassen bekanntgegeben. Jedem diesen Tags wird auch ein *id*-Attribut zugewiesen, mit dem im weiteren Quelltext auf die Klassen zugegriffen wird.

Die Arbeit jeder JSP-Seite im System wird von der Bean *Page* unterstützt. Die Bean wird mit der Anweisung

```
currentpage.setPageNr(int iPageNr);
```

in der Zeile 13 initialisiert, wobei der Parameter *iPageNr* die eindeutige Nummer der Seite im System darstellt. In diesem Beispiel ist *iPageNr* mit dem Wert *12* versehen. Die Verarbeitungsschritte, die durch dieser Zeile innerhalb der Bean *Page* ausgelöst werden, werden in Kürze genauer angegangen. Wei-

[51] Mögliche Attributwerte sind *page*, *request*, *session* und *application*.

terhin folgen auf der JSP-Seite *include-Tags*, die bestimmte Inhalte der Seite darstellen. Sie fügen den Inhalt folgender Seiten ein:

- *include_files/head.html*,
- *include_files/body_begin.html*,
- *menue.jsp*,
- *include_files/xmldoc_withprms.jsp*,
- *copyright.html*,
- *include_files/body_end.html*.

Das den Dateinamen vorgesetzte *include_files/* verweist auf den Ordner *include_files* der Anwendung, in dem sich die entsprechenden Dateien befinden.

Zunächst wird in der Zeile 15 der JSP-Seite die Seite *head.html* integriert. Ihr Inhalt ist auf Listing 3.5 dargestellt.

Listing 3.5: Quelltext des *head*-Abschnitts der HTML-Seiten

```
1 <html>
2    <head>
3       <title>
4          <jsp:getProperty name="currentpage"
5                           property="title" />
6       </title>
7       <link rel="stylesheet" type="text/css"
8                              href="stylesheet.css"/>
9    </head>
```

In der ersten Zeile des Listings wird zuerst der Start-*html*-Tag eingefügt. Zwischen den Zeilen 2 und 9 befindet sich der Inhalt des *head*-Tags. Dieser besteht aus einem *title*-Tag und einem *link*-Tag. Der *title*-Tag gibt den Titel der Seite an, welcher von der Bean *Page* zur Verfügung gestellt wird. Der *link*-Tag in den Zeilen 7 und 8 gibt die css-Stylesheet-Datei[52] bekannt, welche Regeln zur Seitenformatierung enthält.

Die weitere von der *productlist.jsp* integrierte *body_begin.html*-Seite (siehe Zeile 16 in Listing 3.4 auf Seite 94) wird auf Listing 3.6 dargestellt. Sie öffnet

[52]Die css-Datei definiert Formatierungsbereiche, mit deren Hilfe Formatierungsregel der Anwendung zentral gesteuert werden können.

den *body*-Tag und legt eine Tabelle an, deren Inhalt aus einer Zeile und einer Spalte besteht. Die Tabelle ist zentriert und nimmt 90% der Breite des Bildschirms in Anspruch. Dies wird durch das *width*-Attribut in der Zeile vier eingegeben. Hier wird auch der weiße Hintergrund mit dem Attribut *bgcolor* vorgegeben. Der Inhalt der Tabellen-Zelle wird durch den *div*-Tag in der Zeile sieben zentriert.

Listing 3.6: Quelltext des *body-begin*-Abschnitts der HTML-Seiten

```
<body background="images/hintergrund.gif">
   <div align="center">
      <table border="1" cellpadding="0" cellspacing="2"
                        width="90%" bgcolor="#FFFFFF">
         <tr>
            <td bgcolor="#FFFFFF">
               <div align="center">
```

Alle geöffneten und nicht geschlossenen Tags innerhalb der Seiten *head.html* und *body_begin.html* werden mit dem Quelltext der Seite *body_end.html* auf Listing 3.7 geschlossen.

Listing 3.7: Quelltext des *body-end*-Abschnitts der HTML-Seiten

```
                  </div>
               </td>
            </tr>
         </table>
      </div>
   </body>
</html>
```

Der für den Benutzer sichtbare Inhalt der Seiten besteht aus dem Menü des Systems, aus dem dynamisch erzeugten Inhalt und aus einer Zeile mit den Copyright-Informationen. Zur besseren Übersicht der JSP-Seiten sind diese Inhalte mit *include*-Tags eingefügt.[53] Das Menü der Anwendung wird von der Seite *menue.jsp* zur Verfügung gestellt. Ihr Quelltext ist auf Listing 3.8 dargestellt.

Listing 3.8: Quelltext der *menue.jsp*-Seite

```
<%@ page import="myapplication.bean.CustomerSession" %>
<table border="0">
<tr>
    <td colspan="7" align="center">
```

[53] Siehe Zeilen 17, 20 und 23 in Listing 3.4 auf Seite 94.

```
5     <img src="images/name_index.jpg" alt="" border="0"/>
6     </td>
7 </tr>
8 <tr>
9     <td class="menue"><a href="register.jsp">
10    <img src="images/png/neuanmeldung.png"
11         alt="Refistrierung" border="0"/></a>
12    </td>
13 <%
14 CustomerSession cust_session =
15         (CustomerSession)session.getAttribute("user");
16 if(cust_session == null){
17    out.println("<td class='menue'><a href='login.jsp'>
18                 <img src='images/png/anmelden.png'
19                  alt='Anmelden' border='0'/></a></td>");
20 }else{
21    out.println("<td class='menue'><a href='logout.jsp'>
22                 <img src='images/png/abmelden.png'
23                  alt='Abmelden' border='0'/></a></td>");
24 }
25 %>
26    <td class="menue"><a href="buy.jsp">
27        <img src="images/png/kaufen.png" alt="Kaufen"
28                                         border="0"/></a>
29    </td>
30    <td class="menue"><a href="sell.jsp">
31        <img src="images/png/verkaufen.png" alt="Verkaufen"
32        border="0"/></a>
33    </td>
34    <td class="menue"><a href="prodtype.jsp">
35        <img src="images/png/katalog.png" alt="Katalog"
36                                          border="0"/></a>
37    </td>
38    <td class="menue"><a
39    href="http://127.0.0.1:8080/myapplication/fopservlet">
40    <img src="images/png/pdf-report.png" alt="PDF-Report"
41                                         border="0"/></a>
42    </td>
43    <td class="menue"><a href="wstest.jsp">
44    <img src="images/png/ws_test.png" alt="WS-Test"
45                                         border="0"/></a>
46    </td>
```

```
47 </tr>
48 <tr>
49     <td colspan="7" align="center">
50     <form name="FormName" action="google_resultlist.jsp"
51                                                     method="get">
52         <div align="right">
53         <font color="#3366CC"><b>Google WS Suche:</b></font>
54         <input type="text" name="search_string" size="24">
55         <input type="submit" name="Submit" value="Suchen">
56         </div>
57     </form>
58     </td>
59 </tr>
60 </table>
61 <table>
62 <tr>
63     <td>
64     <img src="images/punkt.gif" height="50" width="10"
65                                                     border="0"/>
66     </td>
67 </tr>
68 </table>
```

Diese JSP-Seite importiert zuerst in der ersten Zeile die Bean *CusomerSession*, die über die Daten der angemeldeten Benutzer verfügt. Weiter werden im Quelltext in den Zeilen 2-60 und 61-68 zwei Tabellen erstellt. Die erste Tabelle stellt alle Menü-Links zur Verfügung; die zweite Tabelle legt die Distanz zwischen dem Menü und dem dynamischen Inhalt der Seite fest. Die Menü-Einträge *Anmelden* und *Abmelden* treten nicht gleichzeitig auf einer Seite auf. Deren Erscheinung hängt davon ab, ob der Benutzer im System angemeldet ist oder nicht (siehe Zeilen 16-24). Dabei wird das *CustomerSession*-Objekt aus dem *session*-Objekt[54] gelesen und geprüft, ob dieses den Wert *null* besitzt. Der *null*-Wert deutet darauf hin, dass der Benutzer nicht angemeldet ist und lässt den Button *Anmelden* erscheinen. Im anderen Fall ist der Benutzer erfolgreich angemeldet und hat nur die Möglichkeit, sich abzumelden, was durch den Button *Abmelden* zur Verfügung gestellt wird.

Nachdem das Menü des Systems erstellt wurde, wird der dynamisch erzeugte Inhalt der Seite angezeigt. Bei den JSP-Seiten mit XML-Daten werden die

[54] JSPs stellen vordefinierte Variablen wie *request (HttpServletRequest)*, *response (HttpServletRespponse)* und *session (HttpSession)* zur Verfügung.

Inhalte auch mit einem *include*-Tag, wie auf Listing 3.4 auf Seite 94, Zeile 20 dargestellt, eingebunden. Das Erzeugen des dynamischen Inhalts erfolgt mit der Seite

- *xmldoc_withprms.jsp*, wenn Parameter für das Erstellen der XML-Daten benötigt werden oder alternativ mit der Seite
- *xmldoc.jsp*[55], wenn das Erzeugen der XML-Daten keine Parameter benötigt.

Auf den Inhalt von *xmldoc_withprms.jsp* wird auf Seite 143 im Kapitel 4.2.2 näher eingegangen, wo das Bearbeiten der XML-Daten innerhalb der Anwendung analysiert wird.

Um den Inhalt der *productlist.jsp*-Seite abzuschließen wird die Seite *copyright.html* eingefügt. Diese ist aus Listing 3.9 ersichtlich.

Listing 3.9: Quelltext der *copyright.html*-Seite

```
1  <table border="0">
2  <tr>
3     <td><img src="images/punkt.gif" height="20"
4                                      width="10" border="0"/>
5     </td>
6  </tr>
7  <tr>
8     <td>
9        <p align="center">&copy; 2003-2004 Daniela Stoykova</p>
10    </td>
11 </tr>
12 <tr>
13    <td><img src="images/punkt.gif" height="10"
14                                      width="10" border="0"/>
15    </td>
16 </tr>
17 </table>
```

Der Quelltext der Seite erstellt eine Tabelle mit Abständen vor und nach der Copyright-Information, die aus Copyright-Symbol und -Daten besteht.

Die Bean Page

Die Funktionalität jeder JSP-Seite im System wird von der Bean *Page* unterstützt. Die Bean versorgt die JSP-Seiten mit Daten aus der Datenbank. Die

[55]Siehe Supplementum, Listing B.7 auf Seite 16.

Hilfstabellen in der Datenbank sind TCAT_PAGE und TCAT_PAGEPARAM, die aus der Abbildung 3.7) auf Seite 80 ersichtlich sind.

Die Tabelle TCAT_PAGE stellt Informationen zu jeder JSP-Seite zur Verfügung. Sie enthält Daten wie den Namen der JSP-Seite, deren Titel, den Typ der Seite und Informationen zu den XML-Daten. So enthält das Attribut XML-DOC_CODE die eindeutige Nummer des XML-Dokuments mit den Nutzdaten für die JSP-Seite. Die Spalte TRANSFORMFILE_PATH enthält den Pfad zur transformierenden XSL-Datei innerhalb der Anwendung.

Die Tabelle TCAT_PAGEPARAM enthält für jede Seite, die die Übergabe von Parametern benötigt, die eindeutige *Page*-Nummer, den Namen des Parameters der aufrufenden JSP-Seite und den dazugehörigen PK des System-Parameters. Einer Seite können mehrere Parameter zugeordnet werden.

Als Bindeglied zwischen den obigen zwei Tabellen und der Bean *Page* dient die Prozedur P_PAGEDATA. Sie erwartet den eindeutigen Schlüssel der JSP-Seite als Eingabeparameter und bereitet die von der Bean *Page* benötigten Daten vor.

Wie aus Zeile 13 in Listing 3.4 auf Seite 94 ersichtlich, stellt die JSP-Seite ihre Beziehung zur Bean *Page* mit ihrer Methode *setPageNr(int iPageNr)* zur Verfügung. Sie erwartet die eindeutige Nummer *iPageNr* der JSP-Seite im System. Innerhalb dieser Methode wird die *init()*-Methode der Bean *Page* aufgerufen, welche auf Listing 3.10 dargestellt ist.

Listing 3.10: Die *init()*-Methode der Bean *Page*

```
1  private void init(int page_nr) throws Exception {
2     String sql = "select * from p_pagedata("
3                        + String.valueOf(page_nr) + ")";
4     rsPage = db.executeQuery(sql);
5     rsPage.next();
6     this.setTitle(rsPage.getObject("TITLE").toString());
7     if(rsPage.getObject("XMLDOC_NAME").toString()
8                                          .compareTo("")!= 0){
9        this.setXmlDocName(rsPage.getObject("XMLDOC_NAME")
10                                                .toString());
11    }
12    if(rsPage.getObject("TRANSFORMFILE_PATH").toString()
13                                         .compareTo("")!= 0){
14       this.setTransformFilePath(rsPage
```

```
15                  .getObject("TRANSFORMFILE_PATH").toString());
16      }
17      if(rsPage.getObject("REQUESTPARAM_NAME")
18                                  .toString().compareTo("")!= 0){
19          String[] prm = new String[2];
20          Vector v = new Vector();
21          do{
22              prm[0] = rsPage.getObject("XMLPRM_NAME")
23                                                      .toString();
24              prm[1] = rsPage.getObject("REQUESTPARAM_NAME")
25                                                      .toString();
26              v.add(prm);
27          }while(rsPage.next());
28          this.setDocParams(v);
29      }
30      rsPage.close();
31  }
```

Diese Methode führt zunächst in Zeile 2 die Abfrage

```
select * from p_pagedata(int iPageNr)
```

auf die Datenbank aus und nutzt damit die bereits angesprochene Prozedur P_PAGEDATA. Das Ausführen der Abfrage in Zeile 4 liefert das *ResultSet*-Objekt *rsPage* zurück. Zeile 5 springt zum ersten Datensatz von *rsPage* und Zeile 6 versieht das *title*-Attribut der Bean *Page* mit einem Wert, welches damit den Inhalt des *title*-Tags der HTML-Seite enthält. Wenn für den Namen des zu erstellenden XML-Dokuments und für den dazugehörigen System-Pfad zur Transformationsdatei keinen *null*-Wert vom *ResultSet rsPage* geliefert wurde, werden in den Zeilen 7-16 die entsprechenden Attribute im Bean *Page* auch mit einem Wert versehen.

Bevor das *ResultSet*-Objekt geschlossen wird, werden in den Zeilen 17-29 die Informationen zu den benötigten Parametern für die XML-Daten gelesen und in dem Vektor-Attribut *v* der Bean *Page* gespeichert. Wenn die JSP-Seite mehr als einen Parameter benötigt, enthält das *ResultSet*-Objekt *rsPage* mehrere Zeilen. Die *do-while*-Schleife in den Zeilen 21-27 wird so oft durchlaufen bis alle Zeilen und damit alle Parameter eingelesen sind.

Nach dem Ausführen der *init()*-Methode enthalten alle benötigten Attribute der Bean *Page* jeweils einen Wert, mit dessen Hilfe die vollständige JSP-Seite aufgebaut werden kann.

Registrierung des Benutzers

Die Erstanmeldung eines Benutzers im Auktionssystem wird durch die JSP-Seiten

- *register.jsp* und
- *complete_registration.jsp*[56]

durchgeführt.

Die *register.jsp*-Seite stellt das Anmeldeformular zur Verfügung. Hier werden Felder für die notwendigen Daten sowie die möglichen Auswahlfelder für das Geschlecht und das Land zur Verfügung gestellt. Sie befinden sich in den Datenbanktabellen TCAT_GENDER und TCAT_COUNTRY. Dadurch wird eine Trennung von Datenpräsentation und Datenverarbeitung erreicht. Das Auswahlfeld für das Land eines Benutzers kann durch Modifikation der Tabelle TCAT_COUNTRY mit weiteren Einträgen erweitert oder gekürzt werden, was die Datenbankpflege vereinfacht.

Nach dem Ausfüllen und Absenden des Formulars durch den Benutzer werden die Daten von der Seite *complete_registration.jsp* verarbeitet. Die dazugehörige Bean ist *NewRegSession*[57] im Paket *myapplication.bean*. Durch den unter diesem Abschnitt stehenden Quelltextausschnitt aus der Seite *complete_registration.jsp* wird mit dem Befehl *useBean* die Bean *NewRegSession* im Gültigkeitsbereich *page* zur Verfügung gestellt. Mit dem Befehl *setProperty* werden alle Variablen der Bean mit den Werten des ausgefüllten Formulars versehen:

```
<jsp:useBean id="regSession" class=
        "myapplication.bean.NewRegSession" scope="page" />
<jsp:setProperty name="regSession" property="*"/>
```

Bevor das Einfügen der Daten des neuen Benutzers in das Datenbanksystem stattfindet, werden die Daten des Formulars analysiert. Wie aus der Abbildung 3.11 auf Seite 104 ersichtlich, wird nach dem Absenden der Anmeldedaten zuerst geprüft, ob alle obligatorischen Felder ausgefüllt sind. Falls dies nicht der Fall ist, wird die *register.jsp*-Seite erneut aufgerufen, die den Benutzer auffordert, die unausgefüllten Felder mit Werten zu versehen. Die Aufforderung wird über ein Attribut mit der Methode *setAttribute()* des Objektes *request*

[56]Siehe Supplementum, Listing B.1 auf Seite 11.
[57]Siehe Supplementum, Listing B.14 auf Seite 33.

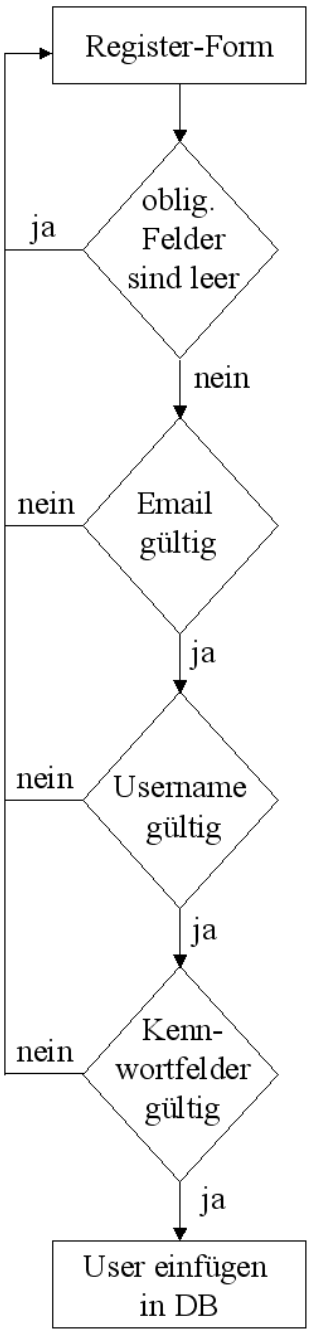

Abbildung 3.11: Der Registrierungsprozess eines neuen Benutzers

weitergegeben.[58]

Wenn alle obligatorischen Felder ausgefüllt sind, wird die Gültigkeit der Email-Adresse geprüft.[59] Eine ungültige Email-Adresse führt erneut zur *register.jsp*-Seite mit der Bemerkung, dass das Email-Feld nicht gültig ist. Die Werte der bereits ausgefüllten Felder bleiben vorhanden, um ein neues Ausfüllen des Formulars zu vermeiden.

Wenn die Email-Adresse gültig ist, wird eine Abfrage auf den vorhandenen Benutzernamen in der Datenbank ausgeführt. Wenn der gewünschte Benutzername bereits vorhanden ist, wird erneut die *register.jsp* mit der Aufforderung aufgerufen, einen neuen Namen für die Registrierung einzutragen.[60]

Wenn der Benutzername vom System akzeptiert wird, wird weiterhin geprüft, ob das eingegebene Kennwort und die Kennwort-Bestätigung identisch sind und ob das gewünschte Kennwort mindestens sechs Zeichen enthält. Wenn das nicht der Fall ist, wird erneut die Seite *register.jsp* aufgerufen.[61] Im anderen Fall wird der Benutzer in die Datenbank eingetragen. Abbildung 3.12 auf Seite 106 zeigt ein Beispiel eines Registrierungsvorgangs, der zu einer Fehlermeldung geführt hat.

Anmelden des Benutzers am System

Der Ablauf der Anmeldung eines Benutzers am System wird durch die Abbildung 3.13 auf Seite 108 dargestellt. Das Anmelden eines Benutzers am Auktionssystem wird durch die JSP-Seiten

- *login.jsp* und
- *complete_login.jsp*[62]

realisiert.

Die *login.jsp*-Seite stellt das Formular für die Anmeldung zur Verfügung. Hier müssen der Benutzername und das Kennwort eingetragen werden, um eine Verifizierung des Benutzers starten zu können.

[58] Siehe Supplementum, Zeilen 11-12 in Listing B.1 auf Seite 11.
[59] Siehe Supplementum, Zeile 16 in Listing B.1 auf Seite 11.
[60] Siehe Supplementum, Zeilen 24-30 in Listing B.1 auf Seite 11.
[61] Siehe Supplementum, Zeilen 33-39 in Listing B.1 auf Seite 11.
[62] Siehe Supplementum, Listing B.8 auf Seite 16.

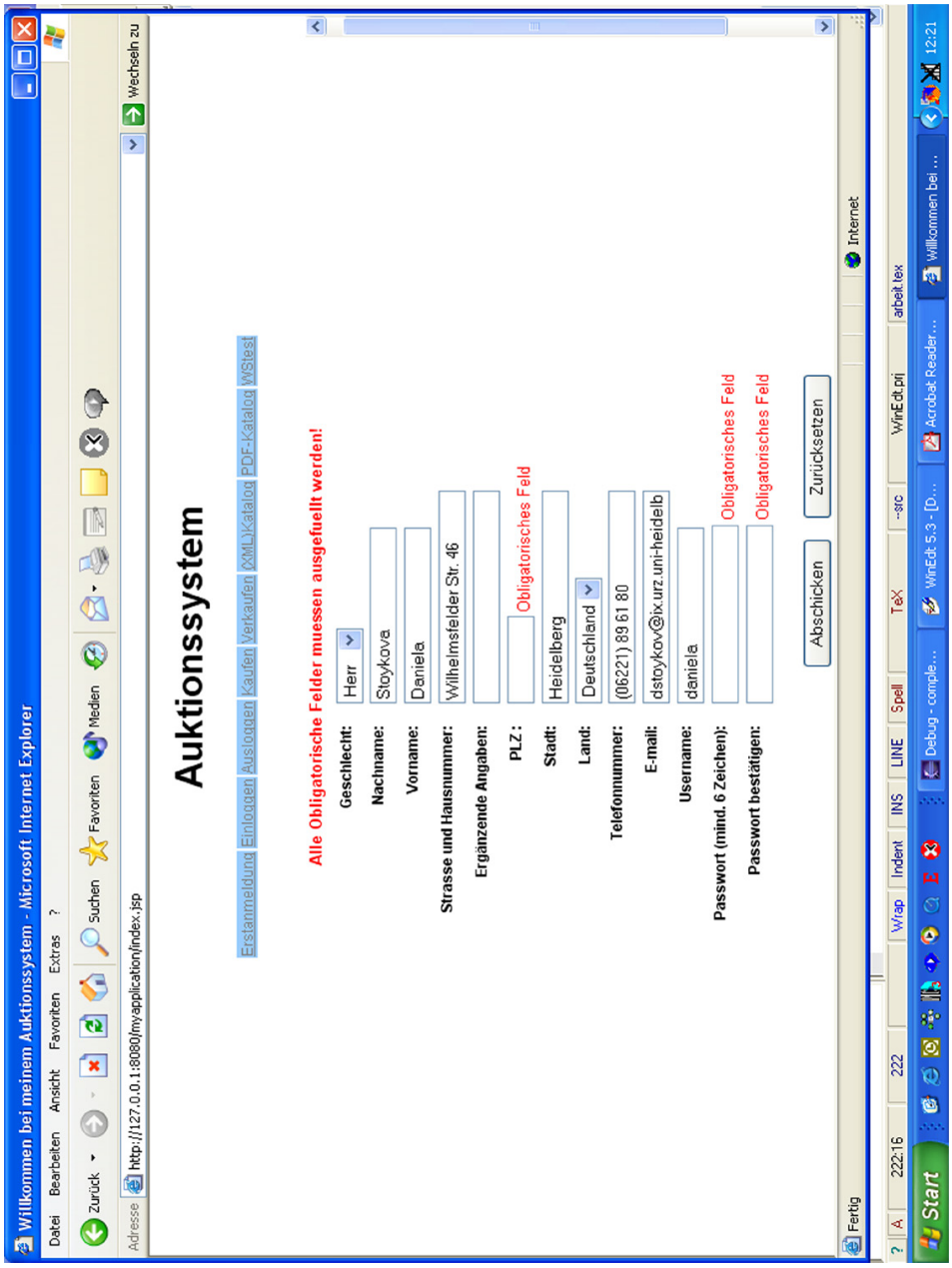

Abbildung 3.12: Registrierung eines neuen Benutzers

Nach Abgabe der Anmeldedaten des Benutzers findet ein Aufruf der Seite *complete_login.jsp* statt. Die dazugehörige Bean ist *CustomerSession*[63] und wird durch folgenden Quelltext, ähnlich wie beim Registrieren des Benutzers, in die JSP-Seite eingebunden:

```
<jsp:useBean id = "customer_session" class=
        "myapplication.bean.CustomerSession" scope="page"/>
<jsp:setProperty name="customer_session" property="*" />
```

Die letzte Zeile versieht die Parameter für den Benutzernamen und für das Passwort innerhalb der Bean *CustomerSession* mit den im Formular eingegebenen Werten. Als nächstes wird die Methode *validateUser()* der Bean *CustomerSession* aufgerufen.[64] Diese Methode prüft die Gültigkeit des Accounts und liefert ein *CustomerSession*-Objekt zurück.

Weiterhin wird mit der Methode *getParameter()* des *request*-Objektes innerhalb der *complete_login.jsp* geprüft, ob der Parameter *forward_page* einen Wert enthält.[65] Wenn er keinen Wert besitzt, wird der String *show* in Zeile 12 auf den Wert *buy.jsp* gesetzt - das ist der Fall, wenn die *complete_login.jsp* von der *login.jsp* aufgerufen wird. Ziel dieses Teils des Quelltextes ist es, nach einer erfolgreichen Anmeldung die nächste Seite zu bestimmen. So werden die *login.jsp* und anschließend die *complete_login.jsp* aufgerufen, wenn ein Benutzer ein Gebot zu einem beliebigen Artikel abgeben möchte, aber im System noch nicht angemeldet ist. Im Parameter *forward_page* des *request*-Objektes wird der gewünschte Artikel gehalten, bis die Anmeldung erfolgreich ausgeführt wird. Danach kann das Gebot durchgeführt werden, ohne dass der Benutzer den Artikel im Katalog nochmal suchen muss.

Als nächstes wird in *complete_login.jsp* in Zeile 16 die Information über die Gültigkeit des Benutzerkontos von der Bean *CustomerSession* aufgerufen. Wenn das Benutzerkonto existiert, wird in der Zeile 17 aus Sicherheitsgründen die IP-Adresse abgerufen und gespeichert. In einem *HashMap*-Objekt werden alle geöffneten Sitzungen gehalten. Bevor eine neue Sitzung gestartet wird, wird zuerst in den Zeilen 18-22 geprüft, ob die alte bereits geschlossen wurde:

```
if(monitor.containsKey(customer_session.getUserid())){
  HttpSession oldSession =
```

[63] Siehe Supplementum, Listing B.13 auf Seite 29.
[64] Siehe Supplementum, Zeilen 133-142 in Listing B.13 auf Seite 29.
[65] Siehe Supplementum, Zeilen 11-15 in Listing B.8 auf Seite 16.

```
        (HttpSession)monitor.get(customer_session.getUserid());
  oldSession.invalidate();
}
```

Wenn eine Sitzung zu diesem Benutzer im *HashMap*-Objekt *monitor* existiert, wird vor dem Erstellen einer neuen Sitzung die alte Sitzung vernichtet. Danach wird die neue Sitzung für den Anwendungsnutzer wie folgt erstellt:

```
session = request.getSession(true);
session.setAttribute("user", customer_session);
monitor.put(customer_session, session);
```

Anmelden des Verkaufs eines Produktes

Das Auktionssystem ist ein Marktplatz für holländische Auktionen, auf dem Käufer und Verkäufer zusammentreffen. Daher bietet das System die Möglichkeit sowohl Artikel zu kaufen, indem der Käufer an einer Auktion teilnimmt, als auch Artikel zu verkaufen, indem der Verkäufer eine Auktion für einen Artikel eröffnet. Um dem System die Daten für die anzumeldende Auktion weitergeben zu können, soll zuerst die Kategorie, in der eine Auktion platziert wird, dem System bekanntgegeben werden. Danach wird das Formular zum Eröffnen einer Auktion angezeigt und seine Daten weiterverarbeitet.

Das Eintragen einer neuen Auktion in das Auktionssystem wird von den JSP-Seiten

- *sell.jsp*[66],
- *sellcategory.jsp*[67],
- *sell_form.jsp* und
- *complete_sell.jsp*[68].

ausgeführt.

Die Seite *sell.jsp* prüft, ob der Benutzer bereits im System angemeldet ist.

[66]Siehe Supplementum, Listing B.4 auf Seite 13.
[67]Siehe Supplementum, Listing B.5 auf Seite 14.
[68]Siehe Supplementum, Listing B.6 auf Seite 14.

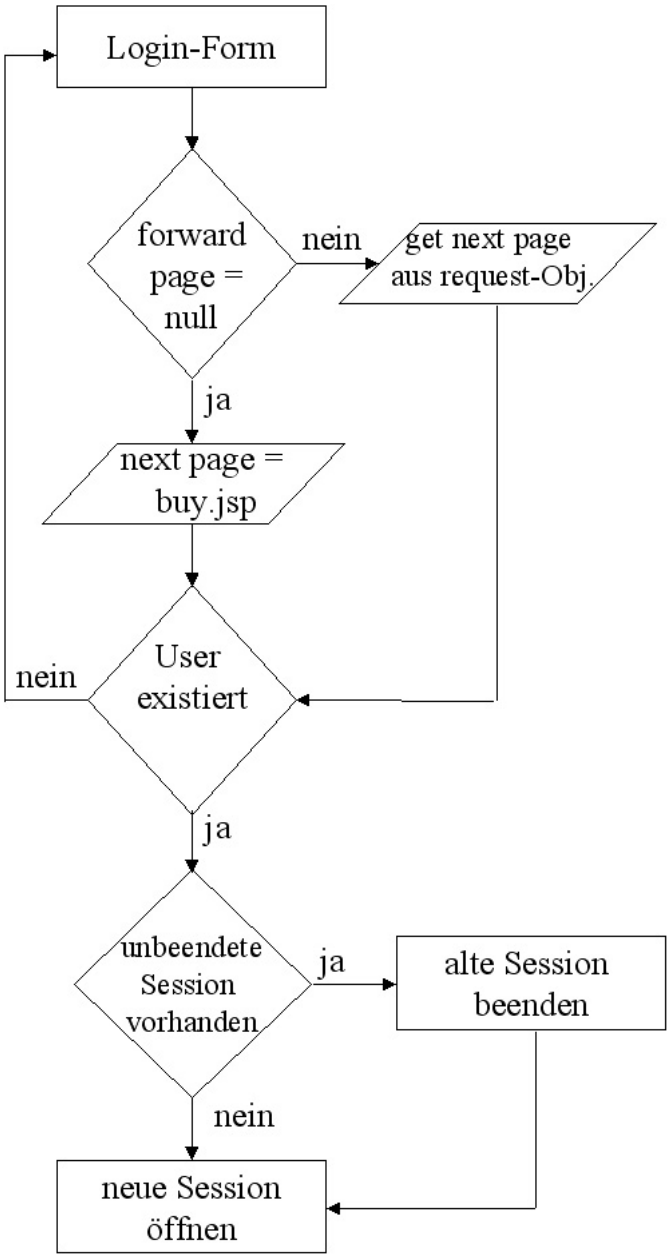

Abbildung 3.13: Der Anmeldeprozess eines Benutzers

Wenn der Anwender keine Anmeldung durchgeführt hat, wird die Seite *login.jsp* geladen und nach erfolgreicher Anmeldung die *sellcategory.jsp* aufgerufen. Wenn eine Anmeldung bereits vorliegt, wird die Seite *sellcategory.jsp* direkt aufgerufen. Sie zeigt, wie auf der Abbildung 3.14 auf Seite 115 zu sehen, alle Hauptkategorien und ihre zugehörgen Kategorien. Sie ermöglicht durch einfachen Mausklick das Auswählen einer beliebigen Kategorie, was die Seite *sell_form.jsp* anzeigt. Sie stellt das Formular zur Anmeldung einer Auktion zur Verfügung. Anschließend wird die Bearbeitung der Auktionsdaten mit Hilfe der Seite *complete_sell.jsp* realisiert. Diese ist für die Überprüfung der Daten vor dem Eintragen der Auktion in das System verantwortlich. Die Arbeit der JSP-Seiten wird hier durch die Java Beans

- *Auction*[69],
- *Page*[70],
- *Product*[71] und
- *SubSubCategory*[72]

unterstützt. Die Beans befinden sich innerhalb des Pakets *myapplication.bean*.

Der Vorgang zum Eintragen einer Auktion im System sieht so aus: Bei ausgeführter Anmeldung und beim Auswählen des Links *Verkaufen* aus dem Menü des Systems wird die Seite *sellcategory.jsp*[73] geladen. Diese importiert in den Zeilen 3 und 4 die von ihr benötigten Klassen und bindet in der Zeile 6 die Bean *Page* wie folgt ein:

```
<jsp:useBean id = "currentpage" class = "myapplication.bean.Page"
                                scope = "session" />
```

Der eigentliche Inhalt der Seite wird mit der Datei *include_files/xmldoc.jsp*[74] eingefügt. Nach dem Ersetzen der benötigten Parameter aus der Bean *currentpage*, nämlich *currentpage.getXmlDocName()* und *currentpage.getTransformFilePath()*, wird der Quelltext wie auf Listing 3.11 angezeigt. Er sorgt für das Erstellen der XML-Datei *ProdTypeWithSubType* mit Hilfe der Datenbank-Daten und der Klassen *XmlDocument*, *XmlProductData* und *XmlTool*. Auf die Funktion dieser Klassen wird in Kapitel 4.2.2 näher eingegangen.

[69]Siehe Supplementum, Listing B.11 auf Seite 20.
[70]Siehe Supplementum, Listing B.15 auf Seite 39.
[71]Siehe Supplementum, Listing B.16 auf Seite 41.
[72]Siehe Supplementum, Listing B.17 auf Seite 44.
[73]Siehe Supplementum, Listing B.5 auf Seite 14.
[74]Siehe Supplementum, Listing B.7 auf Seite 16.

Listing 3.11: Erstellen und Formatieren des XML-Dokuments

```
1  XmlProductData xmldata = new XmlProductData();
2  XmlTool xmltool = new XmlTool();
3  Document xmldoc =
4      xmldata.getDocumentFromDBData("ProdTypeWithSubType");
5  String path = "http://" +
6                  request.getRemoteAddr()+ ":" +
7                  String.valueOf(request.getServerPort()) +
8                  request.getContextPath()+
9                  "/xsl/prodtypewithsubtypesell.xsl";
10 String html = xmltool.transform(path, xmldoc);
```

Die ersten beiden Zeilen von Listing 3.11 erstellen jeweils ein Objekt der Klassen *XmlProductData* und *XmlTool*. Anschließend wird in der Zeile 4 das XML-Dokument mit Hilfe der Methode *getDocumentFromDBData()* aus der Klasse *XmlProductData* erstellt und dem *Dokument*-Objekt *xmldoc* zugewiesen. Es enthält alle Hauptkategorien und deren zugewiesene Kategorien des holländischen Auktionssystems. Der *String*-Parameter *path* in Zeile 5 ermittelt die Adresse der formatierenden XSL-Datei. Sie setzt sich aus folgenden Bestandteilen zusammen:

- *http://*: Das HTTP-Protokoll wird hier als übertragendes Protokoll gewählt.

- *request.getRemoteAddr()*: Das *request*-Objekt hält die IP-Adresse der Anwendung bereit. Diese kann damit dynamisch ermittelt werden, was für die Flexibilität der Anwendung beiträgt. Sie kann damit auf verschiedenen Maschinen laufen, ohne eine Änderung des Quelltextes zu erfordern.

- *request.getServerPort()*: Der Server-Port der Anwendung wird ebenso vom *request*-Objekt ermittelt und auch dynamisch gehalten. Die Addresse der Anwendung und des Server-Ports werden durch ein Semikolon getrennt.

- *request.getContextPath()*: Weiterhin wird der *ContextPath* der Anwendung ermittelt, unter dem diese innerhalb des Servers läuft. Im konkreten Fall ist dies *myapplication*. Der *ContextPath* wird mit dem *Context*-Tag innerhalb der Datei *server.xml* festgelegt. Diese Datei befindet sich im Ordner *conf* des Tomcat Servers. Der Quelltext des *Context*-Tags sieht für die Anwendung wie folgt aus:

```
<Context path="/myapplication" reloadable="true"
docBase="D:\Daniela\promotion\MyApplication\myapplication"
workDir="D:\Daniela\promotion\MyApplication\work\org
                                              \apache\jsp" />
```

- */xsl/prodtypewithsubtypesell.xsl*: Anschließend wird der Pfad zur XSL-Datei bekanntgegeben. Sie heißt *prodtypewithsubtypesell.xsl* und befindet sich im Ordner *xsl* der Anwendung.

Wenn das Documentobjekt *xmldoc* (siehe Zeile 3 und 5, Listing 3.11) mit den Nutzdaten und dem Pfad zur XSL-Datei, die die Regel für die Formatierung der Daten enthält, vorhanden sind, kann die Transformation gestartet werden. Die Methode *transform()* der Klasse *XmlTool* wird in der Zeile 10 des Listings 3.11 aufgerufen. Sie formt eine beliebige XML-Datei in eine HTML-Datei um, wenn die Regel zur Transformation in einer XSL-Datei bereits gegeben ist. Die Methode *transform()* erwartet daher ein XML-Dokumentobjekt und den Pfad zur XSL-Datei. Das Ergebnis ist ein Stringobjekt, das innerhalb der JSP-Seite eingebunden werden kann. Es wird mit der Abbildung 3.14 veranschaulicht.

Wie aus Abbildung 3.14 auf Seite 115 ersichtlich, wird der Name jeder Kategorie auf der erstellten Seite mit einem Link zur *sell_form.jsp*-Seite versehen. An die *sell_form.jsp*-Seite wird der Parameter *SUBPK* übergeben. Er entspricht der Datenbank-Primärschlüssel derjenigen Kategorie, die durch den Benutzer ausgewählt wurde. Ein Teil des Quelltextes ist auf Listing 3.12 dargestellt.

