

MEDIENÜBERGREIFENDES  
PUBLIZIEREN VON  
SCHULUNGSMATERIALIEN  
UNTER VERWENDUNG VON  
DOCBOOK-XML UND XSL



**D I P L O M A R B E I T**

IM FACHBEREICH INFORMATIK  
UND KOMMUNIKATIONSSYSTEME  
AN DER HOCHSCHULE MERSE-  
BURG (FH)

vorgelegt von  
Christian Auspurg  
Matrikel 02KTT  
aus Leipzig

---

Erster Betreuer: Herr Dr. Thomas Meinike  
Zweiter Betreuer: Herr Dipl.-Tech.-Red. (FH) Christian Günther  
Eingereicht am: 26. Juli 2007

Medienübergreifendes Publizieren von  
Schulungsmaterialien unter Verwendung  
von DocBook-XML und XSL

# Inhaltsverzeichnis

<b>1 Einführung</b> .....	<b>1</b>
1.1 Aufgabenstellung .....	2
1.2 Zielstellung .....	3
<b>I Grundlagen und Technologien</b>	
<b>2 Medienübergreifendes Publizieren</b> .....	<b>6</b>
2.1 Merkmale .....	7
2.2 Funktionsweise .....	9
2.3 Realisierung .....	10
2.4 Vorteile und Nachteile .....	12
<b>3 XML – Extensible Markup Language</b> .....	<b>14</b>
3.1 Merkmale .....	15
3.2 Aufbau .....	17
3.3 XML-Anwendungen – XML-Familie .....	18
3.4 Dokumenttyp-Definitionen .....	20
3.5 Namensräume .....	21
3.6 XML und Cross-Media-Publishing .....	22
<b>4 XSL – Extensible Stylesheet Language</b> .....	<b>23</b>
4.1 XSLT – XSL Transformations .....	24
4.1.1 Aufbau von XSLT-Stylesheets .....	25
4.1.2 Template-Regeln .....	26
4.1.3 Transformation .....	26
4.1.4 XSLT-Beispiel .....	28
4.2 XPath – XML Path Language .....	30
4.2.1 Funktionsweise .....	30
4.2.2 XPath-Ausdrücke .....	31

4.3 XSL-FO – XSL Formatting Objects .....	32
4.3.1 Aufbau von XSL-FO-Stylesheets .....	33
4.3.2 Seitenvorlagen .....	34
4.3.3 Seitenfolgen .....	37
4.3.4 Statische Inhalte und Textfluss .....	40
4.3.5 Verarbeitungsprozess .....	41
<b>5 DocBook .....</b>	<b>44</b>
5.1 DocBook-XML .....	46
5.1.1 Merkmale .....	46
5.1.2 DocBook-XML-Beispiel .....	47
5.2 DocBook-XSL .....	48
5.2.1 Anpassungsmöglichkeiten .....	48
5.2.2 Customization-Layer .....	49
 <b>II Praktische Umsetzung</b>	
<b>6 System- und Zielanalyse .....</b>	<b>54</b>
6.1 CMP-Lösung .....	54
6.2 Schulungsmaterialien .....	55
<b>7 Erarbeiten der Cross-Media-Publishing-Lösung ....</b>	<b>57</b>
7.1 DocBook-XML-Publishing-Konzept als Basis .....	57
7.2 Zielformat DocBook-XML .....	58
7.3 Zielformat PDF .....	59
7.4 Zielformat XHTML .....	60
<b>8 Einrichten der Cross-Media-Publishing-Lösung und Testlauf .....</b>	<b>61</b>
8.1 System-Umgebung .....	61
8.2 DocBook-XML .....	63
8.3 DocBook-XSL .....	63

8.4 RenderX XEP .....	63
8.5 xsltproc .....	65
8.6 ooo2dbk .....	65
8.7 Testlauf .....	65
<b>9 Umsetzen des Gestaltungskonzeptes .....</b>	<b>70</b>
9.1 Finden der richtigen Anpassungsmöglichkeiten .....	70
9.2 Druck-Variante der Schulungsmaterialien .....	72
9.2.1 Seiteneinrichtung .....	72
9.2.2 Anders verwirklicht: Das Inhaltsverzeichnis .....	76
9.2.3 Nicht verwirklicht: Seitenlayout mit Illustratio- nen .....	78
9.3 Online-Variante der Schulungsmaterialien .....	80
9.3.1 Einstellungen im Customization-Layer für XHTML .....	81
9.3.2 Einstellungen im Cascading Stylesheet .....	82
<b>10 Integrieren der Inhalte .....</b>	<b>84</b>
10.1 DocBook-XML-Dokumente .....	84
10.1.1 Auswählen geeigneter DocBook-XML-Elemente .....	84
10.1.2 Erstellen von DocBook-XML-Dokumenten .....	86
10.1.3 Unterstützung durch ooo2dbk .....	88
10.2 Bildmaterial .....	90
10.2.1 Hinterlegen des Bildmaterials .....	90
10.2.2 Einfügen in DocBook-XML-Dokumente .....	91
<b>11 Publizieren auf Knopfdruck .....</b>	<b>93</b>
<b>12 Zusammenfassung .....</b>	<b>96</b>
<b>Literaturverzeichnis .....</b>	<b>100</b>
<b>Eidesstattliche Erklärung .....</b>	<b>104</b>

## Abbildungsverzeichnis

2.1 Einfache schematische Darstellung von Cross-Media-Publishing .....	6
2.2 Ansatz einer einfachen CMP-Lösung für Schulungunterlagen .....	10
4.1 Ablauf einer XSLT-Transformation .....	28
4.2 Grundaufbau eines XSL-FO-Stylesheets .....	33
4.3 Seitenmodell von XSL-FO .....	35
4.4 Vorlagen und Zuweisung .....	40
4.5 Verarbeitungsprozess .....	42
4.6 Transformation und Formatierung .....	43
5.1 DocBook-XML-Publishing .....	45
7.1 Cross-Media-Publishing-Lösung für das medienübergreifende Publizieren der Schulungsmaterialien .....	58
8.1 Ordnerstruktur der CMP-Lösung .....	62
8.2 Formatting Settings in XEP .....	66
8.3 Ergebnis des Testlaufs für das Zielformat PDF .....	67
8.4 Ergebnis des Testlaufs für das Zielformat XHTML .....	69
9.1 Layout einer Doppelseite .....	73
9.2 Aufteilung der Regionen .....	74
9.3 Layout für das Inhaltsverzeichnis .....	77
9.4 Layout mit Illustrationen .....	79
9.5 Layout für die Online-Variante .....	80
12.1 Online-Ausgabe der Schulungsmaterialien .....	97
12.2 Druckausgabe der Schulungsmaterialien .....	98

## Beispiele

3.1 Aufbau von XML-Elementen .....	17
3.2 Aufbau von XML-Dokumenten .....	18
3.3 Namensräume .....	21
4.1 Grundaufbau eines XSLT-Stylesheets .....	25
4.2 XSLT-Beispiel .....	28
4.3 XPath-Beispiele .....	31
4.4 Grundaufbau eines XSL-FO-Stylesheets .....	34
4.5 Simple-Page-Master .....	36
4.6 Seitenfolge mit mehreren, bedingten Seitenvorlagen .....	38
5.1 Grundaufbau eines DocBook-XML-Dokumentes .....	47
5.2 Customization-Layer .....	50
8.1 Schriften-Konfiguration in XEP .....	64
9.1 Seitendefinition für eine linke Seite .....	75
9.2 Auszug aus dem Customization-Layer für XHTML .....	81
9.3 Unterschiedliche Gestaltung von <h2>-Elementen .....	83
9.4 CSS-Eigenschaften für die Navigationsleiste .....	83
10.1 Grundgerüst der Schulungsmaterialien .....	86
10.2 Verändertes Grundgerüst .....	87
10.3 Ausgelagertes Dokument vorwort.docb.xml .....	88
10.4 <mediaobject>-Element .....	91
11.1 Makefile .....	94

# 1 Einführung

Technischen Redakteuren steht heute für die Wissensvermittlung und -dokumentation eine Vielzahl von Medien zur Verfügung. Längst sind sie für ihre Publikationen nicht mehr nur auf Bücher, Journale und Broschüren beschränkt. Neben diesen klassischen gedruckten Medien spielen heute elektronische Medien und deren Formate eine zunehmend wichtige Rolle. Auf Computern, Personal Digital Assistants (PDA) und Mobiltelefonen können Internetseiten, Online-Hilfen, elektronische Bücher usw. gelesen und genutzt werden.

Inhalte zu einem bestimmten Thema oder Sachverhalt müssen nun also für verschiedene Medien aufbereitet werden. Für den Technischen Redakteur entsteht so erheblich mehr Aufwand, möchte er zum Beispiel einen wissenschaftlichen Artikel nicht nur in einem Journal, sondern außerdem auch auf einer Internetseite und im Portable Document Format publizieren. Müssen dann noch weitere Inhalte oder Korrekturen in den Artikel eingearbeitet werden, wirken sich diese Änderungen selbstverständlich auf alle drei Medien aus. Und das bedeutet noch mehr Aufwand.

An dieser Stelle entsteht der Wunsch nach einem Konzept, das die Arbeitsabläufe beim so genannten medienübergreifenden Publizieren vereinfacht und automatisiert. Gleichzeitig sollen aber die besonderen Eigenschaften der jeweiligen Medien optimal genutzt werden, zum Beispiel die Lesefreundlichkeit und das anspruchsvolle Layout eines Buches oder aber die schnelle Abrufbarkeit und die Interaktivität einer Internetseite.

## 1.1 Aufgabenstellung

Die Forschungsstelle zur Rehabilitation von Menschen mit kommunikativer Behinderung (FST), ein An-Institut der Martin-Luther-Universität Halle-Wittenberg, möchte Schulungsmaterialien zum Thema »Fair und erfolgreich prüfen mit textoptimierten Prüfungsaufgaben« veröffentlichen. Autorin der Schulungsmaterialien ist Dr. Susanne Wagner, wissenschaftliche Mitarbeiterin der Forschungsstelle.

Es werden folgende Anforderungen gestellt:

- Die Schulungsmaterialien sollen für verschiedene Medien erstellt werden, zunächst in gedruckter Form als Broschüre und in einer für das Internet geeigneten Form. Die Möglichkeit, später weitere Formate wie beispielsweise eine Online-Hilfe zu nutzen, soll offen bleiben.
- Die Schulungsmaterialien sollen ansprechend und benutzerfreundlich für die verschiedenen Medien gestaltet sein. Es wird Wert auf eine Gestaltung gelegt, die den Leser motiviert, sich mit der umfangreichen wissenschaftlichen Thematik zu beschäftigen.
- Die verschiedenen Publikationsformen sollen automatisiert erstellt werden, um den Arbeitsaufwand gegenüber der jeweiligen manuellen Erstellung – Satz und Umbruch bei Drucksachen, »Programmieren« bei Online-Angeboten – zu verringern. Im Idealfall soll »Publizieren auf Knopfdruck« möglich sein.
- Es soll leicht möglich sein, Änderungen an den Schulungsmaterialien vorzunehmen. Ergänzungen und Korrekturen sollen schnell und ohne großen Aufwand integrierbar sein.

Um die Schulungsmaterialien »Fair und erfolgreich prüfen mit textoptimierten Prüfungsaufgaben« zu realisieren, wurden zwei Teilprojekte vergeben. Die Ergebnisse dieser Projekte werden im Rahmen jeweils einer Diplomarbeit dokumentiert.

Die Diplomarbeit »Optimale Gestaltung von Schulungsmaterialien unter dem Aspekt des medienübergreifenden Publizierens« von Susanne Seyfarth befasst sich mit der Konzeption und Erstellung von Layout- und Gestaltungsvorlagen für die Druck- sowie die Online-Ausgabe der Schulungsmaterialien. Dabei werden Aspekte des medienübergreifenden Publizierens berücksichtigt.

Die hier vorliegende Diplomarbeit »Medienübergreifendes Publizieren von Schulungsmaterialien unter Verwendung von DocBook-XML und XSL« beschäftigt sich mit technischen Möglichkeiten zum medienübergreifenden Publizieren der Schulungsmaterialien. Hierbei werden die Layout- und Gestaltungsvorlagen der zuvor genannten Diplomarbeit angewendet.

## 1.2 Zielstellung

Diese Diplomarbeit soll einen Weg aufzeigen, wie sich die Schulungsmaterialien »Fair und erfolgreich prüfen mit textoptimierten Prüfungsaufgaben« mit DocBook-XML und XSL technisch umsetzen lassen.

Zunächst sollen in einem theoretischen Teil Grundlagen des medienübergreifenden Publizierens – Merkmale, Zielstellungen und konzeptionelle Aspekte – besprochen werden. Ein weiterer Bestandteil der theoretischen Betrachtungen sollen Technologien sein, die bei der technischen Umsetzung der Schulungsmaterialien eingesetzt werden können und die das medienübergreifende Publizieren überhaupt erst ermöglichen.

Der praktische Teil dieser Diplomarbeit soll zeigen, wie eine Lösung für medienübergreifendes Publizieren auf der Basis von DocBook-XML und XSL entwickelt wird. Es soll dokumentiert werden, wie mit dieser Lösung eine Druckausgabe und eine Internetausgabe der Schulungsmaterialien entstehen. Anhand wichtiger Schritte sollen Herangehensweisen und Arbeitsabläufe vorgestellt werden, wie Text-, Layout- und Gestaltungsvorlagen angewendet werden, um daraus medienübergreifende Publikationen erstellen zu können.

Eine weitere Zielstellung dieser Diplomarbeit soll es sein, herauszufinden, inwieweit sich DocBook-XML und XSL eignen, die Schulungsmaterialien zu realisieren und Vorgaben zu Layout und Gestaltung umzusetzen. Hierzu soll ein Vergleich zwischen den Vorlagen und den mit DocBook-XML und XSL erstellten Publikationen gezogen werden.

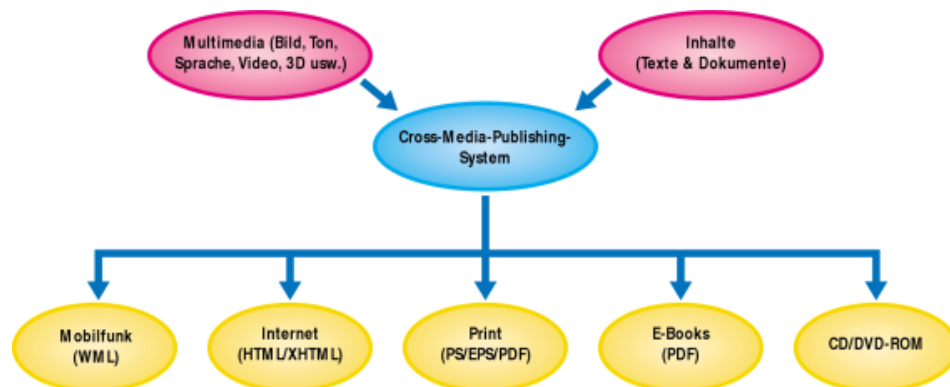
Der Diplomarbeit liegt eine CD-ROM mit Arbeitsmaterialien bei. Die darauf enthaltenen Dokumente können herangezogen werden, um die Ausführungen und Beispiele aus dem zweiten Teil der Arbeit nachzuvollziehen. Die praktischen Ergebnisse – die Schulungsmaterialien, die im Rahmen dieser Diplomarbeit technisch realisiert wurden – befinden sich ebenfalls auf der CD-ROM.

# **I Grundlagen und Technologien**

## 2 Medienübergreifendes Publizieren – Cross-Media-Publishing

Für das medienübergreifende Publizieren werden sehr häufig die englischen Begriffe »Cross-Media-Publishing«, kurz CMP, oder »Single-Source-Publishing« verwendet. Interessant ist, dass beide Begriffe sich vom Namen her auf unterschiedliche Aspekte des medienübergreifenden Publizierens beziehen. So deutet Single-Source-Publishing auf den Ausgangspunkt hin, das Publizieren aus *einer* Quelle, Cross-Media-Publishing dagegen auf das Ergebnis, die medienübergreifenden Publikationen. Und damit ist bereits der Grundgedanke des medienübergreifenden Publizierens benannt – aus *einer* Quelle *mehrere* Ausgabeformate erzeugen.

Abbildung 2.1: Einfache schematische Darstellung von Cross-Media-Publishing



Quelle: [Grunenberg 2001]

Dieser Grundgedanke ist zunächst einmal allen Cross-Media-Publishing-Konzepten gemein. Je nach Zielstellung und Einsatz-Umgebung unterscheiden sich jedoch die Anforderungen, die an ein solches Konzept gestellt werden. Es kann sehr einfach

gehalten, aber auch äußerst komplex sein. Eine Hauptanforderung an Cross-Media-Publishing ist heute ein möglichst hoher Grad an Automation. Das bedeutet, der Anwender erstellt die Quelldokumente, die Erzeugung der medienübergreifenden Publikationen übernimmt das so genannte Cross-Media-Publishing-System. In der Literatur wird deshalb im Zusammenhang mit Cross-Media-Publishing häufig von »Publizieren auf Knopfdruck« gesprochen.

## 2.1 Merkmale

»Das wesentliche Merkmal von CMP-Systemen ist das Überschreiten von Grenzen zwischen den einzelnen Medien, Datenformaten und Systemplattformen.« [Fritsche 2000, 119]

### Medienübergreifendes Publizieren

Ein ganz wichtiges Merkmal wurde bereits als Grundgedanke formuliert: Alle Inhalte einer medienübergreifenden Publikation – Texte, Bilder, Animationen usw. – stammen aus einem gemeinsamen Quell-Datenbestand. Aus diesem heraus werden sie für das jeweilige Zielmedium, z. B. den Druck oder den Bildschirm, in der optimalen Form, also mediengerecht, aufbereitet.

### Plattformunabhängige Formate

Nun soll dieser Ansatz nicht an den Grenzen zwischen Systemplattformen scheitern. Deshalb basieren medienübergreifende Publikationen auf plattformunabhängigen Formaten wie PDF, PostScript, HTML und XML – also Formaten, die auf Linux- und Windows-Rechnern, Macs und anderen Systemen gleichermaßen verwendet werden können.

## Integrationsfähigkeit

Dasselbe gilt auch für die Software, auf der ein CMP-System basiert. Viele Programme, die im Rahmen eines CMP-Systems eingesetzt werden, sind für die unterschiedlichsten Plattformen verfügbar und lassen sich somit in fast jede Systemumgebung integrieren. Anderenfalls »lässt sich eine auf den jeweiligen Bedarf zugeschnittene Lösung erarbeiten.« [Fritsche 2000, 199]

»Medienn neutrale Datenhaltung und Trennung von Inhalt und Layout beziehungsweise mehrfache Verwendung der Daten sind Kernpunkte aller CMP-Konzepte.« [Fritsche 2000, 122]

## Medienn neutrale Datenhaltung

Um Inhalte bzw. Daten für alle möglichen Zielmedien optimal aufbereiten zu können, sollten sie zunächst einmal in einer Form vorliegen, die nicht schon für ein bestimmtes Medium oder einen bestimmten Verwendungszweck zugeschnitten ist. Das ist natürlich nicht für alle Inhalte möglich, denn beispielsweise Videos oder Animationen sind nunmal nicht medienn eutral. In diesem Fall spielt wieder die mediengerechte Aufbereitung eine Rolle.

## Trennung von Inhalt und Layout

Bei medienübergreifenden Publikationen werden Inhalte getrennt von Angaben zur Gestaltung abgelegt. Die Verwendungsmöglichkeiten sind damit sehr flexibel, denn erst wenn die Inhalte für ein bestimmtes Medium aufbereitet werden, bekommen sie eine entsprechende Gestaltungsvorlage zugewiesen. Cross-Media-Publishing mit einem Content-Management-System geht noch einen Schritt weiter. Hier wird sogar eine Trennung von Inhalt, Layout und Struktur vorgenommen.

## Wiederverwendung von Inhalten

»Zurzeit sind Print, CD-ROM und Internet die wichtigsten Medien, doch mit Blick auf die Zukunft sollte man sich auch mit kommenden oder bereits bestehenden Alternativen auseinandersetzen.« [Grunenberg 2001]

Inhalte sollen beim medienübergreifenden Publizieren nicht nur einmal verwendet werden. Anderenfalls wäre schon allein der Begriff »medienübergreifendes Publizieren« gar nicht gerechtfertigt. Inhalte sollen also mehrfach, für viele Publikationen verwendet werden. Durch die zuvor beschriebenen Merkmale von CMP-Systemen ist dies gewährleistet, und es stehen außerdem Möglichkeiten offen, die Inhalte auch in der Zukunft vielseitig nutzen zu können.

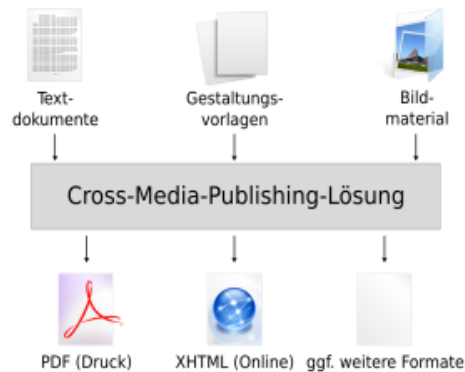
Das Prinzip der Wiederverwendung gilt auch für Gestaltungsvorlagen. So kann beispielsweise eine Gestaltungsvorlage, die für eine bestimmte Publikationsform angelegt worden ist, selbstverständlich auf unterschiedliche Inhalte angewendet werden.

## 2.2 Funktionsweise

In den beiden vorangegangenen Abschnitten wurde bereits besprochen, worum es beim medienübergreifenden Publizieren geht und welche Merkmale CMP-Systeme aufweisen. Es wurde außerdem gesagt, dass CMP-Systeme unter Umständen sehr komplex aufgebaut sein können – beispielsweise mit integrierten Datenbanken und Content-Management-Systemen.

Die folgende Grafik zeigt den Ansatz einer einfachen CMP-Lösung, die für die medienübergreifende Publikation der Schulungsmaterialien »Fair und erfolgreich prüfen mit textoptimierten Prüfungsaufgaben« angewendet werden soll.

Abbildung 2.2: Ansatz einer einfachen CMP-Lösung für Schulungsunterlagen



Diese Grafik wird in ähnlicher Form in dieser Diplomarbeit wiederkehren und durch weitere Merkmale und Details ergänzt werden.

