Xpert.press

Die Reihe **Xpert.press** vermittelt Professionals in den Bereichen Softwareentwicklung, Internettechnologie und IT-Management aktuell und kompetent relevantes Fachwissen über Technologien und Produkte zur Entwicklung und Anwendung moderner Informationstechnologien.

Robert Deutz

Mambo

Installation, Administration, Anwendung und Entwicklung

Mit 146 Abbildungen

Springer

Robert Deutz
Robert Deutz Business Solution
Brüsseler Ring 67
52074 Aachen
www.rdbs.de
mambobuch@rdbs.net

Website zum Buch: www.rdbs.de/mambobuch.html

Bibliografische Information der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über
http://dnb.ddb.de abrufbar.

ISSN 1439-5428
ISBN-10 3-540-22158-1 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-22158-6 Springer Berlin Heidelberg New York

Springer ist ein Unternehmen von Springer Science+Business Media
springer.de

Printed in The Netherlands

Satz: Druckfertige Daten des Autors
Herstellung: LE-TeX Jelonek, Schmidt & Vöckler GbR, Leipzig
Umschlaggestaltung: KünkelLopka Werbeagentur, Heidelberg
Gedruckt auf säurefreiem Papier 33/3142/YL - 5 4 3 2 1 0

Für Silvia

Vorwort

Ein Buch über eine Open Source Software zu schreiben ist eine interessante Aufgabe und vor – allem – spannend. Was heute richtig und gut ist, kann morgen schon wieder vollkommen veraltet sein. In den letzten Monaten hat sich die Release-Politik im Mambo-Projekt allerdings geändert. Die aktuellen Versionen bleiben jetzt inhaltlich und von den Funktionalitäten über eine längere Zeit unverändert. Es findet zwar eine Fehlerbehebung statt, aber es gibt keine strukturellen Änderungen zwischen den Hauptversionen. Dieses Buch dürfte also eine lange Zeit für Mambo-Interessierte hilfreich sein.

Ein Buch ist auch immer die Leistung mehrerer. An erster Stelle ist meine Freundin Silvia Rensonnet zu nennen, die immer die erste Version eines Kapitels zur Korrektur vorgelegt bekam. Die Fehler und merkwürdigen Sätze, die sie verbessert hat, reichen aus um noch zwei weitere Bücher mit Fehlern auszustatten. Ihr danke ich ganz besonders!

Dann gilt mein Dank den Kollegen von „Mambers.com“: Antonio Cambule, Ulrich Eichenseer, Nils Feberwee, Christian Hent und Thomas Kahl für ihre fachlichen Korrekturen und Hinweise. Ich danke außerdem Angie Radtke, für ihre Unterstützung und den Beitrag zur Barrierefreiheit, dem Springer-Verlag und insbesondere Herrn Hermann Engesser, der dieses Projekt von Anfang an unterstützt hat. Last but not least gilt mein Dank Susanne Neugebauer, die dem Buch den letzten Schliff verpasst hat.

Ich hoffe, dass Sie das Buch hilfreich und nützlich finden. Über konstruktive Kritik und natürlich auch Lob freue ich mich sehr, schicken Sie diese Nachrichten bitte an: mambobuch@rdbs.net.

Aachen, Januar 2005
Robert Deutz

Inhaltsverzeichnis

1 Einleitung 1

1.1 Über dieses Buch 2

2 Grundlagen 5

2.1 Die Wurzeln des World Wide Web 5
2.1.1 Wie schreibt man einen Aufsatz? 5
2.1.2 Generic Coding und Markup-Sprachen 6
2.1.3 Hypertext 7
2.1.4 HTML und WWW 7
2.1.5 Dem Medium angepasst ausgeben 8
2.2 Websprachen 9
2.2.1 XML 9
2.2.2 HTML-Aufbau eines Dokuments 10
2.2.3 CSS 12
2.2.4 CSS-Praxis 14
2.3 Cross Browser 19
2.4 Barrierefreiheit – Internet verbindet 19
2.4.1 Übersichtlichkeit 20
2.4.2 Kontraste 20
2.4.3 Grafiken 21
2.4.4 Schriftgrößen 21
2.4.5 Sauberer Code und logische Strukturen 21

3 Installation 23

3.1 Voraussetzungen 23
3.2 Installation MySql-Datenbank 24
3.2.1 Installation unter Unix 24
3.2.2 Installation unter Windows 25
3.3 Installation Apache 26
3.3.1 Installation unter Unix 26
3.3.2 Installation unter Windows 28

3.4 Installation PHP 28
3.4.1 Installation unter Unix 28
3.4.2 Installation unter Windows 30
3.5 Konfiguration 30
3.6 PhpMyAdmin 30
3.7 XAMPP 31
3.8 Mambo-Installation 31
3.8.1 Vorbereitung 31
3.8.2 Webinstaller 31
3.8.3 Manuelle Installation 35
3.9 MSAS – Mambo Stand Alone Server 37

4 Die erste Website 41

4.1 Aufgabenstellung 41
4.2 Anmelden als Administrator 41
4.3 Der Adminbereich: Ein Überblick 42
4.4 Realisierung 43
4.4.1 Einrichten der Informationsbereiche 43
4.4.2 Menüs bearbeiten 50
4.4.3 Einrichten von Web-Links 54
4.4.4 Konfiguration der Frontpage 56

5 Mambo Basics 63

5.1 Mambo erweitern 63
5.1.1 Templates 63
5.1.2 Mambots 65
5.1.3 Modules 66
5.1.4 Components 66
5.1.5 Sonstiges 67
5.2 Gruppen und Berechtigungen 67
5.3 Section und Categories 70
5.4 Static Content vs. All Content 71
5.5 Inhalte bearbeiten 71
5.5.1 Unterschiede im Frontend 78
5.6 Gemeinsames Arbeiten 79