Listing 3.12: *sell_form.jsp*-Seite

```
1 <%@ page contentType="text/html"%>
2 <%@ page errorPage="error.jsp" %>
3 <%@ page import="java.util.*" %>
4 <%
5 out.clear();
6 Vector v = null;
7 String[] sPara = null;
8 %>
9 <jsp:useBean id = "auction"
10              class = "myapplication.bean.Auction"
11              scope = "request" />
12 <jsp:useBean id = "subsubcat" class =
13     "myapplication.bean.SubSubCategory" scope = "session" />
14 <jsp:useBean id = "prod" class = "myapplication.bean.Product"
```

```
15                              scope = "session" />
16 <jsp:useBean id="currentpage"
17 class="myapplication.bean.Page" scope="session" />
18 <%
19     currentpage.setPageNr(6);
20 %>
21 <%@ include file="include_files/head.html" %>
22 <%@ include file="include_files/body_begin.html" %>
23 <%@ include file="menue.jsp" %>
24 <!-- ********** BEGIN CONTENT **********-->
25
26 <table>
27 <tr>
28 <td></td>
29 <td class="heading"><b>Anmeldung Auktion</b></td>
30 <td></td>
31 </tr>
32
33 <tr> <td></td> <td>
34 <img src="images/punkt.gif" alt="" height="20"
35                                 width="10" border="0"/>
36 </td> <td></td> </tr>
37
38 <form action="complete_sell.jsp?SUBPK=
39     <% out.println(request.getParameter("SUBPK")); %>"
40                                             method="post">
41 <tr>
42 <%
43 out.println("<input type='hidden' name='subcat_pk' value=" +
44 request.getParameter("SUBPK") + " border='0'>");
45 %>
46 <table border="0">
47 <%
48 if((request.getAttribute("message")) != null){
49     out.println("<tr><td colspan='3'><div align='center'>
50     <font color='red'><b>" + request.getAttribute("message")
51                         + "</b></font></div></td></tr>");
52 }
53 %>
54 <tr> <td align="right"><div align="right"><b>Unterkategorie:
55 <font color='red'>*</font></b></div></td> <td><img
56 src="images/punkt.gif" alt="" height="10" width="10"
```

```
57 border="0"></td> <td> <select name="subsubcat_pk" size="1">
58 <%
59 v = null;
60 v = subsubcat.
61     getDbSubSubCategoryData(request.getParameter("SUBPK"));
62 sPara = null;
63    for(int i=0; i<v.size(); i++){
64      sPara = (String[])v.elementAt(i);
65      out.println("<option value='" + sPara[0] + "'>" +
66                                      sPara[1] + "</option>");
67    }
68 %>
69 </select>
70 </td>
71 </tr>
72
73 ...
74
75 <tr><td align="right"><div align="right"><b>Einheit:
76 <font color='red'>*</font></b></div></td> <td></td> <td>
77 <select name="unit" size="1">
78 <%
79 v = null;
80 v = prod.getDbUnitData();
81 prod.closeDb();
82 sPara = null;
83    for(int i=0; i<v.size(); i++){
84      sPara = (String[])v.elementAt(i);
85      out.println("<option value='" + sPara[0] + "'>" +
86                                      sPara[1] + "</option>");
87    }
88 %>
89 </select>
90 </td>
91 </tr>
92
93 ...
```

Wie bei allen bisherigen JSP-Seiten werden in der Seite *sell_form.jsp* mit dem Tag *jsp:useBean* die benötigten Beans *SubSubCategory*, *Product*, *Auction* und

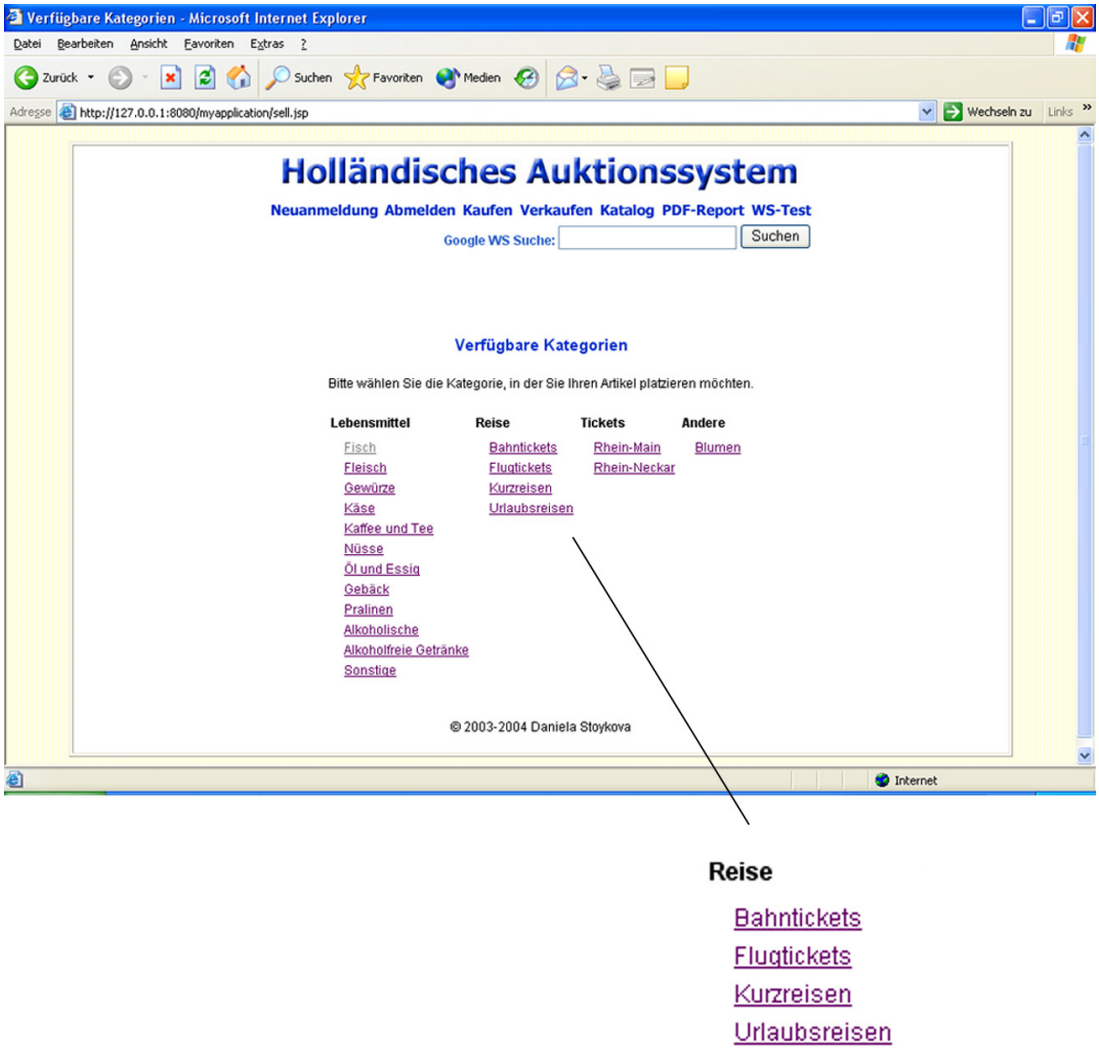

Abbildung 3.14: Transformierte Seite mit den Hauptkategorien und den zugehörigen Kategorien des Auktionssystems

Abbildung 3.15: Anmeldung einer Auktion in der Anwendung

Page eingebunden.[75] Die Bean *SubSubCategory*[76] liefert mit Hilfe der Methode *getDbSubSubCategoryData()* in den Zeilen 58-64 einen Vektor mit den Unterkategorien. Hier wird der übergebene Parameter *SUBPK* in Anspruch genommen. Der Vektor enthält die Unterkategorien zur ausgewählten Kategorie und wird als Dropdown-Liste wie in der Abbildung 3.15 dargestellt. Die Methode *getDbUnitData()* der Bean *Product* liefert die möglichen Einheiten für die Menge des Artikels der neuen Auktion in Form eines Vektors.[77]. Die möglichen Einheiten werden wieder wie bei den Unterkategorien als Dropdown-Liste dargestellt. Die *sell_form.jsp*-Seite stellt das eigentliche Formular zur Anmeldung einer Auktion dar. Er ist in der Abbildung 3.15 dargestellt und erwartet die hier aufgelisteten Pflichtdaten:

- *Unterkategie*: Da vor dem Anzeigen des Formulars die Kategorie des Auktionsartikels ausgewählt wurde, werden im Formular die möglichen Unterkategorien in einer Dropdown-Liste mit Hilfe des Parameters *SUBPK* zur Verfügung gestellt.

- *Artikelname*: Der vollständige Name des Artikels, mit der dieser in der Datenbank geführt und den potenziellen Käufern angezeigt wird.

[75] Siehe Zeilen 9-17 in Listing 3.12 auf Seite 112.
[76] Siehe Supplementum, Listing B.17 auf Seite 44.
[77] Siehe Zeile 80 in Listing 3.12 auf Seite 112.

- *Menge*: Hier wird eine Zahl für die Menge des Artikels, welche für die Auktion angeboten wird, erwartet.

- *Einheit*: Einheit stellt eine Dropdown-Liste, die die möglichen Einheiten (Stück, kg) zur Verfügung stellt.

- *Startpreis*: Das System realisiert das Prinzip der holländischen Auktion. Dazu wird ein maximaler Preis benötigt, zu dem der Artikel beim Start der Auktion angeboten wird. Mit diesem Preis wird die Auktion gestartet.

- *Preisrutsch*: Jede Auktion hat einen Anfangszeitpunkt und ein Ende. Innerhalb dieser Zeitspanne wird der Auktionspreis gesenkt, falls Einheiten des Artikels noch offen stehen. Der Preisrutsch ist die Preiseinheit, um die der Preis in bestimmten Zeitabständen während des Verlaufs der Auktion gesenkt wird.

- *Endpreis*: Das ist der niedrigste Preis, zu dem der Verkäufer bereit ist, den Auktionsartikel einem Käufer zu übergeben. Der niedrigste Preis ist der minimalste Preis, zu dem ein Auktionsartikel erworben werden kann. Im holländischen Auktionssystem wird jeder Artikel eine halbe Stunde vor dem Ablauf der Auktion zum niedrigsten Preis angeboten, falls noch offene Einheiten eines Artikels zur Verfügung stehen.

- *Dauer der Auktion*: Der Besitzer eines Artikels wird bei der Anmeldung einer Auktion aufgefordert, ihre Laufzeit einzutragen. Hier ist eine Eingabe in Tagen und Stunden möglich.

Nach dem Absenden des Formulars werden seine Daten der Seite *complete_sell.jsp* weitergegeben. Sie bindet die Bean *Auction* ein und versieht dort alle Parameter des Formulars mit Werten. Die Anmeldung einer Auktion enthält die Prüfung der Gültigkeit der durch den Benutzer eingegebenen Daten und deren Aufnahme in die Datenbank.

Der Anmeldungsprozess einer Auktion wird auf Abbildung 3.16 auf Seite 119 dargestellt und läuft wie folgt ab:

- Zunächst wird die gewünschte Kategorie ausgewählt, in der der Artikel platziert werden soll. Anhand dieser wird die Seite *sell_form.jsp* angezeigt, welche in einer Dropdown-Liste die zugehörigen Unterkategorien darstellt.

- Nach dem Ausfüllen und Absenden des Formulars werden die Daten des Formulars geprüft. Alle nicht ausgefüllten Felder für Dauer der Auktion (Tage und Stunden) bekommen automatisch den Wert *0* zugewiesen. Alle leeren Felder verursachen das erneute Aufrufen der *sell_form.jsp*-Seite mit einer entsprechenden Bemerkung.

- Weiterhin wird geprüft, ob die Felder für Mengen- und Preisangaben entsprechende gültige Zahlenwerte enthalten.

- Es folgt eine Überprüfung, ob der Startpreis größer als der Endpreis ist und ob der Preisrutsch einen kleineren Wert als der Startpreis darstellt.

- Als nächstes wird geprüft, ob das Feld für die Stundenangaben der Auktionsdauer nicht den Wert 23 überschreitet.

- Anschließend prüft das System, ob die Auktionsdauer mindestens eine Stunde beträgt. Wenn das zutrifft, wird die Auktion durch den Aufruf der Methode *addAuction()* der Bean *Auction* angemeldet.

Die Methode *addAuction()* berechnet zusätzliche Parameter für die weitere Verarbeitung der Auktionsdaten. Folgende Parameter werden berechnet und mit den restlichen durch den Benutzer angegebenen Daten zusammen in das Auktionssystem eingetragen:

- *opening*: Stellt die aktuelle Systemzeit dar. Sie gilt als Startzeit der Auktion.

- *duration*: Dieser Parameter repräsentiert die Dauer der Auktion. Sie wird mit Hilfe der eingetragenen Tage und Stunden durch den Benutzer in Minuten berechnet.

- *termination*: Analog zum Parameter *opening*, existiert auch der Parameter *termination*. Er stellt die Systemzeit für das Beenden der Auktion dar und wird durch das Addieren der Dauer und der Eröffnung der Auktion berechnet.

- *totaltime_pricerange*: Diese Zeitspanne stellt die Zeit dar, in der der Preis für die Auktion geändert wird. Das entspricht der *duration*-Zeit weniger dreißig Minuten, da die Letztere für das Handeln zum niedrigsten Preis genutzt wird:

$$totaltime\_pricerange = duration - 30$$

- *diff_highlow_price*: Dieser Parameter entspricht die Differenz zwischen dem Startpreis und Endpreis der Auktion. Das Ergebnis ist eine positive Zahl, da der Startpreis höher als der Endpreis ist.

$$diff\_highlow\_price = price\_begin - price\_end$$

- *number_price_reductions*: Stellt die Anzahl der Preissenkungen für den Artikel dar. Sie enthält die Differenz zwischen den höchsten und den niedrigsten Preis dividiert durch den Preisrutschwert, der durch den Benutzer angegeben wurde.

$$number\_price\_reductions = diff\_highlow\_price/pricerange$$

- *timelag_pricereduc*: Dieser Parameter enthält die Zeitspanne zwischen zwei Preissenkungen. Er wird im System ermittelt, indem die Zeitspanne für die Preissenkung durch die Anzahl der Preissenkungen dividiert wird:

$$timelag\_pricereduc = totaltime\_pricerange/number\_price\_reductions$$

Katalog des Auktionssystems

Das Auktionssystem verfügt über einen Katalog. Er listet alle Hauptkategorien auf, wie auf Abbildung 3.17 auf Seite 120 angezeigt. Diese Daten werden aus der Datenbank ausgelesen, im XML-Format vorbereitet und ins HTML-Format überführt und dargestellt. Die Seite *prodtype.jsp*[78], welche aus der Abbildung 3.17 auf Seite 120 ersichtlich ist, ist für die Durchführung dieser Aufgaben verantwortlich. Jede Kategorie wird hier mit einem Link zu der Seite *prodsubtype.jsp*[79] versehen. Beim Auswählen der Kategorie wird diese Seite aufgerufen und gleichzeitig der Primärschlüssel der Kategorie als Parameter übergeben. Die Seite *prodsubtype.jsp* stellt, wie auf Abbildung 3.18 auf Seite 121 ersichtlich, die ausgewählten Kategorien und ihre zugehörigen Unterkategorien dar.

Dabei werden zur letzten Unterkategorie die verfügbaren laufenden Auktionen aufgezählt und in runden Klammern dazu die Auktionsanzahl eingetragen. Falls zur jeweiligen Kategorie laufende Auktionen verfügbar sind, wird der Kategoriename mit einem Link zur Seite *productlist.jsp* versehen. An diese Seite

[78] Siehe Supplementum, Listing B.2 auf Seite 12.
[79] Siehe Supplementum, Listing B.3 auf Seite 13.

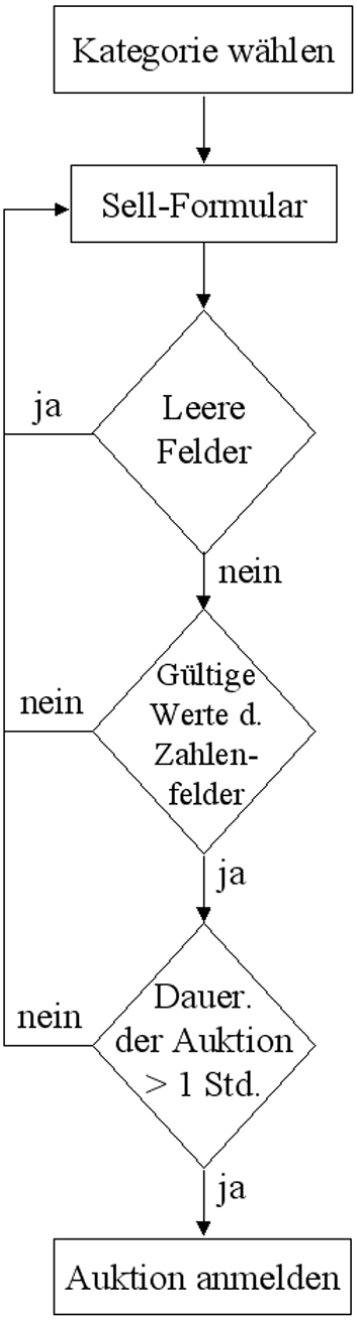

Abbildung 3.16: Anmeldung einer Auktion

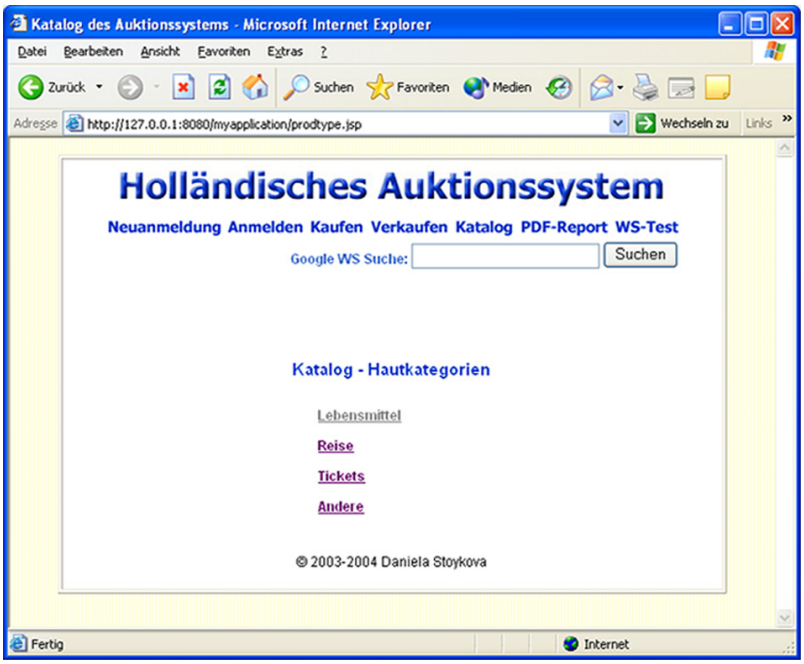

Abbildung 3.17: Katalog des Auktionssystems

wird auch der Primärschlüssel der Unterkategorie als Parameter weitergegeben. Sie stellt alle zur entsprechenden Unterkategorie gehörenden Auktionen in Tabellenform dar.

Abbildung 3.19 auf Seite 122 zeigt eine Liste der verfügbaren Auktionen zu einer beliebigen Unterkategorie an. Hier werden zu jeder Auktion der Name, der Preis, die Anzahl abgegebener Gebote und die verbleibende Zeit der Auktion zur Verfügung gestellt. Der Name der Auktion wird mit einem Link zur Seite *product.jsp* versehen. Weiterhin wird dieser JSP-Seite auch der Primärschlüssel der Auktion als Parameter weitergegeben.

Die JSP-Seite einer beliebigen Auktion ist in Abbildung 3.20 auf Seite 123 dargestellt. Diese Seite stellt weitere Informationen zum Artikel zur Verfügung wie Beginn und Ende der Auktion, Anzahl der bereits getätigten Gebote und Link zu den Gebotsinformationen, wenn mindestens ein Gebot getätigt wurde. Aus dieser Seite aus kann die Abgabe eines Gebots vorgenommen werden.

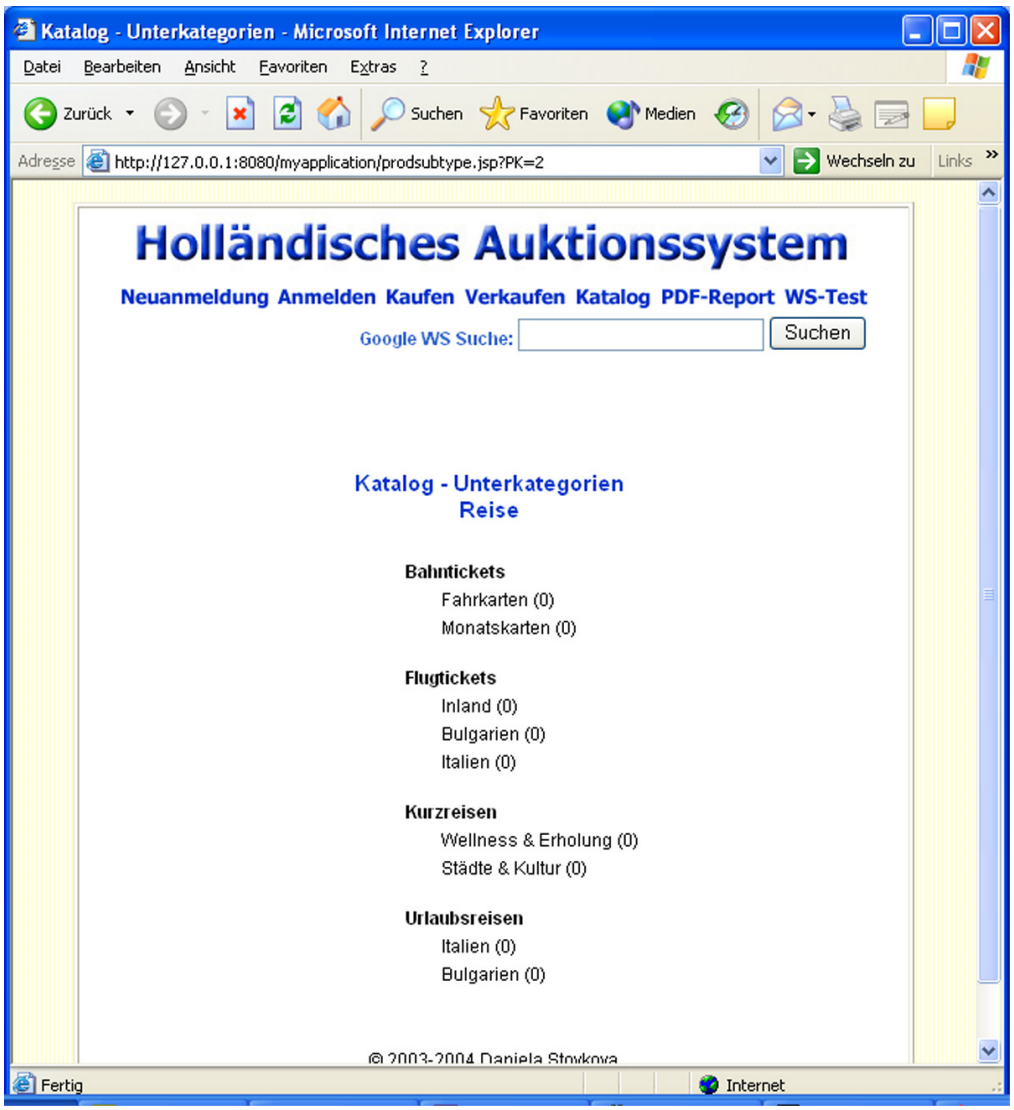

Abbildung 3.18: Unterkategorien des Katalogs

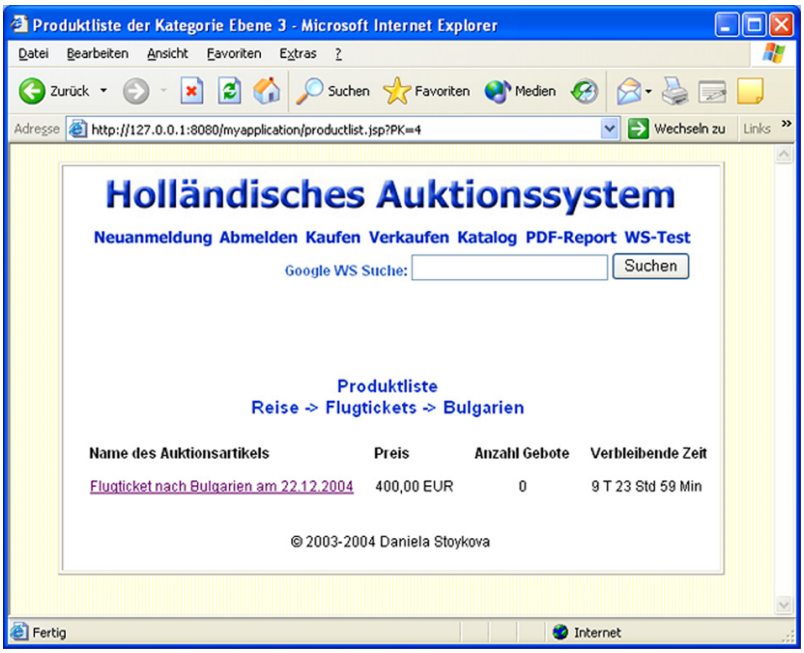

Abbildung 3.19: Liste der verfügbaren Auktionen zu einer Unterkategorie

Gebote für einen Artikel abgeben

Die Abgabe eines Gebots erfolgt durch das Auswählen des Buttons *Bieten*[80] auf der Seite *product.jsp*. Dieser Prozess wird durch die Seiten

- *bid.jsp*[81],
- *bid_form.jsp*[82] und
- *complete_bid.jsp*[83].

ausgeführt.

Die Seite *bid.jsp* prüft, ob der Benutzer angemeldet ist. Die Seite *bid_form.jsp* stellt das Formular zur Abgabe des Gebots zur Verfügung und die Seite *complete_bid.jsp* führt die Abgabe des Gebots aus. Der Prozess wird vom Button

[80] Siehe Abbildung 3.20 auf Seite 123.
[81] Siehe Supplementum, Listing B.9 auf Seite 17.
[82] Die Seite *bid_form.jsp* besteht aus einem Formular mit Felder für die Anzahl der zu ersteigernden Artikel und Gebotshöhe zur Abgabe des Gebots.
[83] Siehe Supplementum, Listing B.10 auf Seite 18.

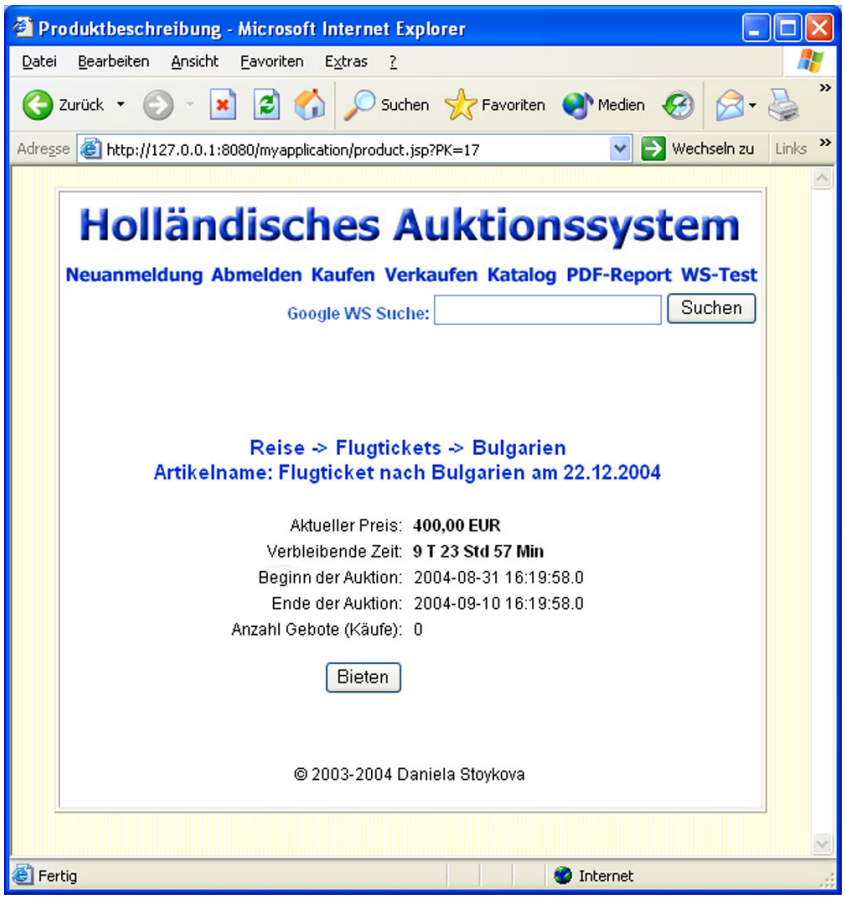

Abbildung 3.20: Produkt-Darstellung des Systems

Bieten der Seite *product.jsp* ausgelöst.[84]. Er wird durch das Ablaufdiagramm in der Abbildung 3.21 auf Seite 126 veranschaulicht. Zunächst wird geprüft, ob der Benutzer am System angemeldet ist. Falls keine Anmeldung vorliegt und damit der Benutzer im System nicht bekannt ist, wird die Seite *login.jsp* mit dem Anmeldeformular aufgerufen, bevor das Gebot abgegeben werden kann. Nach der Anmeldung des Benutzers wird erneut die Seite *product.jsp* angezeigt.

Die benötigten Beans für die Gebotsabgabe sind

- *CustomerSession*[85] und
- *Bid*[86].

Nach dem Betätigen des Buttons *Bieten* auf der Seite *product.jsp* wird die *bid.jsp*-Seite aufgerufen. Sie benutzt die Bean *CustomerSession*. Die Sitzung für den angemeldeten Benutzer wird im *session*-Objekt unter dem Namen *user* abgelegt. Der Quelltext der Seite *bid.jsp*[87] stellt zunächst in der Zeile 2 das *CustomerSession*-Objekt der Sitzung zur Verfügung. Falls keine Anmeldung während der Sitzung stattgefunden hat, ist das *CustomerSession*-Objekt innerhalb der Session leer und besitzt damit den Wert *null*. Das *String*-Objekt *show*, das den Namen der folgenden Seiten halten soll, wird in Zeile 7 auf *login.jsp* gesetzt. In der Zeile 8 wird zuerst geprüft, ob das Objekt *CustomerSession* den Wert *null* besitzt. Wenn das der Fall ist, wird das *request*-Attribut *text* für die Anmerkung der folgenden (*login.jsp*) Seite in Zeile 9 gesetzt. Im *request*-Attribut *forward_page* wird die Seite *bid_form.jsp* in der Zeile 11 gehalten. Das ist die Seite, die nach einem erfolgreichen Anmelden am System aufgerufen wird. Wenn das *CustomerSession*-Objekt nicht den Wert *null* besitzt, wird zuerst das Benutzerkonto verifiziert. Im Falle eines ungültigen Benutzerkontos wird erneut die Seite *login.jsp* mit der Anmerkung für ungültige Benutzerangaben aufgerufen.[88] Das Attribut *forward_page* wird mit dem Wert *bid_form.jsp* versehen (siehe Zeile 17). Wenn das Benutzerkonto gültig ist, wird die *bid_form.jsp* aufgerufen (siehe Zeile 19). Diese Seite enthält das Formular zur Abgabe eines Gebots. Es erwartet die obligatorischen Werte für die gewünschte Menge und Wert des Gebots. Nach der Auswahl des Buttons *Weiter* wird das Gebot an die Seite *complete_bid.jsp* weitergegeben. Das *String*-Objekt *show* ist in dieser Seite auch präsent[89] und wurde auf *bid_form.jsp*

[84]Siehe Abbildung 3.20 auf Seite 123.
[85]Siehe Supplementum, Listing B.13 auf Seite 29.
[86]Siehe Supplementum, Listing B.12 auf Seite 26.
[87]Siehe Supplementum, Listing B.9 auf Seite 17.
[88]Siehe dazu auch Abbildung 3.21 auf Seite 126.
[89]Siehe Supplementum, Zeile 17 in Listing B.10 auf Seite 18.

gesetzt. Bevor das Gebot ausgeführt und in das System eingetragen wird, werden folgenden Aufgaben durchgeführt[90]:

- Es wird geprüft, ob obligatorische Felder leer gelassen wurden.
- Weiterhin wird die Gültigkeit der Werte geprüft. Sowohl der Wert des Gebots als auch die gewünschte Menge müssen positive Werte sein.
- Der Wert der gewünschten Menge soll nicht die verfügbare Menge überschreiten.
- Der Wert des Gebots soll nicht kleiner als der aktuelle Preis für den Artikel sein.

Wenn die bereits aufgelisteten Anforderungen für die Werte des Formulars erfüllt sind, wird die Abgabe des Gebots abgeschlossen. Bei nichterfüllten Anforderungen für die Formularfelder wird erneut die Seite *bid_form.jsp* mit dem Formular für das Gebot aufgerufen.[91] Eine erfolgreiche Gebotsabgabe ruft die Seite *bid_success.jsp* auf. Sie benachrichtigt den Benutzer über die erfolgreiche Gebotsabgabe und zeigt den zu ersteigernden Artikel an.

Abmelden des Benutzers vom System

Das Abmelden eines Benutzers beendet die Sitzung innerhalb des Auktionssystems. Der Prozess wird mit dem Quelltext auf Listing 3.13 durchgeführt. Beim Abmelden wird zuerst in der Zeile 1 geprüft, ob das *session*-Objekt den Wert *null* besitzt. Wenn es einen gültigen Wert hat, wird das *CustomerSession*-Objekt des angemeldeten Benutzers in den Zeilen 2-3 ausgelesen. Wenn das *CustomerSession*-Objekt nicht den Wert *null* entspricht, wird das *HashMap*-Objekt mit den Sitzungsobjekten aufgerufen und das *CustomerSession*-Objekt daraus entfernt.

Listing 3.13: Auszug aus der Seite *logout.jsp*

```
1 if(session!=null){
2     CustomerSession cust_session =
3             (CustomerSession)session.getAttribute("user");
4     if(cust_session != null){
5         HashMap hash =
6             (HashMap)application.getAttribute("monitor");
7         hash.remove(cust_session);
```

[90] Siehe Abbildung 3.21.

[91] Entsprechende Anmerkungen geben dem Benutzer Hinweise auf die obligatorischen Felder.

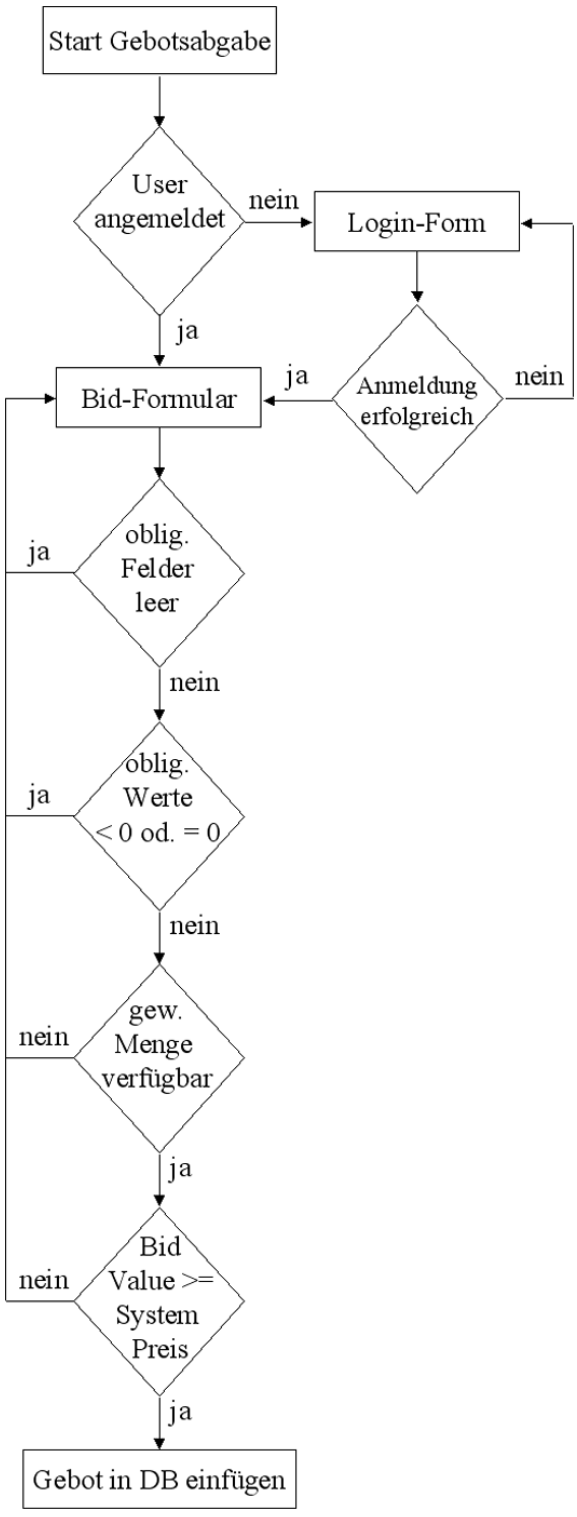

Abbildung 3.21: Abgabe des Gebots im System

```
8           session.invalidate();
9           out.println("<b>Vielen Dank fuer Ihren Besuch!
10              <br/>Sie wurden im dem System abgemeldet!</b>");
11      }else{
12          out.println("Sie waren im System nicht angemeldet!");
13      }
14  }
```

Der Datenbank-Zugriff mit der Klasse Dm

Die Datenbankverbindung ist eine wichtige Komponente innerhalb der Webanwendung. Sie muss vorhanden sein, um die Nutzdaten aus der Datenbank gewinnen zu können. Für diese Aufgabe ist die Klasse *Dm* des Pakets *myapplication.database* zuständig. Im Konstruktor der Klasse *Dm* wird die Methode *jbInit()*[92] aufgerufen, welche auf Listing 3.14 dargestellt ist.

Listing 3.14: Initialisierung des *Datenbank*-Objektes

```
1  private void jbInit () throws Exception {
2      con = null;
3      boolean DriverLoaded = false;
4      org.firebirdsql.jdbc.FBWrappingDataSource wds = null;
5
6      if (!DriverLoaded){
7        try{
8         wds =
9           new org.firebirdsql.jdbc.FBWrappingDataSource();
10        DriverLoaded = true;
11       }catch (ResourceException e){
12           e.printStackTrace();
13       }
14       wds.setDatabase("//127.0.0.1:3050/D:\\AUCTION.GDB");
15       wds.setUserName("SYSDBA");
16       wds.setPassword("masterkey");
17       wds.setIdleTimeoutMinutes(30);
18       wds.setPooling(true);
19       wds.setMinSize(5);
20       wds.setMaxSize(30);
21       try{
22           wds.setLoginTimeout(10);
```

[92] Vgl. http://www.ibphoenix.com/main.nfs?a=ibphoenix&s=1091143881:39644&l=;FAQS;NAME='JayBird'

```
23        }catch (SQLException ex){
24            ex.printStackTrace();
25        }
26       }
27
28      try{
29          con = wds.getConnection();
30      }catch (SQLException e){
31          e.printStackTrace();
32      }
33 con.getTransactionIsolation();
34 }
```

In der Zeile 4 wird zuerst der JDBC-Driver *org.firebirdsql.jdbc.FBDriver* bekanntgegeben. Wenn der Datenbanktreiber vorhanden ist, wird eine Verbindung aufgebaut, welche mit Hilfe der Methode *closeDatabase()* wieder geschlossen werden kann:

```
public void closeDatabase () throws Exception {
   con.close();
}
```

Die Klasse *Dm* stellt weiterhin Methoden zur Verfügung, mit deren Hilfe bei einer bestehenden Verbindung Abfragen und Anweisungen auf die Datenbank ausgeführt werden können.

Die Verarbeitung der Zeit- und Kalenderdaten

Für die Verarbeitung der Zeit- und Kalenderdaten wurden zwei Klassen erstellt:

- *Date*[93] und
- *Time*[94].

Die Zeitverarbeitung ist ein wichtiger Aspekt bei der Datenverarbeitung innerhalb des Auktionssystems. Diese findet beispielsweise bei

- der Anmeldung einer neuen Auktion,
- dem Aufruf der Produktliste einer Unterkategorie und

[93] Siehe Supplementum, Listing B.20 auf Seite 55.
[94] Siehe Supplementum, Listing B.22 auf Seite 62.

- dem Aufruf der Produktdarstellung

statt.

Bei der Anmeldung einer neuen Auktion wird die Startzeit zum Einfügen in die Datenbank benötigt. Diese ist die jeweilige Zeit der Anmeldung der Auktion und wird mit der Methode *getCurrentTime()* der Klasse *Date*[95] ermittelt. Die Endzeit der Auktion kann mit Hilfe der Startzeit und der durch den Benutzer angegebenen Dauer der Auktion berechnet werden. Die Letztere wird in Minuten berechnet. Dazu wird die Methode *getTimeInMinutes()* der Klasse *Time*[96] benutzt. Diese Methode erwartet die Tage und die Stunden und liefert die Gesamtdauer der Zeit in Minuten. Die Endzeit der Auktion wird mit der Methode *getAuctionTerminationTime()* der Klasse *Date* ermittelt.

In der Produktliste und bei der Produktdarstellung wird die verbleibende Zeit der Auktionen angezeigt. Diese Daten ändern sich jede Minute und werden in der Datenbank festgehalten. Die Aktualisierung der Daten wird mit einem Thread[97] realisiert. Der Thread ist ein Programm, welches neben anderen Prozesse in der Anwendung ausgeführt wird und bestimmte Aufgaben durchführt. Der Thread wird in der Klasse *UpdateAuctionStatus*[98] implementiert und erweitert die Klasse *Thread*. Er aktualisiert die Daten aller laufenden Auktionen. Dafür wird in der Zeile 16 die Methode *updateAuctionData()* der Bean *Auction*[99] aufgerufen. Diese benötigt die verbleibende Zeit einer Auktion vom Aufrufzeitpunkt im System aus gesehen. Sie wird mit der Methode *getLeftAuctionDuration_Mins()* der Klasse *Date*[100] berechnet, welche den Primärschlüssel des Artikels erwartet. Mit Hilfe der letzten Methode wird zuerst der Kalenderdatum und die Endzeit der Auktion berechnet. Um anschließend die verbleibende Zeit in Minuten zu berechnen, wird die Methode *minutesBetween()* der Klasse *Date*[101] genutzt. Sie erwartet die aktuelle Zeit und die bereits berechnete Dauer der Auktion.

[95] Siehe Supplementum, Zeilen 12-18 in Listing B.20 auf Seite 55.
[96] Siehe Supplementum, Zeilen 5-10 in Listing B.22 auf Seite 62.
[97] thread: (engl.) Faden
[98] Siehe Supplementum, Listing B.23 auf Seite 63.
[99] Siehe Supplementum, Zeile 261-321 in Listing B.11 auf Seite 20.
[100] Siehe Supplementum, Zeile 151-160 in Listing B.20 auf Seite 55.
[101] Siehe Supplementum, Zeilen 162-166 in Listing B.20 auf Seite 55.

Kapitel 4

Weitere Implementierungsaspekte in Hinsicht auf XML

> Tim Bray: *„XML is more successful than any of us who designed ever thought it would be.“*
> *Becker (2004)*

Der Schwerpunkt dieses Kapitels liegt in der Implementierung des XML-Datenaustausches innerhalb der Anwendung. Es wird dabei auf relevante Standards und auf die konkrete Programmierung eingegangen. Die Implementierungsschritte und Grundsatzideen in diesem Kapitel können auch auf andere Webanwendungen übertragen werden und so die beschriebenen Vorteile von XML[1] nutzen.

Der erste Abschnitt dieses Kapitel beschäftigt sich mit den Grundlagen von XML, gefolgt vom zweiten Abschnitt, in dem die Möglichkeiten zum Prozessieren von XML-Daten behandelt werden. Im dritten Abschnitt wird auf die Spezifikation XPath eingegangen, mit dessen Hilfe im vierten Abschnitt das Formatieren von XML-Dokumenten behandelt wird. Es wird an dieser Stelle die Transformation von XML-Dokumenten mittels XSLT und die Formatierung von XML-Daten mittels XSL:FO beschrieben. Weiterhin wird im fünften Abschnitt auf die Möglichkeiten zur Beschreibung von XML-Dokumenten eingegangen und im sechsten Abschnitt auf einige Sicherheitsstandards für XML-Technologien. Dabei wird auf die innerhalb der Anwendung zur Verfügung gestellten Applets zum Signieren von XML-Daten näher eingegangen. Der letzte Abschnitt wird den zur Verfügung stehenden XML-Spezifikationen der XML-Familie gewidmet.

[1] Siehe Seite 43.

4.1 Grundsätzliches

Im Kapitel 2.3.2 wurde kurz die Geschichte der wichtigsten Auszeichnungssprachen dargestellt und ihre Beziehungen in der Abbildung 2.6 auf Seite 42 veranschaulicht. Der XML-Standard von W3C (2003) befindet sich derzeit in seiner Version 1.1.[2] In diesem Abschnitt werden die Bestandteile eines XML-Dokuments besprochen und anhand von Beispielen erläutert.

4.1.1 Aufbau von XML-Dokumenten

Die Bestandteile eines XML-Dokuments sind:

- der Prolog und
- genau ein XML-Dokument-Element, welches in der Regel weitere Elemente enthält.

Der Prolog des Dokuments besteht aus einer XML-Deklaration, Dokumenttyp-Deklaration, Kommentaren und Processing Instructions. (Rottach und Groß, 2002, S. 11) Das Dokument-Element kann aus weiteren Elementen bestehen, die entweder Knoten oder Zeichendaten enthalten.