## 2.3 Realisierung

Nach Fritsche lassen sich CMP-Konzepte in fünf Schritten realisieren. [Fritsche 2000, 130–133]

### Schritt 1: System- und Zielanalyse

Bevor eine CMP-Lösung entwickelt werden kann, müssen zunächst einmal vorhandene Datenbestände und Gegebenheiten analysiert sowie Ziele bzw. Anforderungen an das CMP-System formuliert werden. In diesem Zusammenhang können beispielsweise folgende Fragen auftreten:

- Welche Inhalte sollen publiziert werden?
- Welche Zielgruppen sollen angesprochen werden?
- Welche Inhalte sind bereits vorhanden?
- Welche Inhalte müssen noch erstellt werden?

- Wer wird das CMP-System anwenden?
- Besteht die Möglichkeit, das CMP-System zu erweitern bzw. an neue Anforderungen anzupassen?

Außerdem sollten Vor- und Nachteile von CMP-Systemen in die Analyse einbezogen werden (siehe hierzu Abschnitt 2.4, Vorteile und Nachteile).

### Schritt 2: Ausarbeiten einer CMP-Lösung

Auf der Grundlage der System- und Zielanalyse aus Schritt 1 kann nun eine geeignete CMP-Lösung ausgearbeitet werden.

### Schritt 3: Übernahme oder Neugestaltung des Designs

Im dritten Schritt werden Gestaltungsvorgaben wie z. B. Corporate Designs u. Ä. in das CMP-System überführt. Konkret werden eine Reihe von Gestaltungsvorlagen erstellt, die auf den entsprechenden Vorgaben beruhen und mit deren Hilfe später Inhalte für die unterschiedlichen Medien aufbereitet werden können.

### Schritt 4: Einrichten des CMP-Systems und Testlauf

Nachdem das CMP-System eingerichtet wurde, wird ein Testlauf mit repräsentativen Beispiel-Inhalten durchgeführt. Anhand dieses Tests soll überprüft werden, ob die Daten korrekt übernommen wurden, ob die Gestaltungsvorlagen richtig funktionieren und ob die medienübergreifenden Publikationen fehlerfrei produziert werden.

## Schritt 5: Integrationsphase

Verläuft die Testphase positiv, so ist das CMP-System einsatzbereit. Im letzten Schritt können alle benötigten Daten eingegeben und die reguläre Arbeit mit dem System aufgenommen werden.

## 2.4 Vorteile und Nachteile

Medienübergreifendes Publizieren bietet dem Anwender viele Vorteile, allerdings auch eine Reihe von Nachteilen. In der folgenden Auflistung werden zunächst einmal einige allgemeine Vor- und Nachteile genannt.

### Vorteile

- Ist ein CMP-System einmal eingerichtet, können damit sehr schnell medienübergreifende Publikationen erstellt werden. Im Idealfall entstehen unterschiedlichste Medienprodukte automatisiert »auf Knopfdruck«.
- Fallen bei Inhalten Änderungen durch Aktualisierungen oder Korrekturen an, so können diese Änderungen auch schnell und unkompliziert in die einzelnen Publikationen übernommen werden.
- Inhalte können für verschiedene Zwecke und Medien verwendet werden, da sie – soweit möglich – medienneutral vorliegen.
- Gestaltungsvorlagen können wiederverwendet, modifiziert und weiterentwickelt werden.

### Nachteile

- Ein CMP-System zu planen und mit allen dazugehörigen Komponenten einzurichten kann einen erheblichen Aufwand mit sich bringen. Bis die ersten Publikationen erstellt werden

können, ist dementsprechend viel Vorarbeit nötig. Der spätere eigentliche Nutzen eines CMP-Systems sollte diese Vorarbeit ausgleichen.

- Ein CMP-System bietet meist weniger individuelle Gestaltungsmöglichkeiten für Publikationen als ein DTP-Programm.

Für jede Cross-Media-Publishing-Lösung ergeben sich weitere, individuelle Vor- und Nachteile. Diese hängen von den Technologien und deren Leistungsmerkmalen ab, die bei der jeweiligen Lösung eingesetzt werden.

## 3 XML – Extensible Markup Language

XML steht für »Extensible Markup Language«, einen englischen Begriff, der sich mit »erweiterbare Auszeichnungssprache« ins Deutsche übersetzen lässt. Zur Begriffsklärung soll nun erläutert werden, was sich hinter dem Ausdruck »Auszeichnungssprache« verbirgt und was in diesem Zusammenhang mit »erweiterbar« gemeint ist.

### Auszeichnungssprache

Schon in der Druckindustrie der vergangenen Jahrhunderte dienten so genannte Auszeichnungen im Text den Schriftsetzern als Setzanweisungen, um anhand eines Manuskriptes eine Druckform herstellen zu können. Dabei wurden einzelnen Textbestandteilen Eigenschaften und Bedeutungen zugewiesen. Genau dies ist das Prinzip einer Auszeichnungssprache – sie beschreibt Inhalte.

### Erweiterbar

Einige Auszeichnungssprachen, z. B. HTML, haben einen fest eingegrenzten Sprachumfang, und nur in dessen Rahmen können sie Inhalte beschreiben. XML hingegen ist »[...] eine Spezifikation für die Schaffung beliebig vieler Auszeichnungssprachen.« [Behme & Mintert 2000, 28] Es handelt sich um eine so genannte Meta-Auszeichnungssprache, eine Sprache, mit der sich andere Auszeichnungssprachen definieren lassen. Der Sprachumfang von XML ist also nicht eingegrenzt, sondern beliebig erweiterbar.

### 3.1 Merkmale

»XML ist ein Satz von Regeln zur Definition von semantischen Tags, die ein Dokument in Teile zerlegen und die verschiedenen Teile des Dokuments beschreiben.« [Harold 2004, 31]

Dieser Satz von Regeln wurde in der Empfehlung des W3C »Extensible Markup Language (XML) 1.0« [W3C 2006] im Februar 1998 erstmals festgeschrieben. – Das W3C, das World Wide Web Consortium, »[...] ist ein internationales Konsortium, in dem Mitgliedsorganisationen, ein fest angestelltes Team, und die Öffentlichkeit gemeinsam daran arbeiten, Web-Standards zu entwickeln.« [W3C 2007] – Neben dem eigentlichen Regelwerk wurden in der Empfehlung auch Entwurfsziele für XML formuliert, die gleichzeitig wichtige Merkmale darstellen.

»Die Entwurfsziele für XML sind:

- XML soll sich im Internet auf einfache Weise nutzen lassen.
- XML soll ein breites Spektrum von Anwendungen unterstützen. [...]
- XML-Dokumente sollten für Menschen lesbar und angemessen verständlich sein. [...]
- XML-Dokumente sollen leicht zu erstellen sein. [...]«

[Behme & Mintert 2000, 389]

#### Meta-Auszeichnungssprache

Bei XML werden Inhalte durch Elemente mit Tags beschrieben und strukturiert. Die Namen der Elemente lassen sich im Rahmen der XML-Spezifikation frei festlegen. Da XML ein Regelwerk zur

Schaffung beliebiger Auszeichnungssprachen ist, lässt sich auch eine Auszeichnungssprache für Schulungsmaterialien definieren.

### Trennung von Inhalt und Layout

XML beschreibt die Bedeutung von Inhalten und bildet deren Struktur ab. Darüber, wie die Inhalte dargestellt werden sollen, sagt XML jedoch nichts aus. Hierfür sind andere Sprachen zuständig, beispielsweise Cascading Stylesheets (CSS) oder die Extensible Stylesheet Language (XSL). Letztere basiert selbst auf XML und soll im folgenden Kapitel näher vorgestellt werden. Inhalte und deren Gestaltung werden bei XML also voneinander getrennt behandelt. Daraus ergeben sich einige weitere Merkmale.

### Mediennutralität

XML ist medienneutral, da die Inhalte weder auf ein bestimmtes Medium noch auf einen bestimmten Verwendungszweck festgelegt sind.

### Medienübergreifende Publikation

XML-Daten können für viele verschiedene Medien aufbereitet werden, indem sie Gestaltungsvorlagen zugewiesen bekommen oder aber sogar in andere Formate umgewandelt werden. Somit lassen sich XML-Daten auch auf unterschiedliche Weise wiederverwenden.

### Weitere Merkmale

XML ist plattformunabhängig und lizenzfrei. Darüber hinaus ist XML hervorragend dokumentiert [W3C 2001]. Es existieren eine

Vielzahl von Büchern sowie Quellen im Internet, die Informationen rund um das Thema XML bereitstellen.

## 3.2 Aufbau

### XML-Elemente

Es wurde bereits geklärt, dass bei XML Inhalte durch Elemente mit Tags beschrieben und strukturiert werden. Es soll nun gezeigt werden, wie solche Elemente aufgebaut sind.

#### Beispiel 3.1: Aufbau von XML-Elementen

```
<absatz sprache="deutsch"> ❶  
  Dies ist ein Beispiel. ❷  
</absatz> ❸
```

- ❶ Start-Tag mit Attribut
- ❷ Inhalt des Elementes
- ❸ End-Tag

Jedes XML-Element besteht aus einem Start-Tag (einer Anfangsmarkierung), dem Element-Inhalt sowie einem End-Tag (einer Schlussmarkierung). Start-Tag und End-Tag umschließen den Element-Inhalt. Das Element im Beispiel trägt den Namen `absatz` und hat den Inhalt »Dies ist ein Beispiel.«. Das Start-Tag kann Attribute enthalten, welche das Element näher beschreiben. Hier wurde ein Attribut `sprache` gewählt, dessen Wert `deutsch` das Element `<absatz>` näher beschreibt.

### XML-Dokumente

Elemente sind die Grundbausteine von XML-Dokumenten. XML-Dokumente können aus einem einzigen oder aus beliebig vielen Elementen bestehen. Das nächste Beispiel zeigt den Grundaufbau.

### Beispiel 3.2: Aufbau von XML-Dokumenten

```
<?xml version="1.0" encoding="UTF-8"?> ❶
<!DOCTYPE buch ...> ❷
<buch> ❸
  <titel>Buchtitel</titel>
  <kapitel>
    <abschnitt> ... </abschnitt>
  </kapitel>
  <!-- weitere Elemente -->
</buch>
```

- ❶ XML-Deklaration
- ❷ Dokumenttyp-Deklaration
- ❸ Wurzelement

Das XML-Dokument beginnt mit der so genannten XML-Deklaration. Diese zeigt an, dass es sich um ein XML-Dokument handelt, und zwar eines der Version 1.0. Sie enthält weiterhin eine Angabe zum Zeichensatz, in dem das Dokument codiert ist, hier UTF-8.

Ein XML-Dokument kann eine Dokumenttyp-Deklaration enthalten. Dadurch wird das Dokument einem bestimmten Dokumenttyp und dessen Regeln zugeordnet (mehr dazu im Abschnitt 3.4, Dokumenttyp-Definitionen).

<buch> ist das Wurzelement des Dokumentes. Ein XML-Dokument besitzt genau ein Wurzelement, welches alle anderen Elemente und Inhalte enthält.

## 3.3 XML-Anwendungen – XML-Familie

Mit XML lassen sich Auszeichnungssprachen für die unterschiedlichsten Anwendungsgebiete entwickeln. Diese werden als XML-Anwendungen bezeichnet. Das W3C stellt bereits eine Reihe solcher XML-Anwendungen für allgemeine Aufgaben zur Verfügung.

»Hinter XML 1.0 steht die ›XML Familie‹ als ein wachsender Satz an Modulen, der nützliche Serviceleistungen für die Verwirklichung wichtiger und häufig angefragter Aufgaben bereithält.« [W3C 2001]

Die folgende Auflistung zeigt einige wichtige Vertreter der »XML-Familie«.

- XSL (Extensible Stylesheet Language), bestehend aus drei Teilen:
  - XSLT (XSL Transformations) zur Umwandlung von XML-Dokumenten
  - XSL-FO (XSL Formatting Objects) zur Formatierung von XML-Dokumenten
  - XPath (XML Path Language) zur Adressierung von Teilen eines XML-Dokumentes
- XLink (XML Linking Language) zur Definition von Hyperlinks in XML-Dokumenten
- XPointer (XML Pointer Language) zum Verweisen auf Teile eines XML-Dokumentes
- XML Schema zur Definition der Struktur von XML-Dokumenten

Außerdem wurden bereits zahlreiche weitere XML-Anwendungen für spezielle Anwendungsgebiete entwickelt. Auch hier seien einige wichtige Vertreter genannt.

- XHTML (Extensible Hypertext Markup Language), der in das XML-Regelwerk überführte Nachfolger von HTML
- SVG (Scalable Vector Graphics) zur Beschreibung zweidimensionaler Vektorgrafiken

- CML (Chemical Markup Language) zur Beschreibung und Verwaltung molekularer Informationen
- MathML (Mathematical Markup Language) zur Beschreibung mathematischer Ausdrücke
- DocBook zur Erstellung strukturierter Textdokumente, insbesondere technischer Dokumentationen
- DITA zur Beschreibung technischer Informationen als Topics [Wikipedia 2007]

### 3.4 Dokumenttyp-Definitionen

Eine Gruppe von gleichartigen Dokumenten bildet einen so genannten Dokumenttyp. So gehören z. B. alle Dokumente, welche in der XML-basierten Sprache SVG formuliert sind, eben dem Dokumenttyp SVG an. Die Gemeinsamkeiten von Dokumenten desselben Typs lassen sich in einer Dokumenttyp-Definition, kurz DTD, beschreiben.

»Eine DTD beschreibt den strukturellen Aufbau und die logischen Elemente einer *Klasse von Dokumenten*, genannt *Dokumenttyp*.« [Behme & Mintert 2000, 53]

Darüber hinaus definiert eine DTD auch die Beziehungen, die zwischen den Bestandteilen eines XML-Dokumentes bestehen. Eine DTD ermöglicht es also, das Regelwerk einer XML-Anwendung zu dokumentieren. Es lässt sich auch überprüfen, ob ein XML-Dokument, welches einer bestimmten XML-Anwendung angehört, der dazugehörigen DTD entspricht. Diese Gültigkeitsüberprüfung wird als Validierung bezeichnet.

Dokumenttyp-Definitionen für XML-Dokumente lassen sich auch über XML-Schemata realisieren. XML-Schemata erfüllen diesel-

ben Aufgaben wie DTDs und besitzen darüber hinaus einen höheren Funktionsumfang. Außerdem sind XML-Schemata – im Gegensatz zu DTDs – selbst XML-Dokumente.

### 3.5 Namensräume

Elementnamen für XML-Elemente sind (weitestgehend) frei wählbar. So kann es schnell vorkommen, dass zwei verschiedene Auszeichnungssprachen gleiche Elementnamen für unterschiedliche Bedeutungen verwenden. Ein gutes Beispiel hierfür ist das Element `<p>`. Im Zusammenhang mit XHTML steht `<p>` für einen Absatz (Paragraph), in einer anderen Sprache kann es aber für etwas ganz anderes stehen, z. B. für Postleitzahl. Namensräume bieten die Möglichkeit, Elemente an Auszeichnungssprachen zu binden und somit die jeweils richtige Bedeutung zu sichern.

»Ein Namensraum ist in XML ein Mechanismus zur Vermeidung von Namenskonflikten und Mehrdeutigkeiten.«

[Schraitle 2004, 31]

Namensräume erlauben es außerdem, Elemente, die zu verschiedenen Auszeichnungssprachen gehören, innerhalb eines Dokumentes zu vermischen. Wie Namensräume eingesetzt werden können, zeigt das nächste Beispiel.

#### Beispiel 3.3: Namensräume

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" ❶
  xmlns:fo="http://www.w3.org/1999/XSL/Format" ❷ ... > ...
  <xsl:template match="para"> ❸
    <fo:block> ... </fo:block>❹
  </xsl:template> ...
</xsl:stylesheet>
```

- ❶ Namensraum-Deklaration für XSLT
- ❷ Namensraum-Deklaration für XSL-FO

- ③ Ein XSLT-Element
- ④ Ein XSL-FO-Element

In einer Namensraum-Deklaration wird ein Präfix – im Beispiel `xsl:` bzw. `fo:` – einem Namensraum zugewiesen, der wiederum durch einen URI gekennzeichnet wird. Diese Präfixe können dann Elementen vorangestellt werden, um deren Zugehörigkeit zu einem bestimmten Namensraum und damit zu einer bestimmten Auszeichnungssprache zu symbolisieren. Elemente mit dem Präfix `xsl:` sind somit XSLT, Elemente mit dem Präfix `fo:` XSL-FO zugeordnet.

### 3.6 XML und Cross-Media-Publishing

»XML ist nicht immer die beste Lösung, aber es lohnt sich immer, XML in Erwägung zu ziehen.« [W3C 2001]

Die Merkmale von XML stimmen in einigen wesentlichen Punkten mit den Merkmalen von Cross-Media-Publishing-Systemen überein, die im vorhergehenden Kapitel erläutert wurden. Dies sind die Trennung von Inhalt und Layout, Medienneutralität, Wiederverwendung von Inhalten sowie die Plattformunabhängigkeit.

Es bietet sich daher an, für das medienübergreifende Publizieren der Schulungsmaterialien »Fair und erfolgreich prüfen mit textoptimierten Prüfungsaufgaben« eine Cross-Media-Publishing-Lösung zu entwickeln, die XML-basierte Dokumente und Technologien verwendet. In den folgenden beiden Kapiteln werden mit XSL und DocBook-XML zwei solche Technologien vorgestellt.

Die Tatsachen, dass XML lizenzfrei und ausführlich dokumentiert ist, sprechen ebenfalls für eine Cross-Media-Publishing-Lösung, die auf XML setzt.

## 4 XSL – Extensible Stylesheet Language

XSL steht für »Extensible Stylesheet Language«. Zu Deutsch heißt das »Erweiterbare Stylesheet-Sprache«, wobei ein Stylesheet als Stil- oder Gestaltungsvorlage zu verstehen ist. XSL ist eine vom W3C entwickelte Familie von Sprachen, der drei Mitglieder angehören:

- XSLT – XSL Transformations

XSLT ermöglicht es, XML-Daten zu transformieren, also umzuwandeln.

- XPath – XML Path Language

XPath wird von XSLT dazu verwendet, bestimmte Teile von XML-Dokumenten zu adressieren, aufzufinden oder auszuwählen.

- XSL-FO – XSL Formatting Objects

XSL-FO stellt Formatierungsobjekte bzw. Stilvorlagen bereit, um XML-Daten eine Gestaltung zuzuweisen.

Diese drei Sprachen können als Familie von Sprachen zusammenwirkend und aufeinander aufbauend, aber auch unabhängig voneinander eingesetzt werden. In diesem Kapitel sollen XSLT, XPath und XSL-FO mit ihren jeweiligen Merkmalen und Funktionsweisen vorgestellt werden.

## 4.1 XSLT – XSL Transformations

Die Extensible Stylesheet Language Transformations, kurz XSLT, ist eine »[...] flexible, leistungsfähige Sprache, um XML-Dokumente in etwas anderes umzuwandeln.« [Tidwell 2002, 1] XSLT-Stylesheets basieren auf dem XML-Regelwerk, sind also gleichzeitig auch XML-Dokumente. Die Ausgabeformate, die sich mit Hilfe von XSLT erzeugen lassen, können selbst XML-basiert sein, beispielsweise XSL-FO, SVG und XHTML, aber auch andere, nicht-XML-basierte Formate wie HTML oder Java-Code sind möglich.

XSLT 1.0 stellt 35 Elemente zur Verfügung und bietet damit umfassende Möglichkeiten, um

- XML-Elementen Eigenschaften zuzuweisen,
- XML-Elemente zu sortieren, zu gruppieren und zu filtern,
- Berechnungen aus XML-Daten durchzuführen,
- Verknüpfungen und Querverweise zwischen XML-Daten herzustellen sowie
- Logik-, Verzweigungs- und Verarbeitungsanweisungen auf XML-Daten anzuwenden. [Tidwell 2002; 2, 50]

Ein XSLT-Stylesheet zu verfassen bedeutet, anhand der gegebenen XSLT-Elemente Regeln festzulegen, die bestimmen, wie ein XML-Dokument in ein anderes Format umgewandelt werden soll.

XSLT als Angehörige der »XML-Familie« (siehe Abschnitt 3.3, XML-Anwendungen – XML-Familie) ist eine Empfehlung des W3C. Im Rahmen dieser Diplomarbeit wird die Version 1.0 besprochen und verwendet, welche im November 1999 veröffent-

lich wurde [W3C 1999]. Seit Januar 2007 liegt XSLT auch in der erweiterten Version 2.0 vor [W3C 2007a].

### 4.1.1 Aufbau von XSLT-Stylesheets

Im folgenden Beispiel sollen der Grundaufbau eines XSLT-Stylesheets gezeigt und die einzelnen Bestandteile erklärt werden.

#### Beispiel 4.1: Grundaufbau eines XSLT-Stylesheets