6 Templates erstellen 81

6.1 Templatedateien und Struktur 81
6.2 Entwurf ohne Tabellen 82
6.3 Entwurf mit Tabellen 100
6.4 Mambo-HTML-Codes 105
6.4.1 Anpassung durch den Styleparameter 105
6.4.2 Mambo-Standards ermitteln 107
6.4.3 Mambo Standard-CSS 109
6.5 Template packen 112

7 Perfekte Website 115

7.1 Internet-Shop 115
7.1.1 Installation 116
7.1.2 Bestellen 117
7.1.3 Bearbeiten der Bestellung 120
7.2 DOCMan 121
7.2.1 Installation und Administration 121
7.2.2 Frontend 124
7.3 Kommentierung von News und Artikeln 125
7.4 Newsletter 127
7.5 Search Engine Friendly URL 130

8 Entwickeln für Mambo 133

8.1 Mambo-Parametersystem 133
8.1.1 Parametertyp „text“ 135
8.1.2 Parametertyp „textarea“ 136
8.1.3 Parametertyp „radio“ 136
8.1.4 Parametertyp „list“ 136
8.1.5 Parametertyp „imagelist“ 137
8.1.6 Parametertyp „spacer“ 137
8.1.7 Parametertypen „mos_menu“, „mos_section“, „mos_category“ 137
8.1.8 Ermittlung der Parameter 137
8.2 Mambots 138
8.3 Modules 145
8.4 Components 148
8.4.1 Installation 148
8.4.2 Backend-Konfigurationsoberfläche 152
8.4.3 Component-Frontend 158

9 Referenz Administration 163

9.1 Überblick..... 163
9.2 Toolbar..... 164
9.3 Infobar..... 164
9.4 Workspace 164
9.5 Menubar..... 164
9.5.1 Home.....164
9.5.2 Site 165
9.5.3 Menu 185
9.5.4 Content.....189
9.5.5 Components 199
9.5.6 Modules 220
9.5.7 Mambots 235
9.5.8 Messages.....239
9.5.9 System.....240
9.5.10 Help.....241
9.6 Menütypen 241
9.6.1 Blog Content Category.....242
9.6.2 Blog Content Category Archive.....244
9.6.3 Blog Content Section.....247
9.6.4 Blog Content Section Archive 249
9.6.5 Component.....252
9.6.6 Link Component Item 259
9.6.7 Link Contact Item.....260
9.6.8 Link Content Item.....260
9.6.9 Link News Feeds 260
9.6.10 Link Static Content.....261
9.6.11 Link URL.....261
9.6.12 List Content Section 261
9.6.13 Separator / Placeholder.....263
9.6.14 Table Contact Category.....263
9.6.15 Table Content Category.....264
9.6.16 Table News Feed Category 266
9.6.17 Table Weblink Category 267
9.6.18 Wrapper 267

10 Wo geht die Reise hin?.....269

11 Anhang **271**

11.1 Websites/Linkverzeichnis 271
11.1.1 Internationale Sites 271
11.1.2 Deutschsprachige Sites 273
11.1.3 Sonstige Sites 273
11.2 Mambo Templates 274
11.2.1 Ohne Tabellen 274
11.2.2 Mit Tabellen 282
11.3 Entwicklungen 300
11.3.1 Mambot 300
11.3.2 Module 304
11.3.3 Component 306
11.4 Begriffe und Formales 321
11.5 Abbildungen 322

Index 327

1 Einleitung

Wenn Sie dieses Buch gekauft haben, werden Sie sich die Frage „Warum ein Content Management System (CMS) einsetzen?“ vielleicht gar nicht mehr stellen. Für die einen kann diese Einleitung eine Bestätigung ihrer Entscheidung sein. Für die anderen, unentschlossenen, bleibt es spannend.

Den meisten CMS sagt man nach, dass sie sehr kompliziert sind, man mehrere Schulungen benötigt um einen einfachen Text einzugeben und Designer in ihrer Freiheit sehr eingeschränkt werden. Wer z.B. schon mal versucht hat eine Site mit typo3 aufzusetzen, wird einige der Aussagen bestätigt sehen. Zur Entschuldigung von typo3 – wir wollen ja nicht sofort mit Schelte anfangen – muss man erwähnen, was der Ansatz bzw. das Konzept bei typo3 ist: typo3 ist eine Boeing, um diese vom Boden zu bekommen, muss man eben zunächst einiges lernen. Im Gegensatz dazu ist Mambo ein Fahrrad, man kommt schnell vom Fleck, aber Mambo kann man zum Flugzeug ausbauen, somit lernt Mambo mit Ihnen fliegen!

Der Spruch „Mambo – Power in simplicity“ kommt nicht von ungefähr. Das Core-Team (die Programmierer der Basisanwendung) legt großen Wert darauf, dass Mambo auch in Zukunft eines der am einfachsten zu bedienenden CMS ist.

Nach dem Ausflug in die Welt der Fortbewegungsmittel kommen wir wieder zur Ausgangsfrage zurück. Warum sollte man ein CMS einsetzen? Zunächst mal aus dem einfachen Grund: Es ist hip. Möchten Sie in einer Gruppe stehen und als einziger noch eine Webpräsenz basierend auf statischen HTML-Seiten betreiben? Aber im Ernst, das ist momentan Stand der Technik, und weil sehr viele auf ein CMS umsteigen, kann es nicht so kompliziert sein, wie ihm nachgesagt wird. Neben den Gründen der sozialen Integration gibt es auch, ganz nüchtern betrachtet, gute Gründe umzusteigen. Ein CMS bietet Ihnen die Trennung von Inhalt und Darstellung. Die von Ihnen eingegebenen Texte können Sie auch an anderer Stelle verwenden und das Design kann geändert werden ohne alle Texte zu überarbeiten. Ein CMS enthält zum Teil sehr umfangreiche Tools

und Funktionen, z.B. Kontaktformulare, Weblink-Verzeichnisse, Möglichkeiten, die Inhalte zu durchsuchen, um nur einige Beispiele zu nennen. Diese Funktionen müssten ansonsten extra eingebunden und ggf. programmiert werden.