In einem XML-Dokument wird zwischen *Metadaten (Markupdaten)* und *Nutzdaten* unterschieden. Die Metadaten haben die Aufgabe, die Nutzdaten mit Markups zu versorgen. Sie dienen damit zur Beschreibung der Nutzdaten und können beispielsweise ein Programm bei der Verarbeitung des XML-Dokuments unterstützen.

XML-Deklaration

Die XML-Deklaration darf nur einmal und an erster Position im Dokument auftreten. (Harold und Means 2002, S. 23; Mintert 2002, S. 51)
Sie besitzt die Form:

```
<?xml version="1.0" encoding ="..." standalone="..."?>
```

Die XML-Deklaration ähnelt einer *Processing Instruction*. Der einzige Unterschied besteht darin, dass die *Processing Instruction* an die Anwendung weitergereicht wird und die XML-Deklaration allein vom XML-Prozessor verarbeitet wird. (Rottach und Groß, 2002, S. 11) Die Parameter der XML-Deklaration

[2] Vgl. Bray (2004).

sind in der Tabelle 4.1 veranschaulicht und erfolgen in der Reihenfolge *version*, *encoding* und *standalone*.

Tabelle 4.1: Parameter der XML-Deklaration

Parameter	obligatorisch	Bedeutung
version	ja	enthält den Wert einer gültigen XML Versionsnummer
encoding	nein	enthält den Wert einer Zeichensatz-Kodierung
standalone	nein	enthält den Wert *ja* oder *nein* und gibt damit bekannt, ob eine externe Ressource wie DTD notwendig ist

Dokumententyp-Deklaration

Die Dokumententyp-Deklaration verweist auf eine externe Dokumentdatei, mit der eine Validierung des XML-Dokuments durchgeführt werden kann. Sie tritt nur einmal innerhalb des Prologs auf und kann wie folgt aussehen:

```
<!DOCTYPE Product SYSTEM "product.dtd">
```

Mit dem Schlüsselwort SYSTEM wird auf den Ort der DTD-Datei verwiesen. Der Ort einer außenstehenden DTD-Datei wird dagegen mit dem Schlüsselwort PUBLIC bekanntgegeben.

Kommentare

Ein XML-Dokument kann Kommentare enthalten, welche mit den Zeichen <!-- eingeleitet und mit --> beendet werden. Sie können an jeder Stelle im XML-Dokument auftreten – außer innerhalb vom Markup. Kommentare können zur Verdeutlichung des Inhalts dienen. Sie können beliebige Zeichen enthalten mit Ausnahme der Einführungszeichen <!-- und der Abschlusszeichen -->.

Processing Instructions

Processing Instructions dürfen auch mehrmals im Prolog des XML-Dokuments erscheinen und geben der Anwendung Hinweise zur Bearbeitung des XML-Dokuments. Sie sind Bestandteile, die über die Grenzen der Definition für die Bestandselemente eines Dokuments springen. (Rottach und Groß, 2002,

S. 12) *Processing Instructions* haben manchmal eine nützliche Rolle in einem solchen *Markup gerichteten* Dokument. Ein Beispiel dafür ist die Einbindung eines CSS-Stylesheets, das beispielsweise für Definitionen der Darstellung des Inhalts dient:

```
<?xml-stylesheet href="....css" type="text/css"?>
```

Elemente und Attribute

Elemente sind die zentralen Bestandteile eines XML-Dokuments. Das Element auf der obersten Ebene ist obligatorisch und wird als *Dokument-Element* oder *root-Element* bzw. *Wurzelelement* bezeichnet. Er enthält meistens noch weitere Elemente, Attribute oder Zeichendaten.

Die Bezeichner, die die Namen der XML-Elemente und XML-Attribute darstellen, beginnen mit einem Buchstaben oder einem Unterstrich. Sie werden von weiteren Buchstaben, Ziffern, Unterstrichen, Bindestrichen, Punkten oder Doppelpunkten gefolgt. Dabei werden die Doppelpunkte ausschließlich dafür genutzt, auf einem Namensraum (engl. *namespace*) zu verweisen, dem das Element gehört. Auf XML-Namensräume wird im Kapitel 4.1.3 eingegangen.

Ein XML-Element kann wie folgt aussehen:

```
<product kat_name = "Fahrkarten">10 HSB Fahrkarten</product>
```

Mintert (2002, S. 57 ff.) unterscheidet bei einem Element drei wichtige Begriffe:

- Element,
- Typ des Elements und
- Element-Tag.

Das obige Element stellt das *product*-Element eines XML-Dokuments dar und ist vom Typ *product*. Dabei bezeichnet der Typ des Elements seine Art und kann nur einmal im Dokument auftreten. Das Element im Gegensatz zu seinem Typ kann beliebig oft als Vertreter seines Typs vorkommen. Anschließend ist noch das Element-Tag vom Element zu unterscheiden. Ein Element kann aus einem Start- und End-Tag bestehen. Der Inhalt der Elemente befindet sich zwischen den beiden Tags. Der Start-Tag beginnt mit einem <-Zeichen, wird vom Namen des Elements gefolgt und endet mit einem >-Zeichen. Der End-Tag hat denselben Aufbau bis auf das /-Zeichen, welches nach dem <-Zeichen

platziert wird.

Der Inhalt jedes XML-Elements besteht aus weiteren Elementen oder Zeichendaten oder beidem. Ein Element kann auch Attribute enthalten. Im obigen Beispiel enthält das Element *product* das Attribut *kat_name*. Das Attribut wird im Start-Tag eines Elements mit einem Leer-Zeichen vom Tag-Namen getrennt und tritt damit innerhalb seines Markups auf. Das Attribut eines Elements besteht aus einem Bezeichner, gefolgt von einem Gleichheitszeichen (=). Nach dem Letzteren wird der Wert des Attributs in Einführungszeichen eingegeben.

4.1.2 Wohlgeformte und gültige XML-Dokumente

XML-Dokumente müssen bestimmten Regeln gehorchen. In einer weiteren Datei können auch zusätzliche Regeln für den Aufbau der Dokumente angegeben werden. Danach können

- wohlgeformte sowie
- wohlgeformte und gültige

XML-Dokumente unterschieden werden.

Ein XML-Dokument ist *wohlgeformt*, wenn es folgende XML-Bedingungen erfüllt (Bitzer 2003, S. 30 ff.; Mintert 2002, S. 45 ff.; Rottach und Groß 2002, S. 8):

- Das XML-Dokument besteht aus genau einem Wurzelknoten (*root*-Element), der beliebig viele Elemente enthalten kann. Diese können wiederum aus weiteren Elementen oder Zeichendaten bestehen.
- Alle Elemente müssen geschlossen werden, d.h. zu jedem öffnenden XML-Tag gehört auch ein schließendes XML-Tag mit dem selben Namen. Leere Elemente werden mit einem „/“ vor der abschließenden Klammer gekennzeichnet.
- Das Überlappen von Tags verschiedener Elemente ist nicht erlaubt.
- Die einzelnen Tags können eindeutige Attribute enthalten, deren Werte in Anführungszeichen eingeschlossen werden müssen.
- Elementnamen müssen aus einem gültigen Zeichensatz bestehen. XML-Dokumente, für die der Zeichensatz UTF-8 definiert wurde, dürfen beispielsweise keine Umlaute enthalten.

Ein XML-Dokument ist dagegen *gültig*, wenn es wohlgeformt ist und wenn es den Regeln gehorcht, die in einem DTD- oder XSD[3]-Stylesheet definiert sind. (Bray, 2004) Die Validierung eines XML-Dokuments ist sehr zeitaufwändig und die Durchführung sollte daher genau überlegt werden. Die Erstellung solcher Dokumente ist vorteilhaft, wenn die Struktur der eigenen XML-Dokumente z.B. einem Partner weitergegeben werden, um den Datenaustausch zu ermöglichen.

4.1.3 XML-Namensraum

Der Vorteil von XML, den Markup eigener Dokumente selbst zu definieren, bringt auch Nachteile mit sich, nämlich die Gefahr von gleichen Bezeichnerdefinitionen. Das trägt dazu bei, dass zwei gleichnamige Markups nicht mehr eindeutig sein können. Um dieser Gefahr zu entkommen, wurden die sogenannten Namensräume (engl. namespace) eingeführt.

> „Ein XML-Namespace stellt eine Ansammlung von Namen und Bezeichnern für Elementtypen und Attributnamen dar, die durch eine IRI-Referenz identifiziert werden.“ (Großwendt, 2002, S. 37)

Ein IRI (International Resource Identifier) existiert seit XML 1.1 und wird anstatt einem URI benutzt. Es erlaubt Zeichen, die in einem URI nicht benutzt werden können.

Eine Gruppe von Elementen wird einem Namensraum zugeordnet. Dies gewährleistet die Eindeutigkeit im Bezug auf ein Element, da zu diesem der genaue Namensraum angegeben wird. Das Beispiel in Listing 4.1 wendet einen XML-Namensraum an.

Listing 4.1: XML-Namespaces

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <Bestellung>
4    <Kunde xmlns:kunde="http://www.xy.de/kunde"
5                                          kundennr="123456">
6      <Anrede>Frau</Anrede>
7      <Name>Schmitt</Name>
8      <Vorname>Petra</Vorname>
9      <Strasse>Kundenstr. 1</Strasse>
10     <PLZ>11111</PLZ>
```

[3]Extensible Stylesheet Definition

```
11      <Ort>Kundenort</Ort>
12    </Kunde>
13    <Atrikelbestellung xmlns:artikel=
14                              "http://www.xy.de/artikel">
15      <Artikel artikelnr="111">
16        <Name>Zahnpasta</Name>
17      </Artikel>
18    </Artikelbestellung>
19  </Bestellung>
```

Hier wird in der Zeile 4 dem Element *Kunde* der Namensraum

`http://www.xy.de/kunde`

zugewiesen. Sein Gültigkeitsbereich bezieht sich auf das komplette Element *Kunde*. Für das Element *Artikelbestellung* wurde ein zweiter Namensraum definiert, welcher sich entsprechend auf das Element *Artikelbestellung* bezieht und innerhalb dieser gültig ist.

In dieser Arbeit werden die Namensräume beispielsweise bei der Transformation der XML-Daten im Kapitel 4.4.1 für die Darstellung im Browser besondere Relevanz bekommen.

4.1.4 Bearbeitung von XML-Dokumenten

Es werden immer häufiger Daten im XML Format abgelegt. Die Gründe dafür sind die bessere Darstellung der Struktur von Daten und damit die Abhängigkeit der Daten innerhalb eines Dokuments. Hinzu kommt noch, dass XML ein Format ist, das sowohl von Mensch als auch von Maschine gelesen werden kann und in vielen verschiedenen Arten und Medien dargestellt werden kann.

Die Entstehung von XML und weiteren XML-basierten Sprachen hat die Entwicklung von mehreren Tools für die Bearbeitung von XML-Daten vorangetrieben. Bitzer (2003, S. 44 ff.) legt die Bearbeitungstools in Schichten wie aus der Abbildung 4.1[4] ersichtlich und ordnet diese in Basistools bis hin zu integrierten Softwarepaketen (engl. *frameworks*) ein.

Die unterste Ebene ist vom Editor besetzt. Dies kann ein einfacher Text-Editor sein, der die XML-Daten bearbeitet. Bitzer (2003, S. 34) unterscheidet zwischen Dokumentorientierten und Datenbankorientierten XML-Daten. Analog dazu stellt Bitzer (2003, S. 45 ff.) auch verschiedene Editoren vor:

[4]Die Abbildung wurde durch den Autor leicht modifiziert.

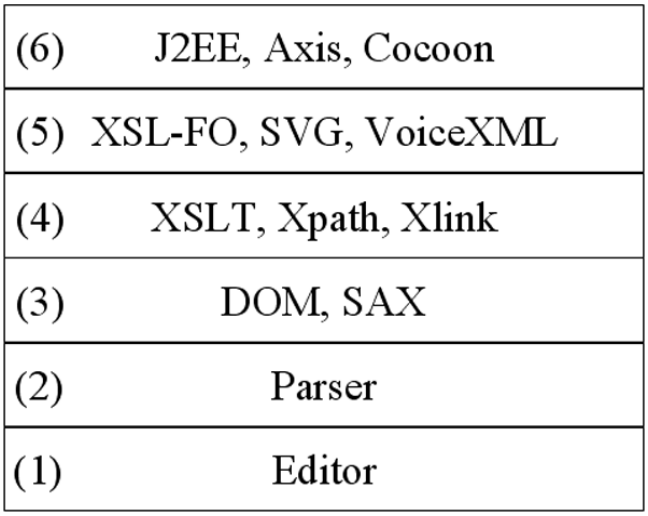

Abbildung 4.1: Schichtenmodell der Tools für die XML-Verarbeitung, Quelle: Bitzer (2003, S. 44)

- die *Tag-Orientierten Editoren*, die sich zum Editieren von Fließtextdokumenten eignen und
- die *Struktur-Editoren*, die sich zur Verarbeitung von XML-Daten ähnlich wie bei Datenbank-Daten eignen.

Auf der zweiten Ebene des Schichtenmodells befindet sich ein unentbehrliches Werkzeug für die Bearbeitung von XML-Dokumenten: Das ist der *XML-Parser*, der auch als *XML-Prozessor* bezeichnet wird. Auf XML-Parser wird im Kapitel 4.2.1 näher eingegangen.

Die dritte Ebene umhüllt die Schnittstelle zwischen dem Parser und den XML-Daten. Dabei wird zwischen den ereignis[5]- und baumorientierten[6] Parsern unterschieden. Dieses Thema wird ab Seite 140 im Kapitel 4.2.1 behandelt.

Die vierte Ebene mit XML-Tools ist für die *Auswahl*, für das *Verlinken* und für die *Transformation* der Daten zuständig. Dazu werden Sprachen wie XPath, XLink[7] und XSLT angewendet. Spezifikationen dieser Ebene werden in diesem Kapitel angegangen und anhand von Beispielen aus der Anwendung näher be-

[5]Siehe *Ereignisorientiertes Parsen mit SAX* auf Seite 140.
[6]Siehe *Baumorientiertes Parsen mit SAX* auf Seite 140.
[7]http://www.w3.org/TR/xlink/

schrieben. So beschäftigen sich Kapitel 4.3 mit der Spezifikation XPath und Kapitel 4.4.1 mit den Möglichkeiten und mit der Anwendung von der Sprache XSLT.

Auf der weiteren, fünften Ebene befinden sich weitere Sprachen, die die Umsetzung speziellerer Aufgaben ermöglichen. In dieser Ebene fügt Bitzer (2003) Sprachen wie XSL-FO, SVG und VoiceXML ein. Die Spezifikation SVG[8] ermöglicht die Darstellung von Grafiken mit der XML-Technologie, VoiceXML[9] soll dagegen das Erstellen von sprachbezogenen Benutzerschnittstellen erinräumen. XSL-FO dient zur Formatierung von XML-Daten. Die von dieser Sprache zur Verfügung gestellten Möglichkeiten werden anhand von Beispielen der Anwendung im Kapitel 4.4.2 dargestellt.

Auf der obersten Ebene sind integrierte Softwarepakete platziert, die die Tools der unteren Ebenen kombinieren und diese in einem Paket zur Verfügung stellen. Beispiele für Frameworks sind die Softwarepakete *Cocoon*[10] und *Axis*[11]. Sie lassen sich unter dem Tomcat Server installieren und verwenden. *Cocoon* besteht beispielsweise aus Paketen wie XML-Parser und XSLT-Prozessor und unterstützt durch deren Zusammenarbeit die einfache Transformation von Daten in Anwendungen.

4.2 Prozessieren von XML-Daten

4.2.1 XML-Parser

Ein XML-Parser ist ein Programm, welches zur Bearbeitung von XML-Dokumenten dient. Innerhalb der Anwendung befindet er sich zwischen den XML-Daten und der Anwendung. Er prüft die XML-Inhalte auf deren Syntax und kann zusätzlich auch auf seine Bestandteile zugreifen. Das Ergebnis der Verarbeitung stellt er der Anwendung zur Verfügung. Die Schnittstelle des Parsers ist vom W3C standardisiert. Es existieren zwei Programmieransätze zur Verarbeitung eines XML-Dokuments, nämlich

- ereignisorientierter und
- baumorientierter Ansatz.

[8] Siehe http://www.w3.org/TR/2003/REC-SVG11-20030114/
[9] Siehe http://www.voicexml.org/
[10] Siehe Seite 47 ff.
[11] Siehe dazu Kapitel 5.4.1 auf Seite 222.

Außerdem existieren Parser, die die Validierung eines XML-Dokuments zulassen und damit die Prüfung nach Wohlgeformtheit und Gültigkeit durchführen.

Es gibt XML-Parser für die meist angewendeten Programmiersprachen wie Java, C, C++, PHP. Die XML-Verarbeitung innerhalb des Prototyps dieser Arbeit wurde mit dem Xerces-Parser[12] von der Apache Software Foundation durchgeführt.

Ereignisorientiertes Parsen mit SAX[13]

Ein ereignisorientiertes API informiert die Anwendung über das Auftreten bestimmter Ereignisse, wie über Anfang und das Ende eines XML-Elements durch direkte Rückfrage (engl. *callback*). Dieser Ansatz benutzt keine internen Strukturen wie z.B. einen Baum mit der Struktur des Dokuments.

SAX (Simple API for XML Access) liegt bereits in der Version 2.0 vor. Das Wesentliche in dieser Version ist die Unterstützung von Namespaces. Weiterhin Unterstützt SAX 2.0 auch die Möglichkeit, bestimmte Eigenschaften des Parsers ein- und auszuschalten und auf definierte Objekte zuzugreifen.

Baumorientiertes Parsen mit DOM

Das *Document Object Model* repräsentiert das Dokument in Form eines hierarchischen Baums bestehend aus Knoten, die genau eine Wurzel besitzen. Der Standard liegt in drei verschiedenen Versionen in Form von Levels vor. Historisch bedingt sollte das DOM Level 1 (Apparao, 1998) Probleme des Dynamic HTML beseitigen.

Das DOM Level 2 (Le Hors, 2001) ist eine Erweiterung des DOM Level 1. Diese Version hat mit dem wachsenden Einsatz des XML-Standard eine grosse Bedeutung gewonnen. DOM Level 2 unterstützt *Namespaces* und stellt weitere Verarbeitungsmöglichkeiten für XML-Dokumente zur Verfügung.

Das DOM Level 3 (Le Hors, 2004) liegt als W3C Recommendation[14] in seiner

[12] http://xml.apache.org/xerces2-j/index.html
[13] Siehe dazu http://www.saxproject.org/
[14] Siehe Seite 41.

Version 1.0 vor[15] und basiert auf dem DOM Level 2. Es ermöglicht das standardisierte Lesen und Schreiben von XML-Dokumenten und den Umgang mit DTD und XML-Schemas.

4.2.2 XML-Datenverarbeitung in der Webanwendung

Der Aufbau und die Struktur der JSP-Seiten in der Webanwendung wurden bereits im Kapitel 3.3.3 ausführlich anhand der Datei *productlist.jsp* beschrieben. Der größte Teil der Seiten besteht aus dynamischen Inhalten, welche im Datenbanksystem gehalten werden. Sie werden in der Auktionssystem-Anwendung selektiert, in das XML-Format überführt und zur Präsentation aufbereitet.

Dieser Abschnitt beschreibt die Generierung der dynamischen XML-Inhalte am Beispiel der bereits erläuterten *productlist.jsp*-Seite. Damit werden die Daten für die Artikelliste zu einer Unterkategorie des Katalogs im XML-Format vorbereitet, in das HTML-Format transformiert und dem Anwender zur Verfügung gestellt.

Die Generierung der XML-Daten wird in der Anwendung mit den Klassen:

- *XmlProductData*[16],
- *XmlDocument*[17] und
- *XmlTool*[18]

realisiert.

Die Klasse *XmlProductData* nutzt die Klasse *XmlDocument*, um die benötigten Daten im XML-Format vorzubereiten. Die Klasse *XmlTool* dient als zusätzliche Hilfsklasse. Sie dient beispielsweise zur erstmaligen Erstellung leerer XML-Dokumente, zum Schreiben eines XML-Dokumentobjektes in eine XML-Datei auf der Festplatte oder zur Transformation eines Dokumentobjektes in HTML-Code.

Die Klasse *XmlProductData* dient als Schnittstelle zwischen den JSP-Seiten und den XML-Daten aus der Datenbank. Sie bedient sich folgender zweier Methoden:

[15] Siehe http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/
[16] Siehe Supplementum, Listing B.18 auf Seite 45.
[17] Siehe Supplementum, Listing B.19 auf Seite 46.
[18] Siehe Seite 160.

- *addParamToProductData()*: Das Gewinnen von Daten innerhalb der Datenbank benötigt in den meisten Fällen Parameter. Bevor ein Dokument erstellt wird, müssen die für das Dokument benötigten Parameter mit einem Wert versehen werden. Mit einem Aufruf der Methode *addParamToProductData()* im System kann dem Parameter ein Wert zugewiesen werden. Dabei müssen der Parametername und der Parameterwert an die Methode übergeben werden. Die Parameternamen haben feste Werte im System und befinden sich in der Tabelle *TCAT_XMLPARAM* im Datenbanksystem. Alle gesetzten Parameter werden in einem Vektor gehalten.

- *getDocumentFromDBData()*: Diese Methode erwartet den Namen der Nachricht, die erstellt werden soll, als Parameter und liefert ein XML-Dokumentobjekt zurück. Dabei wird ein Objekt der Klasse *XmlDocument* erstellt, anschließend die Methode *getDocumentFromDB(String sMessageName, Vector v)* mit dem Namen der Nachricht und deren Parameter aufgerufen, um das Dokument mit den XML-Daten zu erstellen.

Die Zusammenarbeit der XML-bezogenen Klassen mit den JSP-Seiten

Die Seite *productlist.jsp*, anhand dessen die Arbeit der JSP-Seiten mit den XML-bezogenen Klassen erläutert werden soll, liefert eine Liste der verfügbaren laufenden Auktionen einer Unterkategorie des Systems. Die aufgelisteten Auktionen hängen damit von einer bestimmten Unterkategorie ab oder genauer vom Primärschlüssel der Unterkategorie. Dieser Parameter muss im System mit einem Wert versehen werden, um die Daten der laufenden Auktionen ermitteln zu können. Das System ermittelt den Parameterwert über das *request*-Objekt der JSP-Seite. Der Quelltext der *productlist.jsp*-Seite kann aus dem bereits vorgestellten Listing 3.4 auf Seite 94 entnommen werden.

Nach dem Einbinden der Import- und der Bean-Klassen (Zeilen 3, 4, 6-11) in die Seite wird in der Zeile 13 die Methode *setPageNr()* der Bean *Page* ausgeführt, welche die eindeutige aktuelle Seitennummer in der Anwendung festlegt. Mit *setPageNr()* wird die Methode *init(int page_nr)* der Bean *Page* aufgerufen.[19] Sie führt die Initialisierung der Seite mit ihrer eindeutigen Nummer durch und bezieht folgende Daten aus dem Datenbanksystem:

- Titel der Seite,

- Name des zu erstellenden XML-Dokuments mit den Nutzdaten,

[19]Siehe Listing 3.10 auf Seite 101.

Abbildung 4.2: Ergebnis der Prozedur P_PAGEDATA (Aufruf mit Parameter-Wert *12*)

- Benötigter Pfad der XSL-Datei zur Transformation der Daten,
- Name des Parameters zur Erstellung der XML-Datei,
- Name des *request*-Parameters beim Aufruf der JSP-Seite.

Die obengenannten Daten werden mit Hilfe der Prozedur P_PAGEDATA geliefert. Der Aufruf von P_PAGEDATA mit dem Parameter *12*, welches die eindeutige Nummer der aufzurufenden Seite *productlist.jsp* darstellt, liefert beispielsweise das Ergebnis, wie es in der Abbildung 4.2 zu sehen ist.

Der dynamische Inhalt aller JSP-Seiten wird innerhalb der Kommentar-Zeilen

```
<!-- ************** B E G I N  C O N T E N T *************** -->
```

und

```
<!-- ************** E N D  C O N T E N T ****************** -->
```

eingefügt. Die *productlist.jsp*-Seite benötigt für die Zusammenstellung der XML-Daten einen Parameter, nämlich die eindeutige Nummer der ausgewählten Unterkategorie. Dies veranlasst den Aufruf der Seite *xmldoc_withprms.jsp*. Der Quelltext der Seite ist auf Listing 4.2 dargestellt.

Listing 4.2: Die *import*-Seite *xmldoc_withprms.jsp*-Seite

```
1 <%@ page import="java.util.Vector" %>
2 <%
3 XmlProductData xmldata = new XmlProductData();
```

```
4 XmlTool xmltool = new XmlTool();
5 String[] prm = new String[2];
6 Vector v = new Vector();
7 v = currentpage.getDocParams();
8 for(int i = 0; i < v.size(); i++){
9     prm = (String[])v.get(i);
10    xmldata.addParamToProductData(prm[0] ,
11                          request.getParameter(prm[1]));
12 }
13 Document xmldoc =
14         xmldata.getDocumentFromDBData(currentpage
15                                       .getXmlDocName());
16 String path = "http://" +
17         request.getRemoteAddr()+ ":" +
18         String.valueOf(request.getServerPort()) +
19         request.getContextPath()+
20         currentpage.getTransformFilePath();
21 String html = xmltool.transform(path,xmldoc);
22 out.println(html);
23 %>
```

In der ersten Zeile wird zunächst die *Vector*-Klasse vom Paket *java.util* importiert. Nachfolgend werden in den Zeilen 3 und 4 die benötigten Objekte *xmldata* und *xmltool* der bereits in *productlist.jsp* importierten Klassen *XmlProductData* und *XmlTool* erstellt. Das *v*-Objekt enthält die Parameter des zu erstellenden XML-Dokuments.

Nachdem die Methode *setPageNr()* des *page*-Objektes *currentpage* in der *productlist.jsp*-Seite mit dem Parameter *12* aufgerufen wurde, kann hier auch ihre Methode *getDocParams()* in der Zeile 7 in Anspruch genommen werden. Sie füllt den Vektor *v* mit den Parameternamen in Form von String[]-Objekten. String[0] enthält den System-Parameternamen des XML-Parameters, String[1] enthält den Namen des *request*-Parameters, der den Wert des System-Parameters liefert.

Die nachfolgende *for*-Schleife auf Listing 4.2, Zeilen 8-12 liest im *prm*-Objekt die Parameterinformationen und ruft die oben beschriebene Methode *addParamToProductData()* der Klasse *XmlProductData* in der Zeile 10 ab. Dieser Methode werden mit *prm[0]* der Name des XML-Parameters im System und der Wert dieses Parameters mit *request.getParameter(prm[1])* übergeben (Zeilen 10-11). Der letzte kann vom *request*-Objekt mit der Methode *getParame-*

ter() in der Zeile 11 aufgerufen werden. *prm[1]* entspricht damit den Namen des *request*-Parameters.

Nachdem der Durchlauf der *for*-Schleife abgeschlossen ist und alle für das Erstellen des XML-Dokuments benötigten Parameter damit übergeben wurden, kann die eigentliche Erzeugung der Daten beginnen. Zeile 14 ruft die Methode *getDocumentFromDBData()* der Klasse *XmlProductData* auf und übergibt ihr mit Hilfe der Bean *Page* den Namen der zu erstellenden Nachricht. Die Methode *getDocumentFromDBData()* ist auf Listing 4.3 dargestellt. In seiner Zeile 4 wird auf die Klasse *XmlDocument* zugegriffen, welche das XML-Dokument erstellt und ein XML-Dokumentobjekt liefert. Auf die Funktionalität der Klasse *XmlDocument* wird im nächsten Abschnitt näher eingegangen.

Listing 4.3: Die Methode *getDocumentFromDBData()* in der Klasse *XmlProductData*

```
1 public Document getDocumentFromDBData (String sMessageName) {
2     Document doc = null;
3     XmlDocument xmldoc = new XmlDocument();
4     doc = xmldoc.getDocumentFromDB(sMessageName, v);
5     return  doc;
6 }
```

Nachdem das *xmldoc*-Objekt in den Zeilen 13 und 14 des Listings 4.2 in der *productlist.jsp*-Seite die Daten des XML-Dokuments bereits enthält, wird im *String*-Objekt *path* in der Zeile 16 erneut mit Hilfe des *currentpage*-Objektes der Pfad zur zu transformierenden XSL-Datei im System ermittelt. Die Zeile 17 ermittelt die IP-Adresse der Seite und Zeile 18 den Server-Port, welche durch Doppelpunkt getrennt werden.

Anschließend werden die XML-Daten transformiert und mit *out.println(html)* ausgegeben (Zeilen 21, 22). Der Transformationsprozess führt die Transformation in das HTML-Format durch und nutzt dazu die Klasse *XmlTool*. Diese Klasse besitzt die Methode *transform()*, welche den String *path* und das *XML-Dokument*-Objekt übergeben bekommt. Auf die Funktionalität der Klasse *XmlTool* wird auf Seite 160 näher eingegangen.

Erstellen der XML-Dokumente mit der Klasse XmlDocument

Der Prozess der Erstellung eines XML-Objektes läuft innerhalb der Klasse *XmlDocument* ab und wird mit dem Aufruf der Methode *getDocument-*

FromDBData()[20] ausgelöst, welche die Methode *getDocumentFromDB()* aufruft.

Wie aus Listing 4.4 ersichtlich, prüft das System zunächst, ob die übergebenen Parameter für das Erstellen des XML-Dokuments gültig und ausreichend sind. Das wird durch die Methode *checkParameter()* in Zeile 5 ermöglicht.

Listing 4.4: Die Methode *getDocumentFromDB()* in der Klasse *XmlDocument*

```
1 public Document getDocumentFromDB
2                     (String sMessageName, Vector v){
3     Document doc = null;
4     vParam = v;
5     if (checkParameter(sMessageName) == true) {
6         doc = getDocumentDataFromDB(sMessageName);
7     }
8     return  doc;
9 }
```

Die Methode *checkParameter()* kann aus Listing 4.5 entnommen werden. Sie nimmt die Prozedur P_XMLDOC_PARAM der Datenbank in Anspruch, um die notwendigen Parameterdaten für das Erstellen des Dokuments zu bekommen. In der Zeile 8 in Listing 4.5 wird die SQL-Anweisung bestimmt, welche die Prozedur P_XMLDOC_PARAM mit dem Namen des hier zu erstellenden XML-Dokuments aufrufen soll. Im konkreten Beispiel lautet der Name des Dokuments *SubSubTypeProductList* und entspricht dem Wert des Parameternamens *sMessageName* in Zeile 9 in Listing 4.5.

Listing 4.5: Die Methode *checkParameter()* in der Klasse *XmlDocument*

```
1 private boolean checkParameter (String sMessageName) {
2     boolean bAllParamsAvailable = true;
3     boolean bParamAvailable = false;
4     Vector vDbParam = new Vector();
5     String sParamName;
6     String[] sPara = null;
7     ResultSet rset = null;
8     String sql = "SELECT * FROM P_XMLDOC_PARAM
9                         ('" + sMessageName + "')";
10    if(db.getQueryRowCount(sql)==0){
11        bParamAvailable = true;
12    }else{
```

[20] Siehe Zeile 4 in Listing 4.3.

```
13          rset = db.executeQuery(sql);
14          try{
15          rset.next();
16          do {
17              sParamName = "";
18              sParamName = rset.getObject("PARAM_NAME")
19                                                  .toString();
20              vDbParam.addElement(sParamName);
21          } while (rset.next());
22          }catch(Exception e){
23              e.printStackTrace();
24          }
25          for (int i = 0; i < vDbParam.size(); i++) {
26              bParamAvailable = false;
27              String sDbParam = (String)vDbParam.elementAt(i);
28              for (int j = 0; j < vParam.size(); j++) {
29                  sPara = (String[])vParam.elementAt(j);
30                  if (sPara[0].compareTo(sDbParam) == 0) {
31                      bParamAvailable = true;
32                  }
33              }
34              if (!bParamAvailable) {
35                  bAllParamsAvailable = false;
36              }
37          }
38      }
39      return  bAllParamsAvailable;
40  }
```

Die SQL-Abfrage lautet daher

```
SELECT * FROM P_XMLDOC_PARAM('SubSubTypeProductList')
```

und liefert die Namen der erforderlichen System-Parameter zum Erstellen des XML-Dokuments aus der Datenbank. Das Ergebnis des Prozeduraufrufs mit dem Parameterwert *SubSubTypeProductList* ist in der Abbildung 4.3 auf Seite 148 zu entnehmen.

Die von dieser Abfrage gelieferte Ergebnismenge wird als erforderliche System-Parameter im Vektor *vDbParam* abgelegt[21]. Mit diesem Vektor wird in den Zeilen 25-33 in Listing 4.5 geprüft, ob der bereits vorhandenen in der Klasse

[21]Siehe Zeilen 16-21 in Listing 4.5.

Abbildung 4.3: Ergebnis der Prozedur P_XMLDOC_PARAM (Aufruf mit dem Parameterwert *SubSubTypeProductList*)

Vektor *vParam*, der die verfügbaren Parameter und deren Werte enthält, auch die erforderlichen Parameter und ihre Werte für die Erstellung der Nachricht besitzt. Wenn einer der Parameter nicht verfügbar ist, wird der Wert von *bParamAvailable* in der Zeile 26 auf *false* gesetzt und damit in den Zeilen 34-36 der *boolean*-Parameter *bAllParamsAvailable* auch auf *false* gesetzt, welches den Rückgabewert der Methode ergibt.

Wenn die Methode *checkParameter()* aus Listing 4.5 in der Zeile 39 den Wert *true* liefert, was mit der *if*-Anweisung in der Zeile 5 in Listing 4.4 geprüft wird, wird weiterhin in der nächsten Zeile 6 die Methode *getDocumentDataFromDB()* der Klasse *XmlDocument* aufgerufen, welche den Namen des XML-Dokuments als Parameter erwartet. Das ist die wichtigste Methode der Klasse *XmlDocument*. Sie steuert das Erstellen aller XML-Dokumente innerhalb der Anwendung und wird im folgenden Abschnitt daher näher angegangen.

Die Methode getDocumentDataFromDB() in der Klasse XmlDocument

Die Methode *getDocumentDataFromDB()* erwartet den Namen des XML-Dokuments und liefert als Ergebnis ein *Dokument*-Objekt mit den XML-Daten zurück. Die Methode ist in Listing 4.6 dargestellt.

Listing 4.6: Die Methode *getDocumentDataFromDB()* in der Klasse *XmlDocument*

```
1 public Document getDocumentDataFromDB (String messageName) {
2   Document doc= null;
3   Document dDoc = null;
4   ResultSet rs = null;
5   String sDocCode = null;
6   dDoc = xmltool.getEmptyDocument();
7   try {
8   String sql = "select * from p_xmldoc_sqldata('"
```

```
                                    + messageName + "')";
  rs = db.executeQuery(sql);
  boolean b = rs.next();
  sDocCode = rs.getObject("DOC_CODE").toString();
  dDoc.appendChild(dDoc.createElementNS("",
            rs.getObject("MESSAGEDOCELEMENT").toString()));
    do {
     String abfrage = rs.getObject("SQL_CODE").toString();
     Vector vSqlParams = getSqlParamValues(abfrage);
     if (hasParamVectorValidElements(vSqlParams)) {
       if(rs.getObject("XPATH_PARAM") == null){
         doc = getXmlData(vSqlParams,
                    rs.getObject("SQL").toString(),
                    rs.getObject("DOCELEMENT").toString(),
                    rs.getObject("ROWELEMENT").toString(),
                    rs.getObject("PK").toString());
          if (doc != null) {
            Node target_node =
                xmltool.getSingleNodeFromDocument(dDoc,
                rs.getObject("TARGET_NODE").toString());
            NodeList source_nlist =
                doc.getElementsByTagName(
                rs.getObject("SOURCE_NODE").toString());
              for(int i=0; i<source_nlist.getLength(); i++){
                 target_node.appendChild(dDoc.importNode(
                              source_nlist.item(i), true));
              }
          }
      }else{
        String prm_path =
            rs.getObject("XPATH_PARAM").toString();
        NodeList prm_nodelist =
            xmltool.getNodeListFromDocument(dDoc,
                rs.getObject("XPATH_PARAM").toString());
          for(int j=0; j<prm_nodelist.getLength(); j++){
            String prm_wert = prm_nodelist.item(j)
                        .getFirstChild().getNodeValue();
            vSqlParams.addElement(prm_wert);
            doc = getXmlData(vSqlParams,
                rs.getObject("SQL").toString(),
                rs.getObject("DOCELEMENT").toString(),
                rs.getObject("ROWELEMENT").toString(),
```

```
51                  rs.getObject("PK").toString());
52                if (doc != null) {
53                  Node target_node =
54                      xmltool.getSingleNodeFromDocument(dDoc,
55                      rs.getObject("TARGET_NODE").toString(),
56                                                        j);
57                  NodeList source_nlist =
58                      doc.getElementsByTagName(
59                      rs.getObject("SOURCE_NODE").toString());
60                    for(int i=0; i<source_nlist.getLength();
61                                                        i++){
62                      target_node.appendChild(
63                      dDoc.importNode(source_nlist.item(i),
64                                                      true));
65                    }
66                }
67                vSqlParams.clear();
68            }
69        }
70      }
71    } while (rs.next());
72  } catch (Exception ex) {
73      ex.printStackTrace();
74  }
75  dDoc = this.addDocAttributes(sDocCode, dDoc);
76  return  dDoc;
77 }
```

Zunächst wird in der Zeile 10 der Methode *getDocumentDataFromDB()* in Listing 4.6 die Abfrage für die Daten der *productlist.jsp*-Seite ausgeführt:

```
select * from p_xmldoc_sqldata('SubSubTypeProductList')
```

Das Ergebnis der SQL-Anweisung stellt alle Daten aus der Datenbank zur Verfügung, die zum Erstellen des XML-Dokuments notwendig sind. Die gelieferten für das XML-Dokument *SubSubTypeProductList* Daten sind in der Abbildung 4.4 dargestellt.

Mit Hilfe des Parameters SQL_CODE[22], welcher den Primärschlüssel der SQL-Abfrage in der Datenbanktabelle TDAT_XMLSQL liefert, werden zunächst die benötigten Parameter für die Abfrage mit dem Vektor *vSqlParams* in Listing

[22]Siehe Abbildung 4.4

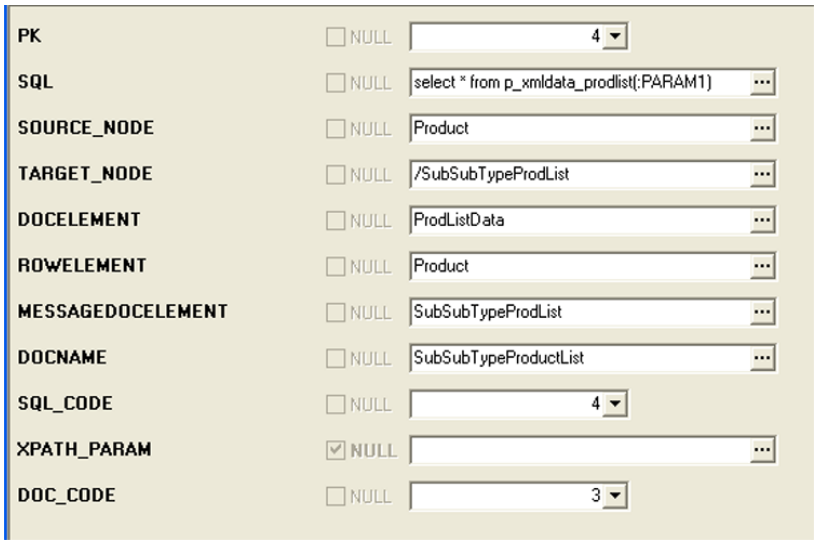

Abbildung 4.4: Die einzige Ergebniszeile der Prozedur P_XMLDOC_SQLDATA (Aufruf mit dem Parameter *SubSubTypeProductList*)

4.6 in der Zeile 17 bereitgestellt. Weiterhin wird in der Zeile 18 geprüft, ob die Parameter für die Erstellung der Abfrage, die die eigentlichen XML-Daten liefert, gültig sind. Diese Aufgabe wird von der Methode *hasParamVectorValidElements()*[23] ausgeführt. Sie erwartet den bereits ermittelten Vektor *vSqlParams* mit den Abfrage-Parametern und liefert einen *boolean*-Wert zurück. Die Methode *hasParamVectorValidElements()* prüft, ob alle Parameter einen Wert ungleich dem leeren Stringwert enthalten. Der Rückgabewert dieser Methode ist *false*, wenn einer der Parameter diese Anforderung nicht erfüllt. Im anderen Fall liefert die Methode den Wert *true* zurück.

Innerhalb der Methode *getDocumentDataFromDB()* in Listing 4.6 wird auch das XML-Dokument erstellt. Das Wurzelelement des XML-Dokuments am Beispiel der Seite *productlist.jsp* trägt den Namen *SubSubTypeProdList*[24]. Es wird in den Zeilen 13-14 in Listing 4.6 erzeugt: Dazu wird der Parameter MESSAGEDOCELEMENT, der mit der Prozedur P_XMLDOC_SQLDATA geliefert wird, in Anspruch genommen. Sein Wert entspricht dem Namen des *root*-Elements vom XML-Dokument. Das *root*-Element ist das übergeordnete Element für alle weiteren XML-Knoten, die innerhalb dieser Methode erstellt

[23]Siehe Supplementum, Zeilen 106-113 in Listing B.19 auf Seite 46.
[24]Siehe das Attribut MESSAGEDOCELEMENT in der Abbildung 4.4.

werden.

Für jede Ergebniszeile der Prozedur P_XMLDOC_SQLDATA werden weiterhin in der Methode *getDocumentDataFromDB()* ein oder mehrere XML-Knoten erzeugt und dem XML-Dokumentobjekt hinzugefügt. Beim Erstellen des XML-Knotens existieren zwei Möglichkeiten:

- Das Ergebnis der Spalte XPATH_PARAM aus der Prozedur P_XML-DOC_SQLDATA besitzt den Wert *null*,
- Das Ergebnis der Spalte XPATH_PARAM aus der Prozedur P_XML-DOC_SQLDATA besitzt einen Wert *ungleich null*.

Im ersten Fall wird mit den von der Prozedur P_XMLDOC_SQLDATA gelieferten Werten für die Attribute SQL, DOCELEMENT, ROWELEMENT und PK[25] der XML-Knoten mit den Datenbankdaten in den Zeilen 20-24 in Listing 4.6 erstellt. Dazu dient die Methode *getXmlData()*[26]. Sie besorgt sich die Daten aus der Datenbank und erstellt ein XML-Dokumentobjekt mit den Nutzdaten. Wenn das erstellte XML-Dokumentobjekt *doc* nicht den Wert *null* und damit einen gültigen Wert besitzt, wird das erstellte XML-Dokument in den Zeilen 25-35 an dem Dokument *dDoc* angehängt.

Im zweiten Fall, in dem die Spalte XPATH_PARAM der Prozedur P_XML-DOC_SQLDATA einen Wert ungleich *null* besitzt, wird eine andere Vorgehensweise beim Erstellen des XML-Dokumentobjektes vorgenommen. Der Unterschied zum ersten Fall ist, dass das Erstellen des XML-Dokumentobjektes einen oder mehrere weitere Parameter *aus dem bestehenden XML-Dokument dDoc* benötigt. Die Anzahl der Parameter hängt von der Anzahl der bereits existierenden Knoten in *dDoc* mit dem Pfad *XPATH_PARAM* ab. Das *NodeList* wird in den Zeilen 40-42 in Listing 4.6 ermittelt. Der Wert des j-ten Parameters entspricht ein Element der Knotenliste und wird in den Zeilen 44-45 bekommen und dem Vektor *vSqlParams* in der Zeile 46 hinzugefügt. Analog zum ersten Fall wird in den Zeilen 47-51 mit Hilfe der Methode *getXmlData()* das XML-Dokument erstellt. Wenn es einen Wert ungleich *null* besitzt[27], wird er in den Zeilen 53-56 an dem j-ten *XPATH_PARAM*-Knoten angehängt.

[25]Siehe Abbildung 4.4 auf Seite 151.
[26]Siehe Seite 153.
[27]Siehe Zeile 52 in Listing 4.6.

Die Methode getXmlData() in der Klasse XmlDocument

Nun soll die Funktionalität der Methode *getXmlData()* näher erläutert werden. Diese Methode findet sich angewendet im Quelltext von Listing 4.6 in Zeile 20 als Funktion wieder. Der Quelltext dieser Methode ist in Listing 4.7 dargestellt.

Listing 4.7: Die Methode *getXmlData()* in der Klasse *XmlDocument*