```
<?xml version="1.0" encoding="UTF-8"?> ❶
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0"> ❷
  <xsl:output method="xml" /> ❸
  <xsl:template match="/" /> ❹
  <!-- weitere Templates -->
</xsl:stylesheet>
```

- ❶ XSLT-Stylesheets sind XML-Dokumente. Sie beginnen deshalb mit der XML-Deklaration.
- ❷ Das Wurzelement eines XSLT-Stylesheets ist `<xsl:stylesheet>`. Es definiert stets den Namensraum sowie die XSLT-Version. Das Namensraum-Präfix für alle XSLT-Elemente ist `xsl:.`
- ❸ Das Element `<xsl:output>` enthält ein Attribut `method`, dessen Wert die Ausgabe-Methode der Transformation angibt. Hier sind `xml`, `html` und `text` möglich.
- ❹ Es folgen nun Template-Regeln (`<xsl:template>`). Diese werden im nächsten Abschnitt näher beschrieben. Auf dieser Ebene des XSLT-Stylesheets können auch weitere so genannte Top-Level-Elemente wie z. B. Parameter (`<xsl:param>`), Attribute-Sets (`<xsl:attribute-set>`, Attribute-Sätze) und Variablen (`<xsl:variable>`) definiert werden.

### 4.1.2 Template-Regeln

Der Teil eines XSLT-Stylesheets, der die eigentlichen Transformationsanweisungen enthält, ist aus so genannten Template-Regeln aufgebaut. Jede Template-Regel enthält ein Muster, welches bestimmt, auf welches Element bzw. welche Elemente des umzuwandelnden XML-Dokumentes die Template-Regel angewendet werden soll. Diese Aufgabe, das Adressieren von Elementen, erinnert an XPath, und tatsächlich handelt es sich bei einem Muster um einen XPath-Ausdruck. Der weitere Inhalt der Template-Regel gibt an, auf welche Art und Weise die ausgewählten XML-Elemente umgewandelt werden sollen. [Schraitle 2004, 277]

Eine Template-Regel ist also eine Art Vorlage oder Schablone, die auf ausgewählte XML-Elemente angewendet wird und festlegt, wie die betreffenden Elemente transformiert werden sollen.

### 4.1.3 Transformation

»Sie schreiben ein XSLT-Stylesheet, um die Regeln für die Umwandlung eines XML-Dokuments festzulegen, und der XSLT-Prozessor erledigt den Rest.« [Tidwell 2002, 1 f.]

Um ein XML-Dokument mit Hilfe eines XSLT-Stylesheets umwandeln zu können, bedarf es einer bestimmten Software, die als XSLT-Prozessor bezeichnet wird. Die folgenden Schritte erläutern, wie der Umwandlungsprozess – die Transformation – abläuft.

#### 1. Parsen des XML-Quelldokumentes

Der XSLT-Prozessor liest das XML-Quelldokument ein, analysiert es und erstellt daraus eine Baumdarstellung. Dieser

Vorgang wird als Parsen (engl.: to parse – zerlegen, analysieren) bezeichnet.

## 2. Parsen des XSLT-Stylesheets

Im zweiten Schritt passiert dasselbe mit dem Stylesheet: Es wird eingelesen und analysiert, und der Prozessor erstellt ebenfalls eine Baumdarstellung, die getrennt von der des XML-Quelldokumentes vorliegt.

## 3. Verarbeitung

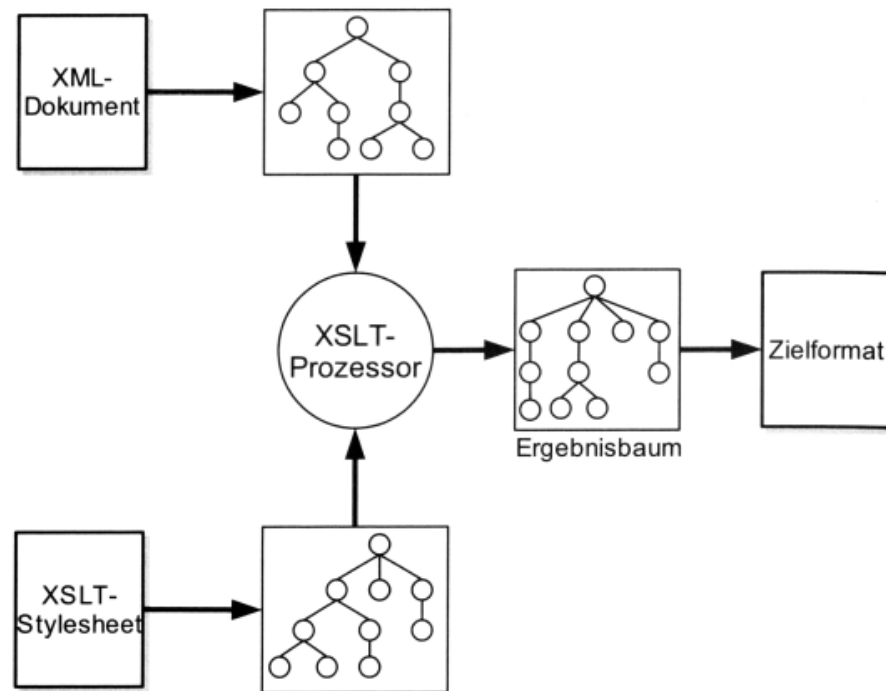
Die Template-Regeln, welche im XSLT-Stylesheet festgelegt sind, werden nacheinander auf das XML-Quelldokument angewendet. Auf diese Weise entsteht eine dritte Baumdarstellung, der Ergebnisbaum.

## 4. Ausgeben des Ergebnisbaums

Zuletzt wird der Ergebnisbaum ausgegeben. Das XML-Quelldokument wurde umgewandelt – die Transformation ist abgeschlossen. [Schraitle 2004, 269; Tidwell 2002, 25 ff.]

Die folgende Abbildung verdeutlicht den Prozess der Transformation noch einmal durch eine grafische Darstellung.

Abbildung 4.1: Ablauf einer XSLT-Transformation



Quelle: [Schraitle 2004, 270]

#### 4.1.4 XSLT-Beispiel

Anhand eines Beispiels soll nun gezeigt werden, wie ein einfaches XML-Dokument mit Hilfe eines XSLT-Stylesheets in ein HTML-Dokument umgewandelt werden kann und wie ein entsprechendes Stylesheet für diesen Zweck aussieht.

##### Beispiel 4.2: XSLT-Beispiel

Dies ist das XML-Quelldokument, das in ein HTML-Dokument umgewandelt werden soll.

```
<?xml version="1.0" encoding="UTF-8"?>
<tier ordnung="Raubtier">Krokodil</tier>
```

Mit Hilfe dieses XSLT-Stylesheets soll die Transformation durchgeführt werden.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0">
  <xsl:output method="html" /> ❶
  <xsl:template match="tier"> ❷
    <html>
      <head>
        <title>
          <xsl:value-of select="." /> ❸
        </title>
      </head>
      <body>
        <p>
          Das <xsl:value-of select="." /> ist ein
          <xsl:value-of select="@ordnung" />. ❹
        </p>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

- ❶ Durch die Transformation soll ein HTML-Dokument erzeugt werden. Die Ausgabemethode ist demzufolge auf html gesetzt.
- ❷ Diese Template-Regel bezieht sich auf das Wurzelement des XML-Quelldokumentes, also auf <tier>.
- ❸ Das Element <xsl:value-of select="." /> gibt den Inhalt des Knotens aus, der gerade bearbeitet wird. Hier ist dies der Inhalt des Elementes <tier>, also »Krokodil«.
- ❹ Hier werden alle Attribut-Werte des aktuell bearbeiteten Elementes ausgegeben. Das Ergebnis ist »Raubtier«. Alle anderen Elemente innerhalb der Template-Regel, die nicht das XSLT-Namensraum-Präfix xsl: aufweisen, werden unverändert in das Zieldokument übertragen.

Die Transformation ergibt schließlich folgenden HTML-Code:

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

```
<title>Krokodil</title>
</head>
<body>
  <p>Das Krokodil ist ein Raubtier.</p>
</body>
</html>
```

## 4.2 XPath – XML Path Language

Das W3C beschreibt XPath in der Empfehlung »XML Path Language (XPath) Version 1.0« vom November 1999 als eine Sprache, welche zur Adressierung von Teilen eines XML-Dokumentes dient [W3C 1999a]. XPath stellt eine Basistechnologie dar, die stets im Zusammenhang mit anderen XML-Sprachen wie beispielsweise XSLT und XPointer verwendet wird. Interessanterweise ist XPath selbst jedoch keine XML-Sprache [Schraitle 2004, 235].

Wie bereits zuvor angesprochen, wird XPath innerhalb von XSLT-Stylesheets benutzt, um einzelne Bestandteile von XML-Dokumenten aufzufinden und auszuwählen. Das Stylesheet kann dann auf den ausgewählten Bestandteil eine entsprechende Umwandlungsregel anwenden.

### 4.2.1 Funktionsweise

»Die Grundstruktur eines XML-Dokuments ist aus XPath-Sicht baumartig.« [Behme & Mintert 2000, 260] Es handelt sich dabei um diejenige Baumdarstellung, die beim Parsen eines XML-Quelldokumentes erstellt wird [Tidwell 2002, 43]. Diese Baumdarstellung besitzt verschiedene Arten von so genannten Knoten, die wiederum die Informationen des XML-Quelldokumentes enthalten:

- Wurzelknoten,
- Elementknoten,
- Attributknoten,
- Textknoten,

- Kommentarknoten,
- Namensraumknoten und
- Verarbeitungsanweisungsknoten. [Schraitle 2004, 237]

Durch Lokalisierungspfade und -schritte können in den Knoten gezielt Informationen aufgefunden werden. XPath ist in der Lage, vier verschiedene Datentypen zurückzugeben. Dies können sein:

- Knotenmengen,
- Boolesche Werte,
- Zahlen und
- Zeichenketten. [Schraitle 2004, 260]

#### 4.2.2 XPath-Ausdrücke

In diesem Abschnitt soll gezeigt werden, wie XPath-Ausdrücke aussehen.

»Die Syntax ist eine Mischung aus einfachen Programmiersprachenausdrücken (wie  $\$x*6$ ) und Unix-artigen Pfadausdrücken (wie `/sonett/autor/nachname`).« [Tidwell 2002, 43]

Im Rahmen dieser Diplomarbeit treten XPath-Ausdrücke ausschließlich im Zusammenhang mit XSLT auf. Es werden nun drei typische Beispiele gezeigt und deren Bedeutung erklärt. Die XPath-Ausdrücke sind kursiv hervorgehoben.

#### Beispiel 4.3: XPath-Beispiele

```
<xsl:template match="chapter/title"> ... </xsl:template> ❶  
<xsl:when test="$sequence='first'"> ... </xsl:when> ❷  
<xsl:value-of select="." /> ❸
```

- ❶ Die Template-Regel trifft auf alle `<title>`-Elemente zu, deren übergeordnetes Element (Elternelement) `<chapter>` ist.
- ❷ Wenn der Wert des Parameters `sequence` gleich `first` ist ...
- ❸ Es wird der Inhalt des Kontextknotens ausgegeben.

### 4.3 XSL-FO – XSL Formatting Objects

Bei Extensible Stylesheet Language Formatting Objects – die Kurzbezeichnung lautet XSL-FO – handelt es sich um eine Seitenbeschreibungssprache. Mit XSL-FO-Stylesheets lässt sich beschreiben, wie Inhalte auf einer Seite angeordnet und formatiert werden sollen. Wie der Name bereits verrät, hält XSL-FO eine Reihe von Formatierungsobjekten bereit. Mit deren Hilfe können sowohl einfache als auch sehr komplexe Layouts für Dokumente – angefangen vom einfachen Informationsblatt bis hin zum umfangreichen Buch – realisiert werden. Mögliche Ausgabeformate sind z. B. PDF oder PostScript, die wiederum für den Druck oder zum Lesen am Bildschirm genutzt werden können.

XSL-FO ist eine XML-Anwendung. Somit folgen auch XSL-FO-Dokumente bzw. -Stylesheets dem XML-Regelwerk. Mit den Elementen, die XSL-FO zur Verfügung stellt, und deren Attributen lassen sich unter anderem folgende Aspekte kontrollieren:

- »Regionen, Ränder und Bereiche einer Seite
- Breite und Höhe von Seiten
- Abfolge von Seiten
- Seitennummerierung
- Rahmen, Abstände, Spalten und Blöcke
- Absätze, Listen und Tabellen
- Textformatierung wie Satzformate und Trennung
- Linien, Bilder und andere Objekte
- und vieles mehr« [WikiPress 2006, 133]

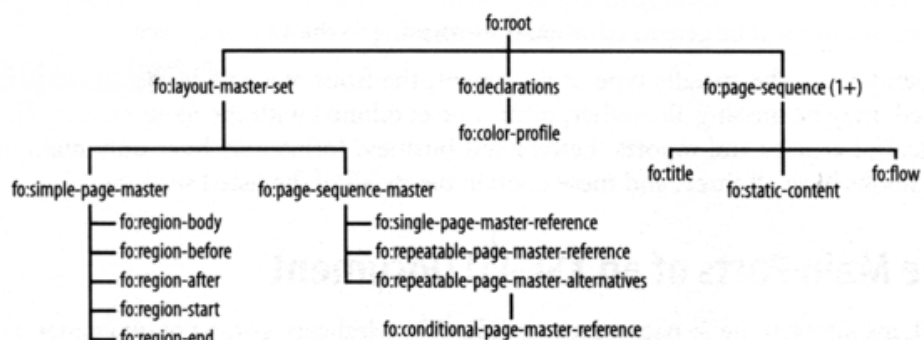
XSL-FO gehört, ebenso wie XSLT, der »XML-Familie« an und ist eine Empfehlung des W3C. Die korrekte Bezeichnung der Empfehlung lautet zwar »Extensible Stylesheet Language (XSL)«, was auf die gesamte XSL-Familie hindeutet, tatsächlich werden

aber nur die Formatting Objects, also XSL-FO, im Detail behandelt. Sowohl für XSLT als auch XPath existieren eigene W3C-Empfehlungen (siehe vorangegangene Abschnitte). Version 1.0 der Empfehlung für XSL-FO wurde im Oktober 2001 verabschiedet [W3C 2001a] und im Dezember 2006 durch Version 1.1 [W3C 2006a] erweitert. Im Rahmen dieser Diplomarbeit wird erstere Version besprochen und verwendet.

### 4.3.1 Aufbau von XSL-FO-Stylesheets

Die folgende Abbildung zeigt ein Organigramm der Grundbausteine bzw. Formatierungsobjekte, aus denen ein XSL-FO-Stylesheet aufgebaut sein kann.

Abbildung 4.2: Grundaufbau eines XSL-FO-Stylesheets



Quelle: [Pawson 2002, 32]

Jedes XSL-FO-Stylesheet besitzt das Wurzelement `<fo:root>`, welches ein Layout-Master-Set (`<fo:layout-master-set>`, ein Satz von Gestaltungsmuster-Vorlagen) sowie eine Page-Sequence (`<fo:page-sequence>`, eine Abfolge von Seiten) beinhaltet. Im Layout-Master-Set können Simple-Page-Master (`<fo:simple-page-master>`, Seitenvorlagen) und Page-Sequence-Master (`<fo:page-sequence-master>`, Vorlagen zur Seitenabfolge) definiert werden.

Die Page-Sequence (<fo:page-sequence>) enthält die eigentliche Seitenfolge, welche die Vorlagen aus dem Layout-Master-Set nutzt.

Im einfachsten Fall enthält ein XSL-FO-Stylesheet ein Layout-Master-Set mit *einem* Simple-Page-Master und *eine* Page-Sequence [Skulschus & Wiederstein 2005, 42]. Das Resultat wäre eine einzelne Seite, die auf der Vorlage des Simple-Page-Master beruht.

#### Beispiel 4.4: Grundaufbau eines XSL-FO-Stylesheets

```
<?xml version="1.0" encoding="UTF-8"?> ❶
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format"> ❷
  <fo:layout-master-set>
    <fo:simple-page-master master-name="Seite"> ❸
      <!-- Festlegungen zum Seiten-Layout -->
    </fo:simple-page-master>
  </fo:layout-master-set>
  <fo:page-sequence master-reference="Seite"> ❹
    <!-- Inhalte -->
  </fo:page-sequence>
</fo:root>
```

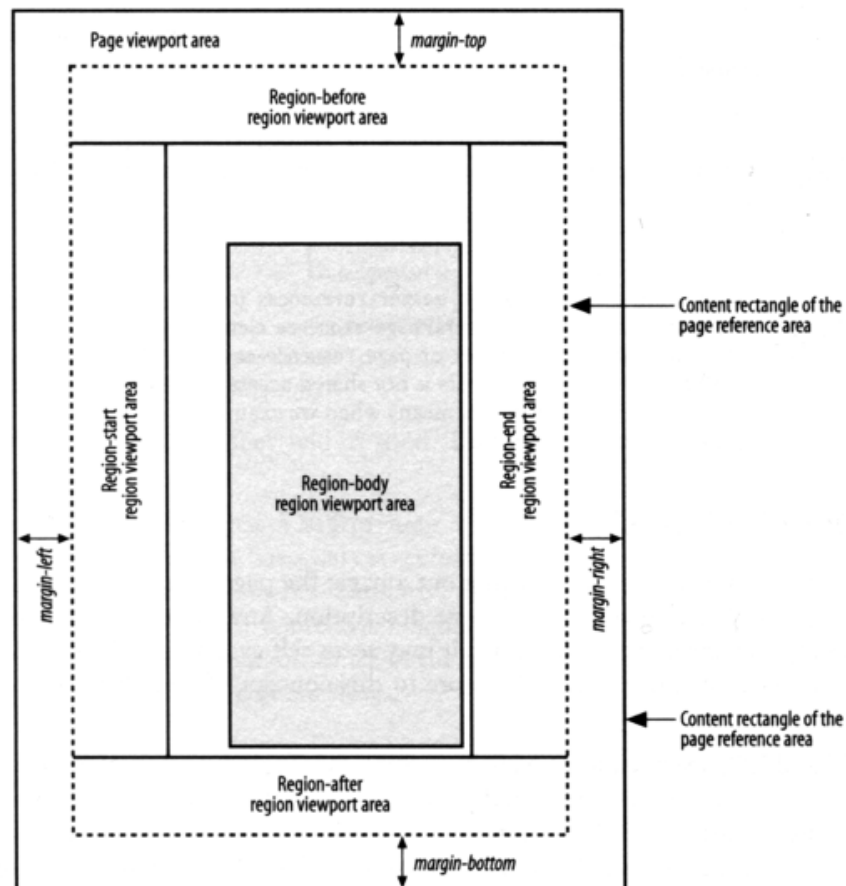
- ❶ XSL-FO-Stylesheets sind XML-Dokumente. Sie beginnen deshalb mit der XML-Deklaration.
- ❷ Das Wurzelement eines XSLT-Stylesheets ist <fo:root>. Es enthält eine Angabe zum Namensraum. Das Namensraum-Präfix für alle XSL-FO-Elemente ist fo:.
- ❸ Im Layout-Master-Set ist ein Simple-Page-Master mit dem Namen Seite definiert.
- ❹ Die Page-Sequence enthält eine Referenz Seite zum gleichnamigen Simple-Page-Master.

#### 4.3.2 Seitenvorlagen

Im Layout-Master-Set können ein oder beliebig viele Simple-Page-Masters, also Seitenvorlagen, angelegt werden. Diese

beschreiben unter anderem Seitenmaße, Ränder und druckbare Bereiche (Regionen).

Abbildung 4.3: Seitenmodell von XSL-FO



Quelle: [Pawson 2002, 34]

Die Abbildung zeigt das Konzept, mit dem bei XSL-FO Seiten beschrieben werden. Es basiert auf fünf Regionen, die insgesamt den druckbaren Bereich einer Seite bilden:

- Region-body, der Hauptbereich,
- Region-before, der Kopfbereich (optional),
- Region-after, der Fußbereich (optional),
- Region-start, der linke Bereich (optional) und

- Region-end, der rechte Bereich (optional).

Diese Anordnung der Regionen gilt selbstverständlich nur für Kulturkreise, in denen Zeilen von links nach rechts und Textblöcke bzw. Spalten von oben nach unten gelesen werden. XSL-FO bietet jedoch auch die Möglichkeit, über das Attribut `writing-mode` innerhalb bestimmter Formatierungsobjekte die Schreibrichtung und damit die Anordnung der Regionen zu verändern [Krüger 2006, 80 f.].

Im folgenden Beispiel wird ein Simple-Page-Master definiert, welcher einer Seitenvorlage für eine Seite dieser Diplomarbeit entspricht.

#### Beispiel 4.5: Simple-Page-Master

```

...
<fo:simple-page-master
  master-name="diplomvorlage"
  page-height="297mm"
  page-width="210mm"
  margin="25mm 30mm 35mm 55mm"> ❶
  <fo:region-body
    region-name="hauptbereich" margin-top="20mm" /> ❷
  <fo:region-before
    region-name="kopfbereich" extent="20mm" /> ❸
</fo:simple-page-master>
...

```

- ❶ Der Simple-Page-Master trägt den Namen `diplomvorlage`. Die Seitenhöhe ist mit 297 mm, die Seitenbreite mit 210 mm angegeben, was dem Format DIN A4 entspricht. Darüber hinaus sind durch das Attribut `margin` die Seitenränder definiert, oben 25 mm, rechts 30 mm, unten 35 mm und links 55 mm.
- ❷ Der Simple-Page-Master enthält einen Hauptbereich `<fo:region-body>`. Für diesen wurde der Name `hauptbereich` gewählt. Der Hauptbereich füllt den gesamten Raum inner-

halb der Seitenränder, abzüglich eines Einzugs um 20 mm von oben.

- ④ Der Kopfbereich `<fo:region-before>` schließt sich an den oberen Seitenrand an und wird vom linken und rechten Seitenrand seitlich begrenzt. Er hat eine Ausdehnung (hier: Höhe) von 20 mm.

### 4.3.3 Seitenfolgen

Seitenfolgen bzw. Page-Sequences greifen auf die Vorlagen zurück, welche im Layout-Master-Set hinterlegt sind. Hierbei bestehen zwei Möglichkeiten. Entweder nutzt die Page-Sequence ein Simple-Page-Master direkt oder aber sie greift auf ein Page-Sequence-Master zu, in welchem bereits eine Abfolge von Simple-Page-Masters definiert ist.

In einem Page-Sequence-Master können verschiedene Arten von Vorlagen für Seitenfolgen angelegt werden.

- Einzelseite

Hierbei besteht die Page-Sequence aus lediglich einer Seite. Der Page-Sequence-Master enthält das Element `<fo:single-page-master-reference>`. [Krüger 2006, 67]

- Seitenfolge mit gleicher Seitenvorlage

Die Page-Sequence besteht aus mehreren Seiten, die alle auf demselben Simple-Page-Master beruhen. Der Page-Sequence-Master enthält das Element `<fo:repeatable-page-master-reference>`. [Krüger 2006, 68]

- Seitenfolge mit mehreren, bedingten Seitenvorlagen

Die Page-Sequence besteht aus mehreren Seiten, welche auf verschiedenen Simple-Page-Masters beruhen, deren Auftreten

wiederum an bestimmte Bedingungen bzw. Ereignisse geknüpft ist. Diese Bedingungen geben beispielsweise an, ob es sich um eine gerade oder ungerade, eine erste oder letzte bzw. eine letzte, leere Seite innerhalb der Seitenfolge handelt. Der Page-Sequence-Master enthält das Element `<fo:repeatable-page-master-alternatives>`, und dieses beinhaltet mehrere `<fo:conditional-page-master-reference>`-Elemente. [Krüger 2006, 68 ff.]

»In textflussorientierten Anwendungen mit variablen Umfängen (Bücher, technische Handbücher) wird typischerweise eine Seitenfolgenvorlage benötigt, in der man unterschiedliche Seitenvorlagen für rechte und linke Seiten spezifiziert [...].« [Krüger 2006, 68]

Für die Schulungsunterlagen »Fair und erfolgreich prüfen mit textoptimierten Prüfungsaufgaben« trifft dieser Sachverhalt ebenfalls zu, da sie unter anderem in Form einer Broschüre publiziert werden sollen. Deshalb soll nun in einem Beispiel ein XSL-FO-Stylesheet betrachtet werden, in dem eine Seitenfolge mit mehreren, bedingten Seitenvorlagen enthalten ist.

Beispiel 4.6: Seitenfolge mit mehreren, bedingten Seitenvorlagen