Sie erhalten mit einem CMS also standardmäßig schon viel, was für eine Website sinnvoll und notwendig ist. Mit Mambo als Auswahl stehen neben den von Hause aus eingebundenen Funktionen noch Hunderte von Erweiterungen zur Verfügung um Ihre Website genau so ins Netz zu stellen, wie Sie es wollen. Allein auf mamboforge.net finden Sie aktuell über 600 Projekte, die Erweiterungen für Mambo beinhalten. Daneben gibt es noch viele kommerzielle Erweiterungen. Sie haben die Qual der Wahl.

1.1 Über dieses Buch

Im Folgenden finden Sie eine kurze Übersicht darüber, was die einzelnen Kapitel zu bieten haben..

Einleitung:

> Darin lesen Sie gerade.

Grundlagen:

> Sie erfahren etwas über die Geschichte des WWW, HTML und CSS. Abschließend finden Sie einige Bemerkungen zur Gestaltung von barrierefreien Websites.

Installation:

> Wie installieren Sie Mambo und was ist der „Mambo Stand Alone Server“.

Die erste Website:

> Auf 21 Seiten lernen Sie die Grundzüge von Mambo kennen, nebenbei erstellen wir Ihre erste Website mit Mambo.

Mambo Basics:

> Welche Ideen stecken hinter dem, was Sie bisher von Mambo kennen. Wir erläutern die Hintergründe.

Template-Erstellung:

Die eigene Website braucht ein individuelles Design; gemeinsam verfolgen wir zwei Strategien zur Erstellung eines Templates.

Perfekte Website:

Hehre Ziele verfolgen wir in diesem Kapitel; wir erweitern die Website um einige Bestandteile und erläutern die Konfiguration.

Entwickeln für Mambo:

Wir erläutern das Parametersystem von Mambo und entwickeln einen Mambot, ein Modul und eine Komponente.

Administrationsbereich:

Eine vollständige Referenz der Mambo-Administration incl. Erläuterung der Standardkomponenten, Module und Mambots.

Wohin die Reise geht:

Ein Ausblick in die nähere Mambozukunft.

Anhang:

Alle Listings, Templates, Literaturhinweise, Index.

2 Grundlagen

Leider geht nichts ohne einige Grundlagen. Der folgende Abschnitt ist für die Leserinnen und Leser gedacht, die mit den Begriffen HTML, XML und CSS nicht viel anfangen können. Es werden Konzepte, Begriffe und Vorgehensweisen erläutert. Eine umfangreiche Aufarbeitung der Themen wäre an dieser Stelle allerdings unangebracht, mehr als ein Anreißen und Einordnen der Themen würde dem Ziel des Buches nicht gerecht werden. Wir wollen schließlich Mambo kennen lernen.

2.1 Die Wurzeln des World Wide Web

2.1.1 Wie schreibt man einen Aufsatz?

Einige werden sich vielleicht noch erinnern. Aufsätze bestehen im Wesentlichen aus drei Teilen. Im ersten Teil führt man den Leser an das Thema heran, im zweiten Teil behandelt man das Thema und im dritten Teil fasst man zusammen und bewertet abschließend. Ein Aufsatz, wie jedes andere Dokument auch, folgt einer gewissen Struktur, diese ist vom Inhalt unabhängig. Ein Dokument besteht also aus zwei Teilen, der Struktur und dem Inhalt. Diese grundlegende Erkenntnis führte in den späten 60er Jahren zur Entwicklung der GML (Generalized Markup Language). Die GML bildet Dokumente in einer verschachtelten Struktur ab und führt das Konzept des Dokumententyps ein. Angewandt auf unser Beispiel wäre der Dokumententyp „Aufsatz“ und als Strukturelemente würden wir Einleitung, Hauptteil und Ende festlegen. Diese Einteilung ist natürlich sehr grob, mit GML kann eine Struktur wesentlich feiner beschrieben werden.

2.1.2 Generic Coding und Markup-Sprachen

Im letzten Abschnitt haben wir schon die Markup-Sprache GML erwähnt. Die Weiterentwicklung ist durch das ISO (International Organization for Standardization) als ISO-Standard 8879 „Standard Generalized Markup Language“ SGML im Jahre 1985 veröffentlicht worden. Das Konzept sieht vor, durch Auszeichnungen im Text, die Struktur des Dokuments abzubilden. Ist ein Dokument in seiner Struktur beschrieben, erreicht man die schon beschriebene Trennung des Inhalts von der Struktur. Damit einher geht ebenfalls eine Trennung von Inhalt und Darstellung. Auf den ersten Blick erscheint dies kein Vorteil zu sein. Riskieren wir einen zweiten Blick.

Dieser Abschnitt hat eine Überschrift, diese ist linksbündig ausgerichtet, die Schrift ist serifenlos, fett gesetzt und etwas größer als der normale Text. Für die Darstellung in diesem Buch ist dies eine gute Wahl, für jeden ist visuell zu erkennen, dass es sich hier um eine Überschrift handelt. Was wäre aber nun, wenn wir den Inhalt des Buches als Sprache ausgeben möchten. In elektronischer Form liegt der Text natürlich vor, das Format ist aber darauf ausgerichtet den Text auszudrucken, es finden sich also Anweisungen, die Schriftgröße, Schriftart und Ähnliches festlegen. Ob der Text aber eine Überschrift ist oder ob es sich nur um größeren, fett gesetzten Text handelt, ist nicht ermittelbar. Die optischen Attribute auszuwerten wird sicherlich nicht sonderlich gut funktionieren. Für die Ausgabe über ein anderes Medium wäre eine Kennzeichnung der Überschrift als Überschrift besser.