```
1  public Document getXmlData (Vector vParam, String sSql,
2      String sDocElement, String sRowElement, String xmlsql){
3    Document result = null;
4    Vector map = null;
5    map = getMappingData(xmlsql);
6    try{
7      Document docNeu = xmltool.getEmptyDocument();
8      Element root = docNeu.createElementNS("", sDocElement);
9      docNeu.appendChild(root);
10     String Sql = getQueryString(sSql, vParam);
11     ResultSet rsxml = db.executeQuery(Sql);
12     String[] value = null;
13     Element oColumn = null;
14     int row = 0;
15     boolean colInMap = false;
16     if(db.getQueryRowCount(Sql) > 0){
17       rsxml.next();
18       do {
19         docNeu.getFirstChild().appendChild(docNeu.
20                                 createElement(sRowElement));
21         Node nRow = docNeu.
22                 getElementsByTagName(sRowElement).item(row);
23         row++;
24         for (int i = 0; i < db.getQueryColumnCount(Sql);
25                                                       i++){
26           if(rsxml.getObject(rsxml.getMetaData().
27                     getColumnName(i+1).toString()).
28                         toString().compareTo("")!= 0){
29             String colname = rsxml.getMetaData().
30                         getColumnName(i+1).toString();
31             colInMap = false;
32             if(map != null){
33               for(int j = 0; j < map.size(); j++){
34                 value = (String[])map.elementAt(j);
35                 if(value[0].compareTo(rsxml.getMetaData().
36                         getColumnName(i+1).toString())==0){
```

```
37                      oColumn = docNeu.createElement(value[1]);
38                      colInMap = true;
39                    }
40                  }
41                }
42                if(!colInMap){
43                    oColumn = docNeu.createElement(rsxml.
44                    getMetaData().getColumnName(i+1).toString());
45                }
46                nRow.appendChild(oColumn);
47                oColumn.appendChild(docNeu.createTextNode(rsxml.
48                    getObject(rsxml.getMetaData().
49                    getColumnName(i+1).toString()).toString()));
50              }
51            }
52          }while (rsxml.next());
53        }
54        else{
55            docNeu = null;
56        }
57        result = docNeu;
58      }catch (Exception ex) {
59        ex.printStackTrace();
60      }
61      return  result;
62  }
```

Die Methode *getXmlData()* erwartet in der Zeile 1 den Vektor *vParam* mit den Parametern, welche für das Erstellen des XML-Dokuments benötigt werden. Weiterhin müssen die SQL-Anweisung *sSql*, die die eigentlichen Daten für das XML-Dokument aus der Datenbank liefert und die Namen des Wurzelelements *sDocElement* und sein Kind-Element *sRowElement* vom zu erstellenden Dokument als *String*-Parameter übergeben werden. Anschließend erwartet die Methode *getXmlData()* den Parameter *xmlsql*, welcher das PK-Attribut der Prozedur P_XMLDOC_SQLDATA[28] entspricht. Dieser Parameter stimmt mit dem PK der Tabelle TDAT_XMLDATA überein, welcher sich auf der Datenbankebene zwischen den Tabellen TDAT_XMLDOC und TDAT_XMLSQL[29] befindet.

[28]Siehe Abbildung 4.4 auf Seite 151.
[29]Siehe Abbildung 3.8 auf Seite 85.

Der zuletzt genannte Parameter *xmlsql* wird in der Zeile 5 auf Listing 4.7 verwendet, um den Vektor mit den Mapping-Daten für die XML-Elementnamen zu ermitteln. Dafür wird die Methode *getMappingData()*[30] aufgerufen. Sie führt die SQL-Abfrage mit dem Wert *4* für XMLDATA_CODE[31] durch

```
select * from TDAT_XMLELEMENT_MAPPING where XMLDATA_CODE = 4
```

Mit den ermittelten Ergebnissen wird ein Vektor mit String[]-Elementen vorbereitet, welcher die Namen der Spalten und die dazugehörigen Elementennamen enthält. Dieser Vektor wird als Ergebnis der Methode *getMappingData()* zurückgeliefert.[32] In der Methode *getXmlData()*[33] entspricht das dem Vektor *map* in der Zeile 5.

In den weiteren Zeilen 7-11 in Listing 4.7 wird das Dokument-Element erstellt und die SQL-Anweisung mit der Methode *getQueryString()* in der Zeile 10 ermittelt. Mit der Methode *getQueryString()*[34] wird geprüft, ob die SQL-Anweisung Parameter integrieren soll und wenn dies der Fall ist, ob es ein oder mehrere Parameter sind. Das Ergebnis ist die ausführbare SQL-Anweisung in der Form eines *String*-Objekts, welches in der Zeile 11 ausgeführt wird. Das Ergebnis ist das *ResultSet rsxml*. In Zeile 16 in Listing 4.7 wird weiterhin geprüft, ob das gelieferte *ResultSet* Daten enthält. Wenn das Ergebnis mindestens eine Zeile liefert, wird in den Zeilen 16-53 das leere XML-Dokument *docNeu* mit den Datenbank-Daten gefüllt.

Das *ResultSet rsxml* aus Zeile 11 enthält die Ergebnisdaten in einer tabellenvergleichbaren Form. Die Daten jeder Spalte einer Zeile werden mit einem eigenen Tag dem XML-Dokument hinzugefügt und alle Tags mit den Nutzdaten werden von einem XML-Tag umhüllt. Das XML-Tag ist ein Kind des Dokument-Elements und besitzt den Namen des Parameters *sRowElement*. In den Zeilen 19-20 in Listing 4.7 wird daher für jedes Tupel des *ResultSets* ein Element mit dem Namen *sRowElement* erzeugt. Diesem werden für jedes der Spalten-Elemente des Tupels Kind-Elemente in den Zeilen 26-50 hinzugefügt. Zunächst wird in den Zeilen 26-28 sicher gestellt, dass der Inhalt der Tabellenzelle nicht leer ist. In den Zeilen 33-40 wird gesucht, ob der Vektor *map* mit den Mapping-Daten einen Eintrag für die entsprechende Spalte enthält. Wenn ein Eintrag vorliegt, wird das Element mit dem im Vektor *map* angegebenen

[30]Siehe Supplementum, Zeilen 402-426 in Listing B.19 auf Seite 46.
[31]Siehe Abbildung 3.9 auf Seite 86.
[32]Siehe Supplementum, Zeilen 425 in Listing B.19 auf Seite 46.
[33]Siehe Seite 153.
[34]Siehe Supplementum, Zeile 312-331 in Listing B.19 auf Seite 46.

Namen erstellt. Im anderen Fall wird in den Zeilen 42-44 das Element mit dem Namen der Tabellenspalte erstellt. Anschließend wird in den Zeilen 46-49 das erstellte Element mit den eigentlichen Daten gefüllt und seinem übergeordneten Element hinzugefügt. Das erstellte XML-Dokument *result* wird in der Zeile 61 an die aufrufende Methode zurückgeliefert.

Die Methode addDocAttributes() in der Klasse XmlDocument

Ein XML-Dokument kann außer Elementen auch Attribute besitzen. Das Erstellen der Attribute wird in der Methode *getDocumentDataFromDB()* aus Listing 4.6 in der Zeile 75 ausgelöst, in der die Methode *addDocAttributes()* aufgerufen wird. Diese Methode erwartet das bereits erstellte Dokument-Element *doc_without_attr* und den Datenbank-PK der Tabelle TCAT_DOCUMENT des zu erstellenden Dokuments *doc_code*. Der Quelltext der Methode *addDocAttributes()* wird in Listing 4.8 dargestellt.

Listing 4.8: Die Methode *addDocAttributes()* in der Klasse *XmlDocument*

```
1  public Document addDocAttributes(String doc_code,
2                                   Document doc_without_attr){
3    Document doc = doc_without_attr;
4    ResultSet attr_rs = null;
5    ResultSet sql_rs = null;
6    String[] sDocParam = null;
7    String sPrmValue = null;
8    Node nTarget = null;
9    String attr_sql = "select * from p_xmldoc_attribute (" +
10                                              doc_code + ")";
11   String sPrmSql = null;
12   String sPrmArt = null;
13   try{
14     attr_rs = db.executeQuery(attr_sql);
15     if (db.getQueryRowCount(attr_sql) > 0) {
16       attr_rs.next();
17       do{
18         sPrmSql = attr_rs.getObject("SQL").toString();
19         sPrmArt = attr_rs.getObject("PARAM_ART").toString();
20         if(sPrmArt.compareTo("SYSPARAM") == 0){
21           nTarget = xmltool.getSingleNodeFromDocument(doc,
22             attr_rs.getObject("PARENT_NODE_PATH").toString());
23           Element el = (Element)nTarget;
24           for (int i = 0; i < vParam.size(); i++) {
```

```
25              sDocParam = (String[])vParam.elementAt(i);
26              if (sDocParam[0].compareTo(attr_rs.
27                  getObject("PARAMNAME").toString()) == 0) {
28                sPrmValue = sDocParam[1];
29              }
30            }
31            if(sPrmSql.compareTo("param") == 0){
32              el.setAttribute(attr_rs.
33                  getObject("ATTRIBUTE_NAME").toString(),
34                      sPrmValue);
35            }else{
36              sql_rs = null;
37              sql_rs = db.executeQuery(sPrmSql.
38                      replaceFirst(":param", sPrmValue));
39              sql_rs.next();
40              el.setAttribute(
41                attr_rs.getObject("ATTRIBUTE_NAME").toString(),
42                sql_rs.getObject("PRMVALUE").toString());
43              sql_rs.close();
44            }
45          }
46          if(sPrmArt.compareTo("XPATHPARAM") == 0){
47            NodeList prm_nodelist = xmltool.
48              getNodeListFromDocument(doc,
49              attr_rs.getObject("XPATH_TO_PARAM").toString());
50            for(int j=0; j<prm_nodelist.getLength(); j++){
51              String prm_wert = prm_nodelist.item(j).
52                                  getFirstChild().getNodeValue();
53              ResultSet rs = db.executeQuery(sPrmSql.
54                              replaceFirst(":param", prm_wert));
55              rs.next();
56              Node target_node = xmltool.
57                  getSingleNodeFromDocument(doc,
58                  attr_rs.getObject("PARENT_NODE_PATH").
59                                                  toString(), j);
60              Element el = (Element)target_node;
61              el.setAttribute(attr_rs.
62                  getObject("ATTRIBUTE_NAME").toString(),
63                      rs.getObject("PRMVALUE").toString());
64              rs.close();
65            }
66          }
```

```
67          }while(attr_rs.next());
68          attr_rs.close();
69        }
70      }catch(Exception e){
71          e.printStackTrace();
72      }
73      return doc;
74  }
```

In der *addDocAttributes()*-Methode wird zunächst die SQL-Abfrage

```
select * from p_xmldoc_attribute ( doc_code )
```

in der Zeile 14 in Listing 4.8 ausgeführt. Für das bereits als Beispiel genommene XML-Dokument *SubSubTypeProductList* mit PK-Wert *3* wird die Abfrage

```
select * from p_xmldoc_attribute (3)
```

ausgeführt. Das Ergebnis dieser Abfrage liefert vier Zeilen, welche in der Abbildung 4.5 auf Seite 159 dargestellt sind. Für jede dieser Zeilen wird die *do*-Schleife in den Zeilen 17-69 in Listing 4.8 ausgeführt, welche ein Attribut in die XML-Datei hinzufügt. Zuerst wird in der Zeile 20, mit der in der Zeile 19 übermittelten Parameterart auf den Attributwert geprüft. Es existieren die zwei Parameterarten

- *System-Parameter* (SYSPARAM): Die System-Parameter sind in der Tabelle TCAT_XMLPARAM aufgelistet.
- *XPath-Parameter* (XPATHPARAM): Dieser Parameter stellt einen Stringwert für einen XPath-Ausdruck dar.

Wenn es sich um einen System-Parameter handelt[35], wird sein Wert vom Übergabeparameter-Vektor der Klasse *XmlDocument* ermittelt[36]. Danach wird die SQL-Spalte der bereits ausgeführten Abfrage geprüft. Falls sie den Wert *param* besitzt, übernimmt das Attribut den Wert des System-Parameters in den Zeilen 32-34 in Listing 4.8. Im anderen Fall wird der Parameterwert genutzt, um die in der SQL-Spalte stehende SQL-Anweisung auszuführen. Der Attributwert wird auf den Wert des Ergebnisses der ausgeführten SQL-Anweisung gesetzt[37].

[35]Siehe Zeile 20 in Listing 4.8.
[36]Siehe Zeilen 24-30 in Listing 4.8.
[37]Siehe Zeilen 36-43 in Listing 4.8.

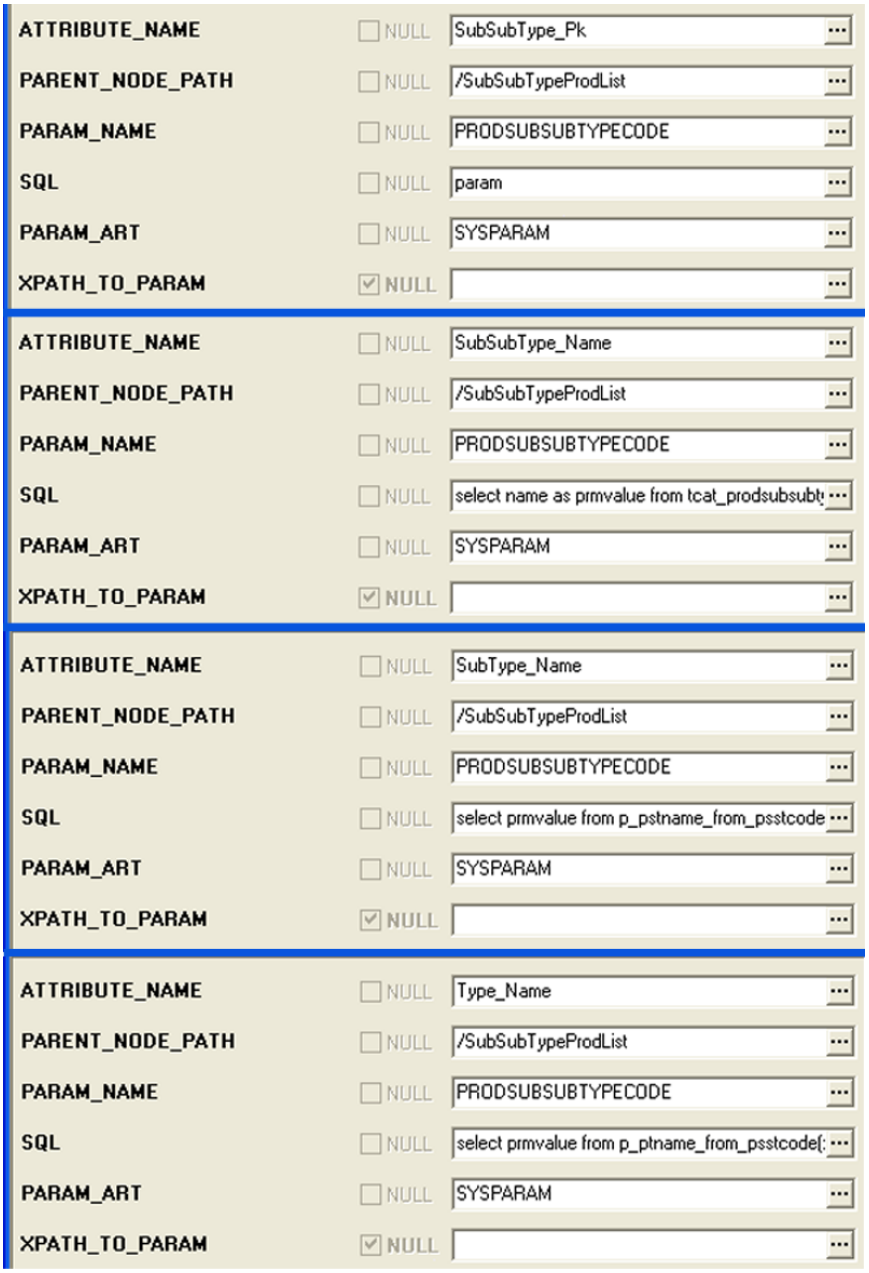

Abbildung 4.5: Ergebnis der Prozedur P_XMLDOC_ATTRIBUTE (Aufruf mit dem Parameterwert *3*)

Wenn es sich anderenfalls um einen XPath-Parameter handelt, wird zuerst eine Knotenliste aus dem XML-Dokument erstellt. Für jeden Knoten wird weiterhin in der *for*-Schleife die SQL-Abfrage mit dem Parameterwert ermittelt und anschließend ausgeführt. Das Ergebnis der Abfrage ist der Wert des neuen Attributs und wird in den Zeilen 61-63 in Listing 4.8 zugewiesen und dem XML-Dokument hinzugefügt.

Die Klasse XmlTool

Die Klasse *XmlTool* ist eine Klasse in der Anwendung, die Grundmethoden zur Verfügung stellt. Deren Rolle ist einfache Aufgaben der Anwendung wie Bereitstellung eines leeren *Dokument*-Objekts oder das Schreiben eines *Dokument*-Objekts in eine Datei zu erledigen. Folgende Methoden werden von der Klasse *XmlTool* zur Verfügung stellt:

- *getEmptyDocument()*: Liefert ein leeres XML-Dokument erwartet dabei keine Übergabeparameter.

- *writeXmlFile()*: Schreibt ein XML-Dokument auf den zugehörigen festen Speicherplatz. Die Methode erwartet ein *XML-Dokument*-Objekt und einen Dateinamen für die XML-Datei.

- *getSingleNodeFromDocument()*: Diese Methode erwartet ein *XML-Dokument*-Objekt und einen XPath-Ausdruck und liefert abhängig vom XPath-Ausdruck einen bestimmten Knoten vom XML-Dokument zurück.

- *transform()*: Transformiert ein XML-Dokument in ein HTML-Dokument. Die erwarteten Parameter sind der Pfad des XSL-Files, der die Regel der Transformation und das *XML-Dokument*-Objekt bereitstellt. Das HTML-Dokument wird als *String*-Objekt zurückgegeben.

- *getDocAsString()*: Liefert aus einem *XML-Dokument*-Objekt ein *String*-Objekt zurück. Daher erwartet die Methode ein *Dokument*-Objekt als Eingabeparameter und liefert ein *String*-Objekt zurück.

- *xml2String()*: Diese Methode erwartet ein *Dokument*-Objekt und liefert das XML-Dokument in Form eines *String*-Objektes zurück.

4.3 Selektieren von XML-Daten mit der XML Path Language (XPath)

Das W3C definiert die XPath-Spezifikation (Clark u.a., 1999), mit der Elemente eines XML-Dokuments wie XML-Knoten, eine Gruppe von XML-Knoten, Attribute oder Kommentare adressiert werden können. Diese Eigenschaft kann beispielsweise von XSLT-Templates für Formatierungszwecke, von XQuery zu Abfragen oder von XPointer zur Verlinkung von XML-Dokumenten genutzt werden.

XPath zeichnet sich durch die hier aufgelisteten Eigenschaften aus:

- Die Groß- und Kleinschreibung ist wie bei XML relevant.
- Ein Elementknoten mit einem Namensraum besitzt auch einen Namensraumknoten.
- Kommentare und Verarbeitungsanweisungen werden auch in diesem abstrakten Modell repräsentiert.
- Dokumenttyp-Deklarationen und CDATA-Abschnitte sind hier nicht abgebildet.
- Jeder Knoten hat eine Menge von Knoten als Kinder und diese Knotenmenge ist jedoch - außer im Fall des Elementknotens - immer leer. Dies begründet das Ergebnis der leeren Knotenmenge beim Versuch, mit XPath die Kinder eines Attributknotens auszuwählen.
- Alle Knoten außer den Wurzelknoten besitzen ein Wurzelelement.

Die XPath-Spezifikation definiert ein abstraktes Dokumentmodell in Form eines Baumes, in dem die folgend aufgelisteten sieben Knotenarten vorkommen (Clark u.a. 1999; Rottach und Groß 2002, S. 67; Eckstein und Eckstein 2004, S. 66 ff.):

- *Wurzelknoten* (*root node*): Der Wurzelknoten tritt nur einmal im Dokument auf. Er enthält alle Knoten des XML-Dokuments wie Verarbeitungsanweisungen, Kommentare und den Dokumentknoten.
- *Elementknoten* (*element node*): Jeder Elementknoten enthält mindestens ein Element, d.h. zu jedem Element existiert ein Elementknoten. Ein Element kann weitere Knoten enthalten.

- *Textknoten* (*text node*): Ein Textknoten stellt den eigentlichen Inhalt der Elemente dar und besteht aus Zeichendaten.

- *Attributknoten* (*attribute node*): Jedes Attribut besitzt einen Attributknoten. Ein Element ist einem Attributknoten übergeordnet (*parent node*), ein Attributknoten ist allerdings kein Kind eines Elements und lässt sich daher nicht von diesem ableiten.

- *Namensraumknoten* (*namespace node*): Namensraumknoten gehören zu einem Namensraum. Jedes Element im Dokument kann mehrere Namensraumknoten enthalten. Ein Element ist einem Namensraumknoten übergeordnet, umgekehrt ist aber ein Namensraumknoten kein Kind seines übergeordneten Knotens und lässt sich nicht von diesem ableiten.

- *Verarbeitungsanweisungsknoten* (*processing instruction node*): Ein Verarbeitungsanweisungsknoten enthält Verarbeitungsanweisungen für das XML-Dokument. Zu jeder Verarbeitungsanweisung existiert ein Verarbeitungsanweisungsknoten.

- *Kommentarknoten* (*comment node*): XML-Dokumente können beliebig viele Kommentare enthalten. Für jeden Kommentar wird im Dokument ein Kommentarknoten gebildet.

Wie auch der Name XPath darauf hinweist, basiert eine XPath-Spezifikation auf Pfaden. Auf einen einzelnen Knoten wird über einen bestimmten Pfad zugegriffen. Er bezeichnet den Weg vom Quell- zum Zielknoten und ist mit einem String beschreibbar. Der Weg fängt beim *Kontextknoten* (*context node*) an, der dadurch der Quellknoten ist, und endet über die *Achsen* beim Zielknoten. Achsen spezifizieren daher den Weg zum Zielknoten ausgehend vom Kontextknoten. XPath kennt 13 Achsen:

- *ancestor*-Achse: Wählt alle Vorfahren des Kontextknotens,

- *ancestor-or-self*-Achse: Wählt alle Vorfahren des Kontextknotens und den Kontextknoten selbst,

- *attribute*-Achse: Enthält alle Attribute des Kontextknotens,

- *child*-Achse: Wählt alle Kinder des Kontextknotens,

- *descendant*-Achse: Wählt alle Nachfolger (alle Kinder und Kindeskinder) des Kontextknotens,

- *descendant-or-self*-Achse: Wählt alle Nachfolger (alle Kinder und Kindeskinder) des Kontextknotens und den Kontextknoten selbst,
- *following*-Achse: Wählt alle Knoten auf derselben Ebene des Kontextknotens (Geschwisterknoten), die nach dem Kontextknoten im Dokument auftreten und deren Nachfolger,
- *following-sibling*-Achse: Wählt alle Knoten auf der gleichen Ebene (Geschwisterknoten), die nach dem Kontextknoten im Dokument auftreten,
- *namespace*-Achse: Enthält alle Namensraumknoten des Kontextknotens,
- *parent*-Achse: Wählt nur einen Knoten, nämlich den Elternknoten des Kontextknotens,
- *preceding*-Achse: Wählt alle Knoten auf der selben Ebene des Kontextknotens, die vor dem Kontextknoten im Dokument auftreten sowie deren Nachfolger,
- *preceding-sibling*-Achse: Wählt alle Knoten auf der gleichen Ebene des Kontextknotens, die vor dem Kontextknoten im Dokument auftreten,
- *self*-Achse: Nur der Kontextknoten selbst wird ausgewählt.

Wütherich (2001, S. 94 ff.) beschreibt drei Konzepte, welche eine XPath-Anfrage formulieren lassen.

- *Der Auswertungskontext*: Dieser stellt das Konzept des Kontextknotens dar, bei dem eine Anfrage relativ in Bezug auf die Baumstruktur eines Dokuments formuliert wird.
- *Die Verwendung von Achsen*: Das Konzept der Achsen beinhaltet die dreizehn oben beschriebenen Achsentypen.
- *Der Knotentest*: Dieser prüft auf Eigenschaften des Knotens wie *name()*, *text()*, *processing-instruction()*.

Darauf aufbauend können für beliebige Teilabschnitte des Dokuments Anfragen erstellt werden. Eine XPath-Anfrage besteht aus mehreren Schritten. Jeder dieser Schritte ist nach Wütherich (2001, S. 95) mit dem folgenden Aufbau strukturiert:

```
Achse::Knotentest[Prädikat]
```

Damit besteht jede Anfrage aus einem Kontextknoten. Von diesem ausgehend werden die einzelnen Schritte bestimmt und mit einem Slash-Zeichen „/“ voneinander getrennt. Der Knotentest wird dabei von der Achse durch zwei Semikolon-Zeichen getrennt. Mit dem optionalen *Prädikat*, welches in eckigen Klammern platziert wird, kann die Auswahl des Knotens weiter eingeschränkt werden.

Einen wichtigen Bestandteil der XPath-Spezifikation stellen die zur Verfügung stehenden Funktionen dar. Mit deren Hilfe kann beispielsweise das erste oder letzte Element eines Knotens ausgewählt werden, eine Teilzeichenkette oder die Länge eines Strings bestimmt werden. Die einzelnen XPath-Funktionen können nicht innerhalb dieser Arbeit behandelt werden, da sie deren Umfang sprengen würden. Eine ausführliche Liste ist unter Selfhtml (2004) vorhanden.

4.4 Formatieren von XML-Dokumenten

XML ist eine Sprache, mit der ausschließlich Daten strukturiert werden. In dieser Form können sie auf unterschiedliche Weise dargestellt werden. Zum Publizieren im World Wide Web werden die Daten in HTML transformiert, zum Drucken wird das PDF-Format verwendet. Das Umformen in weitere Sprachen wie CSV[38] und WML[39] ist auch üblich. Die Formung der Ausgabesprachen wird mit der Extensible Stylesheet Language (XSL)[40] durchgeführt, welche aus drei Teilen besteht:

- XSL Transformations (XSLT)[41],
- XPath[42] und
- XSL Formatting Objects (XSL-FO)[43].

XSL ist eine der wichtigsten Sprachen in der XML-Familie und wurde im Jahre 1999 vom W3C entwickelt. Mit dieser konnten XML-Daten mit Hilfe

[38] CSV (*Comma Separated Values* oder *Character Separated Values*): Das CSV-Format verwendet meist das Komma als Trennzeichen, um tabellarische Daten in einer Text-Datei strukturiert abzulegen.
[39] WML (*Wireless Markup Language*): Eine XML-basierte Sprache, die zur Darstellung von Daten auf Mobilgeräten dient.
[40] Vgl. W3C (2004a).
[41] Siehe Abschnitt 4.4.1.
[42] Siehe Abschnitt 4.3 auf Seite 160.
[43] Siehe Abschnitt 4.4.2 auf Seite 168.

von Transformationsfunktionen in HTML überführt werden. Sie wurde vor der Normierung in zwei Komponenten aufgeteilt: XSLT und XSL. XSLT wurde im November 1999 und XSL gegen Ende 2001 normiert.

4.4.1 XSLT: XSL Transformation Language

XSLT (Clark, 1999) ist eine Skriptsprache und eine der wichtigsten Sprachen der XML-Familie. XSLT stellt grundlegende Funktionen zur Verfügung, mit der XML-Strukturen umgeformt werden können. Mit Hilfe von XSLT können XML-Daten in andere Formate wie XML, HTML, Text und LaTeX[44] transformiert werden. Eine Transformation in das Format HTML ermöglicht beispielsweise die Darstellung der Daten im Browser.

Zur Transformation ist ein XSLT-Prozessor wie Apache Xalan[45], Saxon[46] etc. nötig. Bei Verwendung von J2SE (Java 2 Platform, Standard Edition)[47] 1.4 gibt es einen XSLT-Prozessor, der über eine eigene API angesprochen wird. Die XSLT-API befindet sich im Paket *javax.xml.transform*. Der XSLT-Prozessor wird über die *TransformerFactory*-Klasse erzeugt und der Prozessor wird über die Klasse *Transformer* eingesetzt. Nach Niedermair (2002, S. 172) existieren drei mögliche Orte für die Transformation der XML-Dateien:

- *Transformation durch den Client*: Auf dem Client wird der entsprechende Transformer des Browsers benutzt. Das ist allerdings keine empfehlenswerte Methode, da nicht alle Browser einen solchen Transformer einsetzen.

- *Onlinetransformation durch den Server*: In diesem Fall wird die Transformation auf eine Anfrage des Clients online durchgeführt. Die XML-Dateien werden meist in HTML oder PDF transformiert und zum Client übertragen. Da dies gängige Formate sind, müssen in der Regel keine weiteren Anpassungen auf dem Client stattfinden, was von Vorteil ist.

- *Offlinetransformation*: Die XML-Datei, die bereits mit Hilfe von XSL in HTML transformiert wurde, wird auf dem Server zur Verfügung gestellt. Nachteil dieser Transformation ist, dass bei jeder Änderung der XML oder XSL Dateien eine erneute Transformation stattfinden muss.

[44] LaTeXist ein Softwarepaket, der die vereinfachte Nutzung des Textsatzsystems TeXermöglicht.
[45] Siehe http://xml.apache.org/xalan-j/
[46] Siehe http://saxon.sourceforge.net/
[47] Siehe dazu http://java.sun.com/j2se/

Transformation von XML ins HTML-Format

Die Transformation der XML-Dateien wird innerhalb der entwickelten Anwendung in der Regel von den JSPs angestoßen. Innerhalb dieser wird erst der Pfad zur XSL-Transformationsdatei ermittelt:

```
String path = "http://" +
    request.getRemoteAddr()+ ":" +
    String.valueOf(request.getServerPort()) +
    request.getContextPath() + "/xsl/prodtype.xsl";
```

Anschließend wird die Methode *transform()* der Klasse *XmlTool* aufgerufen. Diese erwartet den bereits ermittelten Pfad zur transformierenden XSL-Datei und das *Dokument*-Objekt mit den Meta- und Nutzdaten und liefert ein *String*-Objekt mit HTML-Output für den Browser:

```
String html = xmltool.transform(path,xmldoc);
```

Das *String*-Objekt *html* wird innerhalb der JSP-Seite integriert und ist hiermit im Browser darstellbar.

Die Transformation ins HTML-Format soll hier innerhalb der Anwendung verdeutlicht werden. In Listing 4.9 sind die Nutzdaten aufgelistet, welche sich im XML-Dokumentobjekt befinden.

Listing 4.9: Die zu transformierende XML-Datei

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ProdTypeData>
   <ProdType>
      <Pk>1</Pk>
      <Name>Lebensmittel</Name>
   </ProdType>
   <ProdType>
      <Pk>2</Pk>
      <Name>Reise</Name>
   </ProdType>
   <ProdType>
      <Pk>3</Pk>
      <Name>Tickets</Name>
   </ProdType>
   <ProdType>
      <Pk>4</Pk>
      <Name>Andere</Name>
```

```
18     </ProdType>
19 </ProdTypeData>
```

Der Pfad zum zugehörigen Stylesheet, der für die Transformation der Daten ins HTML-Format zuständig ist, befindet sich in der Tabelle TCAT_PAGE[48] zur jeweiligen Seite der Anwendung. Damit verfügt die Anwendung über den Pfad zum transformierenden Stylesheet, welcher aus Listing 4.10 entnommen werden kann.

Listing 4.10: Die transformierende XSL-Datei

```
1 <?xml version="1.0" encoding="iso-8859-1"?>
2
3 <xsl:stylesheet version="1.0"
4             xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
5     <xsl:template match="/ProdTypeData">
6
7         <table border="0" cellspacing="2" cellpadding="0">
8         <tr>
9             <td class="heading"
10                colspan="2">Katalog - Kategorien</td>
11            <td></td>
12        </tr> <tr>
13            <td></td>
14            <td><img src="images/punkt.gif" alt=""
15                     height="15" width="10" border="0"/></td>
16            <td></td>
17        </tr>
18
19        <xsl:for-each select="ProdType">
20        <tr>
21            <td></td>
22            <td class="prodtype" valign="center">
23                <xsl:variable name="p_name">
24                <xsl:value-of select="Name"/>
25                </xsl:variable>
26                <xsl:variable name="p_pk">
27                <xsl:value-of select="Pk"/>
28                </xsl:variable>
29                <a href="prodsubtype.jsp?PK={$p_pk}">
30                    <xsl:value-of select="Name"/>
31                </a>
```

[48] Siehe Abbildung 3.9 auf Seite 86.

```
32                </td>
33                <td></td>
34            </tr>
35            </xsl:for-each>
36
37            </table>
38        </xsl:template>
39 </xsl:stylesheet>
```

Das XSL-Stylesheet ist selbst ein XML-Dokument, was mit der Zeile 1 des Listings 4.10 bekanntgegeben wird. In den Zeilen 3-4 befindet sich der öffnende Tag des *root*-Elements. Dieses Element und seine Kind-Elemente befinden sich im *Namensraum xsl* und seine URI ist in Zeile 4 angegeben.

Das *root*-Element des XSL-Stylesheets besitzt ein Template-Element, welches die gesamte Regel für die Formatierung der Daten enthält. Es erstellt eine Tabelle mit zwei Spalten. In die erste Zeile der Tabelle wird der Text „Katalog - Kategorien“ geschrieben.[49] In die zweite Zeile der Tabelle wird mit einer unsichtbaren Grafik Abstand zwischen die erste und dritte Zeile eingeräumt.[50] In den Zeilen 19-35 von Listing 4.10 folgt eine Schleife, die alle *ProdType*-Elemente der XML-Datei durchläuft und ihren Inhalt auswertet. Zunächst wird dabei in den Zeilen 23-25 die Variable *p_name* erzeugt und ihr der Wert *Lebensmittel* des Kind-Elements *Name* zugewiesen. Analog dazu wird in den Zeilen 26-28 die Variable *p_pk* erstellt. Sie bekommt den Wert *1* des Kind-Elements *Pk* vom ersten *ProdType*-Element im Dokument. Anschließend werden die erstellten Variablen in den Zeilen 28-30 angewendet, um den Namen der Hauptkategorie des Katalogs auszugeben und einen Link auf diese zu erstellen, welcher auf die *prodsubtype.jsp*[51]-Seite verweist und dabei ihre eindeutige Nummer in der Anwendung als PK-Parameter weitergibt. Für das erste Element *ProdType* lautet der dabei erzeugte Link: *prodsubtype.jsp?PK=1*. Die restlichen Zeilen 37-39 des Listings schließen die geöffneten Tags.

Die bereits dargestellte XML-Datei wird zusammen mit der XSL-Datei, welche die Transformationsregeln enthält, dem XSLT-Prozessor zur Verfügung gestellt. Abhängig von der Transformationsdatei können Ausgabeformate wie HTML, XML, WML und andere erzielt werden. Der Prozess der Transformation wird in der Abbildung 4.6 veranschaulicht.

[49] Siehe Zeilen 9-10 in Listing 4.10.
[50] Siehe Zeilen 13-14 in Listing 4.10.
[51] Siehe Supplementum, Listing B.3 auf Seite 13.

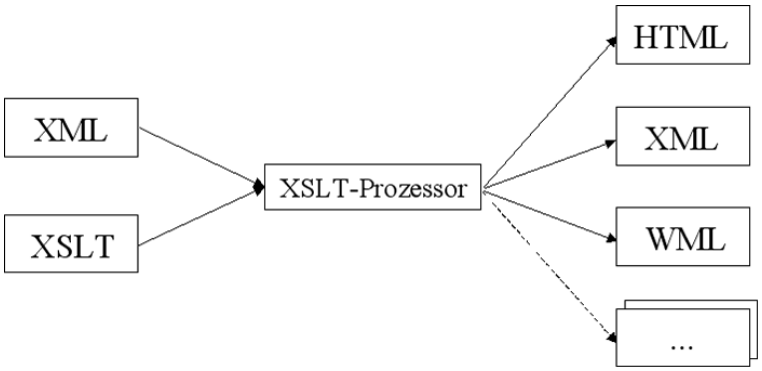

Abbildung 4.6: Transformation mit XSLT

Mit XSLT ist jedoch keine Formatierung möglich. Diese kann aber mit dem Standard XSL-FO erzielt werden. Das Ergebnis sind Formate wie PDF und PostScript.

4.4.2 XSL-FO: Extensible Stylesheet Language

XSL-FO wird auch noch als XML Stylesheet Language bezeichnet. Im unterschied zu XSLT besteht die Aufgabe von XSL-FO (Adler, 2001) darin, die XML-Daten in einem seitenorientierten Format auszugeben. Hiermit ist z.B. eine exakte Positionierung von Objekten gemeint oder die Ausgabe von Rahmen oder Tabellen mit den Daten. Dazu werden XSL und XSLT kombiniert, um die XML-Daten auf der Seite zu ordnen und anschließend zu transformieren.

Die Seitenbereiche eines XSL-FO-Dokuments werden in der Abbildung 4.7 dargestellt. Dabei ist es möglich, die Größe jedes dieser Bereiche flexibel zu bestimmen und pixelweise den Inhalt auf der Seite zu positionieren, worin auch die Stärke von XSL-FO besteht.

Transformation von XML nach PDF-Format

Die Transformation von XML in PDF-Formate dient zum Erstellen von druckbaren Dateien. Das kann z.B. beim Erstellen von Reports wichtig sein. Die praktische Umsetzung in der Auktionssystem-Anwendung wurde mit Hilfe eines Servlets realisiert.

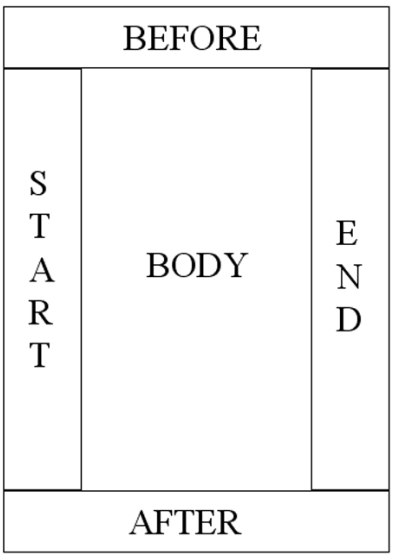

Abbildung 4.7: Die Seitenbereiche eines FO-Dokuments,
Quelle: Knobloch (2002, S. 95)

Es wurde bereits ein Katalog der existierenden Kategorien mit deren zugehörigen Unterkategorien im PDF-Format erstellt. Dabei werden noch die zur Zeit laufenden Auktionen gezählt und zu jeder Unterkategorie aufgelistet.

Die Servlet-Transformation wird durch das Servlet *FopServlet* im Paket *myapplication.xml* realisert, das folgende Schritte durchläuft:

- Zuerst wird das XML-Dokument mit den eigentlichen Daten durch den Aufruf der Methode *getDocumentFromDBData()* von der Klasse *XmlProductData* erstellt.

- Das erstellte XML-Dokument wird in die Datei *catalog.xml* in das Root-Verzeichnis der Anwendung geschrieben.

- In zwei *String*-Objekten werden danach die Pfade zur XML- und zur XSL-Datei zugewiesen.

- Die entsprechenden *String*-Objekte werden einem *XSLTInputHandler* weitergegeben.

Nach dem Programmieren des Servlets muss dieses in der *web.xml*-Datei bekannt gemacht werden, um seinen Aufruf zur ermöglichen. Die relevanten Einträge in der *web.xml*-Datei sind in Listing 4.11 dargestellt.