```

...
<fo:layout-master-set>
  <fo:simple-page-master master-name="Erste"> ❶
    <!-- Festlegungen zum Seiten-Layout -->
  </fo:simple-page-master>
  <fo:simple-page-master master-name="Ungerade">
    <!-- Festlegungen zum Seiten-Layout -->
  </fo:simple-page-master>
  <fo:simple-page-master master-name="Gerade">
    <!-- Festlegungen zum Seiten-Layout -->
  </fo:simple-page-master>
  <!-- weitere Simple-Page-Masters -->
  <fo:page-sequence-master master-name="Kapitel"> ❷
    <fo:repeatable-page-master-alternatives>
      <fo:conditional-page-master-reference
        master-reference="Erste"
        page-position="first" />

```

```

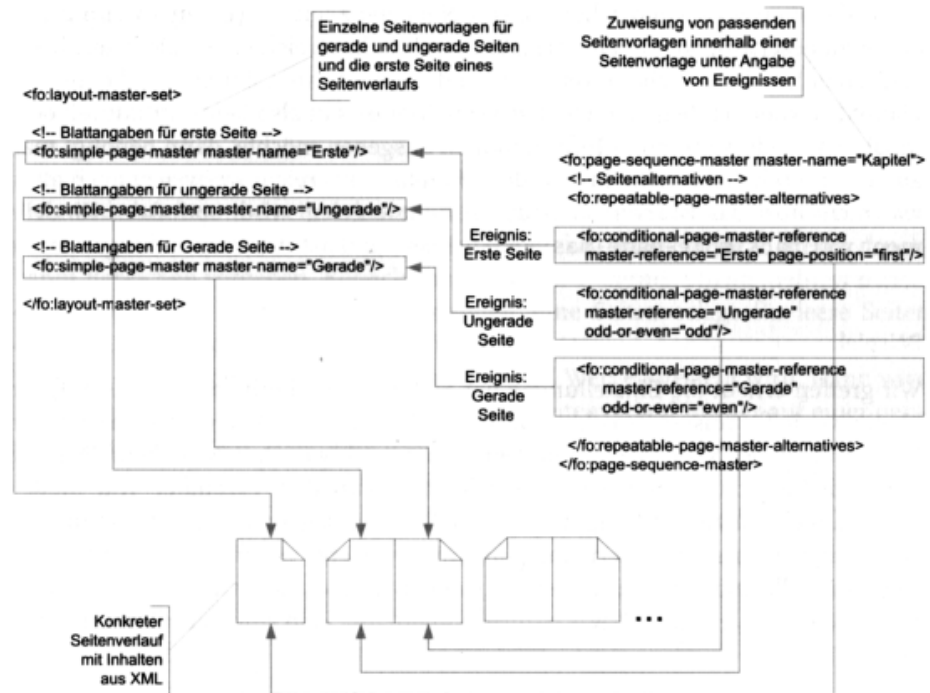
    <fo:conditional-page-master-reference
      master-reference="Ungerade"
      odd-or-even="odd" />
    <fo:conditional-page-master-reference
      master-reference="Gerade"
      odd-or-even="even" />
  </fo:repeatable-page-master-alternatives>
</fo:page-sequence-master />
<!-- weitere Page-Sequence-Masters -->
</fo:layout-master-set>
<fo:page-sequence master-reference="Kapitel"> ❸
  <!-- Inhalte -->
</fo:page-sequence>
<!-- weitere Page-Sequences -->
...

```

- ❶ Es sind verschiedene Simple-Page-Masters für erste, ungerade und gerade Seiten festgelegt.
- ❷ Dies ist ein Page-Sequence-Master für die Seitenfolge Kapitel. Er enthält Angaben über mehrere, bedingte Seitenvorlagen. In den verschiedenen <fo:conditional-page-master-reference>-Elementen wird jeweils einem Simple-Page-Master ein bestimmtes Ereignis zugewiesen. So gilt beispielsweise der Simple-Page-Master Gerade für gerade Seiten, da ihm durch das Attribut odd-or-even das Ereignis even (gerade Seite) zugewiesen wird.
- ❸ Die Page-Sequence ruft die Vorlagen aus dem Page-Sequence-Master Kapitel auf.

Die folgende Abbildung verdeutlicht noch einmal, wie die Vorlagen (Simple-Page-Masters und Page-Sequence-Masters) des eben gezeigten Beispiels zusammenwirken und die Seitenfolge bzw. Page-Sequence formen.

Abbildung 4.4: Vorlagen und Zuweisung



Quelle: [Skulschus & Wiederstein 2005, 61]

#### 4.3.4 Statische Inhalte und Textfluss

Die Page-Sequences nehmen alle Inhalte wie Texte, Listen, Tabellen usw. auf, die das Dokument, welches letztendlich aus dem XSL-FO-Stylesheet erzeugt wird, aufweisen soll. Innerhalb einer Page-Sequence werden die Inhalte den Bereichen (Regionen) zugewiesen, die in den Simple-Page-Masters festgelegt wurden.

Die Bereiche `<fo:region-before>`, `<fo:region-after>`, `<fo:region-start>` und `<fo:region-end>` können durch das Element `<fo:static-content>` so genannte statische Inhalte aufnehmen.

»Sie sind insofern als statisch anzusehen, als sie nicht über die Grenzen ihres Bereichs oder ihrer Seite hinaus fließen können.« [Krüger 2006, 71]

Statische Inhalte sind in den genannten Bereichen also nicht inhaltlich, sondern nur räumlich eingeschränkt. Die Bereiche `<fo:region-before>` und `<fo:region-after>` werden beispielsweise häufig für Inhalte wie Seitenzahlen oder Kolumnentitel verwendet.

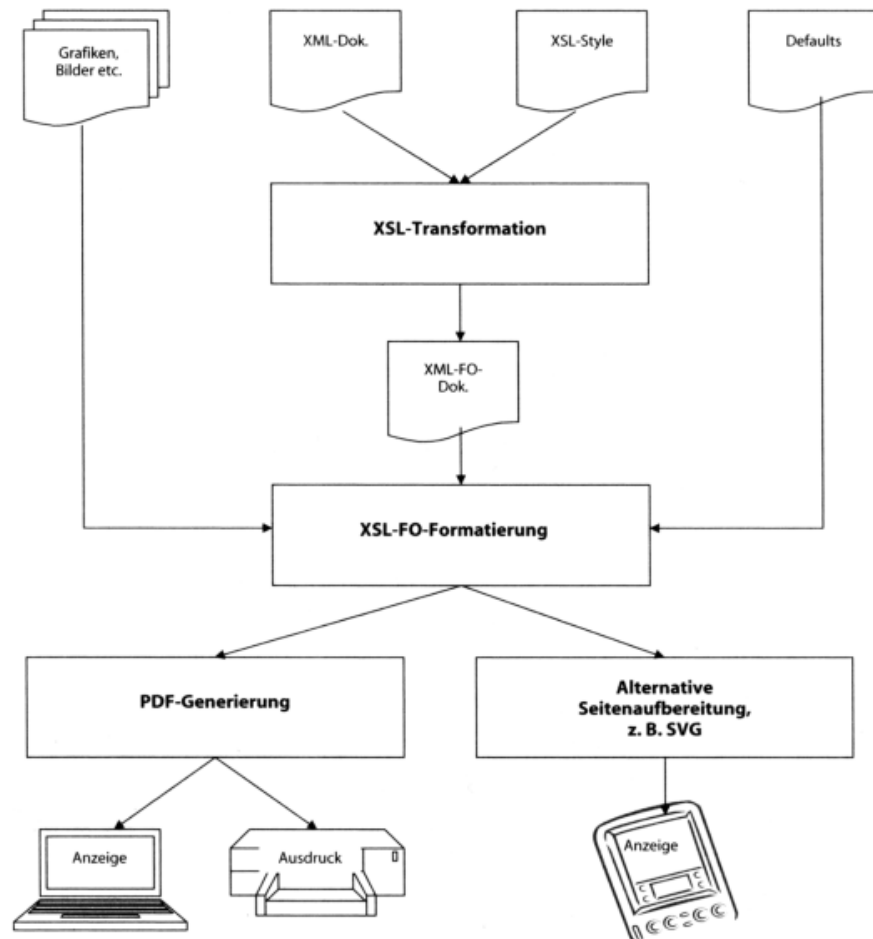
Der Bereich `<fo:region-body>` nimmt durch das Element `<fo:flow>` den Textfluss und damit den Hauptinhalt auf. Der Textfluss ist selbstverständlich nicht räumlich begrenzt, sondern kann über die Seitengrenzen hinaus fließen, d. h. Seitenumbrüche erzeugen. XSL-FO 1.0 lässt innerhalb einer Page-Sequence nur einen Textfluss zu. [Krüger 2006, 74]

#### 4.3.5 Verarbeitungsprozess

Es ist Sinn und Zweck von XSL-FO-Stylesheets, Inhalten eine Gestaltung zuzuweisen, um sie in einem bestimmten Ausgabeformat auf Seiten darzustellen. Nun ist es durchaus möglich, ein XSL-FO-Stylesheet direkt und samt Inhalten zu erstellen. In den meisten Fällen werden XSL-FO-Stylesheets jedoch nicht isoliert verwendet, sondern dienen dazu, im Zusammenspiel mit XSLT (und somit auch XPath) Inhalte zu gestalten, die in *anderen* XML-Dokumenten vorliegen. Und darin besteht auch das Konzept der XSL-Familie.

Hieraus ergibt sich ein Verarbeitungsprozess, wie er in der folgenden Abbildung dargestellt wird.

Abbildung 4.5: Verarbeitungsprozess



Quelle: [Krüger 2006, 9]

### 1. Transformation

Das XML-Quelldokument mit den Inhalten, die für die Darstellung auf Seiten aufbereitet werden sollen, wird zunächst durch einen XSLT-Prozessor transformiert. Das Stylesheet, das die Grundlage für diese Transformation bildet, enthält sowohl Elemente von XSLT als auch von XSL-FO. Das Ergebnis ist ein XSL-FO-Dokument, welches die gewünschten Inhalte aus dem ursprünglichen XML-Quelldokument sowie alle benötigten Vorlagen für deren Gestaltung enthält. Hier, im Rahmen des

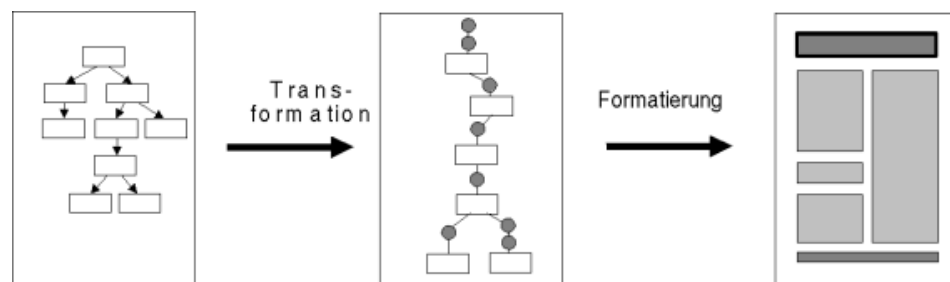
Verarbeitungsprozesses, wird das entstandene XSL-FO-Dokument auch FO-Baum genannt.

## 2. Formatierung

Nun soll das XSL-FO-Dokument bzw. der FO-Baum in ein Ausgabeformat wie PDF oder PostScript überführt werden. Diese Aufgabe übernimmt ein so genannter XSL-FO-Prozessor oder -Formatierer. Er bezieht auch Grafiken und ähnliche Objekte, auf die gegebenenfalls im FO-Baum verwiesen wird, in die Formatierung ein. Die Formatierung ist automatisierter Satz und Umbruch.

»Soweit Spezifikationen, beispielsweise für die Textausrichtung (linksbündig, Blocksatz, mittig zentriert), fehlen, verwendet der Formatierer die im Standard bestimmten Default-Werte (bei Textausrichtung ist dies linksbündig).« [Krüger 2006, 10]

Abbildung 4.6: Transformation und Formatierung



Quelle: [Wikipedia 2007a]

Einige XSL-FO-Prozessoren ermöglichen es auch, den gesamten Verarbeitungsprozess aus Sicht des Anwenders in einem Schritt vorzunehmen. Intern läuft dabei aber derselbe Prozess wie eben beschrieben ab.

## 5 DocBook

DocBook ist eine Auszeichnungssprache. Sie wurde ursprünglich als Standard für die Dokumentation von UNIX-Computersystemen entwickelt [Schraitle 2004, 85]. Die Entwicklung von DocBook begann im Jahre 1991. Damals wurde DocBook nach den Regeln der Standard Generalized Markup Language (SGML) formuliert; XML gab es noch nicht. Heute existiert DocBook auch als XML-Anwendung – DocBook-XML.

Seit dem Beginn seiner Entwicklung hat sich DocBook etablieren können und ist nun, insbesondere im Bereich von Open-Source-Projekten, ein verbreitetes und beliebtes Format:

»DocBook ist mittlerweile zu einem ›Quasi-Industriestandard‹ geworden, der von vielen Institutionen und Unternehmen genutzt wird; z. B. schreiben KDE, GNOME, das Linux Documentation Projekt, die SUSE LINUX AG u. a. ihre Dokumente in diesem Format.« [Schraitle 2004, 86]

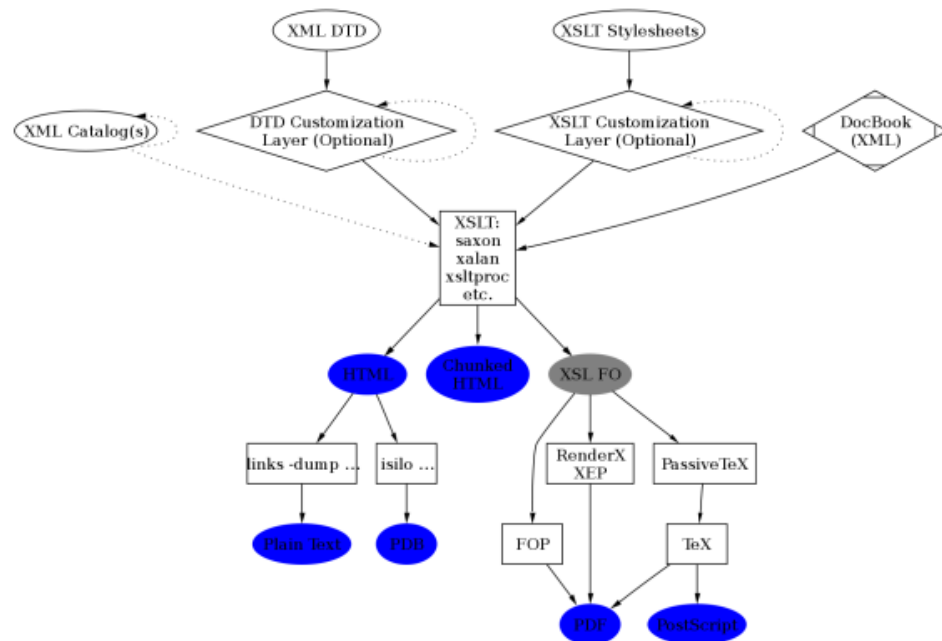
DocBook eignet sich insbesondere zur Erstellung von Publikationen wie Büchern oder Artikeln mit technischem Hintergrund, ist jedoch keinesfalls darauf beschränkt. Es existieren zahlreiche Technologien und Werkzeuge, um DocBook-Dokumente in verschiedenen Formaten und Medien, also medienübergreifend, zu publizieren. DocBook wird deshalb auch als Sammlung von Standards und Werkzeugen verstanden:

»DocBook is a collection of standards and tools for technical publishing.« [Stayton 2005, XV]

So gibt es zum Beispiel spezielle XSL-Stylesheets, die es ermöglichen, Inhalte aus DocBook-XML-Dokumenten in verschie-

denen Druck-, Bildschirm- und Onlineformaten auszugeben. Es ergibt sich folgendes Modell eines Publishing-Systems:

Abbildung 5.1: DocBook-XML-Publishing



Quelle: [Walsh 2006]

Ein solches DocBook-XML-Publishing-System bietet sich insbesondere an für

- umfangreiche Inhalte,
- stark strukturierte Inhalte,
- Inhalte, die automatisiert verarbeitet werden sollen sowie
- Inhalte, die in unterschiedlichen Formaten und Versionen ausgegeben werden sollen. [Stayton 2005, XVI]

Es handelt sich also um ein Konzept, das medienübergreifendes Publizieren ermöglicht.

## 5.1 DocBook-XML

Die Auszeichnungssprache DocBook-XML bildet den Kern des DocBook-Publishing-Konzeptes. Sie wird heute durch das DocBook Technical Committee in der Organization for the Advancement of Structured Information Standards (OASIS) gepflegt und stetig weiterentwickelt. OASIS ist ein internationales, nicht-gewinnorientiertes Konsortium, das die Entwicklung, Zusammenführung und Anpassung von E-Business-Standards vorantreibt [OASIS 2007].

Der Dokumenttyp DocBook-XML ist in einer DTD definiert. Diese liegt derzeit in der Version 4.5 vor, im Rahmen dieser Diplomarbeit kommt die Vorgängerversion 4.4 zum Einsatz. DocBook-XML ist neben der DTD, wenn auch noch nicht offiziell, unter anderem auch durch ein XML Schema definiert [DocBook TC 2007].

### 5.1.1 Merkmale

DocBook-XML ist eine XML-Anwendung. Es vereint daher auch die Merkmale von XML wie Plattformunabhängigkeit, Medienneutralität, Trennung von Inhalt und Layout usw. in sich.

DocBook-XML ist ein offenes und weit verbreitetes Format, und es ist sehr gut dokumentiert. Neben zahlreichen Büchern und Online-Quellen zum Thema gibt es sogar eine offizielle Dokumentation, »DocBook – The Definitive Guide«, die im HTML-Format über das Internet frei zugänglich ist [Walsh & Muellner 2006].

Mit DocBook-XML lassen sich Inhalte für Publikationsformen wie Artikel, Bücher oder ganze Buchreihen mit ihren typischen Bestandteilen strukturieren. Es werden hierfür entsprechende Elemente und Strukturen bereitgestellt. Da DocBook aus dem Bereich der technischen Dokumentation stammt, liegt hier auch ein Schwerpunkt, jedoch kann DocBook-XML auch für andere

Anwendungsbereiche genutzt und, falls notwendig, angepasst werden [Schraitle 2004, 86].

## 5.1.2 DocBook-XML-Beispiel

Das folgende Beispiel zeigt, wie ein typisches DocBook-XML-Dokument aufgebaut sein kann.

### Beispiel 5.1: Grundaufbau eines DocBook-XML-Dokumentes

```
<?xml version="1.0" encoding="UTF-8"?> ❶
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.4//EN"
"http://www.oasis-open.org/docbook/xml/4.4/docbookx.dtd"> ❷

<book> ❸
  <title>Medienübergreifendes Publizieren
    mit DocBook-XML und XSL</title>
  <chapter>
    <title>DocBook</title>
    <section>
      <title>DocBook-XML</title>
      <para>Die Auszeichnungssprache DocBook-XML bildet
        den Kern des DocBook-Publishing-Konzeptes.</para>
      <!-- weitere Absätze, Listen, Abbildungen usw. -->
    </section>
    <!-- weitere Abschnitte -->
  </chapter>
  <!-- weitere Kapitel -->
</book>
```

- ❶ Als XML-Anwendung beginnt das DocBook-XML-Dokument mit der XML-Deklaration.
- ❷ Dies ist die Dokumenttyp-Deklaration. Sie ordnet das Dokument mit dem Wurzelement `<book>` der DTD für DocBook-XML, Version 4.4, und damit auch dem entsprechenden Dokumenttyp zu.
- ❸ Das Wurzelement dieses Dokumentes heißt `book`. Es soll also ein Buch erstellt werden. Es beinhaltet u. a. Überschriften (`<title>`), Kapitel (`<chapter>`), Abschnitte (`<section>`) und Absätze (`<para>`).

## 5.2 DocBook-XSL

Bei DocBook-XSL handelt es sich um eine Reihe von XSL-Stylesheets, die speziell für die Verarbeitung von DocBook-XML-Dokumenten entwickelt wurden. Sie stellen einen weiteren wichtigen Bestandteil des DocBook-XML-Publishing-Konzeptes dar. DocBook-XSL-Stylesheets ermöglichen es, Inhalte aus DocBook-XML-Dokumenten medienübergreifend zu publizieren. Es lassen sich unter anderem folgende Formate erstellen:

- HTML bzw. XHTML
- XSL-FO, und daraus u. a. PDF und PostScript
- HTML Help
- Java Help
- Slides (»Folien« für Präsentationen)

Die DocBook-XSL-Stylesheets wurden von Norman Walsh geschrieben. Sie werden nunmehr im Rahmen eines Open-Source-Projektes bei SourceForge betreut und stetig weiterentwickelt und sind dort auch frei erhältlich [Stayton 2005, 3].

### 5.2.1 Anpassungsmöglichkeiten

Selbst ganz ohne Kenntnisse über XSL lassen sich die DocBook-XSL-Stylesheets bereits anwenden, um hochwertige Ausgabeformate zu erzeugen:

»If you know nothing about XSL, you can still use the stylesheets to generate high-quality output«. [Stayton 2005, XVII]

In den Stylesheets sind bereits Umwandlungs- und Layout-Vorlagen definiert, die in diesem Fall angewendet werden. Die besondere Stärke der DocBook-XSL-Stylesheets liegt jedoch in ihrer Anpassungsfähigkeit. Somit können Umwandlungsregeln und Layout-Vorgaben sehr flexibel an die jeweiligen Erfordernisse

und Bedürfnisse des Anwenders angepasst werden. Hierfür stehen folgende Möglichkeiten zur Verfügung:

- Erstellen eines Customization-Layers
- Setzen von Parameter-Werten
- Ändern von Attribute-Sets
- Vervollständigen von Platzhalter-Templates
- Anpassen von generiertem Text
- Ersetzen von Stylesheet-Templates
- Erstellen eigener Verarbeitungsanweisungen
- Anpassen von Titlepage-Templates [Stayton 2005, 101]

Das Anlegen eines so genannten Customization-Layers ist eine grundlegende Methode, um DocBook-XSL-Stylesheets anzupassen. Durch einen Customization-Layer können auch die meisten der oben genannten Anpassungsmethoden realisiert werden. Das Thema Customization-Layer ist Gegenstand des folgenden Abschnittes, und im weiteren Verlauf dieser Diplomarbeit sollen einige praktische Anwendungsbeispiele vorgestellt werden.

### 5.2.2 Customization-Layer

Ein Customization-Layer (eine Anpassungsebene) ist eine komfortable Methode, um Anpassungen an DocBook-XSL-Stylesheets vorzunehmen, ohne sie tatsächlich verändern zu müssen. Es handelt sich dabei um ein XSLT-Stylesheet, welches alle gewünschten Anpassungen enthält, die dann wiederum entsprechende Eigenschaften der DocBook-XSL-Stylesheets überlagern.

Die allgemeine Vorgehensweise lässt sich wie folgt zusammenfassen:

- Um einen bestimmten Bestandteil (z. B. ein Template) eines DocBook-XSL-Stylesheets anzupassen, wird er in den Customization-Layer übertragen und dort modifiziert.
- Bei der Verarbeitung von DocBook-XML-Dokumenten wird dann anstelle des Original-DocBook-XSL-Stylesheets der Customization Layer aufgerufen.
- Der Customization-Layer importiert das Original-DocBook-XSL-Stylesheet. Es dient somit nach wie vor als Grundlage für die Verarbeitung von DocBook-XML-Dokumenten.
- Anpassungen im Customization Layer ersetzen die entsprechenden Original-Bestandteile des DocBook-XSL-Stylesheets. Sind keine Anpassungen vorhanden, so wird das Original-Stylesheet unverändert angewendet.

Anhand des folgenden Beispiels soll die geschilderte Vorgehensweise demonstriert werden.

### Beispiel 5.2: Customization-Layer

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format"
  xmlns:l="http://docbook.sourceforge.net/xmlns/l10n/1.0" ❶
  version="1.0">

  <!-- DocBook Standard-Stylesheet importieren -->
  <xsl:import href="../fo/docbook.xsl" /> ❷

  <xsl:param name="paper.type">A4</xsl:param> ❸

  ...
  <xsl:attribute-set name="component.title.properties"> ❹
    <xsl:attribute name="font-size">18pt</xsl:attribute>
    <xsl:attribute name="text-align">left</xsl:attribute>
  </xsl:attribute-set>

  ...
  <xsl:template name="page.number.format"> ❺
    <!-- Inhalt des Templates -->
  </xsl:template>

  ...
</xsl:stylesheet>
```

- ❶ Der Customization Layer ist ein XSLT-Stylesheet. Er kann jedoch auch Elemente anderer Sprachen enthalten. Deshalb sind im Wurzelement verschiedene Namensräume angegeben: der von XSLT selbst, der von XSL-FO sowie einer für Sprachanpassungen (Localization).
- ❷ Über das Element `<xsl:import>` wird das DocBook-XSL-Stylesheet aufgerufen, das durch den Customization-Layer angepasst werden soll und gleichzeitig als Grundlage für die Verarbeitung von DocBook-XML-Dokumenten dient. Hier ist es das Stylesheet für XSL-FO.
- ❸ Dies ist ein Parameter mit dem Namen `paper.type` und dem Wert `A4`. Er ersetzt den gleichnamigen Parameter mit seinem dazugehörigen Standard-Wert im DocBook-XSL-Stylesheet.
- ❹ Dies ist ein Attribute-Set mit dem Namen `component.titles.properties`. Es wird mit dem gleichnamigen Attribute-Set des DocBook-XSL-Stylesheets zusammengefügt. Enthaltene gleichnamige Attribute des Customization-Layers überschreiben die des DocBook-XSL-Stylesheets, dort nicht vorhandene Attribute werden ergänzt.
- ❺ Dies ist ein Template mit dem Namen `page.number.format`. Es überschreibt das gleichnamige Template im DocBook-XSL-Stylesheet. Gäbe es dieses Template im Customization-Layer nicht, so würde das Original-Template des DocBook-XSL-Stylesheets verwendet werden. Dasselbe gilt auch für den zuvor genannten Parameter sowie für das Attribute-Set.