Das „generic coding“ (generic, englisch: artmäßig) setzt an dieser Stelle an, ein bestimmter markierter Text wird nach seiner Art gekennzeichnet. Überschriften werden als solche gekennzeichnet; wie dieser Text in einem späteren Schritt wieder ausgegeben wird, ist bei diesem Verfahren zunächst nicht wichtig. Für die verschiedenen Medien gibt es getrennte Beschreibungen, wie mit den markierten Stellen zu verfahren ist. Dies eröffnet eine sehr große Flexibilität, der gleiche Text kann ausgedruckt, am Bildschirm angezeigt oder auch als Sprache ausgegeben werden. Man muss nur dafür sorgen, dass die markierten Textteile dem Medium angemessen ausgegeben werden.

2.1.3 Hypertext

Die Entwicklung des Hypertextes ist ebenfalls eine alte Geschichte; das Konzept, Inhalte miteinander zu verbinden, das heute im World Wide Web genutzt wird, hat seinen Ursprung in Schriften aus dem 17. Jahrhundert. Schon dort wurden durch Querverweise Textstellen miteinander verknüpft. In unserem Jahrhundert muss man Vannevar Bush mit seinem fiktiven System Memex und Douglas Engelbart mit seinen Veröffentlichungen zum Augument-Projekt erwähnen. Der Urvater des Begriffs Hypertext ist jedoch Ted Nelson, 1965 veröffentlichte er seine Ideen und Visionen im Projekt Xanadu. Die Ansätze von Nelson gehen über die heutige Realität des Hypertextes jedoch hinaus. Er selbst formuliert es sogar so: „Einige meinen, Xanadu wäre das Bestreben gewesen etwas wie das World Wide Web aufzubauen. Im Gegenteil: das World Wide Web war genau das was wir verhindern wollten". Das Recht Entwicklungen kritisch zu sehen können Visionäre in Anspruch nehmen. Fakt ist, das World Wide Web, das wir heute haben, ist das beste (und einzige) was wir haben.

2.1.4 HTML und WWW

Die Verbindung der beiden Aspekte „Generic Markup" und Hypertext ist die *Hyper Text Markup Language.* Diese ist für die Beschreibung von Inhalten und deren Verknüpfung entwickelt worden. Die Grundlagen für HTML und auch für das World Wide Web (WWW) wurden im europäischen Labor für Teilchenphysik – dem CERN – in Genf von Tim Berners-Lee gelegt. Weihnachten 1990 stellte Berners-Lee die erste Website ins Netz, das WWW war geboren. Das Konzept für das WWW steht seitdem auf drei Säulen:

- Das HTTP (Hypertext Transfer Protokoll) regelt die Kommunikationen zwischen Webserver und Browser.
- Die Adressierung der Ressourcen im Netz erfolgt über die URI (Uniform Resource Identifier)
- HTML ist die Auszeichnungssprache für die Seiten.

Das WWW und HTML haben sich in der letzten 14 Jahren rasant entwickelt, die Weiterentwicklung der Sprache wird von dem W3C (W3-Konsortium) überwacht und gesteuert. HTML war und ist eine einfach zu erlernende Sprache zur Strukturierung von Inhalten. Der Inhalt wird durch Auszeichnungen in spitzen Klammern ergänzt,

z.B. kennzeichnet „<h1>“ eine Überschrift 1. Ordnung. Einen Bereich kann man aber nur dann kennzeichnen, wenn man Anfang und Ende festlegt. Daher gibt es einen Start-Tag, hinter dem der Bereich anfängt, und einen End-Tag, vor dem der Bereich endet. Zu unterscheiden sind die Tags durch einen „/“. Zur Verdeutlichung ein Beispiel:

```
<h1>Das ist eine Überschrift</h1>
```

Bei einigen Tags ist es erlaubt einen kombinierten Start-/End-Tag zu verwenden. Beispiele dafür sind der Image-Tag und der Break-Tag (Zeilenumbruch).

```
<img src="./bild.jpg" />
<br />
```

Der HTML-Code legt nicht fest, wie eine Überschrift letztlich ausgegeben wird.

2.1.5 Dem Medium angepasst ausgeben

In der frühen Geschichte des WWW sind durch die fehlende Standardisierung viele Stilblüten entstanden; der blinkende Text ist einer dieser Vertreter. Firmen wie Netscape und Microsoft haben versucht ihre Marktposition auszunutzen und HTML mit eigenen Erweiterungen anzureichern. Durch die Gründung des W3-Konsortium und den Beitritt von namhaften Firmen konnte ein weithin akzeptierter Standard geschaffen werden. Heute unterstützen die meisten Browser den HTML-Standard und die Formatierungssprache Cascading Style Sheets (CSS) relativ gut.

Aktuell, verabschiedet und durch Browser unterstützt sind die Cascading Style Sheets, Level 2 vom 12.05.1998. Dieser Standard definiert eine Formatierungssprache, mit der Autoren die Darstellung der Inhalte mediengerecht aufbereiten können. Die möglichen Ausgabegeräte können z.B. visuelle Browser, Drucker, akustische Geräte, Braille-Geräte und Handhelds sein.

Mit CSS kann man für HTML-Sprachelemente die Ausgabe für bestimmte Medien definieren. So lässt sich für die Ausgabe einer Überschrift erster Ordnung „<h1>“ festlegen, dass diese auf dem Bildschirm in Blau und in einem Ausdruck fett und größer ausgegeben wird.

2.2 Websprachen

Im letzten Abschnitt haben wir uns dem WWW von der historischen Seite genähert. In diesem Abschnitt werfen wir nun einen eher technischen Blick auf die Säulen und die Zusammenhänge des WWW.