Listing 4.11: Ausschnitt aus der *web.xml*-Datei

```
1  ...
2 <servlet>
3     <servlet-name>fopservlet</servlet-name>
4     <servlet-class>myapplication.xml.FopServlet</servlet-class>
5 </servlet>
6  ...
7 <servlet-mapping>
8     <servlet-name>fopservlet</servlet-name>
9     <url-pattern>/fopservlet</url-pattern>
10 </servlet-mapping>
11  ...
```

Durch das *servlet*-Tag wird das Servlet *myapplication.xml.FopServlet* und sein Name *fopservlet* bekanntgegeben. Das Mapping des Servlets wird durch ein weiteres Tag, das *servlet-mapping*-Tag in den Zeilen 7-8 von Listing 4.11 realisiert. Es gibt den Namen des Servlets und die URL-Adresse zum Aufrufen des Servlets bekannt.

Der Aufruf in der Anwendung, die auf dem lokalen Server läuft, kann daher mit der Adresse

`http://127.0.0.1:8080/myapplication/fopservlet`

ausgeführt werden.

Ein Beispiel für die zu transformierende XML-Datei ins PDF-Format ist aus Listing 4.12 zu entnehmen.

Listing 4.12: Die ins PDF-Format zu transformierende XML-Datei

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2
3 <Catalog>
4     <ProdTypeData>
5         <ProdType>
6             <PK>1</PK>
7             <NAME>Lebensmittel</NAME>
8         </ProdType>
9         <ProdType>
```

```
10              <PK>2</PK>
11              <NAME>Reise</NAME>
12          </ProdType>
13          <ProdType>
14              <PK>3</PK>
15              <NAME>Tickets</NAME>
16          </ProdType>
17          <ProdType>
18              <PK>4</PK>
19              <NAME>Andere</NAME>
20          </ProdType>
21      </ProdTypeData>
22      <ProdSubSubTypeData>
23          <ProdSubSubType>
24              <SUBTYPE_PK>7</SUBTYPE_PK>
25              <SUBTYPE_NAME>Fisch</SUBTYPE_NAME>
26              <SUBSUBTYPE_PK>15</SUBSUBTYPE_PK>
27              <SUBSUBTYPE_NAME>Lachs</SUBSUBTYPE_NAME>
28              <NUMBER>0</NUMBER>
29              <PRODTYPE_PK>1</PRODTYPE_PK>
30          </ProdSubSubType>
31          <ProdSubSubType>
32              <SUBTYPE_PK>8</SUBTYPE_PK>
33              <SUBTYPE_NAME>Fleisch</SUBTYPE_NAME>
34              <SUBSUBTYPE_PK>16</SUBSUBTYPE_PK>
35              <SUBSUBTYPE_NAME>Schwein</SUBSUBTYPE_NAME>
36              <NUMBER>0</NUMBER>
37              <PRODTYPE_PK>1</PRODTYPE_PK>
38          </ProdSubSubType>
39          ...
40          ...
41      </ProdSubSubTypeData>
42  </Catalog>
```

Der dazugehörige XSL-FO-Stylesheet zur Transformation der XML-Datei ist in Listing 4.13 dargestellt.

Listing 4.13: Die XSL-Datei zur PDF-Transformation

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2
3  <xsl:stylesheet version="1.0"
4  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
5  xmlns:fo="http://www.w3.org/1999/XSL/Format">
```

```
<xsl:template match="Catalog">
   <fo:root>
      <fo:layout-master-set>
         <fo:simple-page-master
         master-name="title-page"
         page-width="21cm" page-height="29.7cm"
         margin-top="6cm" margin-bottom="1.5cm"
         margin-left="5cm" margin-right="2.5cm">
             <fo:region-body margin-top="2cm"
             margin-bottom="2cm" border="1pt"
             border-color="black" padding="20pt"/>
         </fo:simple-page-master>
         <fo:simple-page-master
         master-name="catalog-auctionsystem"
         page-width="21cm" page-height="29.7cm"
         margin-top="1cm" margin-bottom="1.5cm"
         margin-left="2.5cm" margin-right="2.5cm">
             <fo:region-body margin-top="2cm"
                             margin-bottom="2cm"/>
             <fo:region-before extent="2cm"/>
             <fo:region-after extent="1.5cm"
                              padding-before="1cm"/>
         </fo:simple-page-master>
         <fo:page-sequence-master master-name="catalog">
             <fo:single-page-master-reference
             master-reference="title-page"/>
             <fo:repeatable-page-master-reference
             master-reference="catalog-auctionsystem"/>
         </fo:page-sequence-master>
      </fo:layout-master-set>
      <fo:page-sequence master-reference="catalog">
         <fo:static-content flow-name="xsl-region-before"
                            font-size="10">
            <xsl:call-template name="header"/>
         </fo:static-content>
         <fo:static-content flow-name="xsl-region-after">
            <xsl:call-template name="footer"/>
         </fo:static-content>
         <fo:flow flow-name="xsl-region-body">
            <xsl:call-template
                        name="cover-page-content"/>
            <fo:block>
```

```
48                  <xsl:call-template name="prodtype"/>
49              </fo:block>
50              <fo:block id="endofdoc"
51                        color="white">l</fo:block>
52          </fo:flow>
53       </fo:page-sequence>
54    </fo:root>
55 </xsl:template>
56 <xsl:template match="ProdTypeData">
57     <xsl:call-template name="prodtype"/>
58 </xsl:template>
59 <xsl:template name="prodtype">
60     <fo:table width="440pt">
61        <fo:table-column column-width="110pt"
62                         column-number="1"/>
63        <fo:table-column column-width="110pt"
64                         column-number="2"/>
65        <fo:table-column column-width="110pt"
66                         column-number="3"/>
67        <fo:table-header table-omit-header-at-break="false">
68             ... /Tabellenkopf definieren/
69        </fo:table-header>
70        <fo:table-body>
71             ... /Tabelle mit Inhalt fuellen/
72        </fo:table-body>
73      </fo:table>
74 </xsl:template>
75 <xsl:template name="cover-page-content">
76     ...
77 </xsl:template>
78 <xsl:template name="header">
79     ...
80 </xsl:template>
81 <xsl:template name="footer">
82     ...
83 </xsl:template>
84 </xsl:stylesheet>
```

Die erste Zeile des Listings 4.13 deutet darauf hin, dass es sich dabei um eine XML-Datei handelt. Das *fo:root*-Tag in der Zeile 7 der Stylesheet-Datei und seine Kind-Elemente sind die wichtigsten Bestandteile des Stylesheets, die die Struktur der zu erzeugenden Datei festlegen. Es enthält folgende zwei

Elemente:

- *fo:layout-master-set* und
- *fo:page-sequence*.

Mit dem *layout-master-set* können *simple-page-master*-Elemente definiert werden. Diese Elemente besitzen einen Namen und definieren Maße wie Seitengröße, Größe der Seitenränder und Seitenbereiche, Abstände zwischen den Seitenbereichen. In Listing 4.13 sind zwei solche Elemente ab den Zeilen 9 und 18 erkennbar. Das erste definiert die Größe der Titelseite, ihre Ränder und einen *body*-Bereich mit dem Kind-Element *fo:region-body*. Das zweite *simple-page-master*-Element definiert dieselben Größen für alle restlichen Seiten. Zusätzlich sieht es zwei weitere Bereiche in der Seite vor: Einen Kopfbereich, definiert mit dem Element *fo:region-before* in Zeile 25 und einen Fußbereich, definiert mit dem Element *fo:region-after* in Zeile 26.

Mit dem *layout-master-set* wird weiterhin die Abfolge der Seiten festgelegt. Diese Aufgabe übernimmt das Element *fo:page-sequence-master* in der Zeile 29. Es legt fest, dass die Seiten, die mit dem Page-Mater *catalog-auctionsystem* definiert wurden den Seiten folgen, die mit dem Page-Master *title-page* definiert sind.

Das zweite Kind-Element des *fo:root*-Elements ist *fo:page-sequence* und wird ab Zeile 36 definiert. Dieses Element legt den Inhalt für die Seiten und seine Formatierung fest. Das Attribut *master-reference* legt fest, welcher Page-Master verwendet wird. Er verweist in der Zeile 36 auf den Page-Sequence-Master *catalog*.

Das Element *fo:page-sequence* enthält die Elemente *fo:static-content* und *fo:flow*. Das erste Element *fo:static-content* in Zeile 37-38 legt die Schriftgröße 10 für den Kopfbereich der Seiten und seinen Inhalt durch das Template *header* fest. Das Template *header* ist in Zeile 78 definiert. Analog ist in der Zeile 42 im zweiten Element *fo:static-content* die Formatierung und der Inhalt für den Fußbereich der Seite festgelegt. Mit dem Element *fo:flow* in Zeile 44 wird der eigentliche Inhalt der Seite festgelegt. Das Attribut *flow-name* definiert den *body*-Bereich, welcher mit dem Template *cover-page-content* (siehe Zeile 75) festgelegt wird. Weiterhin folgen zwei *fo:block*-Elemente. Das erste verweist auf das Template *prodtype* ab Zeile 59. Hier wird eine Tabelle mit ihren Maßen und Inhalten definiert, welche die eigentlichen Daten mit der Anzahl der Produkte für jede Unterkategorie bekanntgibt.

4.5 Beschreiben von XML-Dokumenten

4.5.1 Document Type Definition (DTD)

Eine *Document Type Definition* eines XML-Dokuments dient zur Definition der Struktur von XML-Daten, indem sie die Bestandteile eines XML-Dokuments und deren Beziehungen zueinander festlegt. Eine DTD kann intern ins XML-Dokument mit den Nutzdaten eingebunden werden oder kann als externe Quelle referenziert werden.

Schöning (2003) stellt drei Vorteile vor, welche eine DTD mit sich bringt:

- *Garantieaspekt*: Das Vorhandensein einer DTD erlaubt die Prüfung, ob ein XML-Dokument einer bestimmten Struktur gehorcht. Eine Anwendung kann sich hiermit vor der Verarbeitung der Daten darauf verlassen, dass diese auch die benötigten Daten liefert.
- *Informationsaspekt*: Über eine DTD können Schlussfolgerungen darüber gemacht werden, aus welchen Daten ein XML-Dokument bestehen soll und wie die Struktur der Daten aussieht. Mit diesen Informationen kann die Entwicklung der Anwendung unterstützt werden.
- *Modularisierungsaspekt*: Die Auslagerung einer DTD in eine externe Datei erlaubt den mehrfachen Verweis auf diese Datei.

Ein Beispiel für ein XML-Dokument mit einer internen DTD-Datei zeigt Listing 4.14.

Listing 4.14: Interne DTD

```
1  <?xml version="1.0"?>
2  <!DOCTYPE products [
3    <!ELEMENT products (product)+>
4    <!ELEMENT product (number, name, type)>
5    <!ELEMENT number  (#PCDATA)>
6    <!ELEMENT name    (#PCDATA)>
7    <!ELEMENT type    (#PCDATA)>
8  ]>
9
10 <products>
11     <product>
12         <number>123</number>
13         <name>Zeitschrift Wirtschaftsinformatik 3/2003</name>
14         <type>Zeitschriften</type>
```

```
15      </product>
16  </products>
```

Die Zeilen 2-8 bestimmen die Struktur der XML-Daten. Die DTD-Datei ist intern, da sie sich innerhalb der XML-Datei, die sie beschreibt, befindet. Das *root*-Element des Dokuments ist das *products*-Element. In Zeile 3 wird festgelegt, dass es aus untergeordneten *product*-Elementen bestehen kann. Durch das +-Zeichen wird bekanntgegeben, dass die Anzahl dieser Elemente beliebig sein kann. Das *product*-Element besteht, wie in Zeile 4 festgelegt, aus den Kind-Elementen *number*, *name* und *type*. In den Zeilen 5-7 werden mit *#PCDATA* diese Elemente als solche mit Textinhalten definiert.

Der Inhalt von XML-Dokumenten kann auch mit einer *externen DTD* beschrieben werden. Die inhaltsbeschreibenden Zeilen können, wie in Listing 4.15 dargestellt, in eine externe DTD-Datei ausgelagert werden.

Listing 4.15: Externe DTD

```
1  <!DOCTYPE products [
2    <!ELEMENT products (product)+>
3    <!ELEMENT product (number, name, type)>
4    <!ELEMENT number  (#PCDATA)>
5    <!ELEMENT name    (#PCDATA)>
6    <!ELEMENT type    (#PCDATA)>
7  ]>
```

Wenn der obige Quelltext in einer Datei mit dem Namen *productlist.dtd* gespeichert ist, wird die XML-Datei mit dem Verweis auf ihre beschreibende Datei wie in Listing 4.16 dargestellt.

Listing 4.16: XML-Dokument mit Verweis auf die DTD-Datei

```
1  <?xml version="1.0"?>
2  <!DOCTYPE Produktliste SYSTEM "productlist.dtd">
3
4  <products>
5      <product>
6          <number>123</number>
7          <name>Zeitschrift Wirtschaftsinformatik 3/2003</name>
8          <type>Zeitschriften</type>
9      </product>
10 </products>
```

Mit dem Element DOCTYPE wird in der Zeile 2 auf eine DTD mit dem Namen *Produktliste* verwiesen. Das Schlüsselwort SYSTEM verweist auf eine interne

Datei. Anschließend wird die Datei *productlist.dtd* festgelegt.

4.5.2 XML Schema

Das XML Schema erlaubt im Unterschied zu einer DTD eine genauere Beschreibung eines XML-Dokuments. Es wurde im Mai 2001 von der W3C in der Form einer Recommendation verabschiedet und besitzt eine XML Syntax.

Das XML Schema ist die komplexeste Empfehlung des W3C und besteht aus drei Teilen:

- *XML Schema Part 0*: Beschreibt die Erstellung eines XML Schemas, welches in den nächsten zwei Teilen der Empfehlung vertieft wird. (Fallside, 2001)
- *XML Schema Part 1*: Ziel dieses Teils ist, Bestandteile eines XML-Dokuments wie Elemente, Datentypen und deren Inhalt, Attribute und Werte zu definieren. (Thompson u.a., 2001)
- *XML Schema Part 2*: Der zweite Teil beschäftigt sich mit der Datentypdefinition. (Biron u.a., 2001)

Ein XML Schema ermöglicht im Unterschied zu einer DTD die Eingabe von Datentypen und Wertebereichen für die Inhalte der Elemente oder Attribute. Es benutzt zwei Datentypen: Die einfachen und die komplexen Datentypen. Schöning (2003, S. 34) definiert diese wie folgt:

- „Ein *einfacher Datentyp (simple type)* im Sinne des XML Schema ist ein Datentyp, der weder Attribute noch Kindelemente beinhaltet – informell gesprochen also ein Datentyp, der geeignet ist, Attributwerte und den Inhalt von Elementen zu beschreiben, die nur textuellen Inhalt haben.“
- „Ein *komplexer Typ (complex type)* besteht hingegen aus einer (ggf. leeren) Menge von Attributdeklarationen und einem Inhaltsmodell. Dieses Inhaltsmodell kann Kindelemente fordern oder verbieten, textuellen Inhalt eines bestimmten einfachen Typs zulassen und auch gemischten Inhalt *(mixed content)* vorsehen.“

Elemente, Attribute und Typen können im XML Schema lokal oder global definiert werden.

- *Lokale Definition*: Eine Definition, die in anderer Definitionen eingebettet und damit nicht wiederverwendbar ist.

- *Globale Definition*: Diese Definition befindet sich auf dem obersten Level des XML Schemas. Sie kann innerhalb des ganzen Schemas referenziert werden und ist damit wiederverwendbar.

Die weitere Vertiefung des Themas XML Schema würde über die gesetzten Grenzen dieser Arbeit hinausgehen. An dieser Stelle soll allein ein Beispiel für eine Produktliste aus der Auktionssystem-Anwendung in Listing 4.17 veranschaulicht werden.

Listing 4.17: Das XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema elementFormDefault="qualified"
 xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="CurrentPrice" type="xs:byte"/>
    <xs:element name="LeftDays" type="xs:byte"/>
    <xs:element name="LeftHours" type="xs:byte"/>
    <xs:element name="LeftMins" type="xs:byte"/>
    <xs:element name="Name" type="xs:string"/>
    <xs:element name="Pk" type="xs:byte"/>
    <xs:complexType name="ProductType">
        <xs:sequence>
            <xs:element ref="Pk"/>
            <xs:element ref="Name"/>
            <xs:element ref="SubSubTypePk"/>
            <xs:element ref="LeftDays"/>
            <xs:element ref="LeftHours"/>
            <xs:element ref="LeftMins"/>
            <xs:element ref="TotalBidNr"/>
            <xs:element ref="CurrentPrice"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="SubSubTypePk" type="xs:boolean"/>
    <xs:element name="SubSubTypeProdList">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Product" type="ProductType"
                maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="SubSubType_Name"
             type="xs:string" use="required"/>
            <xs:attribute name="SubSubType_Pk"
             type="xs:boolean" use="required"/>
            <xs:attribute name="SubType_Name"
```

```
34                type="xs:string" use="required"/>
35              <xs:attribute name="Type_Name"
36                type="xs:string" use="required"/>
37          </xs:complexType>
38      </xs:element>
39      <xs:element name="TotalBidNr" type="xs:boolean"/>
40  </xs:schema>
```

Aus der Zeile 1 in Listing 4.17 ist ersichtlich, dass es sich beim XML Schema um ein XML-Dokument handelt, dessen *root*-Element den Namen *xs:schema* trägt. Es und seine Kind-Elemente gehören dem Namensraum *xs*, dessen URI die Zeile 3 definiert. Die Zeilen 4-9, 22, 39 definieren einfache Datentypen für die Kind-Elemente des *Product*-Elements aus Listing 4.18 und weisen jedem Element einen zulässigen Wertebereich zu. Die Zeilen 10-21 definieren den komplexen Datentyp *ProductType*, welcher eine Sammlung der bereits definierten einfachen Datentypen vornimmt. Die Zeilen 23-38 definieren das Element *SubSubTypeProdList*. Es stellt einen komplexen Typ dar, welcher aus der Folge des bereits definierten *ProductType*-Datentyps in Zeile 26 und aus den obligatorischen Attributen *SubSubType_Name* in Zeile 29, *SubSubType_Pk* in Zeile 31, *SubType_Name* in Zeile 33 und *Type_Name* in der Zeile 35 besteht. Ein Beispiel für ein XML-Dokument, das den Regeln des oben dargestellten Schemas gehorcht, ist in Listing 4.18 dargestellt.

Listing 4.18: Das XML-Dokument zum XML Schema

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <SubSubTypeProdList
3   SubSubType_Name="Fahrkarten" SubSubType_Pk="1"
4   SubType_Name="Bahntickets" Type_Name="Reise"
5   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6   xsi:noNamespaceSchemaLocation="D:\productList.xsd">
7       <Product>
8           <Pk>7</Pk>
9           <Name>Bahnticket nach Berlin am 25.05.05</Name>
10          <SubSubTypePk>1</SubSubTypePk>
11          <LeftDays>4</LeftDays>
12          <LeftHours>23</LeftHours>
13          <LeftMins>56</LeftMins>
14          <TotalBidNr>0</TotalBidNr>
15          <CurrentPrice>120</CurrentPrice>
16      </Product>
17 </SubSubTypeProdList>
```

4.6 Sicherheitsstandards für XML

Dieser Abschnitt der Arbeit beschäftigt sich mit den vorhandenen Standards für den sicheren Austausch von XML-Daten. Es werden zunächst die Anforderungen an eine digitale Signatur dargestellt. Danach werden der Aufbau und die Schritte zur Generierung und Verifizierung einer Signatur erklärt.

Wie im Abschnitt 3.1.3 verdeutlicht, ist die Sicherheit von Daten eine wichtige Anforderung an Webanwendungen. Der Einsatz von XML in Webanwendungen erfordert daher einen möglichst sicheren Datenaustausch.

XML-Daten sind textbasiert und können daher signiert und verschlüsselt werden. Die W3C hat zwei Vorschläge vorbereitet, die die Erstellung von XML-Signaturen und die Verschlüsselung von XML-Daten unterstützen:

- *XML Digital Signature* (Bartel u.a., 2002) und
- *XML Encryption* (Imamura u. a., 2002).

Im Folgenden wird der Aufbau von Digitalen Signaturen mit XML und deren Erstellung mit einem praktischen Beispiel aus der Anwendung vorgestellt.

4.6.1 Digitale Signaturen für XML

Digitale Signaturen für XML sollen nach Kollmorgen u. a. (2003, S. 42) wie alle Signaturen folgende Anforderungen erfüllen:

- „Eine Signatur gehört zu genau einer Person.
- Die Signatur darf nicht unbemerkt gefälscht werden können.
- Ein signiertes Dokument darf nicht unbemerkt geändert werden.“

Die Empfehlung *XML Digital Signature* von W3C stellt ein XML Schema zur Erstellung von Digitalen Signaturen mit XML bereit. Hiermit kann der Urheber seine Unterschrift verbindlich unter die XML-Daten beziehungsweise auf das XML-Dokument legen. Desweiteren ist es möglich, nicht nur Dokumente zu signieren, sondern auch einzelne Bestandteile von XML-Dokumenten. Damit kann beispielsweise ein Prozess, an dem mehrere Teilnehmer beteiligt sind, in einer XML-Datei abgebildet werden. Die Beteiligten können nur diese Teile des Dokuments unterschreiben, für welche sie auch verantwortlich sind. (Ydema, 2001, S. 20)

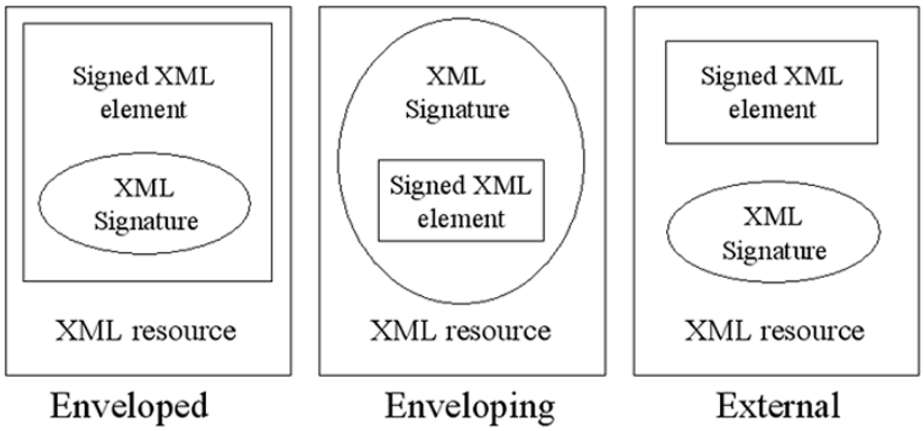

Abbildung 4.8: Position der XML-Signatur, Quelle: Ydema (2001, S. 22)

Hinsichtlich der Platzierung einer XML-Signatur stehen mehrere Möglichkeiten zur Verfügung, welche in Abbildung 4.8 veranschaulicht werden. Generell können in solchen Dokumenten Elemente der signierten Daten und Elemente der Signatur innerhalb des Dokuments identifiziert werden. Nach der Position der Digitalen Signatur im XML-Dokument werden drei Signatur-Arten unterschieden:

- *Enveloped*: Das Signatur-Element befindet sich innerhalb des signierten XML-Elements.
- *Enveloping*: Das signierte XML-Element ist innerhalb des Signatur-Elements platziert.
- *External*: Die Signatur befindet sich außerhalb des Dokuments mit den signierten Daten.

Die Struktur einer XML-Signatur ist in der Abbildung 4.9 dargestellt. Das erste Element der Signatur ist das *SignedInfo*-Element. Es enthält die signierten Nutzdaten und besitzt auch weitere Elemente mit Informationen zu der Signatur. Diese Elemente sind *CanonicalizationMethod*, *SignatureMethod* und *Reference*. Das letzte Element *Reference* kann weitere Kindelemente mit Informationen wie Transformationen von der Berechnung des Hash-Wertes, den Algorithmus und den eigentlichen Hash-Wert der Nachricht enthalten.

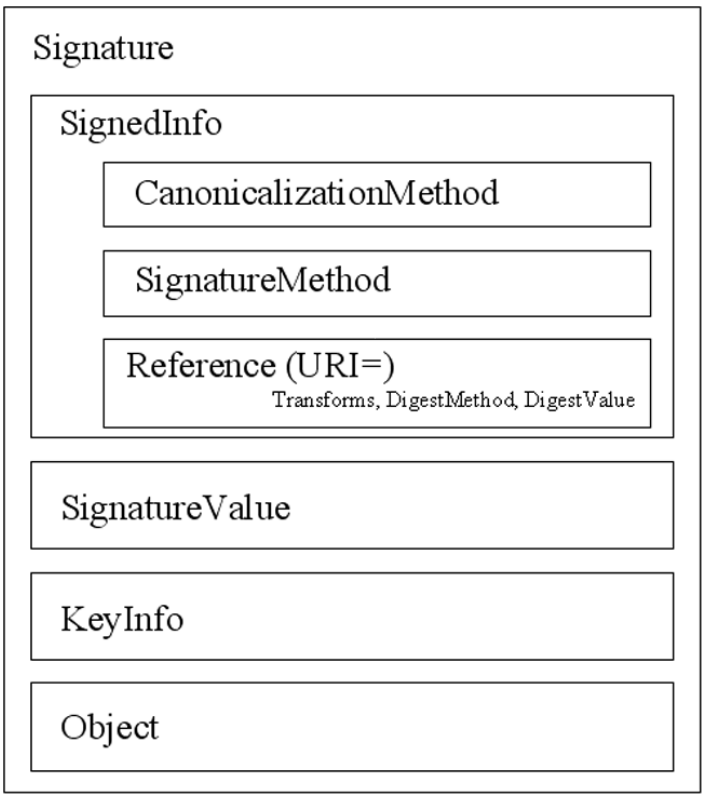

Abbildung 4.9: XML Signatur-Struktur, Quelle: Seemann (2001, S. 91)

Auf der Ebene des *SignedInfo*-Elements befindet sich in der Abbildung 4.9[52] das *SignatureValue*-Element. Es hält die Signatur des Dokuments. Das nächste *KeyInfo*-Element wird dazu genutzt, Informationen über den öffentlichen Schlüssel des Unterzeichners aufzubewahren. Anschließend kann das Element *Object* im XML-Dokument zusätzliche Informationen enthalten wie beispielsweise den Zeitpunkt der Signaturerstellung. Die Informationen in diesem Element sind nicht festgelegt worden und können beliebig sein.

Nachdem die Struktur einer XML-Signatur veranschaulicht wurde, soll hier die Erstellung erläutert werden. Die Abbildung 4.10 von Kollmorgen u. a. (2003) veranschaulicht die Schritte der Generierung einer digitalen Signatur.

Als Erstes wird ein komprimierter Wert der Daten mit Hilfe einer Hash-

[52]Layout der Abbildung wurde vom Autor leicht modifiziert.

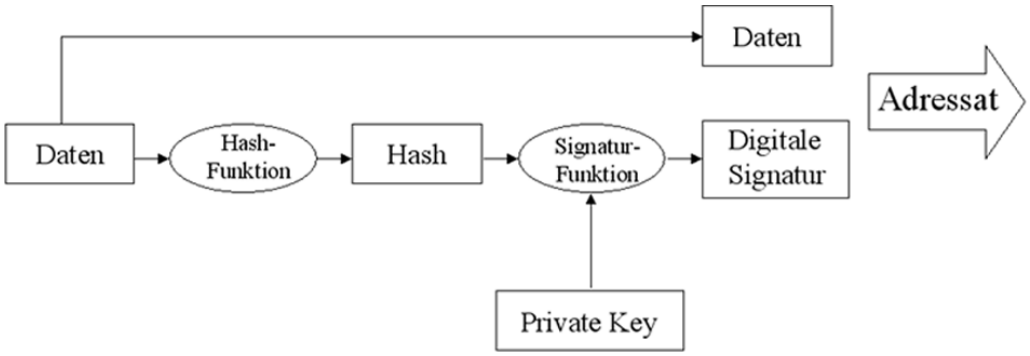

Abbildung 4.10: Generierung einer digitalen Signatur,
Quelle: Kollmorgen u. a. (2003, S. 43)

Funktion berechnet. Hier wird (in der Regel) der Secure Hash[53] Algorithmus SHA[54] eingesetzt. Das Ergebnis ist der *Hash-Wert* der Nachricht, der auch *Message Digest* oder *Fingerprint* genannt wird. Er ist eindeutig zu den Daten zugeordnet und nicht umkehrbar, d.h. eine zweite Berechnung würde zum selben Ergebnis führen, aber ein Rückschluss vom Hash-Wert auf die Daten darf nicht möglich sein.

Als Zweites wird der Hash-Wert der Daten mit Hilfe der Signatur-Funktion mit dem privaten Schlüssel des Urhebers verschlüsselt. Als Signatur-Funktionen werden Verfahren wie RSA[55], DSA[56] eingesetzt. Das Ergebnis der letzten Operation ist der verschlüsselte Hash-Wert der Daten, welcher die digitale Signatur darstellt.

Die Verifizierung der Signatur erfolgt, wie aus Abbildung 4.11 nach Kollmorgen u. a. (2003) ersichtlich, mit dem zum privaten Schlüssel des Urhebers zugehörigen öffentlichen Schlüssel. Dabei wird die Signatur der Daten entschlüsselt, was den Hash-Wert der Daten liefert. Des Weiteren wird von den Daten im Klartext mit der angegebenen Methode der Hash-Wert ermittelt. Dieser Wert muss anschließend mit den entschlüsselten Daten verglichen werden. Eine Übereinstimmung bedeutet Gültigkeit der Signatur, unterschiedliche Hash-Werte dagegen deuten darauf hin, dass die Unterschrift gefälscht wurde.

[53] to hash: (engl.) zerhacken
[54] SHA (Secure Hash Algorithm): Das sind kryptographische Funktionen, die beim Signieren von Daten eingesetzt werden. Vgl. Schmeh (2001, S. 140 ff.).
[55] Vgl. Schmeh (2001, S. 107 ff.).
[56] Vgl. Schmeh (2001, S. 122).

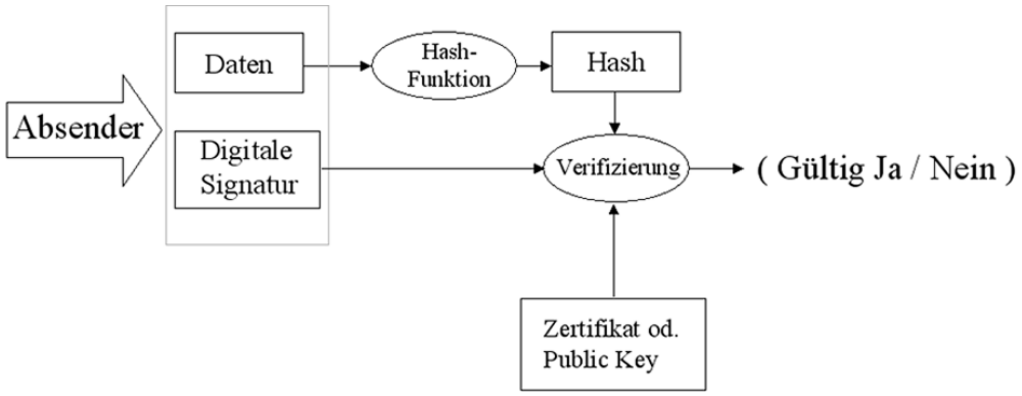

Abbildung 4.11: Validierung einer digitalen Signatur,
Quelle:Kollmorgen u. a. (2003, S. 43)

Ein einfaches Beispiel einer XML-Signatur ist in Listing 4.19 zu sehen.

Listing 4.19: Einfache XML-Signatur

```
<Signature Id="MyFirstSignature"
 xmlns="http://www.w3.org/2000/09/xmldsig#">
   <SignedInfo>
   <CanonicalizationMethod
         Algorithm=
         "http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
   <SignatureMethod
         Algorithm=
         "http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
   <Reference
    URI="http://www.w3.org/TR/2000/REC-xhtml1-20000126/">
     <Transforms>
           <Transform Algorithm=
         "http://www.w3.org/TR/2001
         /REC-xml-c14n-20010315"/>
     </Transforms>
     <DigestMethod Algorithm=
         "http://www.w3.org/2000/09/xmldsig#sha1"/>
     <DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
   </Reference>
 </SignedInfo>
   <SignatureValue>MC0CFFrVLtRlk=...</SignatureValue>
```

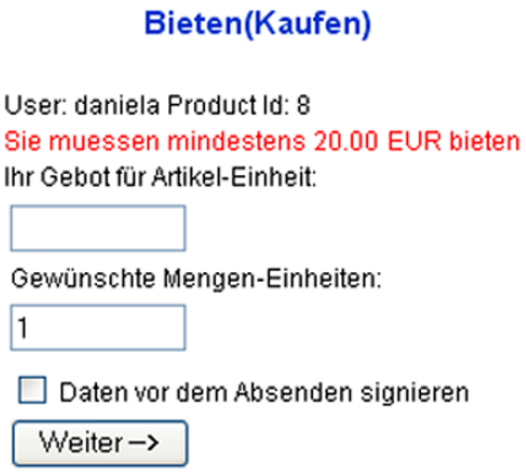

Abbildung 4.12: Das Gebot-Formular mit Signatur-Option

```
23    <KeyInfo>
24     <KeyValue>
25       <DSAKeyValue>
26         <P>...</P><Q>...</Q><G>...</G><Y>...</Y>
27       </DSAKeyValue>
28     </KeyValue>
29    </KeyInfo>
30  </Signature>
```

4.6.2 Erstellen der Gebot-Signatur innerhalb der Anwendung

In der Auktionssystem-Anwendung wurde die XML-Signatur verwendet, um die abgegebenen Gebote zu signieren. Dieser Vorgang leistet die Unabstreitbarkeit der abgegebenen Gebote der Auktionsteilnehmer.

Wie in der Abbildung 4.12 gezeigt, besteht neben der Möglichkeit zur Abgabe der Menge und dem Gebotspreis per Menge auch die Möglichkeit zum Signieren des Gebots beim Auswählen der Check-Box „Daten vor dem Absenden signieren“. Bei der Wahl dieser Funktion wird ein Applet aufgerufen, das für die Signatur der XML-Daten sorgt. Er soll dabei auf eine lokale Datei mit dem privaten und öffentlichen Schlüssel auf die Festplatte zugreifen und mit

diesen das XML-Dokument mit den Gebotsdaten signieren. Die Vorbereitung der Signatur wurde in Anlehnung an Prokhorenko und Prokhorenko (2004) erstellt und hat folgenden Ablauf:

- Das Applet wird programmiert. Auf den Quelltext wird auf späteren Seiten eingegangen.
- Eine JAR-Datei mit dem Applet und allen anderen Klassen der Anwendung, die von diesem beansprucht wurden, wird angelegt. Dazu kann hier die Möglichkeit von Eclipse genutzt werden, auf externe Tools zuzugreifen. Es wird das Programm *jar.exe* vom JDK in Anspruch genommen. Die erstellte JAR-Datei wird *XmlSigApplet.jar* genannt und enthält alle Klassen des Pakets *myapplication.xml*. Ein Bild mit den Einstellungen in Eclipse wird durch die Abbildung 4.13 auf Seite 189 angezeigt.
- Ein Schlüsselpaar wird erzeugt, das zum Signieren der erstellten JAR-Datei dienen soll. Dazu wird das *keytool* aus dem JDK in Anspruch genommen.

  Hier ein Auszug aus dem Vorgang im DOS-Fenster:

```
keytool -genkey -alias XmlSigApplet -validity 1000


Geben Sie das Keystore-Passwort ein:  040531
Wie lautet Ihr Vor- und Nachname?
 [Unknown]:  Daniela Stoykova
Wie lautet der Name Ihrer organisatorischen Einheit?
 [Unknown]:  privat
Wie lautet der Name Ihrer Organisation?
 [Unknown]:  auktionssystem
Wie lautet der Name Ihrer Stadt oder Gemeinde?
 [Unknown]:  hd
Wie lautet der Name Ihres Bundeslandes oder Ihrer
Provinz?
 [Unknown]:  bw
Wie lautet der Landescode (zwei Buchstaben) für diese
Einheit?
 [Unknown]:  de
Ist CN=Daniela Stoykova, OU=privat, O=auktionssystem,
L=hd, ST=bw, C=de richtig?
```

```
 [Nein]:  ja

Geben Sie das Passwort für <XmlSigApplet> ein.
(EINGABETASTE, wenn Passwort dasselbe wie für Keystore):
```

- Die JAR-Datei wird mit den bereits erstellten Schlüsseln signiert. Um die Datei zu signieren, wurde das Tool *jarsigner* aus dem JDK eingesetzt. Es kann im DOS-Fenster mit dem folgenden Befehl aufgerufen werden:

  ```
  jarsigner JAR-Dateiname Alias
  ```

  Damit sieht im konkreten Fall der Befehl wie folgt aus:

  ```
  jarsigner XmlSigApplet.jar XmlSigApplet
  ```

  Dieser Befehl fragt zuerst nach dem Keystore-Passwort und signiert anschließend die JAR-Datei.

Beim Aufrufen des Applets wird zuerst das Fenster mit der Abbildung 4.14 auf Seite 190 angezeigt. Bevor das Applet geladen wird, soll der Benutzer die Installation und das Ausführen des Applets bestätigen. Nachdem die Einwilligung gegeben wurde, erscheint das Fenster mit dem eingebetteten Applet wie auf Abbildung 4.15 auf Seite 191 gezeigt.

Bevor die Signatur durchgeführt werden kann, müssen dem JRE auf dem Client in der Datei $JAVA_HOME/jre/lib/security/java.policy noch folgende SECURITY PERMISSIONS vergeben werden[57]:

```
permission java.util.PropertyPermission
    "org.apache.xml.security.resource.config", "read";
permission java.lang.RuntimePermission
    "accessClassInPackage.sun.security.provider";
```

Außerdem muss im Java Plug-in auf dem Client das installierte JRE ausgewählt werden (falls mehrere installiert wurden) und der Pfad zu den benötigten Applet-Klassen. Diese Einstellungen sind der Abbildung 4.16 auf Seite 192 zu entnehmen.

[57] Vgl. http://java.sun.com/developer/JDCTechTips/2001/tt0130.html (1.06.2004).