Wie bereits angesprochen, werden bei Verwendung eines Customization-Layers Anpassungen nicht direkt in den DocBook-XSL-Stylesheets vorgenommen, sondern separat – eben im Customization-Layer – abgelegt. Dies bietet einige Vorteile:

- Wird eine neue Version der DocBook-XSL-Stylesheets veröffentlicht, so müssen u. U. nur einige Änderungen im Customi-

zation-Layer vorgenommen werden, anstatt alle zuvor getroffenen Anpassungen direkt in die Stylesheets zu reintegrieren.

- Die DocBook-XSL-Stylesheets sind frei zugänglich. Möchte ein Anwender seine Anpassungen für die DocBook-XSL-Stylesheets veröffentlichen, so muss er lediglich seinen Customization-Layer, nicht aber den vollständigen Satz an DocBook-XSL-Stylesheets zur Verfügung stellen. [Stayton 2005, 102]
- Fehlerhafte Anpassungen können in einem Customization-Layer leicht rückgängig gemacht, ggf. gelöscht werden. Die korrekte Funktionsweise der DocBook-XSL-Stylesheets bleibt stets gewahrt.

## **II Praktische Umsetzung**

## 6 System- und Zielanalyse

Es soll eine Cross-Media-Publishing-Lösung erarbeitet werden, die das medienübergreifende Publizieren der Schulungsmaterialien »Fair und erfolgreich prüfen mit textoptimierten Prüfungsaufgaben« ermöglicht. Hierfür werden die fünf Schritte zur Realisierung von CMP-Systemen nach Fritsche zugrunde gelegt. Zuerst ist es notwendig, in einer System- und Zielanalyse verschiedene relevante Vorbedingungen zu analysieren sowie Anforderungen an die CMP-Lösung zu formulieren.

### 6.1 CMP-Lösung

#### Anforderungen an die CMP-Lösung

Die Anforderungen, die an die CMP-Lösung gestellt werden, wurden bereits in der Einführung dieser Diplomarbeit ausführlich formuliert. Die folgende Auflistung fasst noch einmal die wesentlichen Punkte zusammen.

- Es sollen Schulungsmaterialien medienübergreifend publiziert werden, und zwar in Form einer Druck- sowie einer Online-Variante.
- Die Möglichkeit, später weitere Publikationsformen erstellen zu können, soll gegeben sein.
- Es wird »Publizieren auf Knopfdruck« angestrebt.
- Änderungen und Ergänzungen sollen sich leicht integrieren lassen.
- Es wird eine anspruchsvolle Gestaltung gewünscht.

### Anwender der CMP-Lösung

Die CMP-Lösung soll im Rahmen dieser Diplomarbeit entwickelt werden. Durch das medienübergreifende Publizieren der Schulungsmaterialien wird sie im selben Rahmen auch bereits angewendet. Nach Abschluss der Diplomarbeit werden Mitarbeiter der Forschungsstelle die Anwender der CMP-Lösung sein.

### System- bzw. Software-Umgebung

Für die Entwicklung und Anwendung der CMP-Lösung stehen sowohl Windows- als auch Linux-Systeme, ausgestattet mit gängigen Texteditoren, Grafikbearbeitungswerkzeugen und Office-Anwendungen, zur Verfügung.

## 6.2 Schulungsmaterialien

### Inhalte der Schulungsmaterialien

Die Text-Inhalte für die Schulungsmaterialien werden durch die Autorin Dr. Susanne Wagner verfasst. Sie werden im Microsoft-Word-Format zur Verfügung gestellt. Zudem werden durch die Autorin eine Reihe an Abbildungen aus verschiedenen Quellen zusammengestellt.

### Gestaltung der Schulungsmaterialien

Im Rahmen der Diplomarbeit »Optimale Gestaltung von Schulungsmaterialien unter dem Aspekt des medienübergreifenden Publizierens« von Susanne Seyfarth wird ein Gestaltungskonzept für die verschiedenen Publikationsformen erstellt, in denen die Schulungsmaterialien erscheinen sollen. Dies beinhaltet Gestaltungsvorgaben für eine Druck- sowie für eine Online-Variante.

Darüber hinaus werden alle Abbildungen, die in den Schulungsmaterialien verwendet werden sollen, an die Gestaltungsvorgaben angepasst. Sie werden für die CMP-Lösung zunächst in bestmöglicher Qualität, d. h. in »verwendungszweck-freier« Form zur Verfügung gestellt.

## 7 Erarbeiten der Cross-Media-Publishing-Lösung

### 7.1 DocBook-XML-Publishing-Konzept als Basis

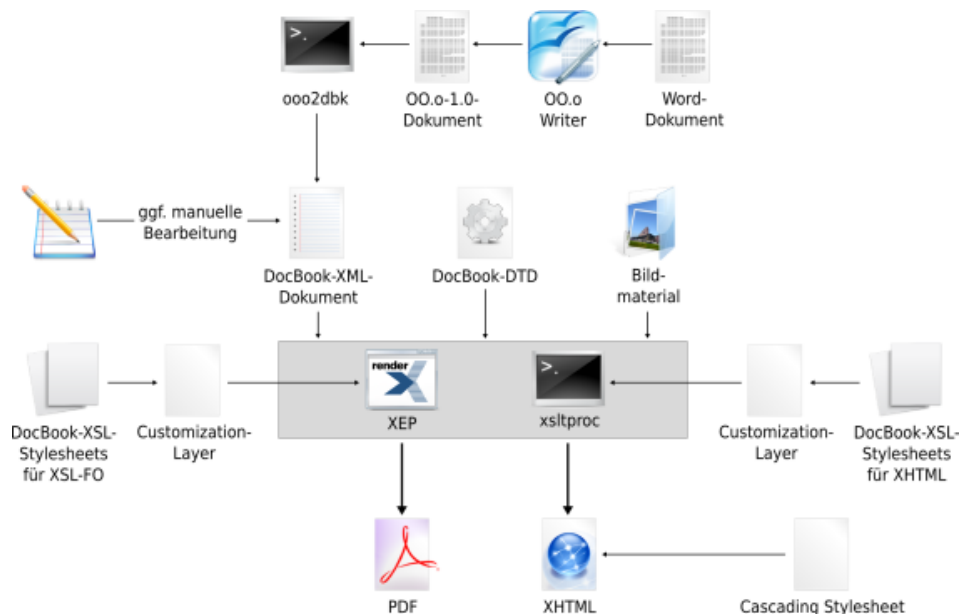
Für das medienübergreifende Publizieren der Schulungsmaterialien »Fair und erfolgreich prüfen mit textoptimierten Prüfungsaufgaben« soll eine Cross-Media-Publishing-Lösung zum Einsatz kommen, die auf dem DocBook-XML-Publishing-Konzept beruht. Dieses ermöglicht die Erstellung verschiedenster Ausgabeformate und darüber hinaus eine automatisierte Verarbeitung von Inhalten.

Die Schulungsmaterialien dokumentieren und vermitteln Wissen, und sie weisen mit Kapiteln und Abschnitten die typischen Strukturen von Büchern auf. Sie befinden sich damit genau im Anwendungsgebiet von DocBook-XML als einer Auszeichnungssprache zum Beschreiben und Strukturieren von Büchern im Bereich der Dokumentation. DocBook-XML wird also eingesetzt werden, um die Inhalte der Schulungsmaterialien zu beschreiben und zu strukturieren.

Die Schulungsmaterialien sollen als Broschüre in einer Druck-Variante sowie in einer Online-Variante veröffentlicht werden. Durch die DocBook-XSL-Stylesheets können die gebräuchlichen Formate PDF für den Druck bzw. XHTML für die Online-Veröffentlichung erstellt werden. Die DocBook-XSL-Stylesheet versprechen hierbei hochwertige Ergebnisse [Stayton 2005, XVII] und sehr flexible Anpassungsmöglichkeiten. Zur Umwandlung von DocBook-XML-Dokumenten in die verschiedenen Ausgabeformate sollen also die DocBook-XSL-Stylesheets angewendet werden.

Die folgende Abbildung zeigt eine schematische Darstellung der Cross-Media-Publishing-Lösung für die Schulungsmaterialien mit all ihren Komponenten.

Abbildung 7.1: Cross-Media-Publishing-Lösung für das medienübergreifende Publizieren der Schulungsmaterialien



## 7.2 Zielformat DocBook-XML

Das Textmaterial für die Schulungsmaterialien wird seitens der Autorin im Microsoft-Word-Format bereitgestellt. Dieses muss also erst einmal in die Auszeichnungssprache DocBook-XML überführt werden.

Eine Möglichkeit hierfür besteht darin, die DocBook-XML-Dokumente manuell zu erstellen und dabei Schritt für Schritt die Inhalte aus den Word-Dokumenten einzuarbeiten. Dies bietet den Vorteil, von Anfang an volle Kontrolle über die Erstellung der DocBook-XML-Dokumente zu haben. Der große Nachteil besteht im enormen Arbeitsaufwand. Zudem soll manuelle

Bearbeitung laut Zielstellung für die CMP-Lösung minimiert werden.

Eine sehr viel komfortablere Möglichkeit Word-Dokumente in DocBook-XML-Dokumente zu überführen bietet sich durch die Office-Suite OpenOffice.org sowie das Script ooo2dbk. Word-Dokumente lassen sich problemlos im OpenOffice.org Writer öffnen und dann als OpenOffice.org 1.0 Dokument (es wird tatsächlich dieses Format, nicht das neuere OpenDocument Text benötigt) abspeichern. Das Script ooo2dbk kann OpenOffice.org 1.0 Dokumente in vereinfachte DocBook-XML-Dokumente umwandeln. Voraussetzung für diese Methode ist, dass in den Word-Dokumenten (oder nachträglich in den OpenOffice.org 1.0 Dokumenten) Formatvorlagen auf die Inhalte angewendet wurden. Die Umwandlung von Word- in DocBook-XML-Dokumente wird im Kapitel 10 noch näher beschrieben.

### 7.3 Zielformat PDF

Mit Hilfe der entsprechenden DocBook-XSL-Stylesheets werden die DocBook-XML-Dokumente in XSL-FO-Dokumente umgewandelt. Hierbei wird auch ein Customization-Layer angewendet, der alle notwendigen Anpassungen enthält, um die gegebenen Layout- und Gestaltungsvorgaben für die Schulungsmaterialien zu realisieren. Dieses Thema wird im Kapitel 9, Umsetzen des Gestaltungskonzeptes, näher betrachtet.

Aus den XSL-FO-Dokumenten werden dann wiederum PDF-Dokumente erstellt. Hierfür wird ein leistungsfähiger XSL-FO-Prozessor benötigt. Zwar werden eine ganze Reihe solcher Prozessoren angeboten, doch unterstützen bis jetzt nur wenige die XSL-FO-Spezifikation in vollem Umfang [Stayton 2005, 9].

In dieser CMP-Lösung soll als XSL-FO-Prozessor RenderX XEP eingesetzt werden. Es handelt sich dabei um ein kommerzielles Produkt, das jedoch für die Erstellung der Schulungsmaterialien seitens des Herstellers mit einer »Academic License« kostenfrei zur Verfügung gestellt wird. Nach Herstellerangaben ist XEP mit den W3C-Spezifikationen für XSL-FO 1.0 konform.

XEP kann sowohl per Kommandozeile als auch über eine grafische Benutzeroberfläche bedient werden. Zudem ermöglicht es XEP, DocBook-XML-Dokumente in einem Schritt in PDF umzuwandeln – der vollständige Verarbeitungsprozess von DocBook-XML über XSL-FO nach PDF läuft intern ab.

## 7.4 Zielformat XHTML

Auch für die Erstellung von XHTML-Dokumenten, durch welche die Online-Variante der Schulungsmaterialien publiziert werden soll, werden die entsprechenden DocBook-XSL-Stylesheets sowie ein Customization-Layer für deren Anpassung angewendet. Auf dieses Thema wird ebenfalls im Kapitel 9 näher eingegangen.

Für die Umwandlung von DocBook-XML-Dokumenten in XHTML-Dokumente wird der freie XSLT-Prozessor `xseltproc` eingesetzt. Er wird als der schnellste unter den XSLT-Prozessoren angesehen und gilt als sehr standardkonform [Stayton 2005, 8].

Während die DocBook-XSL-Stylesheets für XSL-FO sämtliche Gestaltungsmerkmale für die Zieldokumente enthalten, so ist dies bei den Stylesheet für XHTML nicht der Fall. Sie regeln nur, welche Inhalte der DocBook-XML-Dokumente in XHTML-Dokumente umgewandelt werden sollen und auf welche Art und Weise. Die Gestaltung der XHTML-Dokumente wird hauptsächlich über Cascading Stylesheets (CSS) realisiert.

## 8 Einrichten der Cross-Media-Publishing-Lösung und Testlauf

Nachdem bereits eine System- und Zielanalyse vorgenommen und eine geeignete Cross-Media-Publishing-Lösung zum medienübergreifenden Publizieren der Schulungsmaterialien erarbeitet wurde, würde laut Fritsche nun die Übernahme von Gestaltungsvorgaben folgen. Dies entspricht bei dieser CMP-Lösung der Anpassung von DocBook-XSL-Stylesheets auf Grundlage des Gestaltungskonzeptes.

Es erscheint hier jedoch sinnvoller, das Einrichten der Cross-Media-Publishing-Lösung mit anschließendem Testlauf vorzuziehen. So können unter Verwendung eines DocBook-XML-Beispieldokumentes sowie der unveränderten DocBook-XSL-Stylesheets schon einmal PDF- und XHTML-Dokumente erstellt werden. Es ist dann ein direkter Vergleich der Resultate aus dem Testlauf mit dem Gestaltungskonzept der Schulungsmaterialien möglich. Notwendige Anpassungen der DocBook-XSL-Stylesheet können so zielgerichtet vorgenommen und die tatsächlichen Auswirkungen auf die Ausgabeformate genau festgestellt werden.

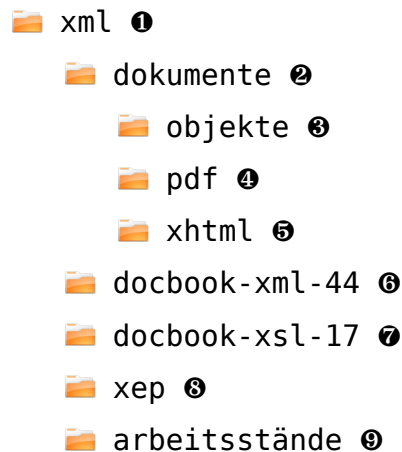
### 8.1 System-Umgebung

Die Cross-Media-Publishing-Lösung soll auf einem Ubuntu-Linux-System eingerichtet werden. Dieses Betriebssystem bietet den Vorteil, dass einige der Komponenten, die in der CMP-Lösung verwendet werden sollen, bereits enthalten sind oder sehr einfach installiert werden können. Grundsätzlich ließe sich die CMP-Lösung aber auch auf Windows oder Mac OS installieren.

Um Java-basierte Programme wie z. B. XEP ausführen zu können, wird eine aktuelle Java Virtual Machine benötigt.

Die folgende Abbildung zeigt die Ordnerstruktur, in der die einzelnen Komponenten der Cross-Media-Publishing-Lösung eingerichtet werden sollen.

Abbildung 8.1: Ordnerstruktur der CMP-Lösung



- ❶ Um effektiv und mit guter Übersicht arbeiten zu können, werden (fast) alle Komponenten in einem gemeinsamen Arbeitsordner abgelegt. Dieser kann sich an einer beliebigen Stelle im Dateisystem befinden. Für Linux-Systeme bietet sich z. B. das Home-Verzeichnis an.
- ❷ In diesem Ordner werden die DocBook-XML-Dokumente und der Customization-Layer abgelegt.
- ❸ Dieser Ordner enthält Objekte, z. B. Abbildungen.
- ❹ Hier werden PDF-Dokumente gespeichert.
- ❺ Hier werden XHTML-Dokumente gespeichert.
- ❻ Hier wird die DocBook-XML-DTD abgelegt.
- ❼ Hier werden die DocBook-XSL-Stylesheets abgelegt.
- ❽ In diesem Ordner wird XEP installiert.
- ❾ In diesem Ordner werden Arbeitsstände gespeichert. Dies ist sinnvoll, um ggf. auf vorherige Versionen zurückgreifen zu können.

## 8.2 DocBook-XML

Die DTD für DocBook-XML ist bei OASIS erhältlich und kann von der folgenden Adresse heruntergeladen werden:

```
http://www.oasis-open.org/docbook/  
xml/4.4/docbook-xml-4.4.zip
```

Das Archiv wird dann in das Verzeichnis docbook-xml-44 entpackt.

## 8.3 DocBook-XSL

Die DocBook-XSL-Stylesheets werden auf der DocBook Project Site bei SourceForge bereitgestellt. Unter der folgenden Adresse findet sich eine Liste der aktuellen Distributionen, u. a. auch die Version 1.70.0, die verwendet werden soll:

```
http://sourceforge.net/project/  
showfiles.php?group_id=21935#files
```

Nachdem das entsprechende Archiv heruntergeladen wurde, wird es in das Verzeichnis docbook-xsl-17 entpackt. Es bietet sich an, außerdem die passende Dokumentation der Stylesheets herunterzuladen und in dasselbe Verzeichnis zu kopieren.

## 8.4 RenderX XEP

Eine Kopie der Software XEP wurde unter einer Academic License durch den Hersteller RenderX bereitgestellt. XEP kann über einen Setup-Assistenten in das Verzeichnis xep im Arbeitsordner installiert werden.

XEP bringt als Grundausstattung an Schriften bereits die so genannten Base 14 Fonts mit. Für die Druckvariante der Schulungsmaterialien sollen aber auch andere Schriften – Futura und Garamond – verwendet werden [Seyfarth 2007]. Zu diesem Zweck

werden alle benötigten Schriftschnitte in das Unterverzeichnis `fonts` des `xep`-Ordners kopiert und anschließend in der Konfigurationsdatei `xep.xml` für XEP registriert.

In dieser Konfigurationsdatei findet sich ein Element namens `fonts`, in dem bereits einige Schriften verzeichnet sind. Hier werden Einträge für alle weiteren Schriften hinzugefügt, im folgenden Beispiel für die Schriftfamilie Futura. Für die Schriftfamilie Garamond wird ebenso vorgegangen.

### Beispiel 8.1: Schriften-Konfiguration in XEP

```
<font-family name="Futura" embed="true" subset="true"> ❶
  <font weight="normal"> ❷
    <font-data ttf="futura.ttf"/> ❸
  </font>
  <font weight="normal" style="italic">
    <font-data ttf="futura.iti.ttf"/>
  </font>
  <font weight="bold">
    <font-data ttf="FutuHv__.ttf"/>
  </font>
  <font weight="bold" style="italic">
    <font-data ttf="FutuHvIt.ttf"/>
  </font>
</font-family>
```

- ❶ Das Element `<font-family>` enthält als Attribut den Namen der Schriftfamilie, Futura. Entsprechende Einträge im Customization-Layer, z. B. `<xsl:attribute name="font-family">Futura</xsl:attribute>`, beziehen sich dann auf den hier beschriebenen Eintrag in der XEP-Konfigurationsdatei. Die Attribute `embed` bzw. `subset` geben an, dass hier die Schriften als Teilsatz in die zu erstellende PDF-Datei eingebettet werden sollen.
- ❷ Durch das Element `<font>` mit den Attributen `weight` bzw. `style` werden der Schriftfamilie nun die einzelnen Schriftschnitte (normal, fett, kursiv usw.) zugewiesen.
- ❸ Das Element `<font-data>` schließlich enthält den Dateinamen des jeweiligen Schriftschnittes.