2.2.1 XML

Die Abkürzung XML steht für „eXtensible Markup Language“. Die folgende Abbildung leistet einen Beitrag zur vollkommenen Verwirrung des Lesers, diese ist aber hoffentlich nur von kurzer Dauer. Nach dem Bild folgt die Erklärung.

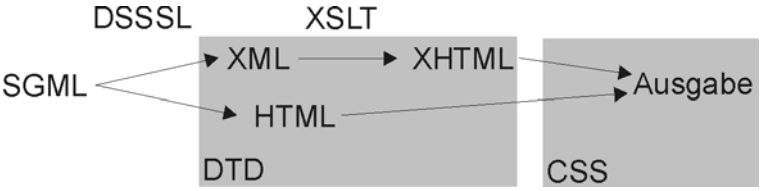

Abb. 2.1 Übersicht Markup-Sprachen

Die Basis für XML und HTML ist jeweils SGML. Der Unterschied besteht darin, dass XML eine Teilmenge von SGML ist und somit für die Formulierung von weiteren Sprachen geeignet ist. HTML hingegen ist eine nicht erweiterbare Sprache, die aus einer festgelegten Anzahl von Elementen (h1 und p zum Beispiel) besteht. Zur Beschreibung der Sprache wird jeweils die DSSSL „Document Style Semantics and Specification Language“ verwendet. Aus XML wiederum entsteht durch Transformierung mittels XSLT „Extensible Stylesheet Language Transformations“ XHTML. Für XML, XHTML und HTML gibt es eine DTD „Document Type Definitions“, diese kennzeichnet das Document. Die DTD findet man am Anfang des Dokuments, für HTML gibt es z.B. die Varianten „Strict“, „Transitional“ und „Frameset“. Aus HTML und XHTML kann mit Hilfe von Cascading Style Sheets (CSS) eine Ausgabe generiert werden.

Mit XML ist aber noch mehr möglich, durch geeignete Transformation bzw. Formatierung kann man aus XML auch PDF und viele andere Formate generieren. Darüber hinaus kann man XML zum Datenaustausch verwenden, die RSS-Feeds basieren auf XML. In den folgenden Abschnitten beschäftigen wir uns genauer mit HTML

und CSS, die Erläuterung zu XML diente nur der Klärung des Gesamtzusammenhangs.

2.2.2 HTML-Aufbau eines Dokuments

Ein einfaches HTML-Dokument besteht aus folgenden Bestandteilen:

- Dokumenttyp-Angabe
- Header
- Body

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Text des Titels</title>
</head>
<body>

</body>
</html>
```

2.2.2.1 Dokumententyp

Der Aufbau eines HTML-Dokuments beginnt immer mit der Dokumententypangabe, mit der die gewählte (X)HTML-Version angegeben wird. Die Version legt fest, welche Elemente und Attribute in einem Dokument vorkommen dürfen. Nur wenn man sich an diese Vorgabe hält, wird das Dokument valide (ist also im Sinne des Dokumententyps gültig). In diesem Beispiel ist XHTML Transitional gewählt. Die unterschiedlichen Dokumententypen scheinen im ersten Augenblick verwirrend, aber eigentlich ist das Festlegen von Standards ein großer Fortschritt, denn so ist es möglich, dass unterschiedliche Ausgabegeräte die Informationen standardisiert verarbeiten können.

2.2.2.2 Header

Im Kopf eines HTML-Dokuments findet man Angaben zum Titel der Seite; dieser wird von den Browsern dann in der Titelleiste des Browserfensters ausgegeben, er wird beim Speichern der Bookmarks verwendet, ist aber auch ein wichtiger Parameter für die gängigen Suchmaschinen, denn viele Suchmaschinen lesen zuerst die URL, anschließend den Titel und dann erst den eigentlichen Inhalt. Häufig wird er innerhalb der Suchmaschine als anklickbarer Link dargestellt. Deshalb ist es besonders wichtig einen aussagekräftigen Titel für jede Seite zu wählen.

Des Weiteren werden im Header die Metatags vereinbart. In den Metatags kann man hilfreiche Angaben für den Webserver, die Browser und die Suchmaschinen unterbringen. Eine meta-Anweisung beginnt immer mit meta und hat mindestens zwei Attribute. Name="mein Name" und content="mein Inhalt". Alternativ zu Name kann auch http-equiv angegeben werden, was direkt vom Webserver ausgelesen wird. Es gibt eine Vielzahl dieser Anweisungen, so dass hier nur die wichtigsten beschrieben werden.

```
Angaben zum Autor
<meta name="author" content="Dein Name">

Angaben zu Zeichensatz
<meta http-equiv="content-type"
content="text/html; charset=ISO-8859-1">

Angaben zur verwendeten Sprache
<meta http-equiv="content-language"
content="de">

Angabe für die Suchmaschinen
<meta name="description" content="Beschreibung
der Seite, die von den Suchmaschinen gefunden
werden soll.">
<meta name="keywords" content="Liste der
Schlüssselbegriffe für die Suchmaschinen, mit
Komma getrennt">

Wann soll eine Suchmaschine wieder vorbei kommen
<meta name="robots" content="follow">
<meta name="revisit-after" content="20 days">

Weiterleitung
<meta http-equiv="refresh" content="5;
URL=http://wohinauchimmer/">
```

Darüber hinaus werden im Header die Stylesheets eingebunden. Es gibt mehrere Möglichkeiten Styles einzubinden: Die Styles können direkt im Seitenkopf, innerhalb der Tags im body oder über eine zentrale Datei eingebunden werden.

2.2.2.3 body

Im body eines HTML-Dokumentes findet man die eigentlichen Inhalte. HTML verwendet sogenannte Tags. Der Inhalt steht immer zwischen einem einleitenden und einem schließenden Tag.