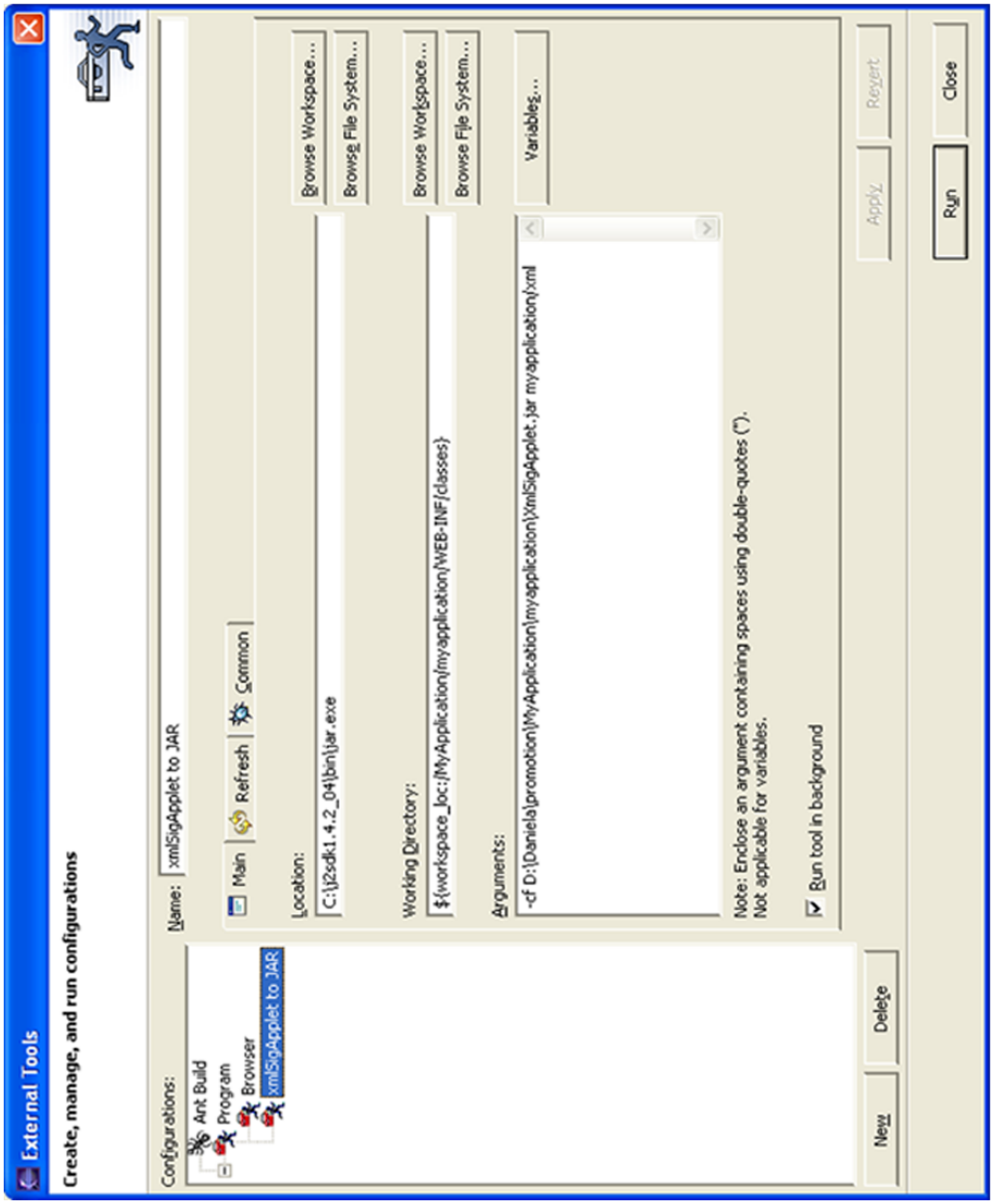

Abbildung 4.13: Erstellen der applet.jar-Datei mit Eclipse

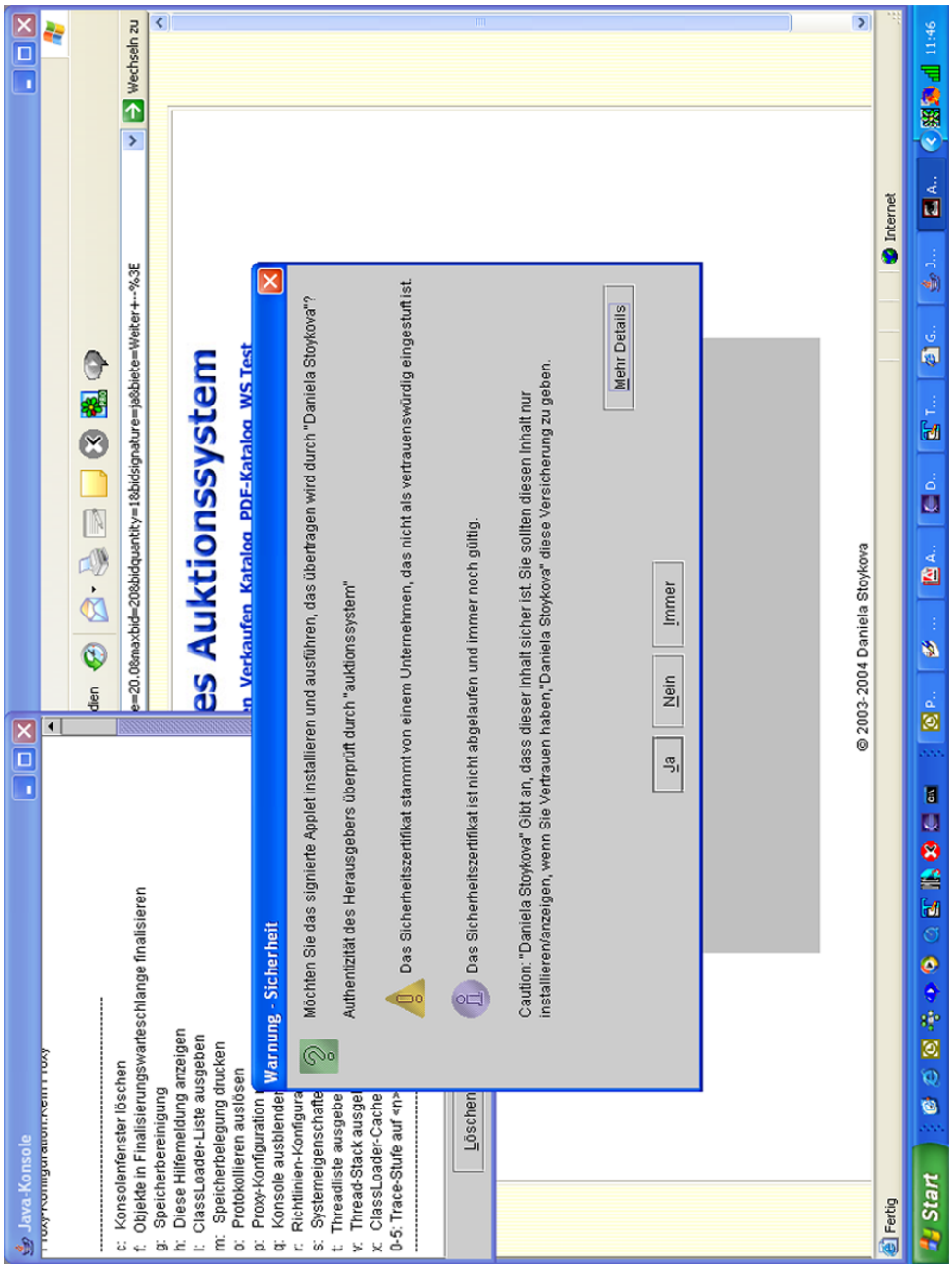

Abbildung 4.14: Aktivieren des Signatur-Applets

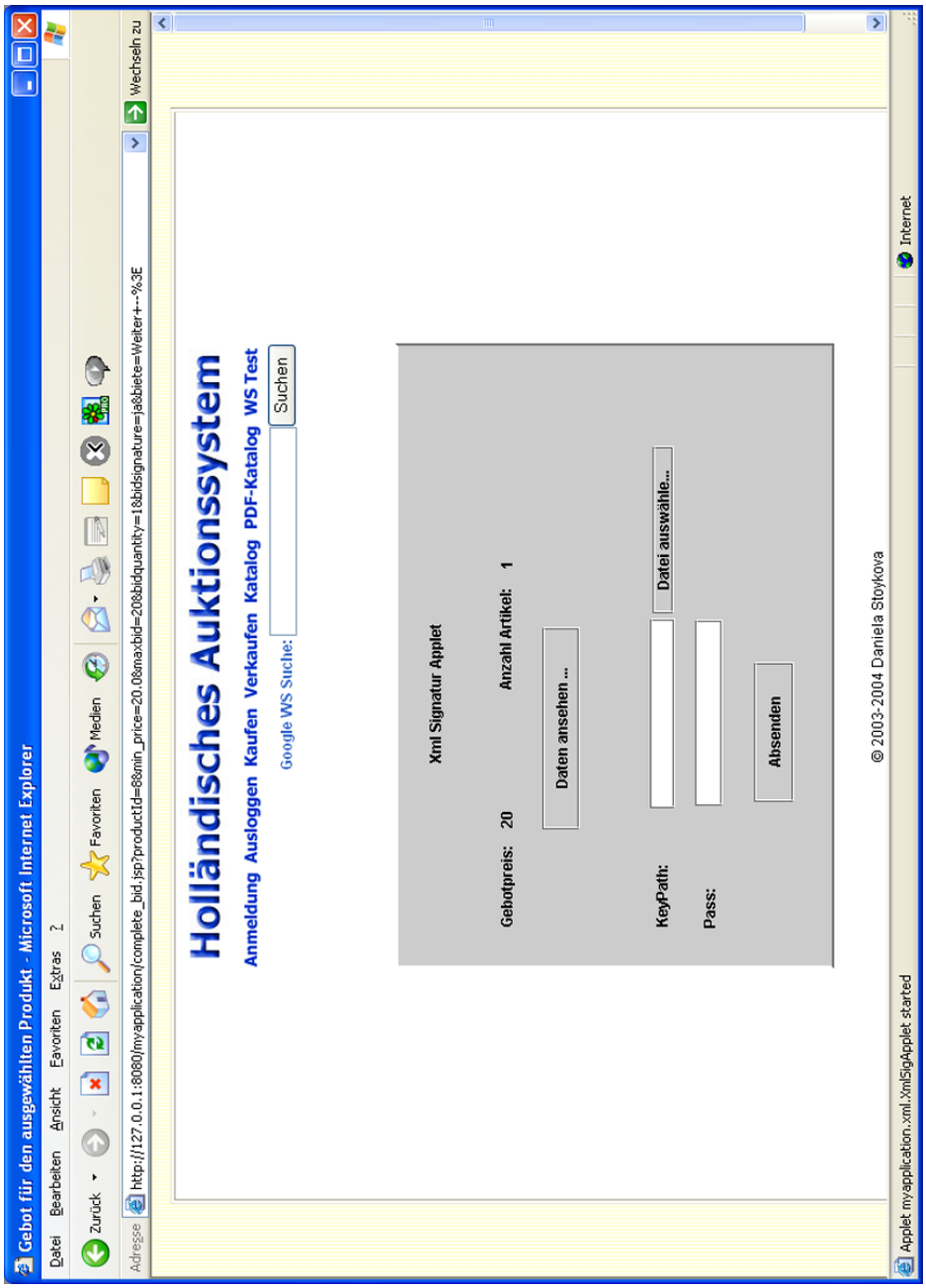

Abbildung 4.15: Signatur-Applet des Gebots

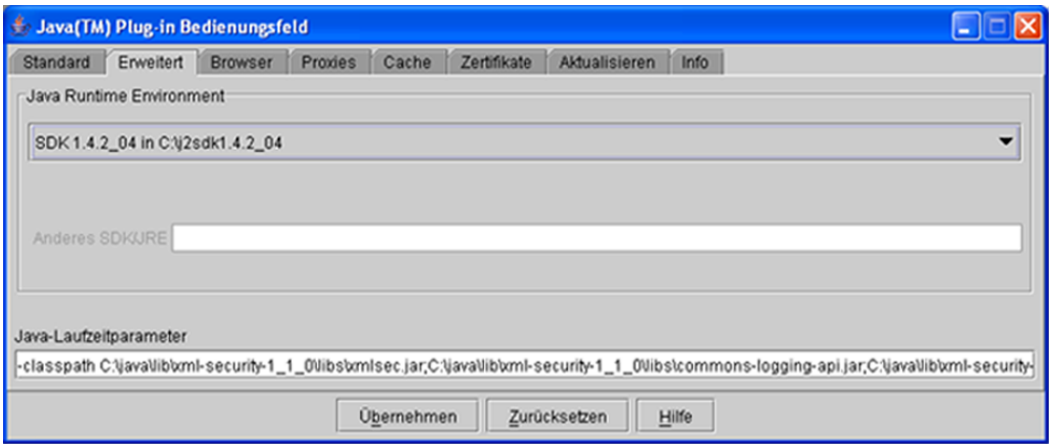

Abbildung 4.16: Java Plug-in-Fenster für die Einstellungen des Java Runtime Environments

Ablauf der Signatur

Für das Starten des Applets wird die Seite *applet.jsp* geladen. Sie bettet das Applet aus Listing 4.20 ein.

Listing 4.20: Quelltext zum Einbetten des Applets aus der Seite *applet.jsp*

```
<OBJECT classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
        width="500" height="400" align="middle">
        <PARAM NAME="code"
               VALUE="myapplication.xml.XmlSigApplet">
        <PARAM NAME="codebase" VALUE=".">
        <PARAM NAME="archive" VALUE="XmlSigApplet.jar">
        <PARAM NAME="prodId" VALUE="<%= sProdId %>">
        <PARAM NAME="bidPrice" VALUE="<%= sBidPrice %>">
        <PARAM NAME="bidQuantity" VALUE="<%= sBidQuantity %>">
        <PARAM NAME="xmlToSign" VALUE="<%= sXmlToSign %>">

      <COMMENT>
      <EMBED type="application/x-java-applet;version=1.4"
               width="500" height="400" align="middle"
               code    = "myapplication.xml.XmlSigApplet"
               ARCHIVE = "XmlSigApplet.jar"

               name    = "Signature"
               codebase = "."
```

```
20              URL = "./homepage.jsp"
21              pluginspage="http://java.sun.com/products/
22                           plugin/1.4/plugin-install.html">
23       <NOEMBED>
24          Um das Applet zu benutzen, muss Java aktiviert sein.
25       </NOEMBED>
26      </EMBED>
27      </COMMENT>
28 </OBJECT>
```

Die Klasse des Applets *XmlSigApplet* im Paket *myapplication.xml* wird in der Zeile 6 bekanntgegeben. Mit den Attributen

```
width="500" height="350" align="middle"
```

des Tags *OBJECT* wird entsprechend die Breite, die Höhe und die horizontale Position des Applet-Fensters festgelegt. In den Zeilen 7-10 werden folgende vier Parameter an das Applet weitergegeben:

- *prodId* – übergibt dem Applet den PK des Produktes,
- *bidPrice* – enthält das Gebot des Auktionators für den Artikel,
- *bidQuantity* – enthält die vom Auktionator gewünschte Menge,
- *xmlToSign* – repräsentiert das XML-Dokument, das die notwendigen Daten enthält, die vom Auktionator signiert werden.

Diese Parameter können im Applet durch das Aufrufen der Methode *getParameter()* ermittelt werden. Die Methode *getParameter()* erwartet den Parameternamen und liefert den Wert des jeweiligen Parameters zurück. Innerhalb der Methode *init()* des Applets, gezeigt in Listing 4.21, werden diese Parameter in den Zeilen 3-6 ausgelesen.

Listing 4.21: Die Methode *init()* des Applets *XmlSigApplet*

```
1 public void init() {
2     try {
3         sXmlToSign = this.getParameter("xmlToSign");
4         sProdId = this.getParameter("prodId");
5         sBidPrice = this.getParameter("bidPrice");
6         sBidQuantity = this.getParameter("bidQuantity");
7         this.setSize(600, 500);
8         this.setContentPane(getJContentPane());
9     } catch (Exception e) {
```

```
10            e.printStackTrace();
11        }
12    }
```

Um die Signatur auszuführen, muss über die Schaltfläche *Datei auswählen* auf der Abbildung 4.15 die Datei mit dem privaten und öffentlichen Schlüssel gewählt werden. Des Weiteren müssen der Keystore-Namen, das Passwort des privaten Schlüssels und sein Alias angegeben werden. Nach dem Tätigen der Schaltfläche *Absenden* wird der Quelltext des Listings 4.22 ausgeführt. Zunächst wird in Zeile 3 rechts vom *Absenden*-Button der Text „Signatur erstellen ...“ angezeigt und anschließend in der Zeile 4 von Listing 4.23 die Methode *signData()* des Applets aufgerufen.

Die Methode *signData()* (Listing 4.23) erstellt in der Zeile 3 ein Objekt der Klasse *BidSignature*, um die Signatur des XML-Dokuments zu ermitteln. In der Zeile 7 wird die Variable *sSignedDoc* mit dem Wert des signierten XML-Dokument-String gesetzt. Auf die Funktionalität der Klasse *BidSignature* wird später eingegangen.

Die Methode *actionPerformed()* in Listing 4.22 bekommt in der Zeile 4 die Signatur mit dem *String*-Objekt *sigdoc* übergeben. Das signierte Dokument soll nun in die Datenbank als Text-Blob gespeichert werden. Dieser Job wird dem Servlet *SigServlet* überlassen. Auf das Eintragen der Daten in das Datenbanksystem innerhalb des Servlets wird später eingegangen. Bevor dies geschieht, muss zunächst die Kommunikation zwischen dem Applet und dem Servlet ermöglicht werden. In den folgenden Abschnitten wird in Anlehnung an Darby (1998) die Programmierung der Applet-Servlet-Kommunikation erläutert.

Die Kommunikation zwischen dem Servlet und dem Applet findet über HTTP statt. Um die Kommunikation zunächst zu ermöglichen, muss eine HTTP Socket[58]-Verbindung geöffnet werden. Sie wird in den Zeilen 5-8 des Listings 4.22 erstellt. In den Zeilen 9-12 wird sichergestellt, dass die Verbindung sowohl Daten nach außen senden als auch welche von außen empfangen kann und zusätzlich das Einsetzen keiner Cash-Werte erlaubt. In der Zeile 13 folgt die Spezifizierung, dass Binärdaten ausgetauscht werden.

Es stehen zwei Methoden für Austausch von Daten zwischen dem Applet und Servlet zur Verfügung:

[58]Socket: Schnittstelle zur Netzwerkkomunikation

- *GET*: Die Parameter des Servlets müssen der aufrufenden URL-Adresse beigefügt werden. Die Adressierung könnte beispielsweise so aussehen:

```
http://127.0.0.1:8080/myapplication/sigservlet?
                    paramName1=wert1\&paramName2=wert2
```

- *POST*: Die POST-Methode ist die mächtigere der beiden Methoden. Sie ermöglicht sowohl den Austausch von Textdaten als auch von Binär- und weiteren Datenformaten. Dazu muss, wie in Listing 4.22 Zeilen 16-18 ersichtlich, die Verbindung bekanntgegeben werden, über die Daten zum *OutputStream* gesendet werden.

Für die Kommunikation wurde hier die POST-Methode gewählt. Sie ist die geeignete Methode, da dem Servlet das gesamte XML-Dokument übergeben werden soll. Daher wird in der Zeile 16 das Objekt *oos* der Klasse *ObjectOutputStream* mit Hilfe der bereits geöffneten Verbindung *servletConnection* erstellt. Die Zeilen 19-21 serialisieren das Objekt *sigdoc*. Das eigentliche Serialisieren geschieht in der Zeile 19, mit deren Hilfe ein Binärobjekt zum *ObjectOutputStream oos* geschrieben wird.

Das Servlet bearbeitet die bereits übermittelten Daten. Ein *InputStream* kann über die Klasse *ObjectInputStream* übermittelt werden. Die Zeilen 25-27 erstellen ein Objekt dieser Klasse. Dieses kann über die Methode *readObjekt()* (Zeile 28) im erstellten Objekt gelesen werden. Anschließend wird in Zeile 29 die Methode *showNextPage()* aufgerufen, mit welcher die Seite *bid_success.jsp* angefordert wird. Der Quelltext dieser Methode ist aus Listing 4.24 ersichtlich. Beim Auftreten eines Fehlers dagegen wird in der Zeile 36 die Methode *showFailurePage()* aufgerufen. Diese wird auf Listing 4.25 angezeigt.

Listing 4.22: Die Methode *actionPerformed()* des Buttons *Absenden* im Applet *XmlSigApplet*

```
1 public void actionPerformed(java.awt.event.ActionEvent e) {
2  if(signData()){
3     try{
4     jLabel7.setText("Signatur erstellen ...");
5     URL servlet = new URL(
6        "http://127.0.0.1:8080/myapplication/sigservlet");
7     URLConnection servletConnection =
8                                 servlet.openConnection();
9     servletConnection.setDoInput(true);
```

```
10      servletConnection.setDoOutput(true);
11      servletConnection.setUseCaches (false);
12      servletConnection.setDefaultUseCaches (false);
13      servletConnection.setRequestProperty("Content-Type",
14                                  "application/octet-stream");
15      try {
16          ObjectOutputStream oos =
17              new ObjectOutputStream(
18                      servletConnection.getOutputStream());
19          oos.writeObject(sSignedDoc);
20          oos.flush();
21          oos.close();
22      } catch (IOException ioex) {
23          ioex.printStackTrace();
24      }
25       ObjectInputStream ois =
26          new ObjectInputStream(
27                      servletConnection.getInputStream());
28       String result = (String) ois.readObject();
29       showNextPage();
30      }catch(Exception ex){
31          ex.printStackTrace();
32      }
33    }else{
34      jLabel7.setForeground(new Color(255, 0, 12));
35      jLabel7.setText("üUngltige Feld-Angaben!");
36      showFailurePage();
37    }
38  }
```

Listing 4.23: Die Methode *signData()* des Applets *XmlSigApplet*

```
1  private String signData(){
2    try{
3      BidSignature bs = new BidSignature(jTextField.getText(),
4                                         sProdId, sXmlToSign);
5      bs.createSig(jKeyPass.getText(),
6       jPasswordField.getText(), jPasswordField1.getText());
7      dSignedDoc = cs.getSignedDoc();
8      sSignedDoc = cs.getSignedDocAsString();
9    }catch(Exception e){
10     e.printStackTrace();
11   } return sSignedDoc;
```

```
12 }
```

Listing 4.24: Die Methode *showNextPage()* des Applets *XmlSigApplet*

```
1 private void showNextPage(){
2   try{
3      getAppletContext().showDocument(new URL(
4                  "http://127.0.0.1:8080/myapplication/
5                   bid_success.jsp?productId=" + sProdId));
6   }catch(Exception e){
7     e.printStackTrace();
8   }
9 }
```

Listing 4.25: Die Methode *showFailurePage()* des Applets *XmlSigApplet*

```
1 private void showFailurePage(){
2   try{
3        getAppletContext().showDocument(new URL(
4        "http://127.0.0.1:8080/myapplication/
5         bid_failure.jsp?productId=" + sProdId));
6   }catch(Exception e){
7       e.printStackTrace();
8   }
9 }
```

Das Signieren von XML-Dokumenten

Wie bereits angesprochen, greift das Applet *XmlSigApplet* auf die Klasse *BidSignature* zu, um die Signatur der Daten zu erstellen. Die wichtigste Methode der Klasse ist *createSig()*, dessen Quelltext in Listing 4.26 gezeigt wird. Diese Methode befasst sich primär mit dem Keystore und mit dem Auslesen seiner Daten (Zeilen 10-25). In den Zeilen 37-38 wird das Objekt *ds* der Klasse *XmlDocSignature* erstellt. Es bekommt das XML-Dokumentobjekt, den privaten Schlüssel und das Zertifikat übergeben. Mit Hilfe der Methode *signMessage()* des *ds*-Objektes kann anschließend mit der Methode *signMessage()* (Listing 4.27) das signierte XML-Dokument ermittelt werden. Die Methode *signMessage()* erstellt die Signatur für das XML-Dokument und hängt das *sig:Signature*-Element an das *root*-Element des Dokuments. Sie benutzt dabei die von Apache zur Verfügung gestellten *XML Security*-Bibliotheken.

Listing 4.26: Die Methode *createSig()* in der Klasse *BidSignature*

```
1  public boolean createSig(String privateKeyPass, String
2      keystorePass, String privateKeyAlias) throws Exception {
3    boolean key_valid = true;
4    String keystoreFile = sKeyPath;
5    String keystoreType = KeyStore.getDefaultType();
6    KeyStore ks = KeyStore.getInstance(keystoreType);
7    FileInputStream fis = new FileInputStream(keystoreFile);
8
9    // keystore laden
10   try{
11     ks.load(fis, keystorePass.toCharArray());
12   }catch(Exception e){
13     e.printStackTrace();
14     key_valid = false;
15   }
16
17   // private key ermitteln
18   PrivateKey privateKey = null;
19   try{
20   privateKey = (PrivateKey) ks.getKey(privateKeyAlias,
21                              privateKeyPass.toCharArray());
22   }catch(Exception e){
23       e.printStackTrace();
24       key_valid = false;
25   }
26
27   X509Certificate cert = null;
28   try{
29   cert = (X509Certificate) ks.
30                             getCertificate(privateKeyAlias);
31   }catch(Exception e){
32         e.printStackTrace();
33         key_valid = false;
34   }
35   XmlTool t = new XmlTool();
36   Document xmldoc = t.getStringAsDoc(sDocument);
37   XmlDocSignature ds =
38             new XmlDocSignature(xmldoc, privateKey, cert);
39   dBid = ds.signMessage();
40   return key_valid;
41 }
```

Listing 4.27: Die Methode *signMessage()* in der Klasse *XmlDocSignature*

```
1  public Document signMessage () {
2    try {
3      Element root = dxmlMessage.getDocumentElement();
4      Constants.setSignatureSpecNSprefix("sig");
5      XMLSignature message_sig = new XMLSignature(dxmlMessage,
6              null, XMLSignature.ALGO_ID_SIGNATURE_RSA);
7      root.appendChild(message_sig.getElement());
8      {
9        Transforms transforms = new Transforms(dxmlMessage);
10       transforms.addTransform(
11                 Transforms.TRANSFORM_ENVELOPED_SIGNATURE);
12       transforms.addTransform(
13                 Transforms.TRANSFORM_C14N_WITH_COMMENTS);
14       message_sig.addDocument(
15             "", transforms, Constants.ALGO_ID_DIGEST_SHA1);
16       message_sig.addKeyInfo(certSigner);
17       message_sig.addKeyInfo(certSigner.getPublicKey());
18       ObjectContainer oc = new ObjectContainer(dxmlMessage);
19       Node sigDate =
20             (Node)dxmlMessage.createElement("sig:Date");
21       java.util.Date date = new java.util.Date(
22                                  System.currentTimeMillis());
23       String sDate = date.toString();
24       sigDate.appendChild(dxmlMessage.createTextNode(sDate));
25       oc.appendChild(sigDate);
26       message_sig.appendObject(oc);
27     }
28     PrivateKey pk = pkSigner;
29     message_sig.sign(pk);
30   }catch (Exception xse) {
31     xse.printStackTrace();
32   }
33   return  dxmlMessage;
34 }
```

Das Eintragen der signierten Dokumente in die Datenbank

Das signierte XML-Dokument muss an einem Ort permanent gehalten werden. Eine sinnvolle Möglichkeit dafür ist das Speichern des Dokuments in der Datenbank im XML-Format. Bei der Abgabe eines Gebots werden die Gebots-

daten in der Tabelle TDAT_BID abgelegt. Diese Tabelle besitzt eine Spalte SIGNED_BID, welche Daten vom Typ Text-Blob enthalten kann. Hier werden die signierten XML-Dokumente gehalten.

Das Einfügen der Dokumente in die Datenbank wird vom Servlet *SigServlet* ausgelöst. Das *SigServlet* bekommt das XML-Dokument übergeben, welches in seiner Methode *doPost()* mit Hilfe des *ObjectInputStream* Objektes *ois* und seiner Methode *readObject()* ausgelesen wird. Anschließend wird die Methode *writeSignedDocInDb()* mit dem *String*-Objekt des XML-Dokuments als Parameter aufgerufen. Diese Methode ist in Listing 4.28 aufgelistet.

Listing 4.28: Die Methode *writeSignedDocInDb()* des Servlets *SigServlet*

```
1  public int writeSignedDocInDb(String sDocument) {
2    if (sDocument.compareTo("") != 0) {
3      try {
4        XmlTool tool = new XmlTool();
5        Document dDoc = tool.getStringAsDoc(sDocument);
6        Dm db = new Dm();
7        Date date = new Date();
8        String time = date.getCurrentTime();
9        boolean bidInDb = false;
10       String xpath_pk = "/ProdBid/ProductData/Pk";
11       String pk =
12         tool.getTextNodeValueFromDocument(dDoc, xpath_pk);
13       String xpath_price =
14         "/ProdBid/ProductData/BidPricePerUnit";
15       String price =
16         tool.getTextNodeValueFromDocument(dDoc, xpath_price);
17       String xpath_quantity =
18         "/ProdBid/ProductData/BidQuantity";
19       String quantity =
20         tool.getTextNodeValueFromDocument(dDoc,
21                                           xpath_quantity);
22       String xpath_customer_code =
23         "/ProdBid/ProductData/CustomerCode";
24       String customer_code =
25         tool.getTextNodeValueFromDocument(dDoc,
26                                           xpath_customer_code);
27       String sql =
28         "execute procedure p_complete_bid(" + pk + "," +
29             customer_code + "," + "'" + time + "'" + "," +
30             price + "," + quantity + "," + "?" + ")";
```

```
31        db.executeStatementWithBlobData(sql, sDocument);
32      } catch (Exception ex) {
33        ex.printStackTrace();
34      }
35    }
36    return  0;
37  }
```

Zeile 2 stellt sicher, dass das in der Datenbank zu speichernde Dokument nicht leer ist. Zeile 5 ermittelt das *Dokument*-Objekt aus der Stringform des XML-Dokuments. Zeilen 10-19 ermitteln Daten aus dem Dokument, welche gleichzeitig mit dem XML-Dokument in der Tabelle TDAT_BID abgelegt werden. Zeile 31 führt den in den Zeilen 27-30 vorbereiteten Prozeduraufruf auf, welcher das Eintragen der Daten im Datenbanksystem auslöst.

Kapitel 5

Web Services

> Archibald MacLeish: *„Our technology, wiser than we, has given us the unforeseen and unforeseeable means of worldwide understanding at the moment when worldwide understanding is the only possible means to lasting peace.“*
> (Deitel u. a., 2003, S. 21)

Web Services sind der neueste und damit der modernste Schritt in Richtung verteilter Systeme. Web Services erlauben Applikationen, die sich auf zwei Computern befinden, miteinander zu kommunizieren.

In diesem Kapitel wird die Technologie von Web Services näher betrachtet. Zunächst wird mit Definitionen des Begriffs, möglichen Szenarien, Eigenschaften und Lebenszyklen von Web Services ein Überblick über die Technologie gegeben. Im darauf folgenden Abschnitt wird auf die zwei Ansätze SOA und Peer-to-Peer-Ansatz zur Bildung von Web Services eingegangen. Der dritte Abschnitt des Kapitels beschäftigt sich mit den wichtigsten Standards von Web Services. Diese werden näher angegangen und mit Beispielen veranschaulicht. Abschnitt vier wird der praktischen Arbeit in Verbindung mit dieser Technologie gewidmet. Hier wird auf die Arbeit mit dem Web Services-Tool *Axis* eingegangen, mit dessen Hilfe die innerhalb der Anwendung erstellten Web Services, publiziert werden können. Die Anwendung nutzt den zur Verfügung gestellten Google Web Service um innerhalb der Anwendung Anfragen an Google zu stellen und damit einen externen Web Service einzubinden.

5.1 Überblick

Die Idee der Web Services und verteilten Anwendungen existiert bereits seit einigen Jahren. Es hat schon vor der Idee der Web Services bereits Ansätze und Implementierungen in diesem Bereich gegeben. Eine solche ist die Idee

von RPC[1], welche Prozeduraufrufe auf entfernten Systemen durchführt. Nach Kuschke und Wölfel (2002, S. 10) unterstützt RPC allerdings nicht moderne Architekturen, Sprachen und Objekte. Um diesen Nachteil zu beheben, wurde CORBA[2] (OMG, 2003) von der OMG[3], DCOM[4] (Microsoft, 1996) von Microsoft®, RMI[5] (Sun, 2003) von Sun® Microsystems und DSOM[6] von IBM entwickelt. DCOM und RMI wurden als ORPC[7] bezeichnet. Der Nachteil der oben genannten Technologien ist allerdings deren Plattformabhängigkeit. Ausnahme ist CORBA, die nahezu alle Plattformen unterstützt. Sie stellt die ganze Funktionalität auf einem Server bereit und kann von geeigneten Clients aufgerufen werden. Der Einsatz von CORBA ist trotz ihrer Vorteile mit einem sehr großen Aufwand verbunden. Die meisten CORBA-Systeme unterstützen, trotz der theoretischen Plattformunabhängigkeit, nur die Programmiersprachen Java und C++, was das Umschreiben einer Anwendung, die in einer anderen Sprache implementiert wurde, notwendig macht.

Was macht die Technologie der Web Services so revolutionär? Die wichtigsten Akronyme sind hier sicher das WWW und XML, welche die sprach- und plattformunabhängige Kommunikation der Anwendungen ermöglichen. Gleichzeitig ist es die Einfachheit der Technologie und der niedrige Kostenfaktor[8], die die Web Services so populär machen.

Bevor auf die Eigenschaften der Web Services eingegangen wird, soll der Begriff selbst zunächst geklärt werden. Es existieren mehrere Definitionen zum Begriff Web Service und einige davon sollen hier vorgestellt werden.

Es folgt zunächst die Definition des W3C:

> „A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL[9]). Other systems interact with the Web service in a man-

[1] Remote Procedure Call
[2] Common Object Request Broker Architecture
[3] Object Management Group
[4] Distributed Component Object Model
[5] Remote Method Invocation
[6] Distributed System Object Model
[7] Object Remote Procedure Call
[8] Darunter fällt der TCO (Total Cost of Ownership). Damit sind hier die Nutzungs- und Wartungskosten von Software gemeint, die durch den Einsatz von Web Services eingespart werden können.
[9] Web Services Description Language

> ner prescribed by its description using SOAP[10] messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.“ (Haas, 2004)

Ein Web Service ist nach Jeckle (2003) eine Komponente, „...die ihre Funktionalität über eine veröffentlichte Schnittstelle anbietet und über ein offenes, im Internet verwendetes Protokoll zugreifbar ist“.

Kossmann und Leymann (2004, S. 117) verstehen unter Web Services

> „... ein Bündel von Technologien zur Beschreibung von Schnittstellen und Eigenschaften von Implementierungen der Schnittstellen, Beschreibung von Datenaustauschformaten und Qualitätseigenschaften des Austauschs, Registrierung von Komponenten, Komposition von Komponenten und Sicherheit im Austausch mit Komponenten“.

Die einfachste Definition, die einen Web Service beschreibt, ist nach Mannes (2003, S. 27):

> „... Web Service is an application that provides a Web API. ... A Web API is an API that lets the applications communicate using XML and the Web.“

Die Definition erweitert Mannes (2003, S. 30) mit der Erklärung der Begriffe *Web* und *Service*:

> „A Web resource is any type of named information object ... that's accessible through the Web. ... A service is an application that exposes its functionality through an application programming interface (API).“

Trotz der Unterschiede in den aufgelisteten Definitionen haben diese eines gemeinsam: Ein Web Service ermöglicht die Kommunikation zwischen Maschinen über das Internet.

Es soll an dieser Stelle verdeutlicht werden, dass Web Services nicht nur über das Internet laufen. Mit Web Services wird die Kommunikation zwischen Maschinen vereinfacht, welche nicht notwendigerweise über das HTTP-Protokoll ermöglicht werden muss. Diese kann beispielsweise auch über andere Protokolle wie z.B. über SMTP stattfinden.

[10] Simple Object Access Protocol

5.1.1 Vorteile und Motivation

Die Stärke von Web Services liegt darin, eine einfache Möglichkeit zur Kommunikation zwischen Software-Systemen zu ermöglichen. Kossmann und Leymann (2004, S. 117) unterscheiden dabei zwischen den Begriffen EAI und EII:

- *Enterprise Application Integration (EAI)*: Durch diese Integration wird die Interaktion zwischen Software-Systemen innerhalb eines Unternehmens oder unternehmensübergreifend ermöglicht.
- *Enterprise Information Integration (EII)*: Durch diese Integration werden die Daten einzelner Software-Systeme verknüpft.

Den letzten zwei Integrationsmöglichkeiten entsprechend können nach Bitzer (2003, S. 224 f.) zwei wichtige Anwendungsfelder von Web Services genannt werden:

- *Information*: Im Mittelpunkt beim Einsatz von Web Services steht der Austausch von Informationen. Wie im Kapitel 2.3 bereits beschrieben, benötigen Daten eine sinnvolle Struktur, um ihre Weiterverarbeitung zu ermöglichen. Auch eine Struktur bei Abfragen der strukturierten Daten trägt zu einer Verbesserung bei. Einen Beitrag dazu leisten die Web Service-Sprachen WSDL und SOAP.
- *Integration*: Es entstehen oft unnötig hohe IT-Kosten, weil viele Softwaresysteme aneinander angepasst werden müssen. Viele Softwareprodukte, sogar innerhalb eines Unternehmens, werden in unterschiedlichen Sprachen realisiert und laufen vielleicht auch noch auf unterschiedlichen Betriebssystemen. Um Daten eines Unternehmenbereichs einem anderen zur Verfügung zu stellen, sind Anpassungen der Software notwendig. Die Integration der Systeme kann:
  - Durch einen Export der Daten z.B. im CSV-Format vom ersten System und Import der Daten im zweiten System stattfinden oder
  - durch direkten Aufruf von Programmen des Systems, das die Daten liefern soll, erfolgen.

  Die letztere Integrationsart ist eleganter, erzielt zugleich eine sinnvolle Schnittstelle, ist aber anspruchsvoller in der Programmierung. Es wird mit Technologien wie CORBA, RPC und RMI realisiert. Sie haben allerdings Nachteile wie die Java Umgebungsabhängigkeit von RMI und hohe Anpassungsnotwendigkeit der eigenen Softwaresysteme an die CORBA-Architektur. Durch die Beschreibung der Schnittstellen im WSDL-Format

und Beschreibung der Nachrichten im SOAP-Format wird die Abstraktion von den eigentlichen Systemen erzielt und damit die Abstraktion von deren Umsetzung und von der Umgebung, in der sie agieren.

5.1.2 Web Services Szenarien

Auf Seite 23 wurden die möglichen Erscheinungsformen des e-Commerce-Geschäftsmodells dargestellt. So konnten von der Seite des e-Business die Bereiche B2B, B2C und B2A definiert werden. Deitel u. a. (2003, S. 61 ff.) und Adam (2004) definieren analog dazu vom Sichtpunkt des Service Providers ausgehend folgende drei Arten von Web Service-Modellen:

- Service-to-Consumer (S2C)
- Service-to-Business (S2B)
- Service-to-Employee (S2E)

Service-to-Consumer

In diesem Bereich existieren eine Reihe von Web Services, die Privatpersonen zur Verfügung gestellt werden. Sie können zum Beispiel Informationen zur Verfügung stellen, die sich sehr oft ändern, wie die aktuellen Nachrichten oder die zeitweilige Temperatur zu einer Postleitzahl.[11] Dieses Modell stellt sowohl kostenlose als auch kostenpflichtige Web Services zur Verfügung.

Service-to-Business

Service-to-Business ist das derzeit bedeutendste Web Service-Modell. Firmen haben schnell die Möglichkeiten erkannt, welche Web Services zur Verfügung stellen und entwickeln Web Services, die sowohl von neuen Kunden als auch von Partnern genutzt werden können. Rhody (2004, S. 3) beschreibt es so: „Integration is one of the most elusive goals of any organization“ und Web Services ist die evolutionäre Technologie, die diese erwartete Flexibilität verspricht. Ein Unternehmen braucht flexible Systeme, die sich ohne Zeitverzögerung an die sie umgebende Businesswelt anpassen.

Bei diesem Modell können Unternehmen Web Services ihren Partnern zur

[11] Solche Web Services können aus http://www.xmethods.net und http://www.salcentral.com entnommen werden.

Verfügung stellen. Dieser Fall entspricht der Konstellation, bei der der Service Provider seinem Business-Partner, nämlich dem Service Requestor, einen freien Service zur Verfügung stellt.

Weiterhin können Unternehmen auch Web Services entwickeln, die von jedem anderen Unternehmen genutzt werden können. Dies könnten sowohl kostenpflichtige Services sein, welche allgemeine Aufgaben durchführen als auch Aufgaben, die einem Partner zur Verfügung gestellt werden.

Service-to-Employee

Diese Art von Web Services werden von Unternehmen für ihre Mitarbeiter entwickelt. Sie stellen bestimmte Informationen zur Verfügung oder unterstützen auch den Austausch von Informationen zwischen den Mitarbeitern im Unternehmen.

5.1.3 Eigenschaften von Web Services

Die Eigenschaften von Web Services sind:

- *Interoperabilität*: Eine der wichtigsten Eigenschaften von Web Services ist die Ermöglichung der Interoperabilität zwischen heterogenen Anwendungen. Sie erlauben auf einfache Weise den Zugriff auf entfernte Anwendungen.
- *Leicht realisierbar*: Der Aufruf bestimmter Funktionen einer entfernten Anwendung mit Web Services erfordert lediglich Kenntnis über die Schnittstelle. Wissen über den konkreten Aufbau und Ablauf der entfernten Funktionen ist nicht notwendig.
- *Selbstbeschreibend*: Web Services werden immer mit einer Beschreibung zur Verfügung gestellt. Sie kann mit Hilfe des WSDL Standards erstellt werden und beschreibt alle zur Verfügung gestellten Funktionen vom Service und die von ihnen erwarteten bzw. zurückgelieferten Parameter. Außerdem wird dort noch die Adresse und der Port angegeben, auf dem der Service auf den Aufruf wartet.
- *Web Services basieren auf dem offenen Standard XML*: Der XML-Standard ist derjenige, der die Grundlage für die Plattformunabhängigkeit und für die Sprachunabhängigkeit der Web Services ermöglicht. Darauf basieren Standards wie WSDL, UDDI[12] und SOAP auf XML. Auf diese

[12]Universal Description, Discovery and Integration

Standards und deren Bedeutung in der Web Services-Welt wird später näher eingegangen.

- *Web Services sind auffindbar*: Web Services werden auf den so genannten „yellow pages“ zur Verfügung gestellt, welche auf dem XML-Standard UDDI basieren können.

- *Informationen können über beliebige Protokolle ausgetauscht werden*: Das meist genutzte Protokoll, über das die Daten ausgetauscht werden, ist HTTP. Für die Unternehmen ist dies ein vertrautes Protokoll, das keine zusätzlichen Kosten verursacht. Weiterhin kann der Austausch auch über die bekannten Protokolle FTP und MIME stattfinden.

- *Austausch von Daten auf der mittleren Ebene von Webanwendungen*: Web Services bilden einen Standard zum Austausch von Daten auf der Business Logik-Ebene von Webanwendungen und bleiben daher für den Endnutzer unsichtbar.

5.1.4 Der Lebenszyklus von Web Services

Der Lebenszyklus eines Web Service besteht nach Clark (2001) aus den folgenden vier Stufen:

- *Erstellen (creation)*: Bei diesem ersten Vorgang wird die *Initiierung und Ausarbeitung* eines Web Service vorgenommen. Danach folgt die eigentliche *Entwicklung* vom Programmierer. Der Web Service muss nach der technischen Implementierung auch mit einer *Dokumentation*, die die Aufgaben und Schnittstellen des Web Service beschreibt, versehen werden. Der Prozess wird mit der Abbildung 5.1 veranschaulicht.

  Nach der Durchführung dieser drei Schritte muss der bereits bestehende Web Service einen weiteren wichtigen Schritt durchleben: Den Prozess des Web Service-Tests. Die Tests können von einer außenstehenden Firma übernommen werden. Sie beinhalten Daten wie Ausfall und Antwortzeiten des Service. Anhand der Ergebnisse dieser Tests kann ein Zertifikat für den Service erstellt werden.

  Anschließend soll der Web Service noch in der Öffentlichkeit bekannt gemacht werden. Diese Aufgabe wird von einem Distributor ausgeführt. Er übernimmt die Kontrolle über das Hosting des Service und über die Daten, die vom Service konsumiert werden.

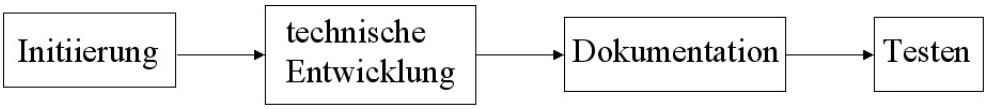

Abbildung 5.1: Erstellen eines Web Services

- *Publizieren (engl. publication)*: Unter Publizieren eines Web Services wird sein eigentliches Deployment im Web verstanden.

- *Werbungsprozess (engl. promotion)*: Diese Aufgabe wird durch Vermittler[13] ausgeführt. Sie müssen sicherstellen, dass der Web Service auffindbar ist.

  Um die Auffindbarkeit eines Web Services zu ermöglichen, wird ein Verzeichnis (*engl. registry*) zur Verfügung gestellt, in dem verfügbare Web Services publiziert werden. Das kann ein öffentliches oder ein privates Verzeichnis[14] sein. Ein verbreiteter öffentlicher Web Service ist der UBR (UDDI Business Registry). Private Verzeichnisse stellen Web Services innerhalb Organisationen oder zwischen Organisationen und deren Partner dar.