## 8.5 xsltproc

xsltproc ist in Ubuntu Linux bereits enthalten und muss nicht erst installiert werden. Das Programm befindet sich allerdings nicht im Arbeitsordner `xml`.

## 8.6 ooo2dbk

Das Script `ooo2dbk` ist im Software-Archiv (Repository) »universe« von Ubuntu Linux verfügbar und kann über die Synaptic Paketverwaltung installiert werden. Das Script befindet sich dann ebenfalls außerhalb des Arbeitsordners `xml`.

## 8.7 Testlauf

Die Lösung zum medienübergreifenden Publizieren der Schulungsmaterialien ist nun eingerichtet und kann getestet werden. Für den Testlauf werden ein DocBook-XML-Beispieldokument sowie die DocBook-XSL-Stylesheets ohne jegliche Anpassung zugrunde gelegt.

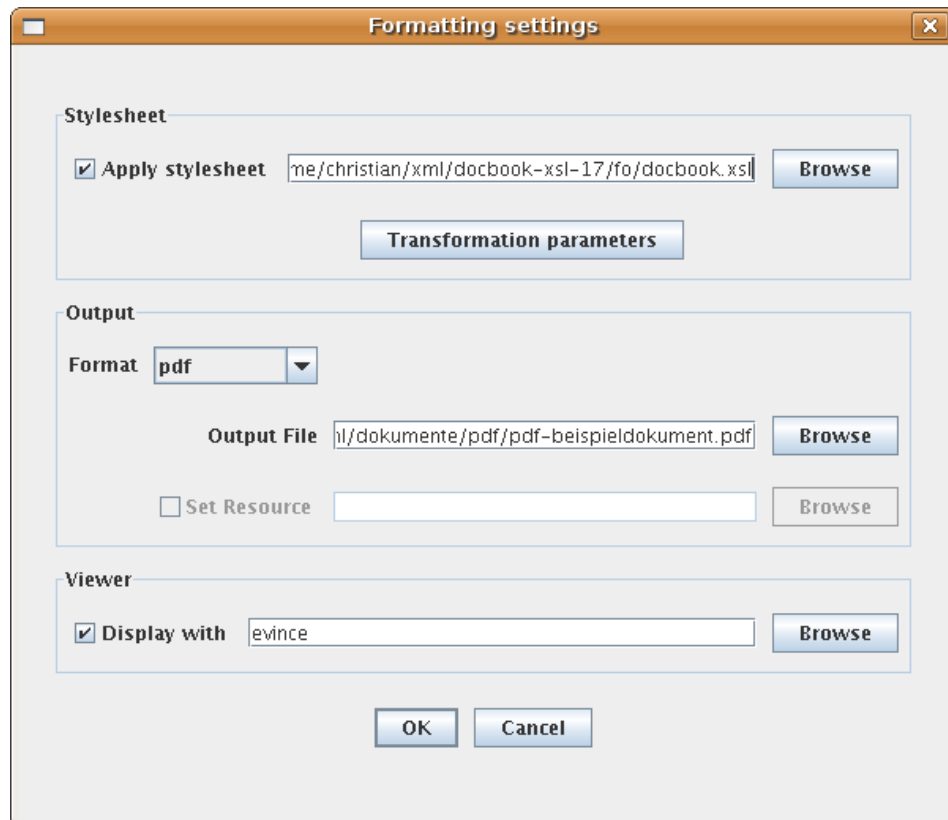
### Zielformat PDF

Um aus dem DocBook-XML-Beispieldokument ein PDF-Dokument zu erzeugen, wird XEP verwendet. Es werden folgende Schritte ausgeführt:

1. Über die Datei `x4u` wird die grafische Benutzeroberfläche von XEP aufgerufen. Es öffnet sich ein zunächst leeres Fenster.
2. Über den Menüpunkt `File → Open` wird nun das DocBook-XML-Beispieldokument geöffnet: `(Arbeitsordner)/dokumente/doc-book-xml-beispieldokument.xml`. Der Inhalt des Dokumentes wird anschließend angezeigt.

3. Es wird der Menüpunkt Formatting → Start ausgewählt. Daraufhin öffnet sich ein neues Fenster mit dem Titel Formatting Settings.

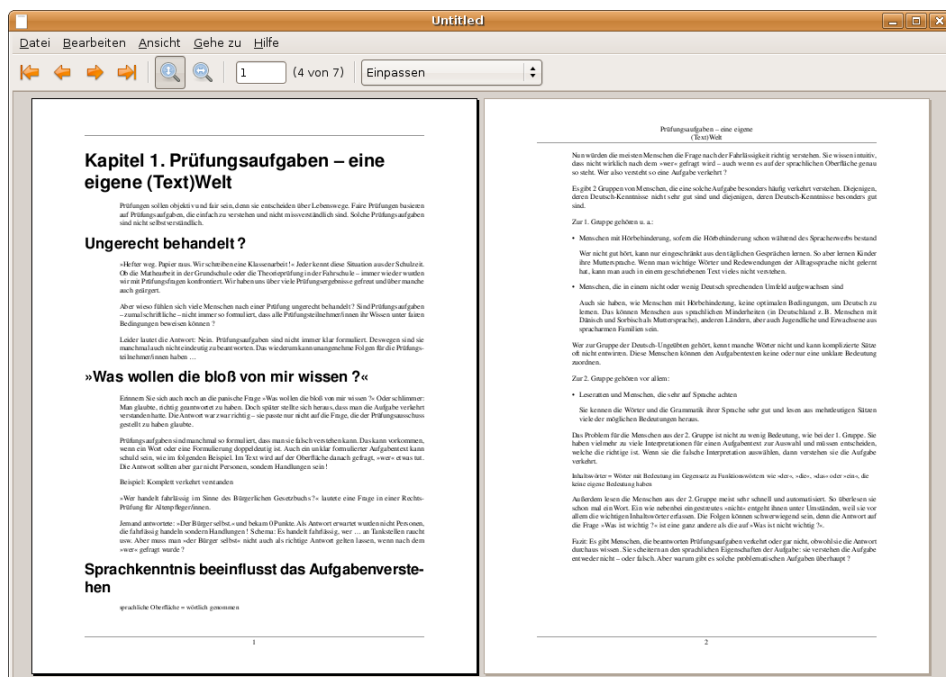
Abbildung 8.2: Formatting Settings in XEP



4. Im Abschnitt Stylesheet wird zunächst die Option Apply Stylesheet aktiviert und anschließend das DocBook-XSL-Stylesheet für XSL-FO ausgewählt: (Arbeitsordner)/docbook-xsl-17/fo/docbook.xsl.
5. Im Abschnitt Output wird das Format PDF ausgewählt und dann der Speicherort sowie ein Name für das zu erstellende PDF-Dokument angegeben: (Arbeitsordner)/dokumente/pdf/pdf-beispieldokument.pdf

6. Im Abschnitt Viewer wird die Option Display with aktiviert und evince als Dokumenten-Betrachter angegeben.
7. Schließlich wird durch die Schaltfläche OK der Verarbeitungsprozess gestartet. Während der Prozess läuft, werden einige Informationen angezeigt. Ist der Verarbeitungsprozess erfolgreich abgeschlossen, so wird das PDF-Dokument im Dokumentenbetrachter Evince geöffnet.

Abbildung 8.3: Ergebnis des Testlaufs für das Zielformat PDF



## Zielformat XHTML

Die DocBook-XSL-Stylesheets für XHTML bieten zwei verschiedene Möglichkeiten der Ausgabe. So kann ein DocBook-XML-Dokument entweder in ein einziges, zusammenhängendes XHTML-Dokument, oder aber in mehrere, kleinere XHTML-Dokumente umgewandelt werden. Letztere Methode wird bei

DocBook als Chunking (Zerstückeln) bezeichnet und bietet sich für umfangreiche Dokumente wie die Schulungsmaterialien an.

Um aus dem DocBook-XML-Beispieldokument XHTML-Dokumente zu erzeugen, wird xsltproc verwendet.

1. Es wird die Konsole für die Kommandozeile geöffnet und in den Arbeitsordner gewechselt.
2. Durch folgende Kommandozeile wird xsltproc ausgeführt:

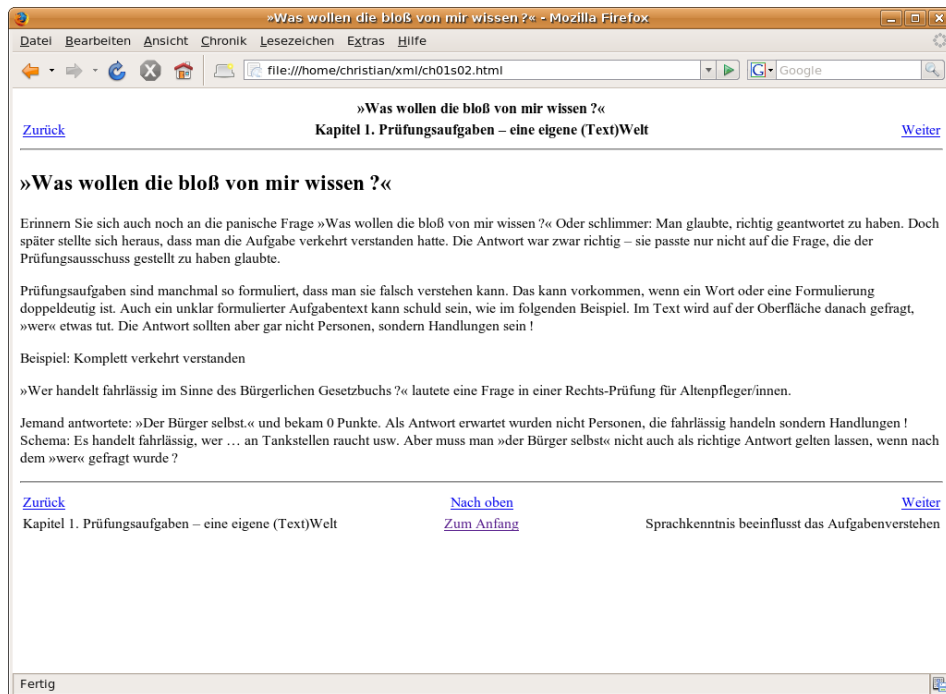
```
xsltproc
  docbook-xsl-17/xhtml/chunk.xsl
  dokumente/docbook-xml-beispieldokument.xml
```

Während der Verarbeitungsprozess läuft, werden einige Informationen angezeigt.

Anschließend befinden sich im Arbeitsordner einige XHTML-Dokumente. Dies ist eigentlich noch nicht der richtige Speicherort. Hierzu werden später noch Einstellungen im Customization-Layer vorgenommen werden müssen.

3. Die Datei `index.html` ist das Start-Dokument und kann nun in einem beliebigen Browser geöffnet werden.

Abbildung 8.4: Ergebnis des Testlaufs für das Zielformat XHTML



Im Testlauf konnten aus dem DocBook-XML-Beispieldokument bereits ein PDF-Dokument sowie eine Reihe von XHTML-Dokumenten erzeugt werden. Die Cross-Media-Publishing-Lösung ist damit erfolgreich eingerichtet.

## 9 Umsetzen des Gestaltungskonzeptes

Nachdem die Cross-Media-Publishing-Lösung erfolgreich eingerichtet wurde, besteht der nächste Schritt darin, das Gestaltungskonzept für die Schulungsmaterialien umzusetzen, welches im Rahmen der Diplomarbeit »Optimale Gestaltung von Schulungsmaterialien unter dem Aspekt des medienübergreifenden Publizierens« erarbeitet wurde. Hierzu müssen umfangreiche Anpassungen an den DocBook-XSL-Stylesheets vorgenommen werden, die für die Gestaltung der medienübergreifenden Publikationen zuständig sind. Für die Online-Variante der Schulungsmaterialien ist es zusätzlich erforderlich, ein Cascading Stylesheet anzulegen.

Es würde erheblich den gegebenen Umfang dieser Diplomarbeit überschreiten, sämtliche notwendigen Anpassungen und Einstellungen zu dokumentieren und zu erläutern. Sie können in den entsprechenden Dokumenten (z. B. den Customization-Layers), die sich auf der beigelegten CD mit Arbeitsmaterialien befinden, eingesehen und anhand der Anmerkungen in den Quelltexten nachvollzogen werden.

In diesem Kapitel sollen ausgewählte, wichtige Beispiele vorgestellt werden, wie sich Vorgaben des Gestaltungskonzeptes innerhalb der Cross-Media-Publishing-Lösung umsetzen lassen.

### 9.1 Finden der richtigen Anpassungsmöglichkeiten

Die DocBook-XSL-Stylesheets besitzen einen sehr großen Leistungsumfang, sind aber auch dementsprechend komplex aufgebaut. Bestehen nicht schon fortgeschrittene Kenntnisse über die Stylesheets, ist es unbedingt notwendig, auf geeignete Informationsquellen zurückzugreifen, um erfolgreich Anpassungen vor-

nehmen zu können. In diesem Zusammenhang müssen folgende Fragen beantwortet werden:

- An welcher Stelle muss eine Anpassung vorgenommen werden, um eine bestimmte Änderung in einem Ausgabedokument hervorzurufen?
- Wie muss die Anpassung erfolgen, um die Änderung im Ausgabedokument zu erreichen?

### DocBook-XSL-Doc

Die Benutzer-Dokumentation der DocBook-XSL-Stylesheets enthält Parameter-Referenzen für die verschiedenen unterstützten Ausgabeformate, darunter auch für XSL-FO und HTML (ebenso gültig für XHTML). Hier werden die Parameter und deren Einstellungsmöglichkeiten beschrieben, die für die Anpassung der Stylesheets zur Verfügung stehen. Es finden sich zudem auch Beschreibungen über einige Attribute-Sets.

### DocBook-XSL – The Complete Guide

»DocBook-XSL – The Complete Guide« von Bob Stayton ist ein »Must-Read« [DocBook Project 2005] für die Anwendung und Anpassung der DocBook-XSL-Stylesheets. Hier sind umfassende Anleitungen über das Publizieren von DocBook-Inhalten zusammengestellt.

»It provides the necessary documentation to realize the full potential of DocBook publishing.« [DocBook.org 2007]

### DocBook-Apps Mailing List

Eine sehr hilfreiche Informationsquelle über die DocBook-XSL-Stylesheets ist die Mailingliste DocBook-Apps [OASIS 2007a].

In deren Archiv finden sich Diskussionen sowie Fragen und Antworten zu ganz speziellen Problemstellungen bei DocBook-XSL und verwandten Themen. Eine Recherche in diesem Archiv kann somit auch Hinweise für eigene Problemstellungen liefern. Andernfalls ist es möglich, nach einer Anmeldung mit diesen Problemstellungen selbst an die Nutzergemeinde der Mailingliste heranzutreten. Auf diese Weise konnte ein Problem bezüglich der Gestaltung des Inhaltsverzeichnisses der Schulungsmaterialien (Druck-Variante) gelöst werden.

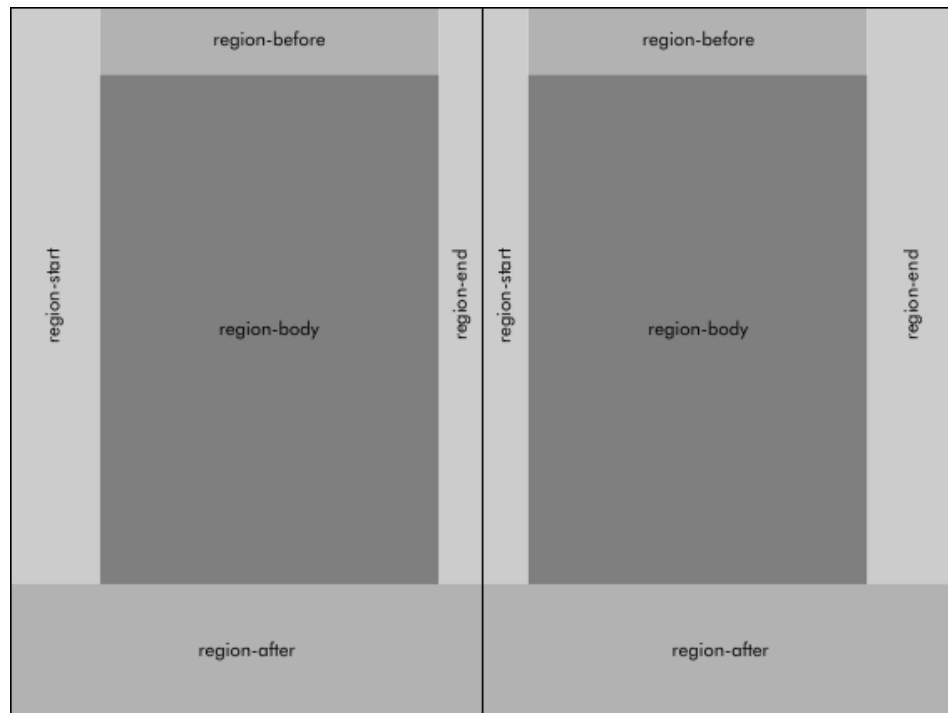
## 9.2 Druck-Variante der Schulungsmaterialien

### 9.2.1 Seiteneinrichtung

Ein wesentlicher Punkt bei der Gestaltung von Drucksachen ist die Seiteneinrichtung, d. h. Einrichtung des Satzspiegels, der Ränder usw. Es soll nun demonstriert werden, wie die Seiteneinrichtung für die Druck-Variante der Schulungsmaterialien vorgenommen wird. Das Gestaltungskonzept sieht folgendes Layout für eine Doppelseite im Hauptteil vor:



Abbildung 9.2: Aufteilung der Regionen



### Hintergrund-Grafiken

Leider ist es mit XSL-FO 1.0 nicht möglich, eine Hintergrund-Grafik festzulegen, die sich über alle Regionen und somit ggf. über die gesamte Seite erstrecken kann. Stattdessen müssen solche Grafiken »zerschnitten« und auf die einzelnen Regionen aufgeteilt werden. Nun sollen die Hintergrund-Grafiken bis an den Rand heran reichen. Um dies zu gewährleisten, müssen also die äußeren Regionen ebenfalls am Rand abschließen, was wiederum bedeutet, dass keine Seitenränder vorhanden sein dürfen.

### Marginalien

Um eine Spalte für Marginalien zu realisieren, liegt zunächst der Ansatz nahe, hierfür die Region-start bzw. Region-end zu benutzen. Marginalien orientieren sich aber sowohl bezüglich

ihres Inhaltes als auch bezüglich ihrer Anordnung an Inhalten im Haupt-Textfluss, der sich in der Region-body befindet. Aus Sicht des Autors lässt sich eine räumliche Beziehung von Textelementen zwischen zwei verschiedenen Regionen mit XSL-FO 1.0 nicht herstellen.

Die Spalte für Marginalien muss demzufolge mit im Haupt-Textfluss untergebracht werden. Hier erhalten alle Textelemente *außer* den Marginalien einen entsprechenden Einzug von links (start-indent) auf linken Seiten bzw. von rechts (end-indent) auf rechten Seiten. Auf diese Weise wird den Marginalien der benötigte Raum der Region zur Verfügung gestellt.

Das folgende Beispiel zeigt die notwendigen Einstellungen für eine linke Seite im Hauptteil der Schulungsmaterialien.

#### Beispiel 9.1: Seitendefinition für eine linke Seite

```
<xsl:template name="user.pagemasters"> ❶
  <fo:simple-page-master ❷
    master-name="body-even-user"
    page-width="{ $page.width }"
    page-height="{ $page.height }"
    margin-top="{ $page.margin.top }"
    margin-bottom="{ $page.margin.bottom }"
    margin-left="{ $page.margin.outer }"
    margin-right="{ $page.margin.inner }">
    <fo:region-body
      margin-bottom="{ $body.margin.bottom }"
      margin-top="{ $body.margin.top }"
      margin-left="{ $body.start.indent }"
      margin-right="- { $body.end.indent } + { $body.margin.inner }"
      column-count="{ $column.count.body }">
    </fo:region-body>
    <fo:region-before
      region-name="xsl-region-before-even"
      extent="{ $region.before.extent }"
      display-align="before"/>
    <fo:region-after
      region-name="xsl-region-after-even"
      extent="{ $region.after.extent }"
      display-align="after"
      background-image="{ $body-even-user.bg.region-after }"
      background-repeat="no-repeat"
      background-position-horizontal="center"
```

```

        background-position-vertical="bottom"
        precedence="true"/>
<fo:region-start
    region-name="xsl-region-start-even"
    extent="{ $region.outer.extent}"
    background-image="{ $body-even-user.bg.region-start}"
    background-repeat="no-repeat"
    background-position-horizontal="right"
    background-position-vertical="top"/>
<fo:region-end
    region-name="xsl-region-end-even"
    extent="{ $region.inner.extent}"
    background-image="{ $body-even-user.bg.region-end}"
    background-repeat="no-repeat"
    background-position-horizontal="left"
    background-position-vertical="top"/>
</fo:simple-page-master>
<!-- weitere Seitendefinitionen -->
</xsl:template>

```

- ❶ Im Customization-Layer werden im Template `user.pagemasters` alle benutzerdefinierten Seiteneinstellungen vorgenommen. Neben dem hier gezeigten Simple-Page-Master für eine linke Seite im Hauptteil befinden sich also noch sehr viel mehr Seiteneinstellungen in diesem Template.
- ❷ Der Simple-Page-Master enthält Einstellungen für alle Regionen der Seite. Viele Attribute erhalten hier ihre Werte nicht durch direkte Eingabe (z. B. `page-width="210mm"`), sondern über Parameter (z. B. `page-width="{ $page.width}"`), die bereits an anderer Stelle im Customization-Layer definiert wurden. Dies hat folgenden Grund: Einstellungen wie beispielsweise die Seitenbreite sind für viele Seitenvorlagen gleich. Sie werden deshalb nur einmal mit Hilfe eines Parameters festgelegt (z. B. `<xsl:param name="page.width">210mm</xsl:param>`), und auf diesen kann dann in verschiedenen Seitenvorlagen zurückgegriffen werden.

### 9.2.2 Anders verwirklicht: Das Inhaltsverzeichnis

Für das Inhaltsverzeichnis der Druck-Variante der Schulungsmaterialien ist im Gestaltungskonzept eine besondere Form vorge-

sehen. Es handelt sich hierbei um eine ausgestaltete Doppelseite. Anstelle der für Inhaltsverzeichnisse meist verwendeten hierarchischen und nummerierten Auflistung aller Kapitel- und Abschnittstitel sollen nur die einzelnen Kapiteltitel mit einer kurzen Zusammenfassung des jeweiligen Kapitelinhaltes erscheinen.