```
<h1>Überschrift der 1.Ordnung </h1>
<p> Ein Absatz <br />
Mit einem Zeilenumbruch</p>
<h2> Überschrift der 2 Ordnung </h2>
<p> Ein anderer Absatz</p>
```

Bei gewöhnlichen HTML spielt es keine Rolle, ob die Tags groß oder klein geschrieben werden, wird jedoch XHTML eingesetzt, müssen alle Tags klein geschrieben werden.

2.2.3 CSS

Mit CSS kann man die Darstellung von HTML-Elementen für ein Ausgabemedium festlegen. Zum Auswählen einer bestimmten Gruppe von HTML-Elementen, für die ein Format gelten soll, gibt es in CSS Selektoren. Wie schon beim Aufbau eines HTML-Dokuments erwähnt, können CSS-Formate entweder in einem zentralen Style-Bereich im Header definiert werden oder in einer externen CSS-Datei. Eine weitere Möglichkeit ist die Definition des Styles direkt im einleitenden Tag eines HTML-Elements. CSS-Formate bestehen aus einer oder mehreren Eigenschaften und Werten. So können Sie z.B. das Format für einen Absatz definieren, indem Sie die Eigenschaften wie Schriftgröße, Schriftfarbe und Abstand festlegen.

Hier ein kleines Beispiel:

```
<html>
<head>
<title>Titel der Datei</title>
<style type="text/css">
<!--
p { color:#cc0000;
   font-family:arial;
   font-size:12px;
   margin :10px }
/* Alle Absätze  haben eine rote Schrift*/
-->
</style>
</head>
<body>
<p> Der rote Absatz</p>
</body>
</html>
```

Ab der CSS-Version 2.0 haben Sie die Möglichkeit Elemente auf einer Website absolut oder relativ zu positionieren, man kann die Breite von Elementen festlegen, Abstände bestimmen, Elemente ein- und ausblenden und vieles mehr.

2.2.3.1 Vorteile von CSS

Dieser Standard definiert eine Formatierungssprache mit der Designer die Darstellung der Inhalte individuell formatieren können, ohne in den HTML-Code eingreifen zu müssen. Zentrales Formatieren und damit schnelle Änderung ist möglich, gleichzeitig verkürzt sich die Ladezeit der Seiten spürbar.

Bei HTML handelt es sich um eine Seitenbeschreibungssprache, die gerade dem Webdesigner nur wenig gestalterischen Freiraum bietet. Jahrelang musste man sich mit Layouttabellen behelfen, um Elemente dort im Browserfenster platzieren zu können, wo man sie gerne gehabt hätte. Da Tabellen aber eigentlich für die Ausgabe tabellarischer Daten bestimmt sind, ist diese Zweckentfremdung semantisch nicht korrekt. Nicht grafische Ausgabegeräte haben ihre Probleme damit, denn sie interpretieren ein Dokument linear d.h. sie lesen es von oben nach unten. Was aber, wenn sie auf verschachtelte Tabellen treffen? Sie verlieren die Orientierung und finden sich in diesem Dokument nur schwer zurecht. Mit CSS ist es nun möglich semantisch korrekte und grafisch ansprechende HTML-Seiten zu erstellen.

2.2.4 CSS-Praxis

Die folgenden Abschnitte sollen Ihnen einen vorsichtigen Einstieg in die CSS-Techniken bieten. Die Darstellung ist natürlich lückenhaft, aber vielleicht motiviert Sie dies, sich an anderer Stelle weiter zu informieren. Hinweis auf Informationsquellen finden Sie im Anhang.

2.2.4.1 Syntax

Ein Style besteht im Wesentlichen aus drei Bestandteilen einem Selector, einer Eigenschaft und einem Wert:

```
SELECTOR { EIGENSCHAFT:WERT }
```

Bei einem Style können auch mehrere Eigenschaften und Werte in einer Anweisung zugeordnet werden, dazu werden die Eigenschaften-Werte-Paare durch Semikolons getrennt:

```
SELECTOR {
   EIGENSCHAFT1:WERT1;
   EIGENSCHAFT2:WERT2;
}
```

Und so sieht das Ganze praktisch – am Beispiel des <p>-Befehls – aus:

```
p {
   font-family:arial;
   color:red;
   font-weight:bold;
}
```

Damit definieren Sie für die p-Tag auf Ihrer Seite standardmäßig die Schriftart Arial, die Farbe Rot und dass die Schrift fett dargestellt werden soll. Sollen Ihre Anweisungen für mehrere Selektoren gelten, können Sie diese durch Kommata getrennt in dieser Form aufzählen:

```
h1, h2, h3, h4 { color:green }
```

Um möglichst flexibel zu sein, kann man allen Elementen im body eine bestimmte *class* zuweisen, die auf unterschiedliche Elemente angewendet werden kann.

Als Style wird Folgendes definiert:

```
.rot {color:#cc0000}
```

In der HTML-Datei nutzt man dieses Style dann so:

```
<p class="rot">mein roter Ansatz</p>
<h1 class="rot">meine rote Überschrift</h1>
```

Eine weitere Möglichkeit ist die Definition einer *id*. Diese kann im Gegensatz zur class nur ein einziges Mal in einem Dokument verwendet werden. Man benutzt Id´s häufig zum Positionieren von Elementen oder Bereichen.

```
#meinDiv {
   position:absolute;
   left:10px;
   top:10px;
   width:100px;
   height:100px;
   border:solid 10px
}
```

Im HTML-Dokument darf dann an einer Stelle stehen:

```
<div id="meinDiv"> Irgend etwas </div>
```

Bisher haben wir die einfachen Selektoren beschrieben, es gibt jedoch auch die Möglichkeit die Styles an die Dokumentenstruktur anzupassen. Diese Selektoren nennt man „kontextabhängige Selektoren".