  Value-added services(VAS) sind mögliche Typen von Services. Solche Services stellen Informationen bereit, wie erreichbar und wie schnell ein Web Service ist. Den Organisationen, die Web Services herstellen und hosten werden Akkreditierungen vergeben, die auf Faktoren wie der Größe einer Organisation oder der Verfügbarkeit des Services basieren. (Clark, 2001)

- *Der Verkauf (engl. selling)*: In dieser letzten Stufe des Web Services Lebenszyklus gibt es zwei Beteiligte. Das sind der *Auditor* und der *Account Manager*.

  Der *Auditor* eines Web Services ist für die Lauffähigkeit eines Services verantwortlich. Er hat die Aufgabe den Service zu testen, da nach dem Publizieren des Services weiterhin sicher gestellt werden muss, dass der Service seine Aufgaben korrekt erfüllt.

[13] Diese werden auch noch als *broker* bezeichnet.

[14] Bekannte Web Services Verzeichnisse sind die bereits oben erwähnten *http://www.xmethods.net* und *http://www.salcentral.com*.

Der *Account Manager* ist der zweite Beteiligte im Verkaufsprozess des Web Services. Er bekommt die Einnahmen von den Web Service-Abonnenten und verteilt diese zwischen den beteiligten Partnern, die für das Bereitstellen des lauffähigen Services verantwortlich sind.

5.2 Web Services-Modell

In diesem Abschnitt werden Architekturen, mit der die Web Services-Provider ihre Services zur Verfügung stellen, dargestellt. Dabei werden zwei Ansätze dargestellt – die Service Orientierte Architektur und den Peer-to-Peer Ansatz.

5.2.1 Service Orientierte Architektur (SOA) - Ansatz

Die Service-Orientierte Architektur wurde von IBMs Web Services *Architecture Team* im September 2000 entwickelt (Chappell und Jewell, 2002, S. 14) und auf der Web Seite von DeveloperWorks[15] publiziert. Die Komponenten dieser Architektur sind wie aus der Abbildung 5.2 ersichtlich der Service Provider, die Service Registry und der Service Requestor.

Der *Service Provider* ist ein Dienstanbieter,

- der den Web Service in einer Programmiersprache implementiert und
- über eine XML-basierte Schnittstellenbeschreibung des Web Services wie ein WSDL-Dokument verfügt.

WSDL ist die meist verbreitete Spezifikation zum Beschreiben eines Web Services. Das WSDL-Dokument ist ein XML-Dokument, welches die Schnittstelle eines Web Services mit folgenden Parametern exakt beschreibt:

- *Protokolle*, mit denen auf den Web Service zugegriffen werden kann. Dies können Protokolle wie HTTP, FTP, SMTP und POP sein.
- *Ein-* und *Ausgabe-Parameter* der einzelnen Operationen werden spezifiziert.

Auf die Spezifikation WSDL wird im Kapitel 5.3.2 detaillierter eingegangen.

Die *Service Registry* ist eine Art Verzeichnis für Web Services, bei der der

[15] http://www.ibm.com/developerWorks

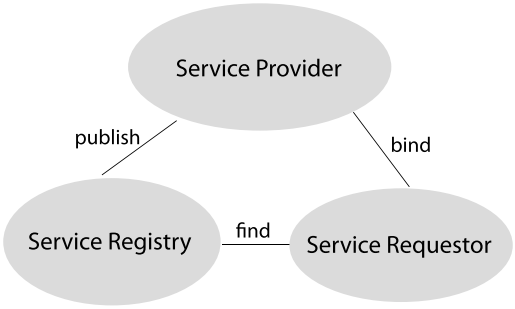

Abbildung 5.2: Komponenten eines Service Orientierten Ansatzes

Service Provider Informationen über sich und seine angebotenen Dienste hinterlegen kann. Diese Art von Verzeichnis kann mit einer UDDI realisiert werden. Das UDDI ist ein offener Rahmen,

- um Services zu beschreiben,
- um vorhandene Services zu finden und
- um verschiedene Services miteinander zu verbinden.

Der Service Provider registriert sich bei der Service Registry (UDDI). Dabei werden Infos in *yellow*, *white* und *green pages* zur Verfügung gestellt.

- Die *yellow pages* geben Auskunft zu den unterstützten Branchen, Diensten und Produkten.
- Die *white pages* beinhalten Namen, Adressen und Ansprechpartner der Service Anbieter.
- Die *green pages* enthalten technische Details zur Kommunikation mit den registrierten Web Services.

Der *Service Requestor* ist der eigentliche Nutzer eines Web Services. Dies ist eine Software-Anwendung; Es kann aber auch eine Person sein, die über ein Webinterface auf den Server zugreift.

5.2.2 Peer-to-Peer (P2P) - Ansatz

Der Peer-to-Peer-Ansatz ist eine andere Möglichkeit, Web Services zur Verfügung zu stellen. Dieser unterscheidet sich vom SOA-Ansatz, indem hier keine Rollenverteilung stattfindet. In diesem Modell kann jeder im Netzwerk

einen Service anbieten und in Anspruch nehmen. Daher kann ein Netzwerk-Teilnehmer sowohl die Rolle eines Service Providers als auch die Rolle eines Service Requestors annehmen. Chappell und Jewell (2002, S. 24) geben folgende zwei Gründe an, die für diesen Ansatz sprechen:

- *Effizienteres Nutzen der Netzwerkbandbreite*: Die Konzentration fällt nicht auf einen Server, auf dem der Service zur Verfügung gestellt wird. Wenn ein Peer zu langsam ist, kann ein anderer seine Aufgabe übernehmen und schnell die Antwort geben. Beim Ausfall eines Peers kann die Aufgabe genauso von einem anderen Peer übernommen werden.

- *Größeres Vorhandensein*: Der am nächsten stehende Peer kann den Service ausführen und die Ergebnisse zur Verfügung stellen.

5.3 Web Services-Standards

Die Entstehung und die Entwicklung der Web Service Technologie nimmt mit schnellen Schritten zu. Auf diese Technologie wird enorme Hoffnung seitens vieler großer Unternehmen gelegt. Sie setzen sie in Pilotprojekte ein und untersuchen dort ihre Möglichkeiten und Funktionalität. Dieses Interesse gibt Anlass für die schnelle Entwicklung von Standards, die Web Services unterstützen sollen. Das wird gleichzeitig auch von Unternehmen durch Anregungen, Vorschläge, Beteiligung an Gruppen, die Standards entwickeln, immens fortgetrieben.

Die bedeutendsten Standards für Web Services sind WSDL, UDDI und SOAP. Im Rahmen dieses Abschnitts werden diese Standards und deren Einsatz erläutert.

5.3.1 Simple Object Access Protocol (SOAP)

SOAP (Mitra, 2003) steht für Service Object Access Protocol und wird beim W3C weiterentwickelt. Es ist ein XML-basierender Kommunikationsprotokoll und ein Kodierungsformat für Kommunikation zwischen Applikationen. SOAP ist ein offener Standard und damit ein Rückgrat für offene, plattformunabhängige und sprachunabhängige Internet-Anwendungen.

Nach Wüstner und Buxmann (2002) kann SOAP nach den zwei folgenden Einsatzarten verwendet werden:

- *Remote Procedure Calls*: Bei diesem Einsatz werden Methoden auf entfernten Rechnern aufgerufen. Beim Aufruf sollen dabei der Name der aufzurufenden Methode und die von dieser erwarteten Parametern übergeben werden. Der Server führt die Methode mit den Werten der entsprechenden Parameter aus und liefert die entsprechenden Ergebnisse zurück.

- *Dokumentenbasierter Nachrichtenaustausch*: In diesem Fall wird keine Methode aufgerufen. Hier werden Nachrichten im XML-Format zum Server geschickt und dort verarbeitet. Eine strukturierte Antwort wird in Form einer Nachricht zurückgesendet.

Aufbau einer SOAP-Nachricht

Der Aufbau einer SOAP-Nachricht ist mit der Abbildung 5.3 veranschaulicht. Sie besteht, wie aus Listing 5.1 ersichtlich, aus einem *xml*-Element und aus einem *Envelope*-Element.

Eine SOAP-Nachricht besteht aus einem *SOAP-Envelope*, welcher sich aus einem optionalen SOAP-Header und aus einem SOAP-Body zusammensetzt. Der *SOAP-Body* (Zeilen 6-14) ist Pflicht für eine SOAP-Nachricht und enthält die eigentlichen Daten, die vermittelt werden sollen. Der *SOAP-Header* dagegen enthält Daten, die Informationen zur Nachrichtenverarbeitung liefern.

Listing 5.1: SOAP-Nachricht

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <soapenv:Envelope
3      xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
4      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
5      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
6   <soapenv:Body>
7    <ns1:getProductNameResponse soapenv:encodingStyle=
8          "http://schemas.xmlsoap.org/soap/encoding/"
9          xmlns:ns1="impl:FirstWsSoapBinding">
10    <ns1:getProductNameReturn xsi:type="xsd:string">
11          Flugticket nach Bulgarien am 12.12.2003
12          </ns1:getProductNameReturn>
13   </ns1:getProductNameResponse>
14  </soapenv:Body>
15 </soapenv:Envelope>
```

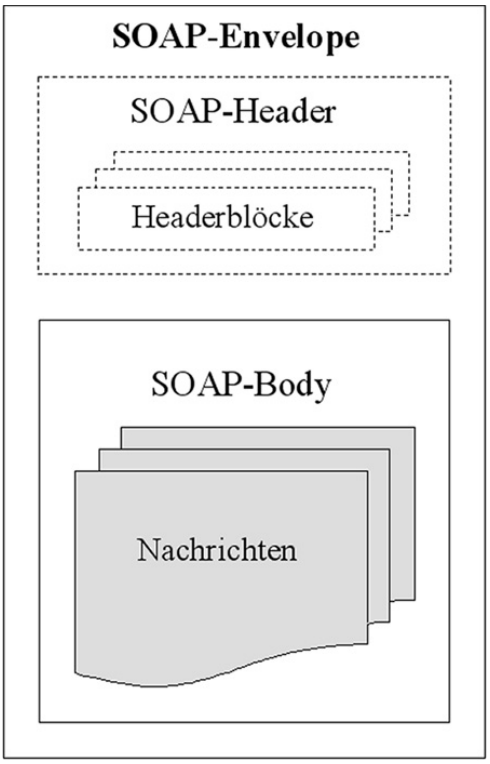

Abbildung 5.3: Architektur einer SOAP-Nachricht

Das SOAP-Kommunikationsmodell

Im Listing 5.2 soll nach dem Beispiel vom W3C (Mitra, 2003) das Prozessieren einer SOAP-Nachricht erläutert werden.

Listing 5.2: SOAP-Kommunikationsmodell

```
<?xml version="1.0" ?>
 <env:Envelope
 xmlns:env="http://www.w3.org/2003/05/soap-envelope">
   <env:Header>
     <p:oneBlock xmlns:p="http://example.com"
             env:role="http://example.com/Log">
      ...
      ...
     </p:oneBlock>
     <q:anotherBlock xmlns:q="http://example.com"
       env:role=
       "http://www.w3.org/2003/05/soap-envelope/role/next">
      ...
      ...
     </q:anotherBlock>
     <r:aThirdBlock xmlns:r="http://example.com">
      ...
      ...
     </r:aThirdBlock>
   </env:Header>
   <env:Body >
      ...
      ...
   </env:Body>
 </env:Envelope>
```

Eine SOAP-Nachricht kann über mehrere Zwischenstationen auf dem Weg vom Absender zum Empfänger laufen. Diese werden den *Header*-Block der SOAP-Nachricht interpretieren und eventuell auch ändern.

5.3.2 Web Services Description Language (WSDL)

Vor dem eigentlichen Zugriff einer Web Anwendung auf einen Web Service muss diese zuerst Informationen über den Web Service erhalten. Zum Ausführen dieser Aufgabe wurden bereits mehrere Technologien entwickelt. Die Namen zweier davon sind das *Resource Description Framework* und die *DARPA Agent*

Markup Language (DAML). Eine weitere Technologie, die sich durchgesetzt hat, ist die XML-basierte Web Services Description Language. Ein wesentlicher Vorteil der WSDL gegenüber RDF (Resource Description Framework) und DAML ist die Einfachheit dieser Sprache.

Die Sprache WSDL ist ein W3C Standard, welcher aus drei Teilen besteht:

- *Core Language* (Chinnici u.a., 2004),
- *Message Exchange Patterns* (Gudgin u.a. 2004, 2004) und
- *Bindings* (Moreau u.a., 2003).

WSDL ist hiermit die Beschreibungssprache für Web Services und eine der meist genutzten Technologien zum Publizieren von Web Services. WSDL-Dateien definieren die Formate der Web Service-Nachrichten und können genauso wie SOAP per HTTP, FTP und MIME angesprochen und übertragen werden.

Die Bestandteile einer WSDL-Datei sind im Folgenden aufgelistet:

Definitions Dies ist das Wurzelelement eines Web Services und enthält alle seine Elemente. Hier werden alle Namensräume des Web Services definiert.

Types Der Types-Knoten des WSDL-Dokuments definiert *einfache* und *zusammengesetzte* Typen, die eine SOAP-Nachricht enthalten darf. Das ist ein optionales Element und muss nicht vorkommen, wenn ein Standard XSD-Typensystem angewendet wird. Bei komplexen Typen können diese basierend auf dem XML Schema definiert werden.

Messages Dieser Teil des WSDL-Dokuments definiert abstrakt die zu übertragende Nachricht. Die *Message*-Elemente enthalten *Part*-Elemente, welche ein eindeutiges *name*-Attribut besitzen. Weitere Attribute des *Part*-Elements sind *element* oder *type*. Das *type*-Attribut definiert den zugehörigen einfachen oder komplexen Datentyp aus dem XML Schema und das *element*-Attribut verweist auf ein XSD-Element.

PortTypes Der PortType-Knoten gibt eine abstrakte Beschreibung der Funktionen der Web Service-Anwendung und besitzt ein eindeutiges *name*-Attribut. Jede Operation/Funktion wird mit dem Element *operation* definiert. Dieses kann Elemente vom Typ *input*, *output* und *fault* enthalten. Hiermit lassen sich folgende vier Nachrichten-Konstellationen beschreiben:

- *One-Way*: Die Operation enthält nur eine *input*-Nachricht, die vom Service zum Client gesendet wird.
- *Request-Response*: Der Client sendet hier eine *input*-Nachricht zum Service, welche empfangen und gelesen wird. Auf diese wird mit einer *output*-Nachricht reagiert. Die Definition einer *fault*-Nachricht ist in dieser Konstellation auch möglich.
- *Solicit-Response*: Hier erfolgt zuerst eine *output*-Nachricht vom Service zum Client und als Antwort sendet der Client eine Nachricht an den Service. Eine *fault*-Nachricht kann hier ebenfalls enthalten sein.
- *Notification*: Der Service sendet dem Client eine *output*-Nachricht. Weiterer Nachrichtenaustausch findet hier nicht statt.

Operations Dies ist ein Kind-Element von *PortType*. Es enthält die eigentlichen *input*-, *output*- und *fault*-Nachrichten.

Bindings Das *Binding* eines Services bindet die Funktionen eines Services mit der Adresse des Services an. Der WSDL-Standard definiert Bindings für SOAP, HTTP GET/POST und MIME.

Ports *Port* ist ein Kind-Element vom *Service*-Element und gibt die Adresse des Web Services an.

Service Gibt die genaue Beschreibung der Web Service-Adressierung bekannt.

Die Struktur eines WSDL-Dokuments wird auf der Abbildung 5.4 auf Seite 219 veranschaulicht. Ein Beispiel für ein WSDL-Dokument ist aus Listing 5.3 ersichtlich.

Listing 5.3: WSDL-Dokument, Quelle: Chinnici u.a. (2004)

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <wsdl:definitions
3     targetNamespace="http://example.org/TicketAgent.wsdl20"
4     xmlns:xsTicketAgent="http://example.org/TicketAgent.xsd"
5     xmlns:wsdl="http://www.w3.org/2004/03/wsdl"
6     xmlns:xs="http://www.w3.org/2001/XMLSchema"
7     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
8     xsi:schemaLocation=
9     "http://www.w3.org/2004/03/wsdl wsdl20.xsd">
10
11    <wsdl:types>
```

```
12          <xs:import schemaLocation="TicketAgent.xsd"
13          namespace="http://example.org/TicketAgent.xsd" />
14      </wsdl:types>
15
16      <wsdl:interface name="TicketAgent">
17          <wsdl:operation name="listFlights"
18          pattern="http://www.w3.org/@@@@/@@/wsdl/in-out">
19              <wsdl:input
20              element="xsTicketAgent:listFlightsRequest"/>
21              <wsdl:output
22              element="xsTicketAgent:listFlightsResponse"/>
23          </wsdl:operation>
24
25          <wsdl:operation name="reserveFlight"
26          pattern="http://www.w3.org/@@@@/@@/wsdl/in-out">
27              <wsdl:input
28              element="xsTicketAgent:reserveFlightRequest"/>
29              <wsdl:output
30              element="xsTicketAgent:reserveFlightResponse"/>
31          </wsdl:operation>
32      </wsdl:interface>
33  </wsdl:definitions>
```

Kossmann und Leymann (2004, S. 121) stellen mit der Abbildung 5.5 die wesentlichen Konzepte der Schnittstellenbeschreibung dar, welche die drei folgenden Fragen beantwortet:

- *Was*: „Was bedeutet der Web Service?“ Hiermit wird die Frage nach den Operationen, die der Web Service zur Verfügung stellt, beantwortet. Dabei werden die Input- und Outputnachrichten des Service bekanntgegeben.

- *Wie*: „Welche Protokolle werden zum Nachrichtenaustausch verwendet und wie werden die Nachrichten kodiert?“

- *Wo*: „Wie heißt der Web Service und unter welcher Internet-Adresse kann man Nachrichten an den Web Service schicken?“

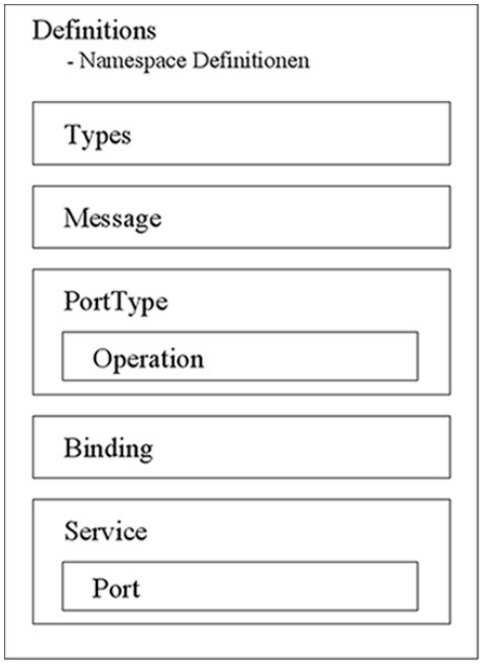

Abbildung 5.4: Bestandteile eines WSDL-Dokuments

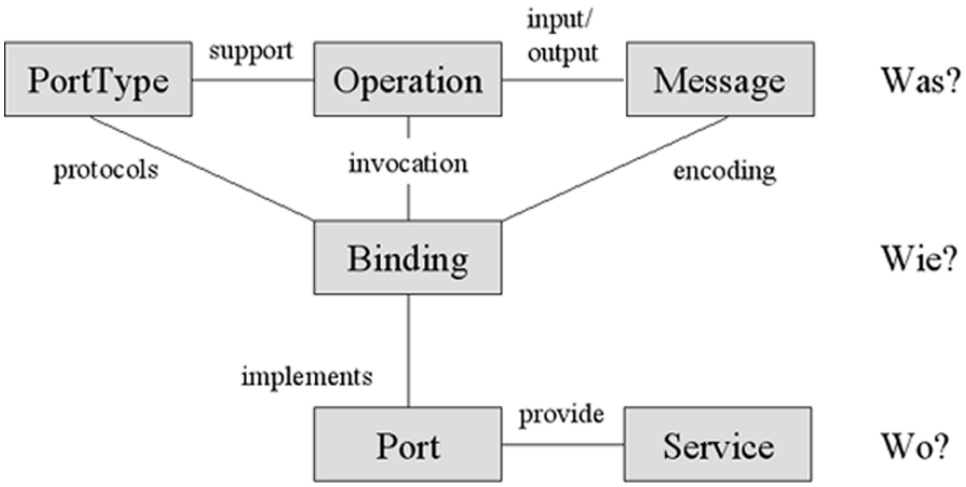

Abbildung 5.5: Übersicht über WSDL-Konzepte,
Quelle: Kossmann und Leymann (2004, S. 121)

5.3.3 Universal Description, Discovery and Integration-Standard (UDDI)

Die Existenz eines Services muss der Öffentlichkeit bekanntgegeben werden. Diese Aufgabe soll durch den UDDI-Standard (OASIS, 2004) gewährleistet werden. Publizierte Web Services sollen in einem Verzeichnis auffindbar gemacht werden. Nun haben alle Befürworter von UDDI wie Microsoft und IBM ihre eigenen UDDI-Verzeichnisse aufgebaut, in welchen die publizierten Web Services gesucht werden können.

Um ein UDDI-Verzeichnis bereitzustellen, sind nach Kuschke und Wölfel (2002, S. 77) folgende Komponenten notwendig:

- *Definition der Struktur für ein universelles Verzeichnis*: Hier können Unternehmen sich und die von ihnen angebotenen Dienstleistungen beschreiben.

- *Funktionen des Verzeichnisses*: Hier werden Funktionen definiert, die das Suchen und die Modifikation der Daten innerhalb des Verzeichnisses erlauben.

Zunächst soll die Architektur eines UDDI-Verzeichnises am Beispiel einer Web Service Anmeldung kurz erläutert werden. Wie aus Abbildung 5.6 ersichtlich, ist eine Kernkomponente das Universal Business Registry (UBR) oder das UDDI Business Registry und besteht aus UDDI-Operatoren. UDDI-Operatoren sind Unternehmen, die die Sicherheit, Verfügbarkeit und Performance eines Verzeichnisses garantieren. Weiterhin agieren die UDDI-Operatoren miteinander, indem sie ihre Verzeichnisse synchronisieren.

Ein Service-Anbieter kann direkt seinen Service beim Operator veröffentlichen. Eine weitere Möglichkeit, den Service zu publizieren, hat der Anbieter, indem er die Anmeldung über einen Registrar realisiert. Ein *Registrar* ist hier ein Unternehmen, das Oberflächen zum UDDI-Verzeichnis zur Verfügung stellt und sich damit zwischen dem Operator und dem Service-Anbieter positioniert. Anschließend befinden sich auf der Ebene der UBR die privaten UDDI-Verzeichnisse. Diese gehören nicht zum UBR, stellen allerdings eine zusätzliche Möglichkeit der Service Anbieter zur Service-Publikation dar. (Hauser und Löwer, 2004, S. 102 ff.)

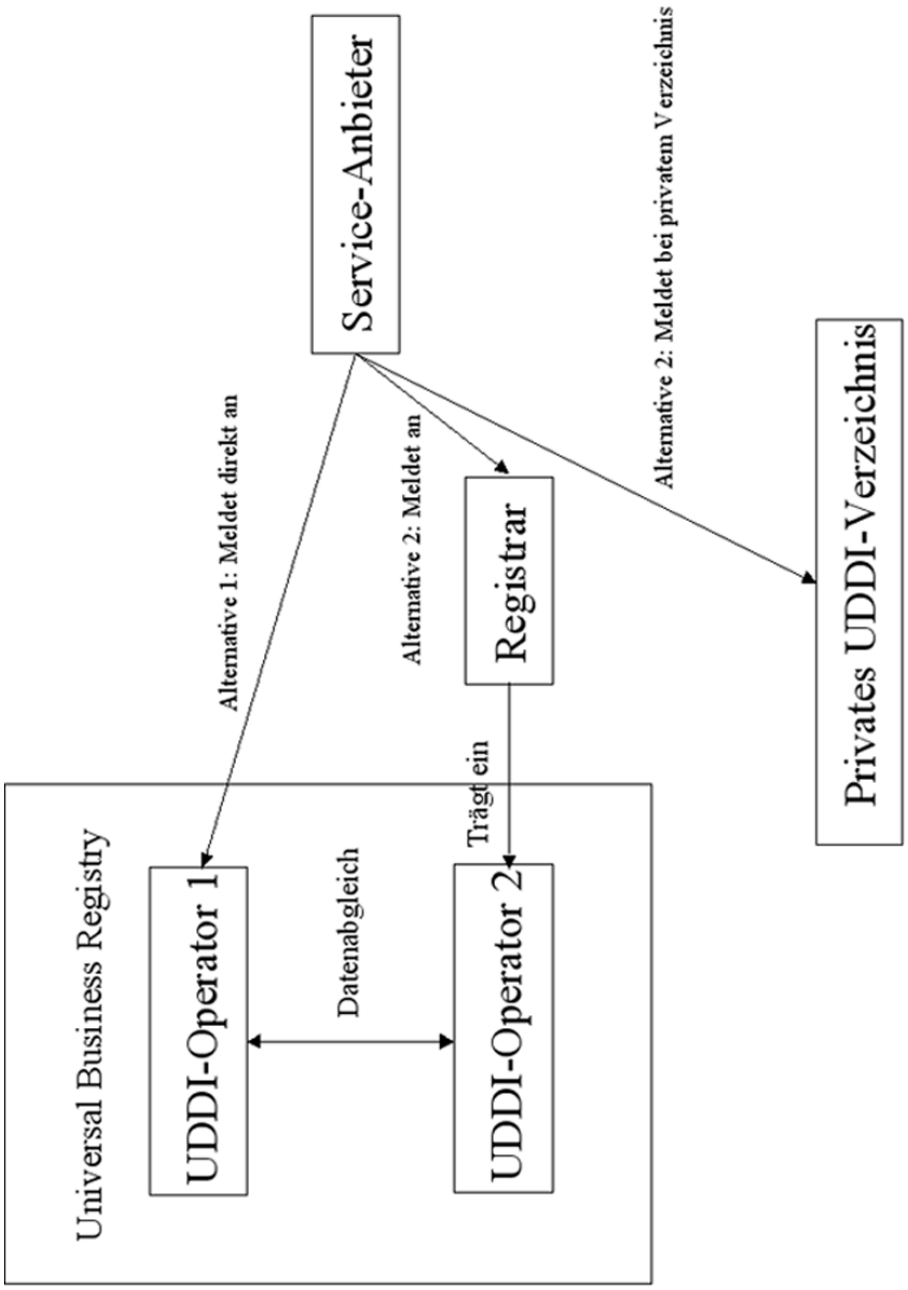

Abbildung 5.6: Die UDDI-Architektur am Beispiel der Anmeldung eines Web Service, Quelle: Hauser und Löwer (2004, S. 104)

5.4 Web Services in der Auktionssystem-Anwendung

In diesem Abschnitt wird zunächst die Erstellung von Web Services mit Apache *Axis*[16] erläutert. Dabei wird auf das Publizieren mit JWS[17] (Java Web Service)- und WSDD[18] (Web Service Deployment Descriptor)-Dateien eingegangen. Des Weiteren wird sowohl die Implementierung eines Web Services als auch die Anbindung eines externen zur Verfügung stehenden Web Services beschrieben.

5.4.1 Web Services mit Apache Axis

Folgende Möglichkeiten können in Anspruch genommen werden, um Web Services mit *Axis* zu publizieren:

- *JWS* (Java Web Service) Deployment und
- Deployment durch *Deployment Descriptor*

Publizieren mit JWS Dateien

Die einfachste Möglichkeit, einen Web Service mit *Axis* (Apache, 2003) durch SOAP zur Verfügung zu stellen, ist der Einsatz von JWS-Dateien. Zuerst sollte die eigentliche Funktion des Web Services implementiert und in einer Java-Datei abgelegt werden. Diese Datei sollte im Web Application-Ordner gespeichert werden, z.B. unter <webapp-root>/axis/SomeWSImplFile.jws. Angenommen, der Server läuft auf Port 8080, dann kann der Web Service durch die Adresse http://localhost:8080/axis/SomeWSImplFile.jws aufgerufen werden. *Axis* erkennt automatisch die Datei, kompiliert die Klasse und konvertiert SOAP-Aufrufe in Java-Aufrufe des Web Services.

[16] Apache *Axis* der Apache Software Foundation ist eine Open Source Implementierung des SOAP-Standards. Er besteht hauptsächlich aus drei Servlets (Axis Servlet, der Services aufnimmt und ihnen Anfragen weitergibt; Admin Servlet zum Installieren und Deinstallieren von Web Services; SOAP Monitor Servlet gibt die Möglichkeit zur Überwachung der SOAP-Nachrichten), welche in einem Web Container, z. B. wie Tomcat, installiert werden können. Für weitere Informationen dazu siehe http://ws.apache.org/axis/.

[17] JWS-Dateien bieten eine schnelle Möglichkeit, Web Services zu publizieren. Weiteres dazu folgt im nächsten Unterabschnitt.

[18] WSDD-Dateien sind XML-Dateien, welche genauere Angaben über einen Web Service definieren. Weiteres dazu folgt im nächsten Unterabschnitt.

Diese Art von Web Services sind einfach zu realisieren. Sie sollten daher zum Implementieren einfacher Web Services eingesetzt werden, weil

- keine bereits vorhandenen Pakete importiert werden müssen und
- der Quelltext zur Laufzeit kompiliert wird und damit keine Fehler erkannt werden können, bis das Deployment nicht abgeschlossen ist.

Publizieren mit WSDD (Web Service Deployment Descriptor)

Das Web Service Deployment Descriptor ist wie bereits erwähnt eine XML-Datei. Deren Funktionalität soll mit Hilfe des Beispiels auf Listing 5.4 erklärt werden.

Listing 5.4: Beispiel einer WSDD-Datei

```
1 <deployment xmlns="http://xml.apache.org/axis/wsdd/"
2  xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
3
4  <handler name="service_handler" type="java:samples.Logger">
5   <parameter name="filename" value="SampleService.log"/>
6  </handler>
7
8  <service name="SampleService" provider="java:RPC">
9   <requestFlow>
10   <handler type="service_handler"/>
11  </requestFlow>
12  <parameter name="className" value="samples.SampleService"/>
13  <parameter name="allowedMethods" value="*"/>
14  <parameter name="scope" value="Request"/>
15 </service>
16 </deployment>
```

Durch das *service*-Element (Zeilen 8-15) wird hier der eigentliche Service definiert. Durch sein *provider*-Attribut in der Zeile 8 wird ein *java:RPC*-Service bekanntgegeben. Mit dem ersten untergesetzten *parameter*-Tag in der Zeile 12 wird dem Provider die *Service*-Klasse bekanntgegeben. Mit dem weiteren *parameter*-Tag wird der Aufruf jeder *public*-Klasse des Services erlaubt. Hier könnten die Namen der zum Aufruf erlaubten Methoden, getrennt durch Leerzeichen oder Komma, eingetragen werden. Das nächste *parameter*-Element definiert die Umgebung, in der die Objekte gültig sind. Es sind folgende Werte zugelassen:

- *Request*: Bei jedem Aufruf des Services wird ein neues Objekt des Services erstellt.
- *Application*: Für alle Aufrufe dieses Services wird ein einziges Objekt des Services genutzt.
- *Session*: Hier wird ein neues Objekt zu jeder neuen Sitzung erstellt.

Dieser Service wird weiterhin mit einem *Handler* erweitert. Die Zeilen 4-6 definieren einen Handler, der *service_handler* genannt ist. Bei der Definition wird die implementierende Klasse *Logger* bekanntgegeben. Mit dem *paramater*-Element in der Zeile 5 wird der Name der Log-Datei angegeben. Innerhalb des *service*-Elements wird der Handler mit dem Element *requestFlow* eingebunden. Das stellt sicher, dass beim Aufruf des Web Services die Nachricht in der angegebenen Datei protokolliert wird.

5.4.2 Web Services Implementierung

Innerhalb der Anwendung wurde ein Web Service zu Demo-Zwecken erstellt. Er hat die Aufgabe, zu einer bestimmten Artikelnummer den Namen des Produktes zurückzuliefern. Die Vorgehensweise soll im Folgenden beschrieben werden. Weiterhin wurde auch ein externer Web Service eingebunden. Dieser Vorgang wird in diesem Abschnitt genauer beschrieben.

Web Services entwickeln

Zunächst soll der Service implementiert werden. Für diesen Zweck wurde die Klasse *FirstWs* im Paket *myapplication.test* erstellt. Sie implementiert die Methode *getProductName()*, welche einen Wert für die Artikelnummer erwartet und einen Namen des Artikels, wie aus Listing 5.6 ersichtlich, zurückliefert.

Listing 5.5: Die Methode *getProductName()* in der Klasse *FirstWs*

```
1  public String getProductName(String sProdNr){
2      String sProdName = "";
3      String sql = "select name from tdat_product where
4                           pk=" + String.valueOf(sProdNr);
5      try{
6      rs = db.executeQuery(sql);
7      rs.next();
8      sProdName = rs.getObject("NAME").toString();
9      }catch(Exception e){
10         e.printStackTrace();
```

```
11      }
12      return sProdName;
13  }
```

Um die Service-Funktion zu testen wurde das *Axis*-Framework unter dem Tomcat Server installiert. Der Web Service soll nun zur Verfügung gestellt werden. Der Programmcode wird im Verzeichnis

```
axis/WEB-INF/classes
```

auf dem Tomcat Server platziert. Als nächstes wird die Deployment-Datei *deploy_FirstWs.wsdd* erstellt. Sie ist aus Listing 5.6 zu entnehmen.

Listing 5.6: Die Methode *sProdNr()* in der Klasse *FirstWs*

```
1 <deployment xmlns="http://xml.apache.org/axis/wsdd/"
2     xmlns:java=
3      "http://xml.apache.org/axis/wsdd/providers/java">
4     <service name="FirstWs" provider="java:RPC">
5        <parameter name="className"
6                   value="myapplication.test.FirstWs"/>
7        <parameter name="methodName" value="getProductName"/>
8     </service>
9 </deployment>
```

In der Zeile 4 wird mit dem Attribut *name* der Name des Web Service und mit dem Attribut *provider* seine Art vom Typ RPC bekanntgegeben. Zeilen 5 und 6 geben entsprechend die Klasse *FirstWs* und die in dieser Klasse aufzurufende Methode *getProductName()* bekannt.

Um das eigentliche Deployen durchzuführen, kann im DOS-Fenster der folgende Befehl ausgeführt werden:

```
java org.apache.axis.client.AdminClient deploy_FirstWs.wsdd
```

Der aufgestellte Service *FirstWs* kann in einer Liste der verfügbaren Services nach dem Starten des Tomcat Server und Aufruf der Adresse

```
http://127.0.0.1:8080/axis/servlet/AxisServlet
```

gesehen werden. Das Ergebnis kann wie auf der Abbildung 5.7 aussehen. Eine Auswahl des Links, der auf dem Text *(wsdl)* neben dem Servicenamen *FirstWs* zu sehen ist, zeigt die WSDL-Datei des Web Services an. Der Browser-Bildschirm, der die WSDL-Datei aufgerufen hat, sieht wie in der Abbildung 5.8 angezeigt

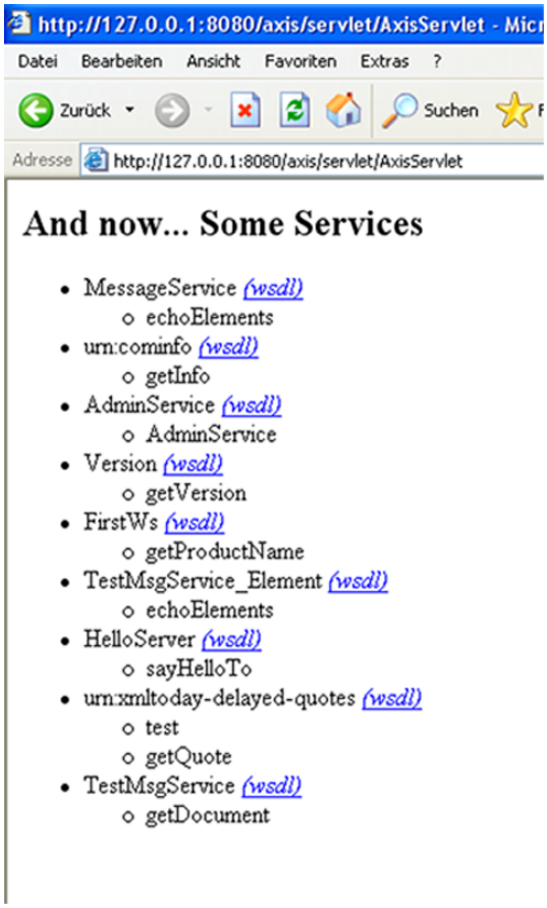

Abbildung 5.7: Liste der unter Apache *Axis* zur Zeit zur Verfügung gestellten Web Services

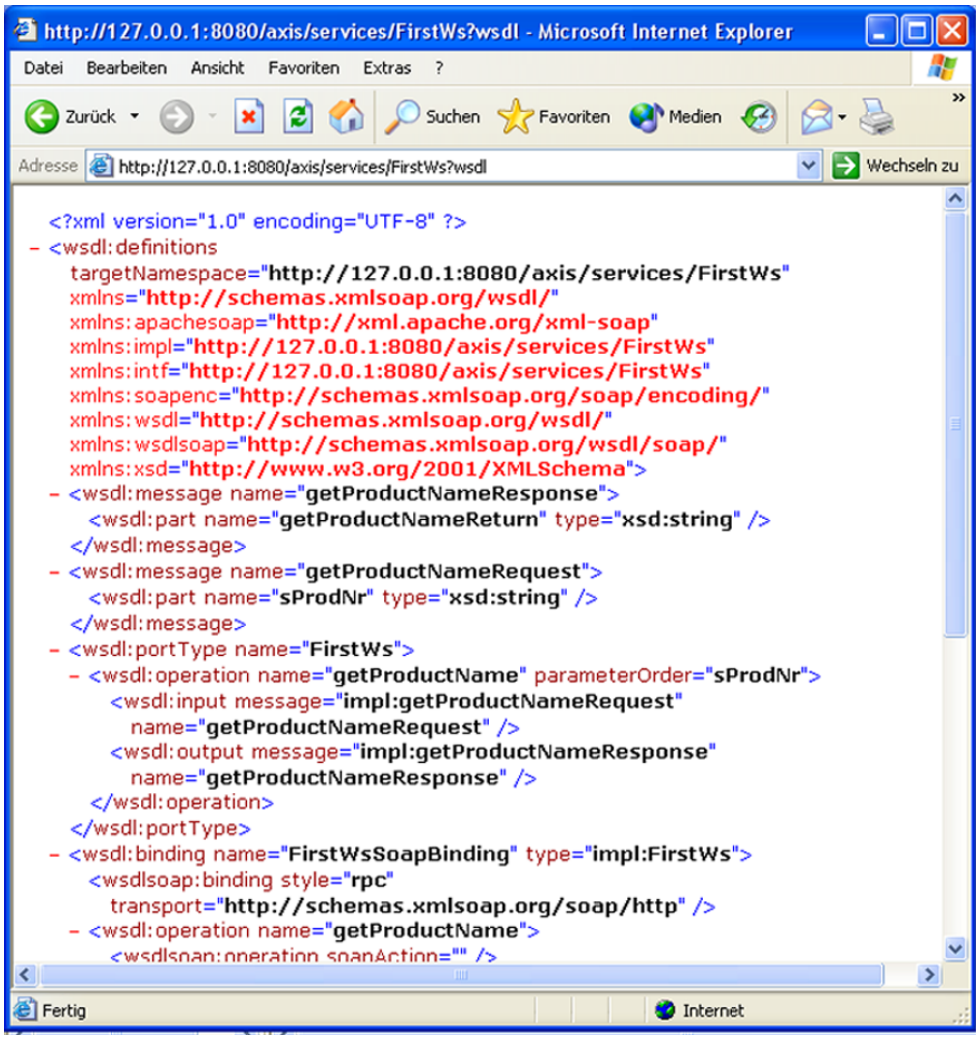

Abbildung 5.8: WSDL-Datei des Web Services *FirstWs*

aus. Damit ist die Installation des Web Services beendet und dieser kann nun vom Client aufgerufen werden.

Für den Client-Aufruf des Web Services wurde die Methode *getProductName()* in der Klasse *GetInfo* des Pakets *myapplication.wsclient* erstellt, welche auf Listing 5.7 angezeigt wird.

Listing 5.7: Die Methode *getProductName()* in der Klasse *GetInfo*

```
1  public void getProductName(String args) {
2    try {
3       Service  service = new Service();
4       Call call = (Call) service.createCall();
5       call.setTargetEndpointAddress(
6          "http://127.0.0.1:8080/axis/services/FirstWs");
7       call.setOperationName(
8          new QName("
9               impl:FirstWsSoapBinding", "getProductName"));
10      call.addParameter(
11         "sProdNr", XMLType.XSD_STRING, ParameterMode.IN );
12      call.setReturnType( XMLType.XSD_STRING);
13
14      String res = (String) call.invoke(new Object[]{args});
15
16      Message msg = call.getResponseMessage();
17      SOAPEnvelope env = msg.getSOAPEnvelope();
18      Document doc = env.getAsDocument();
19      XmlTool tool = new XmlTool();
20      tool.writeXmlFile(doc, "D:\\SOAPMessage.xml");
21   } catch( Exception e ) {
22      e.printStackTrace();
23   }
24 }
```

Zunächst wird in der Zeile 4 ein *Call*-Objekt für den Service erstellt. Die Zeilen 5-12 legen die Adresse, die Methode, den Typ und den Namen des Parameters fest. Zeile 14 ruft den Service auf und speichert das Ergebnis im *String*-Objekt *res*. Die Zeilen 16-20 dienen dazu, die SOAP-Antwortnachricht zu ermitteln und in einer XML-Datei zu speichern. Die entsprechende SOAP-Antwort kann dem Listing 5.8 entnommen werden.