Abbildung 9.3: Layout für das Inhaltsverzeichnis

12 Inhaltsverzeichnis	Inhaltsverzeichnis 12
Inhaltsverzeichnis	
<p>1 Prüfungsaufgaben – eine eigene (Text)Welt</p> <p>15 Faire Prüfungsaufgaben sind einfach zu verstehen und nicht missverständlich. Solche Prüfungsaufgaben sind nicht selbstverständlich. Eine Einführung in die Sprache des Prüfers.</p> <p>2 Prüfungsaufgaben – Entstehung von Texten und Sprach-Barrieren</p> <p>25 Derselbe Aufgaben-Inhalt kann ganz verschieden ausgedrückt werden. Doch nicht jede inhaltlich korrekte Formulierung ist auch gut zu verstehen. Über Sprach-Barrieren in Prüfungsaufgaben.</p> <p>3 Basis der Textoptimierung – das Wissen vom Lesen und Verstehen</p> <p>35 Wenn eine Aufgabe nicht oder falsch verstanden wird, dann geht meist beim Lesen des Aufgabentextes etwas schief. Leseprobleme zeigen, wo Textoptimierung ansetzen muss.</p>	<p>4 Sprach-Barrieren in Prüfungstexten</p> <p>45 Sprach-Barrieren für Prüfungsaufgaben gibt es auf Wort-, Satz- und Textebene. Viele lassen sich grammatikalisch kategorisieren. Eine Sammlung.</p> <p>5 Textoptimierungs-Strategien</p> <p>53 Sprach-Barrieren zu finden ist eine Sache und bessere Formulierungen zu finden die andere Seite der Textoptimierungsmedaille. Strategien zur barrierearmen Formulierung von Aufgaben.</p> <p>6 Die Forschung hinter der Textoptimierung</p> <p>63 Die Erfahrung langjähriger Prüferinnen und Prüfer ebenso wie psycholinguistische Experimente zeigen: Textoptimierung funktioniert.</p> <p>Glossar 74</p> <p>Literaturverzeichnis 76</p> <p>Index 78</p>

Quelle: [Seyfarth 2007]

Mit Hilfe der DocBook-XSL-Stylesheets können Inhaltsverzeichnisse für Dokumente automatisch generiert werden. Es werden vielfältige Möglichkeiten geboten, diese Funktionalität sowohl gestalterisch als auch inhaltlich zu beeinflussen. Diese Möglichkeiten reichen jedoch bei weitem nicht aus, um die geforderte Form zu realisieren. Die Art und Weise, wie durch die DocBook-XSL-Stylesheets Inhaltsverzeichnisse erstellt werden, müsste

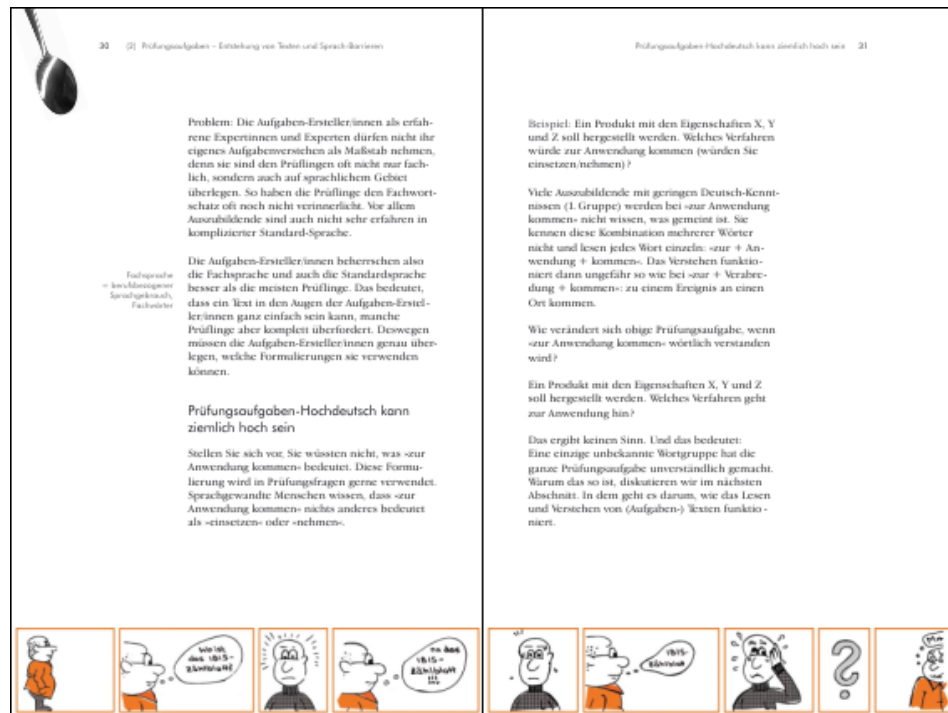
stark modifiziert werden. Dieser Aufwand ist im Rahmen dieser Diplomarbeit jedoch nicht zu bewältigen.

Es wurde daher entschieden, im Rahmen der CMP-Lösung doch auf »traditionelle« Inhaltsverzeichnisse und damit auf die standardmäßige Funktionalität der DocBook-XSL-Stylesheets zurückzugreifen. Die Kurz-Zusammenfassungen zu den jeweiligen Kapitelinhalten sollen in einem gesonderten Abschnitt zum Aufbau der Schulungsmaterialien innerhalb des Vorwortes eingebracht werden. Diese Lösung bietet gegenüber der ursprünglich vorgesehenen Form den Vorteil, dass sie gleichermaßen auf die Druck-Variante sowie auf die Online-Variante der Schulungsmaterialien angewendet werden kann.

### 9.2.3 Nicht verwirklicht: Seitenlayout mit Illustrationen

Eine erweiterte Fassung des Gestaltungskonzeptes für die Druck-Variante der Schulungsmaterialien sieht Illustrationen im Fußbereich der Seiten vor. Diese Illustrationen beziehen sich auf die Inhalte der jeweiligen Seiten und sollen auf eine amüsante Art und Weise Sachverhalte aus dem Text bildlich darstellen.

Abbildung 9.4: Layout mit Illustrationen



Quelle: [Seyfarth 2007]

Leider lässt sich dieser Gestaltungsentwurf im Rahmen der Cross-Media-Publishing-Lösung nicht verwirklichen. Er basiert auf einer Vorgehensweise, die mit dem automatisierten Satz und Umbruch nicht vereinbar ist: Die DocBook-XML-Dokumente müssten zunächst den gesamten Verarbeitungsprozess durchlaufen und als PDF-Dokumente vorliegen. Erst im Anschluss könnte analysiert werden, wie die Inhalte auf die einzelnen Seiten verteilt wurden, und dann wäre es möglich, entsprechend der jeweiligen Inhalte einer Seite eine dazu passende Illustration zu gestalten und diese zu platzieren. Dies wäre aber mit manuellen Eingriffen verbunden, die innerhalb der Cross-Media-Publishing-Lösung weder vorgesehen noch erwünscht sind.

Es wäre durchaus möglich gewesen, auch mit der CMP-Lösung Illustrationen wie im Gestaltungsentwurf zu realisieren, allerdings *ohne* Bezug zur entsprechenden Seite und deren Inhalt. Genau jener Bezug erschien aber als entscheidend, da die Illustrationen andernfalls ihren eigentlichen Zweck verloren hätten, nämlich Sachverhalte der aktuellen Seite darzustellen. Aus diesem Grund wurde entschieden, bei allen Dokumenten, die mit der CMP-Lösung erstellt werden, gänzlich auf die Illustrationen zu verzichten.

### 9.3 Online-Variante der Schulungsmaterialien

Die folgende Abbildung zeigt das Gestaltungskonzept für die Online-Variante der Schulungsmaterialien.

Abbildung 9.5: Layout für die Online-Variante



Quelle: [Seyfarth 2007]

Wichtige Merkmale sind hierbei die Navigationsleiste mit einer festen Position (d. h. kein Scrollen mit dem restlichen Inhalt) am unteren Seitenrand, das Breadcrumb-Menü im oberen Seitenbe-

reich sowie die Spalte für Marginalien rechts neben dem Haupt-Textfluss.

### 9.3.1 Einstellungen im Customization-Layer für XHTML

Um dieses Gestaltungskonzept zu realisieren, bedarf es zunächst einiger Anpassungen der DocBook-XSL-Stylesheets mit Hilfe eines Customization-Layers. Dabei werden noch keine gestalterischen Aspekte im eigentlichen Sinne (z. B. Schriftgrößen, Farben usw.) berührt. Vielmehr werden allgemeine Einstellungen für die XHTML-Ausgabe sowie Festlegungen zur Struktur der zu erstellenden XHTML-Dokumente getroffen. Der folgende Auszug aus dem Customization-Layer für XHTML enthält einige Beispiele.

#### Beispiel 9.2: Auszug aus dem Customization-Layer für XHTML

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet ...>
  <xsl:import href="../docbook-xsl-17/xhtml/chunk.xsl" /> ❶
  <xsl:param name="html.stylesheet"
    select="'aussehen/aussehen.css'" /> ❷
  <xsl:param name="base.dir"
    select="'/home/christian/xml/dokumente/html/'" /> ❸
  <!-- weitere Parameter -->
  <xsl:template name="chunk-element-content"> ❹
    ""
    <xsl:param name="content">
      <xsl:apply-imports/>
    </xsl:param>
    <html>
      <xsl:call-template name="html.head">...
      </xsl:call-template>
      <body>
        <xsl:call-template name="body.attributes"/>
        <xsl:call-template name="header.navigation">...
        </xsl:call-template>
        <xsl:call-template name="user.header.content"/> ❺
        <div>
          <xsl:attribute name="id">container</xsl:attribute> ❻
          <div>
            <xsl:attribute name="id">breadcrumb</xsl:attribute> ❼
            <xsl:call-template name="tw.generate.breadcrumb">
              <xsl:with-param name="current.node" select="."/>
            </xsl:call-template>
```

```

        </div>
        <xsl:copy-of select="$content"/>
    </div>
    <xsl:call-template name="footer.navigation">...
</xsl:call-template>
</body>
</html>
</xsl:template>
<!-- weitere Templates -->
</xsl:stylesheet>

```

- ❶ Importieren des DocBook-XSL-Stylesheets für XHTML mit Chunking
- ❷ Parameter, der den Ort des Cascading Stylesheets für die XHTML-Dokumente angibt
- ❸ Parameter, der den Speicherort für die zu erstellenden XHTML-Dokumente angibt
- ❹ Template zur Grundstruktur der zu erstellenden XHTML-Dokumente; aus den DocBook-XSL-Stylesheets übernommen und um einige Einstellungen erweitert
- ❺ Aufrufen des Templates `user.header.content` mit benutzerdefinierten Einstellungen für den Seitenkopf
- ❻ Einführen eines zusätzlichen `div`-Elementes als Container für Seiteninhalte
- ❼ Einführen eines zusätzlichen `div`-Elementes für das Breadcrumb-Menü und anschließendes Aufrufen des entsprechenden Templates

Das Ziel der Anpassungen im Customization-Layer ist es, die erforderliche Struktur und die gewünschten Inhalte für XHTML-Dokumente und somit für die Online-Variante der Schulungsmaterialien herzustellen.

### 9.3.2 Einstellungen im Cascading Stylesheet

Die tatsächliche Gestaltung der XHTML-Dokumente, d. h. deren Erscheinungsbild, wird schließlich über ein Cascading Stylesheet (CSS) realisiert.

In den XHTML-Dokumenten werden die Inhalte der Schulungsmaterialien durch einen sehr viel geringeren Umfang an Auszeichnungselementen repräsentiert als in den DocBook-XML-Quelldokumenten. Um dennoch differenzierte Gestaltungsmöglichkeiten zu gewährleisten, besitzen die XHTML-Elemente häufig zusätzlich ein `class`-Attribut, dessen jeweiliger Wert dem Namen eines DocBook-XML-Elementes entspricht, beispielsweise `<div class="titlepage">` oder `<div class="example">`. Auf diese Weise können z. B. Inhalte von Titelseiten oder Beispielen unterschiedlich gestaltet werden, obwohl sie beide in einem `div`-Element enthalten sind. Darüber hinaus sind auch unterschiedliche Gestaltungsmöglichkeiten in Abhängigkeit von übergeordneten Elementen gegeben:

### Beispiel 9.3: Unterschiedliche Gestaltung von `<h2>`-Elementen

```
div.chapter div.titlepage h2 { font-size: 22px; }
div.chapter div.section div.titlepage h2 { font-size: 19px; }
```

Eine besondere Forderung des Gestaltungskonzeptes für die Online-Variante der Schulungsmaterialien ist die am unteren Seitenrand feststehende Navigationsleiste. Sie soll dem Nutzer stets an derselben Position zur Verfügung stehen und nicht durch Scrollen in einen nicht sichtbaren Bereich der Seite geraten. Auf diese Weise soll das »Blättern« zwischen den einzelnen Seiten komfortabel gestaltet werden.

### Beispiel 9.4: CSS-Eigenschaften für die Navigationsleiste

```
.navfooter
{ position: fixed;
  bottom: 0;
  left: 0;
  margin: 0;
  padding: 0;
  width: 1000px;
  background: orangered;
  font-size: 12px;
  font-weight: normal;
  line-height: 120%; }
```

## 10 Integrieren der Inhalte

Im letzten Schritt der Umsetzung der Schulungsmaterialien geht es darum, alle Inhalte – Text- und Bildmaterial – in die Cross-Media-Publishing-Lösung zu integrieren. Es müssen DocBook-XML-Dokumente erstellt und das Bildmaterial hinterlegt werden.

### 10.1 DocBook-XML-Dokumente

#### 10.1.1 Auswählen geeigneter DocBook-XML-Elemente

Die Auszeichnungssprache DocBook-XML besitzt über 300 Elemente. Es wird aber nur ein Teil davon für die Schulungsmaterialien benötigt. Zunächst muss analysiert werden, wie die Schulungsmaterialien aufgebaut sein sollen und welche Textelemente im vorliegenden Textmaterial vorhanden sind. Anschließend können geeignete DocBook-XML-Elemente ausgewählt werden, durch welche die Inhalte der Schulungsmaterialien beschrieben und strukturiert werden sollen.

Um aus der großen Anzahl von DocBook-XML-Elementen die passenden herauszufinden und richtig anzuwenden, wird eine umfassende Referenz benötigt. Eine solche findet sich in »DocBook – The Definitive Guide« [Walsh & Muellner 2006]. Hier werden alle DocBook-XML-Elemente ausführlich beschrieben und ihre korrekte Anwendung anhand zahlreicher Beispiele demonstriert.

Die folgende Tabelle zeigt die aus dem vorliegenden Textmaterial ermittelten Bestandteile der Schulungsmaterialien (Vorlagen) und die jeweils geeigneten DocBook-XML-Elemente.

Tabelle 10.1. Übersicht aller Absatzformate

<b>Vorlagen</b>	<b>DocBook-XML-Elemente</b>
Buch	<book>
Bibliografische Angaben	<bookinfo>
Inhaltsverzeichnis	wird automatisch generiert
Vorwort	<preface>
Kapitel	<chapter>
Literaturverzeichnis	<bibliography>
Titel, Überschrift	<title>
Abschnitt	<section>
Absatz	<para>
Marginalie	<sidebar>
Liste, Aufzählung	<itemizedlist>
Liste, nummeriert	<orderedlist>
Liste, einfach	<simplelist>
Zusammenfassung	<abstract>
Literaturverweis, -verzeichnis- seintrag	<bibliomixed>
Zitate (eigenständig)	<blockquote>
Beispiele, ohne Titel	<informalexample>
Abbildungen	<mediaobject>

Tatsächlich wird aber eine wesentlich höhere Anzahl an DocBook-XML-Elementen verwendet als in der Tabelle gezeigt, denn die meisten der genannten Elemente enthalten wiederum weitere Elemente. Auf eine vollständige Aufzählung aller DocBook-XML-Elemente, die für die Schulungsmaterialien verwendet werden, wird an dieser Stelle verzichtet. Stattdessen sei hier auf die

DocBook-XML-Dokumente verwiesen, die auf der beiliegenden CD mit Arbeitsmaterialien enthalten sind. Dort kann direkt nachvollzogen werden, wie die einzelnen Elemente eingesetzt werden, um die Inhalte der Schulungsmaterialien zu beschreiben und zu strukturieren.

### 10.1.2 Erstellen von DocBook-XML-Dokumenten

Auf Grundlage der ausgewählten DocBook-XML-Elemente und mit Hilfe der Elemente-Referenz können nun die DocBook-XML-Dokumente für die Schulungsmaterialien erstellt werden. Das folgende Beispiel zeigt das Grundgerüst.

#### Beispiel 10.1: Grundgerüst der Schulungsmaterialien

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.4//EN"
  "http://www.oasis-open.org/docbook/xml/4.4/docbookx.dtd">

<book lang="de">
  <title>Fair und erfolgreich prüfen mit
    textoptimierten Prüfungsaufgaben</title>
  <bookinfo><!-- Bibliografische Angaben --></bookinfo>
  <preface><!-- Inhalt des Vorwortes --></preface>
  <chapter><!-- Inhalt des 1. Kapitels --></chapter>
  <chapter><!-- Inhalt des 2. Kapitels --></chapter>
  <chapter><!-- Inhalt des 3. Kapitels --></chapter>
  <chapter><!-- Inhalt des 4. Kapitels --></chapter>
  <chapter><!-- Inhalt des 5. Kapitels --></chapter>
  <chapter><!-- Inhalt des 6. Kapitels --></chapter>
  <bibliography><!-- Inhalt des Literaturverz. --></bibliography>
</book>
```

Prinzipiell können die Inhalte der Schulungsmaterialien in nur einem DocBook-XML-Dokument enthalten sein. Aus Gründen der Übersichtlichkeit und der einfacheren Handhabung bietet es sich jedoch an, hier Unterteilungen vorzunehmen: Das Vorwort, die einzelnen Kapitel sowie das Literaturverzeichnis werden jeweils in eigene Dokumente ausgelagert und können so separat bearbeitet werden. Das zuvor gezeigte Grundgerüst ändert sich dann wie folgt:

## Beispiel 10.2: Verändertes Grundgerüst

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.4//EN"
  "http://www.oasis-open.org/docbook/xml/4.4/docbookx.dtd"
[<!ENTITY vorwort SYSTEM "vorwort.docb.xml"> ❶
 <!ENTITY kapitel_1 SYSTEM "kapitel_1.docb.xml">
 <!ENTITY kapitel_2 SYSTEM "kapitel_2.docb.xml">
 <!ENTITY kapitel_3 SYSTEM "kapitel_3.docb.xml">
 <!ENTITY kapitel_4 SYSTEM "kapitel_4.docb.xml">
 <!ENTITY kapitel_5 SYSTEM "kapitel_5.docb.xml">
 <!ENTITY kapitel_6 SYSTEM "kapitel_6.docb.xml">
 <!ENTITY literatur SYSTEM "literatur.docb.xml">]>

<book lang="de">
  <title>Fair und erfolgreich prüfen mit
    textoptimierten Prüfungsaufgaben</title>
  <bookinfo><!-- Bibliografische Angaben --></bookinfo>
  &vorwort; ❷
  &kapitel_01;
  &kapitel_02;
  &kapitel_03;
  &kapitel_04;
  &kapitel_05;
  &kapitel_06;
  &literatur;
</book>
```

- ❶ Für jeden Bestandteil der Schulungsmaterialien, der ausgelagert werden soll, wird eine so genannte externe Entität deklariert. Dadurch wird eine Referenz zu einem entsprechenden externen Dokument hergestellt, in dem sich der ausgelagerte Inhalt befindet. So enthält beispielsweise das Dokument `vorwort.docb.xml` den Inhalt des Vorwortes.
- ❷ Dort, wo später der ausgelagerte Inhalt erscheinen soll, wird ein Verweis zur entsprechenden Entität angegeben. Beim Verarbeitungsprozess wird dieser durch den Inhalt des Dokumentes ersetzt, zu dem in der Entitäten-Deklaration eine Referenz hergestellt wurde. `&vorwort;` wird dementsprechend durch den Inhalt des Dokumentes `vorwort.docb.xml` ersetzt.

Auch wenn das Vorwort, die Kapitel und das Literaturverzeichnis in separaten Dokumenten vorliegen, muss stets beachtet werden,

dass sie Teil eines Ganzen sind und dementsprechend verarbeitet werden. Bei Referenzierung durch externe Entitäten dürfen die ausgelagerten Dokumente deshalb keine eigene Dokumenttyp-Deklaration haben. So besitzt beispielsweise das Dokument, welches das Vorwort beinhaltet, folgenden Inhalt:

Beispiel 10.3: Ausgelagertes Dokument `vorwort.docb.xml`

```
<?xml version="1.0" encoding="UTF-8" ?>
<preface>
  <title>Vorwort</title>
  <!-- Inhalt des Vorwortes -->
</preface>
```

### 10.1.3 Unterstützung durch ooo2dbk

Die Kapitel der Schulungsmaterialien liegen als Textmaterial in Form einzelner Microsoft-Word-Dokumente vor. Im Abschnitt 7.2 dieser Diplomarbeit wurde bereits eine Möglichkeit vorgestellt, wie diese Word-Dokumente mit Hilfe von OpenOffice.org und des Scriptes ooo2dbk in DocBook-XML-Dokumente umgewandelt werden können. Auch diese Vorgehensweise erfordert manuelle Nachbearbeitungen. Trotzdem kann gegenüber der vollständigen manuellen Erstellung von DocBook-XML-Dokumenten viel Zeit gespart werden.

Für jedes der Word-Dokumente werden folgende Schritte ausgeführt – hier am Beispiel des ersten Kapitels, für alle weiteren Kapitel analog:

1. Das Word-Dokument wird im OpenOffice.org Writer geöffnet.
2. Für Überschriften, den Textkörper und Listen werden, sofern nicht schon vorhanden, die entsprechenden Formatvorlagen zugewiesen. Daraus kann dann ein DocBook-XML-Dokument mit der richtigen Grundstruktur (geschachtelte Abschnitte, Überschriften, Absätze) erstellt werden, und auf dessen

Grundlage sind weitere Verfeinerungen möglich. Die Zuweisung speziellerer Formatvorlagen, z. B. für Beispiele, lohnt aus Sicht des Autors nicht, da sie nicht korrekt bzw. in der erforderlichen Form in DocBook-XML-Elemente umgewandelt werden.

3. Das Dokument wird nun als OpenOffice.org 1.0 Textdokument `kapitel_1.sxw` im Dokumenten-Ordner der CMP-Lösung abgespeichert.
4. Anschließend wird die Konsole für die Kommandozeile geöffnet und in den Dokumenten-Ordner der CMP-Lösung gewechselt.
5. Dort wird folgender Befehl ausgeführt:

```
ooo2dbk -a kapitel_1.sxw
```

Das Script erstellt im Dokumenten-Ordner das DocBook-XML-Dokument `kapitel_1.docb.xml`.

Im neu erstellten DocBook-XML-Dokument müssen anschließend noch einige manuelle Eingriffe vorgenommen werden.

1. Das Dokument wurde als DocBook Article erstellt. Es wird aber ein Kapitel, also ein Chapter, benötigt. Dementsprechend wird das Wurzelement `<article>` ganz einfach gegen `<chapter>` ausgetauscht.
2. Die Dokumenttyp-Deklaration sowie das Element `<articleinfo>` samt seiner Inhalte werden entfernt.
3. Vor Fragezeichen und Ausrufezeichen werden durch die numerische Zeichenreferenz `&#8201;` schmale Leerzeichen eingefügt. Ebenso werden die normalen Leerzeichen in Abkürzungen wie »z. B.«, »d. h.« oder »u. a.« durch schmale Leerzeichen ersetzt.

4. Auf der Grundlage des erstellten DocBook-XML-Dokumentes können nun entsprechende weitere Elemente eingefügt werden, z. B. für Abbildungen, Beispiele usw.

## 10.2 Bildmaterial

Um das Bildmaterial für die Schulungsmaterialien zu integrieren, müssen die Bilddateien innerhalb der Cross-Media-Publishing-Lösung hinterlegt und entsprechende Verweise und Meta-Informationen in den DocBook-XML-Dokumenten eingefügt werden.

### 10.2.1 Hinterlegen des Bildmaterials

Die Schulungsmaterialien sollen in den Ausgabeformaten PDF für die Druck-Variante und XHTML für die Online-Variante publiziert werden. Jedes dieser beiden Formate stellt andere Anforderungen an das Bildmaterial. Es geht also darum, das Bildmaterial mediengerecht aufzubereiten. Dies kann die Cross-Media-Publishing-Lösung leider nicht leisten, d. h. eine Aufbereitung muss bereits im Voraus stattfinden.

Alle Abbildungen für die Druck-Variante der Schulungsmaterialien werden im Objekte-Ordner der CMP-Lösung hinterlegt. Für die Online-Variante ist dies nicht sinnvoll, da sie später an anderer Stelle veröffentlicht werden wird, z. B. auf einem Webserver, wo nicht die Ordnerstruktur der CMP-Lösung zur Verfügung steht. Deshalb werden Abbildungen für die Online-Variante in einem Unterordner `bilder` innerhalb des Ordners für die XHTML-Dokumente abgelegt. Wird dann der `xhtml`-Ordner veröffentlicht, so sind die Abbildungen gleich enthalten.

## 10.2.2 Einfügen in DocBook-XML-Dokumente

Für jede Abbildung, die in den Schulungsmaterialien erscheinen soll, wird an entsprechender Stelle im DocBook-XML-Dokument ein Element `<mediaobject>` eingefügt. Dieses enthält Meta-Informationen (Verweis auf eine Bilddatei, Dateiformat, Angaben zu Abmessungen usw.) für verschiedene, alternative Medienobjekte, die in Abhängigkeit des gewünschten Zielformates bzw. Zielmediums verwendet werden.

Das folgende Beispiel zeigt die Anwendung eines `<mediaobject>`-Elementes in den DocBook-XML-Dokumenten der Schulungsmaterialien.

### Beispiel 10.4: `<mediaobject>`-Element

```
<mediaobject>
  <imageobject role="pdf"> ❶
    <imagedata
      width="100%"
      scalefit="1"
      align="center"
      fileref="objekte/fixation_sakkade.pdf"
      format="PDF"/>
    </imageobject>
    <imageobject role="xhtml"> ❷
      <imagedata
        width="100%"
        scalefit="1"
        align="center"
        fileref="bilder/fixation_sakkade.jpg"
        format="JPEG"/>
      </imageobject>
    <textobject> ❸
      <phrase>Grafik »Lesen: Wechsel von
        Fixation und Sakkade«</phrase>
    </textobject>
  </mediaobject>
```

- ❶ Hier sind Informationen für eine Abbildung enthalten, die im Zielformat PDF, also in der Druck-Variante der Schulungsmaterialien, verwendet werden soll. Dies wird durch das Attribut `role="pdf"` kenntlich gemacht. Die Auswertung von `role`-Attributen und somit die Auswahl der erforderlichen

Abbildung wird beim Verarbeitungsprozess anhand der DocBook-XSL-Stylesheets durchgeführt.

- ② In diesem Element mit dem Attribut `role="xhtml"` hingegen sind Informationen für eine Abbildung enthalten, die im Zielformat XHTML, also in der Online-Variante der Schulungsmaterialien, verwendet werden soll.
- ③ Die Angaben in diesem Element werden für Alternativ-Beschreibungen bei Abbildungen im Zielformat XHTML verwendet.

## 11 Publizieren auf Knopfdruck

Im zweiten Teil dieser Diplomarbeit wurde die Entwicklung einer Cross-Media-Publishing-Lösung dokumentiert, mit der die Schulungsunterlagen »Fair und erfolgreich prüfen mit textoptimierten Prüfungsaufgaben« in einer Druck-Variante sowie in einer Online-Variante publiziert werden können. Die Cross-Media-Publishing-Lösung ist nun einsatzbereit.

Wie bereits im Abschnitt 8.7 demonstriert, können die gewünschten Zielformate erstellt werden:

- Um die Druck-Variante im Format PDF zu erzeugen, werden dem XSL-FO-Prozessor XEP das DocBook-XML-Dokument `buch.docb.xml` sowie der Customization-Layer für XSL-FO, `custom-layer-fo.xsl`, aus dem Dokumenten-Ordner der CMP-Lösung übergeben.
- Um die Online-Variante im Format XHTML zu erzeugen, werden dem XSLT-Prozessor `xsltproc` das DocBook-XML-Dokument `buch.docb.xml` sowie der Customization-Layer für XHTML, `custom-layer-xhtml.xsl`, übergeben.

Nach jeweils einem kurzen Verarbeitungsprozess liegen die fertigen medienübergreifenden Publikationen vor.

Die Vorgehensweise kann aber noch mehr vereinfacht werden, um die Forderung nach »Publizieren auf Knopfdruck« bestmöglich zu erfüllen. Die Linux-Umgebung, in welcher die Cross-Media-Publishing-Lösung eingerichtet wurde, stellt das Werkzeug »make« zur Verfügung. In einem so genannten Makefile können verschiedene Ausgabeziele definiert und diesen jeweils komplexe Befehle zugeordnet werden.

»A Makefile is like a script or a batch file, but it can have several make ›targets‹ to run different command sequences from the same file.« [Stayton 2005, 35]

Das folgende Beispiel zeigt ein Makefile, das die Erstellung der medienübergreifenden Publikationen in der CMP-Lösung weiter vereinfacht.

### Beispiel 11.1: Makefile

```
# PDF erstellen
pdf: ❶
    /home/christian/xml/xep/xep
    -xml /home/christian/xml/dokumente/buch.xml
    -xsl /home/christian/xml/dokumente/custom-layer-fo.xsl
    -pdf /home/christian/xml/dokumente/pdf/buch.pdf ❷

# XHTML erstellen
xhtml: ❸
    xsltproc
    /home/christian/xml/dokumente/custom-layer-xhtml.xsl
    /home/christian/xml/dokumente/buch.xml ❹
```

- ❶ Hier wird das Ausgabeziel pdf festgelegt.
- ❷ Mit diesem Befehl wird XEP angesprochen, um die Druck-Variante der Schulungsmaterialien zu erstellen. Dieser Befehl wird hier dem Ausgabeziel pdf zugeordnet.
- ❸ Hier wird das Ausgabeziel xhtml festgelegt.
- ❹ Mit diesem Befehl wird xsltproc angesprochen, um die Online-Variante der Schulungsmaterialien zu erstellen. Er wird dem Ausgabeziel xhtml zugeordnet.

Dieses Makefile wird unter dem Dateinamen makefile im Arbeitsordner xml der CMP-Lösung abgelegt. Anschließend wird die Konsole für die Kommandozeile geöffnet und in das Arbeitsverzeichnis gewechselt. Dort kann das Makefile wie folgt ausgeführt werden:

- `make pdf`, um die Druck-Variante der Schulungsmaterialien zu erzeugen,
- `make xhtml`, um die Online-Variante der Schulungsmaterialien zu erzeugen oder
- `make pdf xhtml`, um beide Varianten zu erzeugen.

Anhand eines ganz einfachen Befehls können somit vollständige medienübergreifende Publikationen erzeugt werden – Publizieren auf Knopfdruck.

## 12 Zusammenfassung

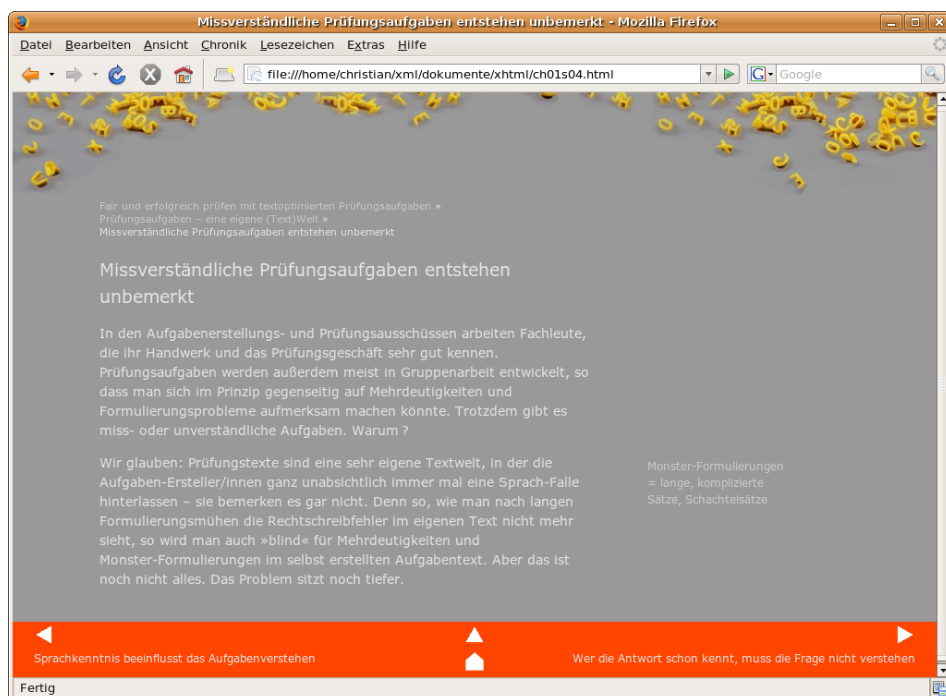
Im Rahmen der vorliegenden Diplomarbeit sollte eine technische Möglichkeit entwickelt werden, Schulungsmaterialien zum Thema »Fair und erfolgreich prüfen mit textoptimierten Prüfungsaufgaben« als medienübergreifende Publikationen zu realisieren. Dabei war »Publizieren auf Knopfdruck«, d. h. die automatisierte Verarbeitung von Inhalten, eine zentrale Forderung. Mit XML und XSL wurden zwei grundlegende Technologien vorgestellt, die medienübergreifendes Publizieren ermöglichen.

Das DocBook-XML-Publishing-Konzept setzt auf diese beiden Technologien. Es bietet umfassende Möglichkeiten, Publikationen im Bereich der Dokumentation für zahlreiche unterschiedliche Zielformate bzw. -medien zu erstellen. Es ist bewährt und darüber hinaus hervorragend dokumentiert. Aus diesem Grunde wurde für die Realisierung der Schulungsmaterialien eine Cross-Media-Publishing-Lösung auf der Grundlage des DocBook-XML-Publishing-Konzeptes entwickelt.

Die Auszeichnungssprache und XML-Anwendung DocBook-XML erwies sich als Basis zur Beschreibung und Strukturierung der Inhalte der Schulungsmaterialien als sehr geeignet. Für alle Inhalte stehen passende DocBook-XML-Elemente zur Verfügung.

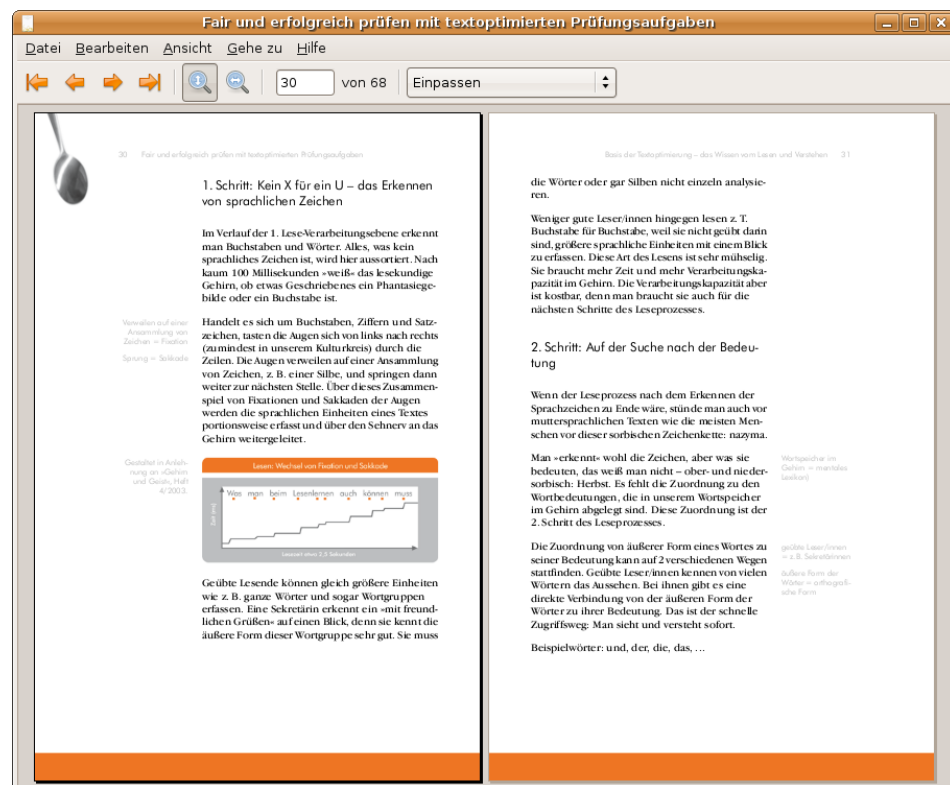
Auf die Gestaltung der Schulungsmaterialien wurde besonderer Wert gelegt. Es wurde versucht, mit Hilfe der DocBook-XSL-Stylesheets und ihrer umfangreichen Anpassungsmöglichkeiten das vorgesehene Gestaltungskonzept [siehe Seyfarth 2007] in der bestmöglichen Weise umzusetzen. Für die Online-Ausgabe der Schulungsmaterialien war dies ohne Einschränkungen möglich.

Abbildung 12.1: Online-Ausgabe der Schulungsmaterialien



Bei der Druckausgabe konnte das Gestaltungskonzept sehr weitgehend, jedoch mit einigen Einschränkungen realisiert werden. Hinsichtlich des Layouts des Inhaltsverzeichnisses musste auf eine andere Lösung zurückgegriffen werden. Der erweiterte Gestaltungsentwurf mit Illustrationen ließ sich nicht umsetzen.

Abbildung 12.2: Druckausgabe der Schulungsmaterialien



Mit Hilfe der Cross-Media-Publishing-Lösung konnten die Anforderungen, welche an die technische Umsetzung der Schulungsmaterialien gestellt wurden, erfüllt werden. Die Schulungsmaterialien wurden in zwei Publikationsformen, einer Druckausgabe im Format PDF sowie einer Online-Ausgabe im Format XHTML, erfolgreich erstellt. Deren Inhalte können auf einfache Weise erweitert und verändert werden. Durch einen kurzen Eingabe-Befehl – »auf Knopfdruck« – ist es möglich, innerhalb von Sekunden die medienübergreifenden Publikationen neu zu erstellen.

Die Cross-Media-Publishing-Lösung ist dank der verwendeten Technologien beliebig erweiterbar. Bereits mit dem vorhandenen Leistungsumfang können neben PDF und XHTML weitere Zielformate erstellt werden. Durch Hinzufügen neuer Stylesheets oder Anpassungen lässt sich der Leistungsumfang noch mehr steigern, weitere Publikationsformen und Gestaltungsvorlagen sind möglich.

## Literaturverzeichnis

### **[Behme & Mintert 2000]**

BEHME, Henning & MINTERT, Stefan: *XML in der Praxis – Professionelles Web-Publishing mit der Extensible Markup Language*. 2., erw. Aufl. München u. a.: Addison-Wesley, 2000.

### **[DocBook.org 2007]**

DocBook.org: *Hello and welcome!* 2007. URL: <http://www.docbook.org/> (2007-07-19).

### **[DocBook Project 2005]**

The DocBook Project: *The DocBook Project*. 2005. URL: <http://docbook.sourceforge.net/> (2007-07-19).

### **[DocBook TC 2007]**

DocBook Technical Committee: *DocBook Technical Committee Document Repository*. k. A. URL: <http://www.oasis-open.org/docbook/> (2007-07-19).

### **[Fritsche 2000]**

FRITSCH, Hans P.: *Cross Media Publishing – Konzepte, Grundlagen und Praxis*. Bonn: Galileo-Press, 2000.

### **[Grunenberg 2001]**

GRUNENBERG, Christian: *Schöne neue Welten: Publishing zwischen Auf- und Umbruch*. In: *Publishing Praxis*, (2001) 01, S. 38 f.

### **[Harold 2004]**

HAROLD, Elliotte R.: *XML – Das mitp-Standardwerk zur professionellen Programmierung mit XML*. IT-Studienausg., 2. Aufl. Bonn: mitp, 2004.

**[Krüger 2006]**

KRÜGER, Manfred: *XSL-FO verstehen und anwenden – XML-Verarbeitung für PDF und Druck*. Heidelberg: dpunkt, 2006.

**[OASIS 2007]**

OASIS: *About OASIS*. 2007. URL: <http://www.oasis-open.org/who/> (2007-07-19).

**[OASIS 2007a]**

OASIS: *Monthly Archives for docbook-apps*. k. A. URL: <http://lists.oasis-open.org/archives/docbook-apps/> (2007-07-19).

**[Pawson 2002]**

PAWSON, Dave: *XSL-FO – Making XML look good in print*. Beijing u. a.: O'Reilly, 2002.

**[Schraitle 2004]**

SCHRAITL, Thomas: *DocBook-XML – Medienneutrales und plattformunabhängiges Publizieren*. Nürnberg: SuSE-Press, 2004.

**[Seyfarth 2007]**

SEYFARTH, Susanne: *Optimale Gestaltung von Schulungsmaterialien unter dem Gesichtspunkt des medienübergreifenden Publizierens*. Merseburg, Hochschule Merseburg (FH), Fachbereich Informatik und Kommunikationssysteme, Diplomarbeit, 2007.

**[Skulschus & Wiederstein 2005]**

SKULSCHUS, Marco & WIEDERSTEIN, Marcus: *XSL-FO für PDF und Druck*. Bonn: mitp, 2005.

**[Stayton 2005]**

STAYTON, Bob: *DocBook XSL – The complete guide*. 3rd ed. Sagehill Enterprises, 2005.

**[Tidwell 2002]**

TIDWELL, Doug: *XSLT – XML-Dokumente transformieren*. Beijing u. a.: O'Reilly, 2002.

**[W3C 1999]**

W3C: *XSL Transformations (XSLT) Version 1.0*. 1999. URL: <http://www.w3.org/TR/1999/REC-xslt-19991116> (2007-07-19).

**[W3C 1999a]**

W3C: *XML Path Language (XPath) Version 1.0*. 1999. URL: <http://www.w3.org/TR/1999/REC-xpath-19991116> (2007-07-19).

**[W3C 2001]**

W3C Deutsch-Österr. Büro: *XML in 10 Punkten*. 2001. URL: <http://www.w3c.de/Misc/XML-in-10-points.html> (2007-06-12).

**[W3C 2001a]**

W3C: *Extensible Stylesheet Language (XSL) Version 1.0*. 2001. URL: <http://www.w3.org/TR/2001/REC-xsl-20011015/> (2007-07-19).

**[W3C 2006]**

W3C: *Extensible Markup Language (XML) 1.0 (Fourth Edition)*. 2006. URL: <http://www.w3.org/TR/2006/REC-xml-20060816/> (2007-07-19).

**[W3C 2006a]**

W3C: *Extensible Stylesheet Language (XSL) Version 1.1*. 2006. URL: <http://www.w3.org/TR/2006/REC-xsl11-20061205/> (2007-07-19).

**[W3C 2007]**

W3C: *Über das World Wide Web Consortium (W3C)*. 2007. URL: <http://www.w3c.de/about/overview.html> (2007-07-19).

**[W3C 2007a]**

W3C: *XSL Transformations (XSLT) Version 2.0*. 2007. URL: <http://www.w3.org/TR/2007/REC-xslt20-20070123/> (2007-07-19).

**[Walsh 2006]**

WALSH, Norman: *DocBook Publishing Model*. 2006. URL: <http://nwalsh.com/docbook/procdiagram/> (2007-07-19).

**[Walsh & Muellner 2006]**

WALSH, Norman & MUELLNER, Leonard: *DocBook: The Definitive Guide*. 2006. URL: <http://www.docbook.org/tdg/en/html/docbook.html> (2007-07-19).

**[Wikipedia 2007]**

WIKIPEDIA: *Darwin Information Typing Architecture*. 2007. URL: [http://de.wikipedia.org/wiki/Darwin\\_Information\\_Typing\\_Architecture](http://de.wikipedia.org/wiki/Darwin_Information_Typing_Architecture) (2007-06-12).

**[Wikipedia 2007a]**

WIKIPEDIA: *Extensible Stylesheet Language – Formatting Objects*. 2007. URL: [http://de.wikipedia.org/wiki/Extensible\\_Stylesheet\\_Language\\_-\\_Formatting\\_Objects](http://de.wikipedia.org/wiki/Extensible_Stylesheet_Language_-_Formatting_Objects) (2007-07-19).

**[WikiPress 2006]**

WIKIPRESS: *DTP Professionell – Grundlagen, Standards und Perspektiven; aus der freien Enzyklopädie Wikipedia zusammengestellt von Andreas-M. Salignow*. Berlin: Zenodot, 2006.

## Eidesstattliche Erklärung

Ich versichere, dass ich die Arbeit selbstständig angefertigt, keine anderen als die angegebenen Hilfsmittel benutzt und alle wörtlichen oder sinngemäßen Entlehnungen deutlich als solche gekennzeichnet habe.

Leipzig, 26.07.2007

Christian Auspurg