```
#meinDiv h1 { color: #cc0000 }
```

In diesem Beispiel werden jetzt alle h1, die sich im Bereich #meinDiv befinden, in roter Schriftfarbe dargestellt, während alle Überschriften der ersten Ordnung, die außerhalb liegen, davon nicht beeinflusst werden. Zwischen „#meinDiv" und „h1" ist ein Leerzeichen!

In CSS-Dateien gibt es die Möglichkeit Kommentare einzubinden. Da CSS-Dokumente schnell sehr lang werden können, ist dies eine hilfreiche Methode, um sich hinterher wieder darin zurechtzufinden.

```
/* Hier steht ein Kommentar */
```

2.2.4.2 Vererbung

HTML-Dokumente bilden, wenn sie „ordentlich“ aufgebaut sind, einen Baum. In diesem Baum erben die untergeordneten Elemente die Eigenschaften der darüber liegenden Elemente. Diesen Vorgang nennt man Vererbung. Haben Sie z.B. für den body die Schrift auf 12px gesetzt, so erhalten alle innerhalb des body-Tags liegenden Inhalte ebenfalls die Schriftgröße 12px.

Den Vorgang der Vererbung können Sie umgehen, indem Sie bei einem Element die Schriftgröße explizit angeben. Zum Problem kann die Vererbung werden, wenn Sie relative Angaben verwenden. Haben Sie vereinbart, dass die class „small“ eine Schriftgröße von 75 % hat, ist diese Schrift, wenn Sie ein <p class="small"> definieren, gerade noch lesbar. Kommt allerdings innerhalb dieses p-tags ein Bereich vor, der ebenfalls die class „small“ hat, ist die Schrift nur noch schwer bis gar nicht mehr zu entziffern. Bei dem ersten small waren wir schon bei 9 Punkten, das zweite reduziert die Schrift noch mal um 25 %, dann liegen wir bei ca. 7 Punkten, diese Schrift ist nur sehr schlecht auf einem Bildschirm lesbar.

Obwohl relative Angaben zum Problem werden können, sind diese immer eine gute Wahl. Sie legen damit die Verhältnisse der Elemente zueinander fest. Diese sollten auch Gültigkeit besitzen, wenn die Basisschriftgröße verändert wird.

Achtung, es werden nicht alle Eigenschaften vererbt. Die Eigenschaft „margin“, also der Außenabstand, wird z.B. nicht vererbt.

2.2.4.3 Boxmodel

Im CSS werden alle Elemente als rechteckige Boxen behandelt, zu unterscheiden sind Block- und Inline-Elemente. Ein Block-Element beginnt immer in einer neuen Zeile, hat man die Eigenschaften nicht anders definiert, wird teilweise (je nach Browser) ein Abstand eingefügt. Ein Block-Element kann Inline-Elemente oder andere Block-Elemente beinhalten, nicht aber umgekehrt.
Jede Box hat einen Inhaltsbereich, einen Abstand nach innen (padding), einen Abstand nach außen (margin) und einen Rahmen (border). Wenn Sie die Breite eines Blocks angeben, ist immer der innere Bereich gemeint. Die nachfolgende Grafik zeigt den Zusammenhang:

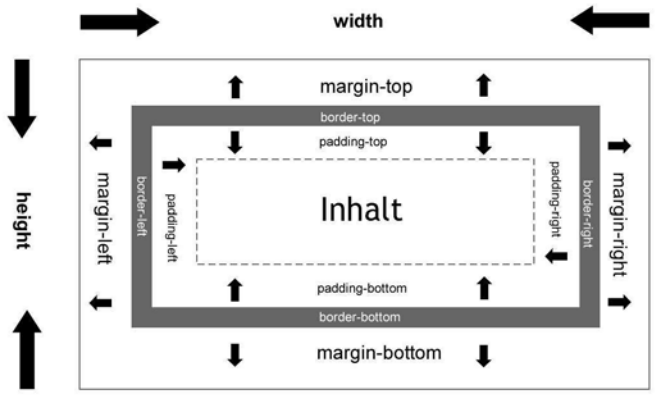

Abb. 2.2
Boxmodell

An dieser Stelle ein Hinweis auf einen Fehler im Internet Explorer: Der IE interpretiert die Angabe „width“ so, als ob in dieser Angabe das padding, margin und die border mit enthalten wären. Das führt dann dazu, dass Bereiche im IE entweder zu klein sind oder dass Bereiche in anderen Browsern zu breit sind.

Inline-Elemente verbleiben im Textfluss und bilden keinen Absatz, ein Beispiel ist der em-Tag, der eine Textpassage betont ausgibt.

2.2.4.4 Positionierung

Mit der Eigenschaft *Position* können Sie Elemente auf der Webseite positionieren. Es werden folgende Positionierungsarten unterschieden:

static: Die entsprechende Box wird in den normalen Dokumentenfluss integriert

relative: Die Box wird relativ zu ihrer statischen Position verschoben. Ein Element div zum Beispiel, das mit

```
div {
   position: relative;
   top: 10px
   left: 10px;
}
```

positioniert wurde, wird 10px weiter unten und 10px weiter rechts positioniert, als wenn es nicht relativ positioniert werden würde. Nachfolgende Elemente erkennen dies jedoch nicht, so dass es zu Überschneidungen kommen kann.

absolute: Die Box wird relativ zu ihrem Elternelement positioniert. Dabei gilt es, Folgendes zu beachten: Der umschließende Block für eine absolut positionierte Box wird durch das nächste positionierte Elternelement oder, falls es kein solches gibt, durch den body gebildet. Ein durch die Regel