Listing 5.8: SOAP-Antwort des Web Services

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <soapenv:Envelope
3  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
4  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
5  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
6  <soapenv:Body>
7   <ns1:getProductNameResponse
8      soapenv:encodingStyle=
9           "http://schemas.xmlsoap.org/soap/encoding/"
10       xmlns:ns1="impl:FirstWsSoapBinding">
11    <ns1:getProductNameReturn xsi:type="xsd:string">
12     Flugticket nach Bulgarien am 12.12.2003
13    </ns1:getProductNameReturn>
14   </ns1:getProductNameResponse>
15  </soapenv:Body>
16 </soapenv:Envelope>
```

Aus den Zeilen 2-16 des Listings 5.8 ist das SOAP-Envelope-Element ersichtlich. Es enthält einen SOAP-Body in den Zeilen 6-15. Dieser besitzt das Kind-Element *getProductNameReturn*, welches das *String*-Objekt mit dem Namen des Artikels zurückliefert.

Externe Web Services einbinden

In der Anwendung wurde die Anbindung eines externen Web Services am Beispiel der zur Verfügung gestellten Suchmöglichkeit mit Web Services von Google[19] implementiert. Das benötigte API wurde heruntergeladen und im System eingebunden. Nach dem Erstellen eines Accounts bei Google[20] wurde weiterhin ein freier Lizenzschlüssel beantragt, welcher bis zu 1000 Abfragen täglich ermöglicht. Anschließend soll der Aufruf des Web Services implementiert werden.

Der Aufruf des Web Services wurde in der Klasse *Google* vom Paket *myapplication.wsclient* durch die Methode *googleSearch()* realisiert. Sie erwartet ein *String*-Objekt mit den Stichworten für die Abfrage und liefert ein Objekt vom Typ *GoogleSearchResultElement[]* mit den ersten zehn Ergebnissen für die Anfrage zurück. Die Methode wird in Listing 5.9 angezeigt.

[19] http://www.google.com/apis/

[20] https://www.google.com/accounts/NewAccount?continue=
http://api.google.com/createkey&followup=http://api.google.com/createkey

Listing 5.9: Die Methode *googleSearch()* in der Klasse *Google*

```
1  public void googleSearch(String sSearch){
2     try {
3        String clientKey = "lPuo6EhQFHLFNhk66I0+tQhgn8uKvtYf";
4        GoogleSearch s = new GoogleSearch();
5        s.setKey(clientKey);
6        s.setQueryString(sSearch);
7
8        GoogleSearchResult r = s.doSearch();
9
10       System.out.println("Google Search Results:");
11       System.out.println("======================");
12       System.out.println(r.toString());
13
14       int resNr = r.getEstimatedTotalResultsCount();
15       GoogleSearchResultElement[] rsElements =
16                              r.getResultElements();
17
18       for(int i = 0; i < 10; i++){
19          GoogleSearchResultElement rsElement =
20                      new GoogleSearchResultElement();
21          rsElement = rsElements[i];
22          System.out.println(rsElement.getTitle());
23       }
24    } catch (GoogleSearchFault f) {
25       System.out.println("Call failed:");
26       System.out.println(f.toString()); }
27 }
```

Zunächst wird in der Zeile 3 der erworbene Client-Schlüssel bekanntgegeben, wonach das Bilden des *GoogleSearch*-Objektes *s* folgt. Zeilen 5 und 6 geben dem *s*-Objekt den Schlüssel und die Abfrage-Stichworte bekannt. Das eigentliche Suchen der Ergebnisse geschieht in der Zeile 8. Sie werden im Objekt *GoogleSearchResult r* gespeichert. In Zeile 16 wird mit der Methode *getResultElements()* das Objekt *GoogleSearchResultElement[] rsElements* ermittelt. Es enthält die benötigten Ergebnisse und ermöglicht deren weitere Bearbeitung bzw. deren Darstellung für den Benutzer.

Kapitel 6

Schlussbetrachtung und Ausblick

Das Ziel der vorliegenden Arbeit ist es, den Einsatz des XML-Standards innerhalb von Webanwendungen zu untersuchen. Als Anwendungsbeispiel diente die Entwicklung eines Prototyps für holländische Auktionen.

Der Beginn der Arbeit beschäftigt sich im zweiten Kapitel mit den theoretischen Aspekten des elektronischen Handels. Die ausschlaggebenden Vorteile, die sich für die Parteien Käufer und Verkäufer ergeben, sind wesentliche Gründe zur Durchführung von Geschäften über das Internet. Diese werden durch das Internet und seine besondere Eigenschaften zur Verfügung gestellt. Diese Aspekte wurden analysiert. Anschließend wurden auch die Begriffe e-Commerce und e-Business abgegrenzt und in Folge davon der Begriff des Geschäftsmodells ausführlich charakterisiert. Sinngebend sind dabei Merkmale wie Zielgruppe, Objekte des Handels, Preisfindung und Art der Einkünfte. Diese müssen im Voraus genau überlegt werden, um einen echten Mehrwert im Wertschöpfungsnetzwerk zu liefern.

Damit wurde die Grundlage für den nächsten Abschnitt gelegt. Er beschreibt die Auktionsarten, -funktionen, -nachteile und -vorteile und stellt daraufhin ein Geschäftsmodell zusammen. Der letzte Abschnitt liefert Grundlagen, die den XML-Standard und seine Einsatzmöglichkeiten im Unternehmen betreffen. Kapitel drei liefert die Grundlagen für die zu entwickelnde Webanwendung. Hier werden einige Aspekte wie Struktur, vorhandene Technologien und allgemeine Anforderungen an die Anwendung dargestellt und darauffolgend der Bedarf an das zu entwickelnde System zusammengestellt. Anschließend wird konkret auf die Vorgehensweise der Programmierung der Anwendungsprozesse eingegangen. Die Dynamik der Anwendung wird durch JSP-Seiten zur Verfügung gestellt. Die dynamischen Daten des Systems werden mittels des XML-Formats vom Datenbanksystem an die Anwendung weitergeleitet. Es wurden dafür spezielle Klassen entwickelt, die beliebige Konvertierungen der Datenbankdaten in beliebige XML-Dokumente realisieren. Struktur und Inhalt der XML-Daten hängen allein von den Definitionen ab, die in den Systemtabellen der Daten-

bank vorgenommen wurden. Jede JSP-Seite des Systems findet auch ihre eigene Definition in der Datenbank und kann mit den Daten eines XML-Dokuments verknüpft werden. Damit bleiben die XML-Dokumente unabhängig von den vom Benutzer aufgerufenen Daten und von den Datenbankdaten.

Kapitel vier geht ins Detail der Entwicklung der XML-Schnittstelle. Es wird dabei ein flexibles System geschaffen, das die Nutzdaten des vorhandenen Datenbanksystems in das standardisierte XML-Format konvertiert. Die Definition der aufzurufenden Daten wurde an dieser Stelle anhand von Beispieldaten vorgestellt. Die vorhandenen Definitionen von XML-Daten können auch mehrmals genutzt werden und stehen sowohl internen als auch externen Systemen zur Verfügung. Des Weiteren wurden die Möglichkeiten zur Transformation von XML-Daten erläutert und auf deren spezifischen Eigenschaften eingegangen. Anhand von Beispielen wurden XML-Daten anhand der Standards XSLT, XSL und XSL-FO transformiert. Ferner wurde das Signieren von XML-Daten dargestellt und anhand der vorhandenen Implementierung die technische Vorgehensweise und Vorteile näher erläutert.

Web Services sind die erfolgversprechende Entwicklung, was die Integration von Systemen angeht. Die Möglichkeit der einfachen Systemintegration, die durch Plattform- und Sprachunabhängigkeit ermöglicht wird, verspricht einen echten Mehrwert in der zukünftigen IT-Welt. Kapitel fünf der Arbeit sorgt für einen allgemeinen Überblick über diese wichtige zukünftige Technologie und beschreibt das Web Services-Modell. Ferner werden wichtige verbreitete Standards wie SOAP, WSDL und UDDI dargestellt. Anschließend wird mit dem Einsatz von Apache Axis das Publizieren von Web Services erläutert. Damit werden interne Web Services einem externen System zur Verfügung gestellt. Danach wurde die Anbindung von Google Web Services als bereits bestehendem Web Service dargestellt. Die Anbindung und dessen Nutzung zeigt die Einfachheit der Anbindung von bestehenden Services, ohne ihre Funktionalität zu kennen.

In Zukunft werden weitere XML-Standards entwickelt, die für die Vereinfachung und Verbesserung von Web-Anwendungen sorgen. Neue Standards wie XForms beispielsweise sollen Webformulare vereinfachen. Auf diesen Standard wurde in der Arbeit nicht eingegangen, da er seit Oktober 2003 in seiner ersten Version existiert und von Standard-Browsern nicht erkannt wird. Die W3C Spezifikation XForms (Dubinko u.a., 2003) hat die Aufgabe, einen Standard zu entwickeln, welcher darauf ausgerichtet ist, die herkömmlichen HTML-Formulare zu ersetzen. Sie realisiert die klare Trennung zwischen dem

Zweck eines Formulars (XForms Modell), seiner Darstellung (XForms Benutzerinterface) und Ergebnissen („instance data“). Sie stellt folgende Vorteile zur Verfügung:

- „Wiederverwendbarkeit: XForms Module können unabhängig von den Informationen, die sie sammeln, unabhängig wiederverwendet werden.
- Geräteunabhängigkeit: Benutzerinterfaces sind abstrakt, d.h. es werden nur ihre generischen Eigenschaften angezeigt, so dass sie einfach in verschiedenen Geräten mit unterschiedlichen Fähigkeiten eingesetzt werden können.
- Zugänglichkeit: Die Trennung von Inhalten und deren Darstellung macht die eigentliche Information für Nutzer mit assistiver Technologie lesbarer. Darüber hinaus kapseln die Benutzerinterfaces alle relevanten Metadaten (z.B. Labels) und ermöglichen dadurch eine Zugänglichkeit zu Anwendungen, die verschiedene Modalitäten benutzen.“ (Daly u.a., 2003)

Literaturverzeichnis

[Adam 2004] Adam, C.: *Web Services Business Modells*. URL `http://www.webservices.org/print.php?sid=361`, 2004. – [26.01.2004]

[Adler 2001] Adler, Sharon ; Berglund, Anders ; Caruso, Jeff ; Deach, Stephen ; Graham, Tony ; Grosso, Paul ; Gutentag, Eduardo ; Milowski, Alex ; Parnell, Scott ; Richman, Jeremy ; Zilles, Steve: *Extensible Stylesheet Language (XSL) Version 1.0*. URL `http://www.w3.org/TR/2001/REC-xsl-20011015/`, 2001. – [11.05.2004]

[Albers 2000] Albers, Sönke: *Was kauft sich im Internet? – Produkte und Leistungen*. S. 21–36. In: *S. Albers, M. Clement, K. Peters, B. Skiera (Hrsg.): eCommerce. Einstieg, Strategie und Umsetzung im Unternehmen.*, F.A.Z.-Institut, 2000

[Albers u. a. 2000a] Albers, Sönke ; Clement, Michael ; Peters, Kay ; Skiera, Bernd: *Warum ins Internet? – Erlösmodelle für einen neuen Kommunikations- und Distributionskanal*. S. 9–19. In: *S. Albers, M. Clement, K. Peters, B. Skiera (Hrsg.): eCommerce. Einstieg, Strategie und Umsetzung im Unternehmen.*, F.A.Z.-Institut, 2000

[Albers u. a. 2000b] Albers, Sönke ; Clement, Michael ; Skiera, Bernd: *Wie sollen Produkte vertrieben werden? – Distributionspolitik*. S. 79–95. In: *S. Albers, M. Clement, K. Peters, B. Skiera (Hrsg.): eCommerce. Einstieg, Strategie und Umsetzung im Unternehmen.*, F.A.Z.-Institut, 2000

[Amor 2000] Amor, Daniel: *Dynamic Commerce*. Galileo Press GmbH, 2000

[Apache 2003] Foundation, The Apache S.: *Apache Wiki: AxisProjectPages/Compare*. URL `http://ws.apache.org/axis/java/user-guide.html`, 2003. – [30.10.2003]

[Apparao 1998] Apparao, Vidur ; Byrne, Steve ; Champion, Mike ; Isaacs, Scott ; Jacobs, Ian ; Hors, Arnaud L. ; Nicol, Gavin ; Robie, Jonathan ; Sutor, Robert ; Wilson, Chris ; Wood, Lauren: *Document Object*

Model (DOM) Level 1 Specification. URL `http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/`, 1998. – [07.05.2004]

[Bartel u.a. 2002] Bartel, Mark ; Boyer, John ; Fox, Barb ; LaMacchia, Brian ; Simon, Ed: *XML-Signature Syntax and Processing*. URL `http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/`, 2002. – [17.05.2004]

[Becker 2004] Becker, David: *XML: Too much of a good thing? — CNET News.com*. URL `http://news.com.com/XML%3A+Too+much+of+a+good+thing%3F/2100-1013_3-5347309.html?tag=st.pop`, 2004. – [07.09.2004]

[Böhm und Felt 2001] Böhm, Andreas ; Felt, Elisabeth: *E-Commerce kompakt*. 2001

[Biron u.a. 2001] Biron, Paul V. ; Malhotra, Ashok(Hrsg.): *XML Schema Part 2: Datatypes, W3C Recommendation, 2 May 2001*. URL `http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/`, 2001. – [20.05.2004]

[Bitzer 2003] Bitzer, Frank: *XML im Unternehmen*. Galileo Computing, 2003

[BMWi 2001] BMWi: Chancen und Risiken inverser Auktionen im Internet für Aufträge der öffentlichen Hand / Im Auftrag des Bundesministeriums für Wirtschaft und Technologie. URL `http://bmwi.de`, 2001 (496). – Abschlussbericht

[Brandstetter und Fries 2002] Brandstetter, Clemens ; Fries, Marc: *E-Business im Vertrieb*. Carl Hanser Verlag, 2002

[Bray 2004] Bray, Tim ; Paoli, Jean ; Sperberg-McQueen, C. M. ; Maler, Eve ; Yergeau, François ; Cowan, John: *Extensible Markup Language (XML) 1.1*. URL `http://www.w3.org/TR/2004/REC-xml11-20040204/`, 2004. – [07.05.2004]

[Bruns und Dunkel 2003] Bruns, Ralf ; Dunkel, Jürgen: *Ein eLearning-Portal unter Einsatz von XML und XSLT*. S. 85–111. In: *P. Mertens (Hrsg.): XML-Komponenten in der Praxis*, Springer Verlag, 2003

[Bryden 2003] Bryden, Alan: *Open and global standards for achieving an enclusive information society*. URL `http://www.iso.ch/iso/en/commcentre/presentations/secgen/2003/ajb2003SISTspeech.pdf`, 2003. – [8.02.2004]

[Buhl 2004] Buhl, Axel: *Grundkurs Software-Projektmanagement*. Hanser Verlag, 2004

[Chappell und Jewell 2002] Chappell, David ; Jewell, Tyler: *Java Web Services*. O'Reilly, 2002

[Chinnici u.a. 2004] Chinnici, Roberto ; Gudgin, Martin ; Moreau, Jean-Jacques ; Schlimmer, Jeffrey ; Weerawarana, Sanjiva(Hrsg.): *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, W3C Working Draft, 26 March, 2004*. URL `http://www.w3.org/TR/2004/WD-wsdl20-20040326/`, 2004. – [25.05.2004]

[Clark 1999] Clark, James: *XSL Transformations (XSLT) Version 1.0*. URL `http://www.w3.org/TR/1999/REC-xslt-19991116`, 1999. – [11.05.2004]

[Clark 2001] Clark, Mike: *Business Architecture for a Web Services Brokerage: Understanding the Business Context of Web Services*. URL `http://www.webservicesarchitect.com/content/articles/clark01.asp`, 2001. – [25.05.2004]

[Clark u.a. 1999] Clark, James ; DeRose, Steve: *XML Path Language (XPath) Version 1.0*. URL `http://www.w3.org/TR/1999/REC-xpath-19991116`, 1999. – [8.12.2004]

[Clement u. a. 2001] Clement, Michel ; Peters, Kay ; Preiss, Friedrich J.: *Electronic Commerce*. S. 49–64. In: *S. Albers, M. Clement, K. Peters (Hrsg.): Marketing mit Interaktiven Medien. Strategien zum Markterfolg.*, IMK, 2001

[computerlexicon.com 2004] *DCL (Data Control Language)*. URL `http://www.computerlexikon.com/begriff.php?id=1643`, 2004. – [31.07.2004]

[Daly u.a. 2003] Daly, Janet ; Forgue, Marie-Claire ; Hirakawa, Yasuyuki: *World Wide Web Consortium gibt XForms 1.0 als W3C Recommendation frei*. URL `http://www.w3c.de/Press/xforms-pressrelease.html`, 2003. – [19.07.2004]

[Darby 1998] Darby, Chád: Applet and Servlet Communication. In: *Java Developer's Journal* ? (1998), September, Nr. 9, S. ?–?. – URL `http://www.j-nine.com/pubs/applet2servlet/Applet2Servlet.html`. – [31.05.2004]

[Deitel u. a. 2003] Deitel, Harvey ; Deitel, Paul ; B.DuWaldt ; Trees, L.: *Web Services: A Technical Intoduction*. Prentice Hall, 2003

[Dubinko u.a. 2003] Dubinko, Micah ; Klotz, Leigh L. ; Merrick, Roland ; Raman, T. V.: *XForms 1.0 W3C Recommendation, 14 Oktober, 2003*. URL `http://www.w3.org/TR/2003/REC-xforms-20031014/`, 2003. – [19.07.2004]

[Ebner 1999] Ebner, Michael: *SQL lernen*. Addison-Wesley Verlag, 1999

[Eckstein und Eckstein 2004] Eckstein, Rainer ; Eckstein, Silke: *XML und Datenmodellierung*. Dpunkt Verlag, 2004

[Evans und Wurster 2000] Evans, Philip B. ; Wurster, Thomas S.: *Die Informationsrevolution: Alte Geschäfte vergehen, neue entstehen*. S. 42–58. In: *Don Tapscott (Hrsg.): Erfolg im E-Business*, Carl Hanser Verlag, 2000

[Fallside 2001] Fallside, D.C.(Hrsg.): *XML Schema Part 0: Primer, W3C Recommendation, 2 May 2001*. URL `http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/`, 2001. – [20.05.2004]

[Feld und Hoffmann 2000] Feld, Thomas ; Hoffmann, Michael: *Viertuelle Marktplätze: Die dritte Dimension des Online-Handelns*. S. 193–214. In: *A. Scheer (Hrsg.): E-Business: Wer geht? Wer Bleibt? Wer kommt?*, Physica-Verlag, 2000

[Grüne und Kneuper 2002] Grüne, Markus ; Kneuper, Ralf: Web Engineering. In: *Wirtschaftsinformatik* 44 (2002), Juni, Nr. 3, S. 269–275

[Großwendt 2002] Grosswendt, Volkmar: Namespaces, die zweite ... Neuerungen in Namespaces in XML 1.1. In: *XML & Web Services Magazin* 1 (2002), Oktober–November, Nr. 3, S. 36–38

[Gudgin u.a. 2004 2004] Gudgin, Martin ; Lewis, Amy ; Schlimmer, Jeffrey(Hrsg.): *Web Services Description Language (WSDL) Version 2.0 Part 2: Message Exchange Patterns, W3C Working Draft, 26 March, 2004*. URL `http://www.w3.org/TR/2004/WD-wsdl20-patterns-20040326/`, 2004. – [25.05.2004]

[Gundavaram 1996] Gundavaram, Shishir: *CGI Programmierung*. O'Reilly Verlag, 1996

[Haas 2004] Haas, Hugo: *W3C Technologies as a key for interoperability - slide „Web Services: definition“*. URL `http://www.w3.org/2004/Talks/0226-jtc1-hh/slide5-0.html`, 2004. – [07.08.2004]

[Hansen und Neumann 2001] Hansen, Hans R. ; Neumann, Gustaf: *Wirtschaftsinformatik*. Lucius & Lucius Verlag, 2001

[Harold und Means 2002] Harold, Elliotte R. ; Means, W. S.: *XML in a Nutshell*. O'Reilly, 2002

[Hauser und Löwer 2004] Hauser, Tobias ; Löwer, Ulrich M.: *Web Services. Die Standards*. Galileo Computing, 2004

[Heinzmann 2002] Heinzmann, Peter: *Internet – Die Kommunikationsplattform des 21. Jahrhunderts*. S. 41–77. In: *R. Weiber (Hrsg.): Handbuch Electronic Business*, Gabler Verlag, 2002

[Hofmann 2002] Hofmann, Josephine: *Multimedia — Information und Kommunikation: Voraussetzungen für die digitale Wertschöpfung*. S. 27–40. In: *Haasis K., Strommer W., Zerfaß A. (Hrsg.): Digitale Wertschöpfung. Internet und E-Business als Chance für den Mittelstand*, Dpunkt Verlag, 2002

[Hohenstein 2003] Hohenstein, Uwe: Generierung von XML aus relationalen Daten. In: *Datenbank Spektrum* 3 (2003), (10/2003), Nr. 7, S. 52–61

[Hoppe u. a. 2002] Hoppe, Gabriela ; Brüggemann, Tobias ; Schwarze, Jochen: E-Commerce und LAMP-Architektur. In: *Das Wirtschaftsstudium* 31 (2002), (11/2002), Nr. 11, S. 1426–1433

[Imamura u. a. 2002] Imamura, Takeshi ; Dillaway, Blair ; Simon, Ed: *XML Encryption Syntax and Processing*. URL `http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/`, 2002. – [9.07.2003]

[internet-manual.de 2004] o.V.: *E-Business – Übersicht*. URL `http://www.internet-manual.de/ebusiness.htm`, 2004. – [03.02.2004]

[Janowicz 2002] Janowicz, Krzysztof: *Sicherheit im Internet*. O'Reilly Verlag, 2002

[Jeckle 2003] Jeckle, Mario: *Web Services Begriffsdefinitionen*. URL `http://www.jeckle.de/webServices/index.html#def`, 2003. – [28.10.2003]

[Jeckle 2004] Jeckle, Mario: *Scriptum zur Vorlesung Datenbanken*. URL `http://www.jeckle.de/vorlesung/datenbanken/script.html`, 2004. – [05.03.2004]

[Joung 2004] Joung, Philip: Bulletproof Web Application Deployments. In: *Web Services Journal* 4 (2004), January, Nr. 1, S. 14–17

[Kappel u. a. 2003a] Kappel, Gerti ; Pröll, Birgit ; Reich, Siegfried: *Web Engineering*. Dpunkt Verlag, 2003

[Kappel u. a. 2003b] Kappel, Gerti ; Retschitzegger, Werner ; Pröll, Birgit ; Unland, Rainer ; Vojdani, Bahram: *Architektur von Web-Informationssystemen*. S. 101–134. In: *E. Rahm, G. Vossen (Hrsg.): Web & Datenbanken. Konzepte, Architekturen, Anwendungen*, Dpunkt Verlag, 2003

[Kauffels 2001] Kauffels, Franz-Joachim: *E-Business*. mitp-Verlag, 2001

[Kauffels 2002] Kauffels, Franz-Joachim: *Durchblick im Netz*. mitp-Verlag, 2002

[Klein 1997] Klein, Stefan: Introduction to Electronic Auctions. In: *B. F. Schmid, S. Klein: EM - Electronic Auctions. EM - Electronic Markets* 7 (1997), (12/1997), Nr. 4, S. 3–6. – URL `http://www.electronicmarkets.org/modules/pub/view.php/electronicmarkets-233`. – [18.02.2004]

[Knobloch 2002] Knobloch, Manfred: Seitengeometrie. Umwandlung von XML in PDF mit XSL Formatting Objects. In: *Der Entwickler* 7? (2002), März–April, Nr. 2, S. 93–97

[Kollmorgen u. a. 2003] Kollmorgen, René ; Dieter Kessler, Eckehard H. ; Jung, Frank: Gezeichnet ... XML. Digitale Signaturen in XML. In: *XML & Web Services Magazin* 2 (2003), Februar–März, Nr. 2, S. 42–46

[Kossmann und Leymann 2004] Kossmann, Donald ; Leymann, Frank: Web Services. In: *Informatik Spektrum* 27 (2004), April–Juni, Nr. 2, S. 117–128

[Kuschke und Wölfel 2002] Kuschke, Michael ; Wölfel, Ludger: *Web Services kompakt*. Spektrum akademischer Verlag, 2002

[Langham und Ziegeler 2002] Langham, Matthiew ; Ziegeler, Carsten: Cocoon2: Neue Architekturen und neue Features unter der Lupe. In: *Javamagazin* 5? (2002), März, Nr. 3, S. 41–48

[Langner 2002] Langner, Torsten: *Verteilte Anwendungen mit Java: Enterprice-Architekturen im Web mit CORBA, XML/SOAP, JSP, (E)JB und JDBC*. Markt+Technik Verlag, 2002

[Le Hors 2001] Hors, Arnaud L. ; Hégaret, Philippe L. ; Wood, Lauren ; Nicol, Gavin ; Robie, Jonathan ; Champion, Mike ; Byrne, Steve: *Document Object Model (DOM) Level 2 Core Specification*. URL `http://www.w3.org/TR/2000/REC-DOM-Level-2-Core-20001113/`, 2001. – [07.05.2004]

[Le Hors 2004] Hors, Arnaud L. ; Hégaret, Philippe L. ; Wood, Lauren ; Nicol, Gavin ; Robie, Jonathan ; Champion, Mike ; Byrne, Steve: *Document Object Model (DOM) Level 3 Core Specification*. URL `http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/`, 2004. – [07.05.2004]

[Lyman u.a. 2003] Lyman, Peter ; Varian, Hal R.: *Internet*. URL `http://www.sims.berkeley.edu/research/projects/how-much-info-2003/internet.htm`, 2003. – [02.07.2004]

[Mannes 2003] Mannes, Anne T.: *Web Services: A Manger's Guide*. Addison-Wesley Verlag, 2003

[Matthiessen und Unterstein 2003] Matthiessen, Günter ; Unterstein, Michael: *Relationale Datenbanken und SQL. Konzepte der Entwicklung und Anwendung*. Addison-Wesley Verlag, 2003

[Meier 2004] Meier, Andreas: *Relationale Datenbanken. Leitfaden für die Praxis*. Springer Verlag, 2004

[Meyer 2003] Meyer, Jens: *Lexitron - Das Fachlexikon der IT-Begriffe*. URL `http://www.lexitron.de/main.php?detail=true&eintrag=529`, 2003. – [30.07.2004]

[Microsoft 1996] *DCOM Technical Overview, Microsoft Corporation, November 1996*. URL `http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndcom/html/msdn_dcomtec.asp`, 1996. – [14.12.2003]

[Mintert 2002] Mintert, Stefan: *XML & Co*. Addison-Wesley Verlag, 2002

[Mitra 2003] Mitra, Nilo(Hrsg.): *SOAP Version 1.2 Part 0: Primer, W3C Recommendation, 24 June, 2003*. URL `http://www.w3.org/TR/2003/REC-soap12-part0-20030624/`, 2003. – [08.12.2003]

[Moreau u.a. 2003] Moreau, Jean-Jacques ; Schlimmer, Jeffrey(Hrsg.): *Web Services Description Language (WSDL) Version 1.2 Part 3: Bindings, W3C Working Draft, 11 June, 2003*. URL `http://www.w3.org/TR/2003/WD-wsdl12-bindings-20030611/`, 2003. – [25.05.2004]

[net-lexikon.de 2003] akademie.de: *Netlexicon - Anwendung Definition Erklärung Bedeutung*. URL `http://www.net-lexikon.de/Anwendung.html`, 2003. – [03.12.2003]

[net-lexikon.de 2004a] akademie.de: *Netlexicon - ERP Definition Erklärung Bedeutung*. URL `http://www.net-lexikon.de/ERP.html`, 2004. – [27.07.2004]

[net-lexikon.de 2004b] akademie.de: *Netlexicon - Online-Banking Definition Erklärung Bedeutung*. URL `http://www.net-lexikon.de/Online-Banking.html`, 2004. – [29.07.2004]

[net-lexikon.de 2004c] akademie.de: *Netlexicon - Router Definition Erklärung Bedeutung*. URL `http://www.net-lexikon.de/Router.html`, 2004. – [27.07.2004]

[Niedermair 2002] Niedermair, Ele: *XML für Print und Screen*. Franizis Verlag, 2002

[NRW Medien GmbH 2003] GmbH, NRW M.: *E-Commerce (Januar 2003)*. URL `http://www.media.nrw.de/infopool/marktdaten/_abb/E-Commerce_Januar_2003.pdf`, 2003. – [08.12.2003]

[OASIS 2004] *UDDI.org*. URL `http://www.uddi.org`, 2004. – [25.26.2004]

[Ockenfels 2002] Ockenfels, Axel: Reputationsmechanismen auf Internet-Marktplattformen / Max-Planck-Institut zur Erforschung von Wirtschaftssystemen. 2002 (46-2002). – Discussion paper

[Ockenfels und Roth 2002] Ockenfels, Axel ; Roth, Alvin E.: The Timing of Bids in Internet Auctions: Market Design, Bidder Behavior and Artificial Agents / Max Planck Institute for Research into Economic Systems, Strategic Interaction Group. 2002 (33-2002). – Discussion paper

[OMG 2003] *CORBA FAQ*. URL `http://www.omg.org/gettingstarted/corbafaq.htm`, 2003. – [14.12.2003]

[o.V. 2004] *Elektronischer Handel*. URL `http://wwwi.wu-wien.ac.at/public/nmec/E_Maerkte_kl.pdf`, 2004. – [06.10.2004]

[Peterßen und Janzik 2004] Peterssen, Stefan ; Janzik, Florian: *Darstellung und Wirtschaftsanalyse der Internet-Handelsplattform des Online-Auktionshauses eBay*. Schriftenreihe des Fachbereichs Wirtschaft der Hochschule Bremen, 2004

[Peters 2000] Peters, Ralf: Technologien des E-Commerce. In: *Das Wirtschaftsstudium* 29 (2000), (7/2000), Nr. 7, S. 961–970

[Pickruhn 2003] Pickruhn, Helmut: *XML in der betrieblichen Praxis*. S. 187–194. In: *P. Mertens (Hrsg.): XML-Komponenten in der Praxis*, Springer Verlag, 2003

[Picot 2004] Picot, Arnold: *Zehn Eigenschaften der Internet-Ökonomie*. URL `http://www.competence-site.de/ebusiness.nsf/9A49EFF5CEE4D033C125697A005B4FFD/$File/eigenschaften_der_internet_oekonomie.pdf`, 2004. – [12.03.2004]

[Pidde 2004] Pidde, Andreas: *Transmission Control Protocol/Internet Protocol (TCP/IP)*. URL `http://www.uni-koblenz.de/~pidde/lupe/tcpip.html`, 2004. – [10.03.2004]

[Piller 1998] Piller, Frank T.: *Mass Customization – kundenindividuelle Massenproduktion*. Carl Hanser Verlag, 1998. – URL `http://www.mass-customization.de/wasist.htm`. – [04.07.2004]

[Porter 1999] Porter, Michael E.: *Wettbewerbsvorteile*. Campus Verlag, 1999

[Prokhorenko und Prokhorenko 2004] Prokhorenko, Alexander ; Prokhorenko, Olexiy: *Creating a Trusted Applet with Local File System Access Rights*. URL `http://www.developer.com/java/ent/article.php/3303561`, 2004. – [31.05.2004]

[Purschke und Wurdack 2000] Purschke, Detlef ; Wurdack, Alexander: *Internet und Gesellschaft*. S. 257–273. In: *A.-W. Scheer (Hrsg.): E-Business: Wer geht? Wer Bleibt? Wer kommt?*, Physica-Verlag, 2000

[Rayport und Sviokla 2000] Rayport, Jeffrey F. ; Sviokla, John J.: *Die virtuelle Wertschöpfungskette – kein fauler Zauber*. S. 59–76. In: *D. Tapscott (Hrsg.): Erfolg im E-Business*, Carl Hanser Verlag, 2000

[RFC1945 2004] Berners-Lee, T. ; Fielding, R. ; Frystyk, H.: *Hypertext Transfer Protocol*. URL `http://www.ietf.org/rfc/rfc1945.txt`, 2004. – [21.04.2004]

[RFC2616 2004] Fielding, R. ; Gettys, J. ; Mogul, J. ; Frystyk, H. ; Masinter, L. ; Leach, P. ; Berners-Lee, T.: *Hypertext Transfer Protocol*. URL `http://www.ietf.org/rfc/rfc2616.txt`, 2004. – [21.04.2004]

[RFC2660 2004] Rescorla, E. ; Schiffman, A.: *Secure Hypertext Transfer Protocol*. URL `http://www.ietf.org/rfc/rfc2660.txt`, 2004. – [21.04.2004]

[RFC793 2004] *Transmission Control Protocol*. URL `ftp://ftp.nic.de/pub/rfc/rfc793.txt`, 2004. – [21.04.2004]

[Rhody 2004] Rhody, Sean: Why Web Services. In: *Web Services Journal* 4 (2004), January, Nr. 1, S. 3

[Roth und Ockenfels 2002] Roth, Alvon E. ; Ockenfels, Axel: *Last-Minute Bidding and the Rules for Ending Second-Price Auctions: Evidence from eBay and Amazon Auctions on the Internet*. URL `http://www.stanford.edu/class/cs206/roth-ockenfels.pdf`, 2002. – [31.03.2005]

[Rottach und Groß 2002] Rottach, Thilo ; Gross, Sascha: *XML kompakt: die wichtigsten Standards*. Spektrum akademischer Verlag, 2002

[Sauer 2002] Sauer, Hermann: *Relationale Datenbanken. Theorie und Praxis*. Addison-Wesley Verlag, 2002

[Scheer u. a. 2000] Scheer, August-Wilhelm ; Erbach, Fabian ; Thomas, Oliver: *E-Business — Wer geht? Wer Bleibt? Wer kommt?* S. 3–45. In: *A. Scheer (Hrsg.): E-Business — Wer geht? Wer Bleibt? Wer kommt?*, Physica-Verlag, 2000

[Scheer und Loos 2004] Scheer, Christian ; Loos, Peter: *Internetbasierte Geschäftsmodelle. Neue Möglichkeiten der Wertschöpfungsorganisation in der Internet-Ökonomie*. URL `http://chris.scheer.bei.t-online.de/docs/scheer02_wertschoepfung_paper.pdf`, 2004. – [14.02.2004]

[Schildhauer 2003] Schildhauer, Thomas: *Lexikon Electronic Business*. Oldenburg Verlag, 2003

[Schinzer und Thome 1999] Schinzer, Heiko ; Thome, Rainer: Extensible Markup Language. In: *Das Wirtschaftsstudium* 28 (1999), (2/1999), Nr. 2, S. 208–215

[Schmeh 2001] Schmeh, Klaus: *Kryptografie und Public-Key-Infrastrukturen im Internet*. Dpunkt Verlag, 2001

[Schmidt 2002] Schmidt, Carsten: *Supply Chain Management*. S. 509–527. In: *R. Weiber (Hrsg.): Handbuch Electronic Business*, Gabler Verlag, 2002

[Schöning 2003] Schöning, Harald: *XML und Datenbanken. Konzepte und Systeme*. Carl Hanser Verlag, 2003

[Schomakers 2001] Schomakers, Jürgen: *Customer Relationship Management stellt den Kunden in den Mittelpunkt des Handelns*. S. 145–157. In: *J. Frischmuth, W. Karrlein, J. Knop (Hrsg.): Strategien und Prozesse für neue Geschäftsmodelle. Praxisleitfaden für E- und Mobile Business*, Springer Verlag, 2001

[SECUDE GmbH 1999] SECUDE Sicherheitstechnologie Informationssysteme GmbH (Veranst.): *Kurze Einführung in die Sicherheitstechnologie*. 1999

[Seemann 2001] Seemann, Michael: XAlice meets XBob. XML-basierte Kryptografie und PKI. In: *Der Entwickler* 6? (2001), November/Dezember, Nr. 6, S. 87–92

[Seemann 2004] Seemann, Michael: XML kompakt. Oder wie man den Umgang mit XML ressourcenschonender gestalten kann. In: *XML & Web Services Magazin* 3 (2004), Januar, Nr. 1, S. 30–33

[Selfhtml 2004] *XPath-Funktionen*. URL `http://selfhtml.teamone.de/xml/darstellung/xpathfunktionen.htm`, 2004. – [11.05.2004]

[Shapiro und Varian 1999] Shapiro, Carl ; Varian, Hal R.: *Information Rules: A Strategic Guide to the Network Economy*. Harvard Business School Press, 1999

[Skiera 1998] Skiera, Bernd: *Auktionen*. S. 297–310. In: *S. Albers, M. Clement, K. Peters (Hrsg.): Marketing mit Interaktiven Medien. Strategien zum Markterfolg.*, IMK, 1998

[Skiera und Lambrecht 2004] Skiera, Bernd ; Lambrecht, Anja: *Erlösmodelle im Internet*. URL `http://www.ecommerce.wiwi.uni-frankfurt.de/skiera/publications/Erloesmodell.pdf`, 2004. – [04.02.2004]

[Skiera und Spann 2002] Skiera, Bernd ; Spann, Martin: *Flexible Preisgestaltung im Electronic Business*. S. 659–707. In: *R. Weiber (Hrsg.): Handbuch Electronic Business*, Gabler Verlag, 2002

[Stelzer 2000] Stelzer, Dirk: Digitale Güter und ihre Bedeutung in der Internet-Ökonomie. In: *Das Wirtschaftsstudium* 29 (2000), (6/2000), Nr. 6, S. 835–842

[Stähler 2002] Stähler, Patrick: *Geschäftsmodelle in der digitalen Ökonomie: Merkmale, Strategien und Auswirkungen*. Josef Eul Verlag, 2002. – URL `http://www.business-model-innovation.com`. – [03.02.2004]

[Strauß und Schoder 2000] Strauss, Ralf E. ; Schoder, Detlef: *Wie werden die Produkte den Kundenwünschen angepasst? – Massenhafte Individualisierung*. S. 111–121. In: *S. Albers, M. Clement, K. Peters, B. Skiera (Hrsg.): eCommerce. Einstieg, Strategie und Umsetzung im Unternehmen.*, F.A.Z.-Institut, 2000

[Sun 2003] *Java Remote Method Invocation (Java RMI)*. URL `http://java.sun.com/products/jdk/rmi/index.jsp`, 2003. – [14.12.2003]

[Sun 2004] Sun, Microsystems: *JavaServer Pages Technology*. URL `http://java.sun.com/products/jsp/`, 2004. – [26.07.2004]

[Tapscott 1996] Tapscott, Don: *Die digitale Revolution*. Gabler Verlag, 1996

[von Thienen 1995] Thienen, Wolfhard von: *Client/Server. Technologie und Realisierung im Unternehmen*. Vieweg Verlag, 1995

[Thompson u.a. 2001] Thompson, Henry S. ; Beech, David ; Maloney, Murray ; Mendelsohn, Noah(Hrsg.): *XML Schema Part 1: Structures, W3C Recommendation, 2 May 2001*. URL `http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/`, 2001. – [20.05.2004]

[Timmers 1998] Timmers, Paul: Business Models for Electronic Markets. In: *Y. Gadient, B. F. Schmid, D. Selz: EM - electronic Commerce in Europe. EM - Electronic Markets* 8 (1998), (7/1998), Nr. 2, S. 3–8

[Turban 1997] Turban, Efraim: Auctions and Bidding on the Internet: An Assessment. In: *B. F. Schmid, S. Klein: EM - Electronic Auctions. EM - Electronic Markets* 7 (1997), (12/1997), Nr. 4, S. 7–11. – URL `http://www.electronicmarkets.org/modules/pub/view.php/electronicmarkets-236`. – [18.02.2004]

[W3C 2003] W3C: *Extensible Markup Language (XML)*. URL `http://www.w3c.org/XML`, 2003. – [7.12.2003]

[W3C 2004a] W3C: *The Extensible Stylesheet Language Family (XSL)*. URL `http://www.w3c.org/Style/XSL/`, 2004. – [11.05.2004]

[W3C 2004b] W3C: *Hypertext Transfer Protocol Overview*. URL `http://www.w3.org/Protocols/`, 2004. – [22.04.2004]

[Weitz 2002] Weitz, Wolfgang: Basisarchitekturen Web-basierter Informationssysteme. In: *Wirtschaftsinformatik* 44 (2002), (?2/2002), Nr. 3, S. 207–216

[Wikipedia 2004a] Wikipedia.de: *Parser - Wikipedia*. URL `http://de.wikipedia.org/wiki/Parser`, 2004. – [18.11.2004]

[Wikipedia 2004b] Wikipedia.de: *Prozessor - Wikipedia*. URL `http://de.wikipedia.org/wiki/Prozessor`, 2004. – [18.11.2004]

[Wikipedia 2004c] Wikipedia.de: *Relationale Datenbank - Wikipedia*. URL `http://de.wikipedia.org/wiki/Relationale_Datenbank`, 2004. – [03.09.2004]

[Winzerling 2002] Winzerling, Werner: *Systemarchitektur von Online-Anwendungen*. S. 9–40. In: *H. Dohmann, G. Fuchs, K Khakzar (Hrsg.): Die Praxis des E-Business. Technische, betriebswirtschaftliche und rechtliche Aspekte*, Vieweg Verlag, 2002

[Wirtz 2001] Wirtz, Bernd W.: *Electronic Business*. Gabler Verlag, 2001

[Wüstner und Buxmann 2002] Wüstner, Erik ; Buxmann, Peter: SOAP für Web Services. In: *Das Wirtschaftsstudium* 31 (2002), (11/2002), Nr. 11, S. 1434–1441

[Wütherich 2001] Wütherich, Gerd: XML-Pfadfinder. Eine Einführung in das Arbeiten mit XPath. In: *Der Entwickler* 6? (2001), November–Dezember, Nr. 6, S. 93–96

[Ydema 2001] Ydema, Hans: Sicherheitsrelevante XML-Anwendungen und -Erweiterungen. In: *KES* ? (2001), Nr. 3, S. 20–24

[Zerfaß und Haasis 2002] Zerfass, Ansgar ; Haasis, Klaus: *Internet und E-Business im Mittelstand: Anwendungsfelder, Chancen, Handlungsmöglichkeiten*. S. 5–25. In: *Haasis K., Strommer W., Zerfaß A. (Hrsg.): Digitale Wertschöpfung. Internet und E-Business als Chance für den Mittelstand*, Dpunkt Verlag, 2002

[Zwerger und Paulus 2002] ZWERGER, Florian ; PAULUS, Sachar: *E-Business Projekte. Warum sie scheitern und wie man sie zum Erfolg führt.* Galileo Press, 2002

[Zwißler 2002] ZWISSLER, Sonja: *Electronic Commerce Electronic Business.* Springer Verlag, 2002