```
img {
   position: absolute;
   top: 10px;
   right: 10px;
}
```

positioniertes Bild wird also mit 10px Abstand nach oben und rechts von der oberen rechten Ecke des Elternelementes aus positioniert. Der Block wird dabei aus dem gewöhnlichen Elementfluss herausgenommen, nachfolgende Elemente er-kennen ihn nicht.

fixed: Die Box wird absolut positioniert und bleibt beim Scrollen der Seite stehen.

2.2.4.5 Gefloatete Elemente

Eigentlich kann jedes Element floaten, was so viel wie schwimmen oder fließen bedeutet. Sobald man einem Element die Eigenschaft float zugeordnet hat, wird es automatisch zu einem Block-Element. Man sollte dem Element immer eine bestimmte Breite zuordnen, weil das Element sonst immer so groß wie das Elternelement oder wie der Inhalt wird. Gefloatete Elemente sind aus dem normalen Textfluss herausgenommen, haben aber den Vorteil, dass sie direkt nach dem vorherigen Block-Element angeordnet werden.

Die Elemente können entweder die Eigenschaft float:left; float:right oder float:none bekommen. Wenn ein Elternelement ein gefloatetes Kind enthält, „weiß“ es nichts über seine Ausmaße und ist nur so groß wie der Inhalt ohne das gefloatete Element, weil das gefloatete Element ja eigentlich aus dem normalen Dokumentenfluss herausgenommen ist, auch wenn es im Quelltext nicht so scheint. Um dies zu verhindern, benötigt das Elternelement die Eigenschaft clear:both, clear:left oder clear:right.

2.3 Cross Browser

Das größte Problem ist die Kodierung der Website mit HTML und CSS, so dass die Site auf unterschiedlichen Browsern gleich aussieht. Man sollte zwar meinen, dass durch die Verwendung der Standards Webbrowser den HTML-Code immer gleich interpretieren; dass dies nicht so ist, können Sie sich schon denken. Ein Grund ist, dass ein Browser ein Stück Software ist und Software ist nie fehlerfrei. So kommt es neben der Tatsache, dass Browser nicht alle Standards vollständig unterstützen, auch noch zu Fehlinterpretationen. Je nach Dokumententyp verhalten sich die Browser unterschiedlich. Es kann also quasi als die hohe Kunst der Website-Erstellung angesehen werden eine Site browserübergreifend „ordentlich" hinzubekommen. Die sicher nicht vollständige Liste für die Erstellung von Websites (nicht nur im Hinblick auf Cross Browser) kann Ihnen dabei helfen:

- Bauen Sie Ihre HTML-Seiten logisch strukturiert auf.
- Geben Sie immer den Dokumententyp an.
- Verwenden Sie CSS.
- Vermeiden Sie, wenn möglich, Tabellen um Elemente zu positionieren.
- Wenn Sie Tabellen nutzen, schachteln Sie nicht zu viele ineinander, Ihre Website wird dadurch langsamer.
- Vermeiden Sie exotische CSS-Eigenschaften (z.B. font-stretch).
- Entwickeln Sie Ihre Site zunächst mit nur einem Browser.
- Informieren Sie sich über die Fehler der einzelnen Browser und wenden Sie die entsprechenden „Hacks" an.
- Lassen Sie sich nicht zu schnell entmutigen, aller Anfang ist schwer.

2.4 Barrierefreiheit – Internet verbindet

Doch leider profitieren davon nicht alle. Behinderte Menschen scheitern oft an Barrieren, die ihnen das Navigieren auf Websites unmöglich machen. Damit möglichst alle Menschen von der Verbindung Internet profitieren können, sollten Websites möglichst barrierearm sein.

Sicher denken Sie bei Barrierefreiheit zunächst an eine Person im Rollstuhl, welche vor einer Treppe im Kaufhaus steht. Tatsächlich ist es so ähnlich. Denn das Web ermöglicht den Behinderten Zugang zu Informationen, die ihnen sonst verschlossen bleiben. Ein blinder Mensch hat nicht die Möglichkeit, morgens die Zeitung aufzuschlagen um an die News des Tages zu kommen, auch kann er nicht im Telefonbuch nach einer benötigten Nummer suchen. Dinge, die für uns selbstverständlich und alltäglich sind, bleiben Behinderten verschlossen. Das hat sich mit der Nutzung des Internets zum Glück geändert.

Es ist unsere Aufgabe, Websites so zu erstellen, dass Informationen allen Menschen zugänglich sind, dabei sollte man aber auf jede Art von Dogmatismus verzichten und immer wieder darauf hinweisen, dass man Kompromisse machen muss, um eine größtmögliche Zielgruppe zu erreichen.

Was kann man aber konkret tun um Barrieren im Web abzubauen?

Zunächst sollte man auf einen validen Code achten und auf Tabellen zum Layout verzichten. Darüber hinaus sollten die Seiten semantisch korrekt aufgebaut sein, eine Überschrift 1. Ordnung kommt vor einer Überschrift 2. Ordnung, ein p-Tag liegt nicht in einem span-Bereich usw.
Es ist gar nicht so schwer, wenn Sie einen Brief schreiben, bauen Sie diesen auch nach gewissen formalen Kriterien und in einer bestimmten Reihenfolge auf. Warum also nicht auch im Web? In den folgenden Abschnitten finden Sie einige weitere Hinweise, die man bei der Entwicklung beachten sollte.

2.4.1 Übersichtlichkeit

Das Layout Ihrer Website sollte generell eine einfache und schnelle Aufnahme aller Informationen gewährleisten. Das hat nichts mit behinderten Menschen explizit zu tun, sondern gehört zum Standard einer guten Website.

2.4.2 Kontraste

Wichtig ist eine ausreichend kontrastreiche Darstellung der gesamten Präsentation. Die Kombination von Hintergrund- und Schriftfarbe sollte so ausgewählt werden, dass zwischen ihnen ein deutlicher