inkl. CD!

MobileDevCon: Infos ab Seite 49

Deutschland € 6,50 Österreich € 7,00 Schweiz sFr 13,40

Infos zur EKON 14 ab Seite 61

entwickler magazin

entwickler magazin

Software, Systems & Development

www.entwickler-magazin.de | Juli/August | 4.10

CD-INHALT:

Trial

Embarcadero RAD Studio 2010: Die Development Suite für High-Performance- und .NET-Applikationen, mit Support für Touch- und gestenbasierte Interfaces.

Doc-To-Help 2010: Onlinehilfen, Projektdokumentationen und gedruckte Handbücher einfach und schnell erstellen. Inklusive XML-basierter Textverarbeitung und vieler Designvorlagen.

Software

Qt SDK for Windows: Cross Platform Framework für die Entwicklung Desktop- und Embedded-basierter Anwendungen, inklusive der Qt-Creator-IDE.

EpubCheck: IDPF-EPUB-Dateien validieren und debuggen, inkl. Support für OCF-Container, OPF und OPS Mark-up sowie Konsistenzprüfung.

Sigil: Der Multiplattform-WYSIWYG-E-Book-Editor zur Erstellung von Büchern im EPUB-Format. Mit Unicode-Unterstützung, Metadaten-Editor und umfangreicher Exportfunktionen.

Bonus

Zusätzlich finden Sie auf unserer aktuellen Magazin-CD wieder ausgewählte Tools sowie natürlich alle Quellcodes zu den Artikeln im Heft!

▸▸▸ Alle Infos auf Seite 3

Weitere Artikel

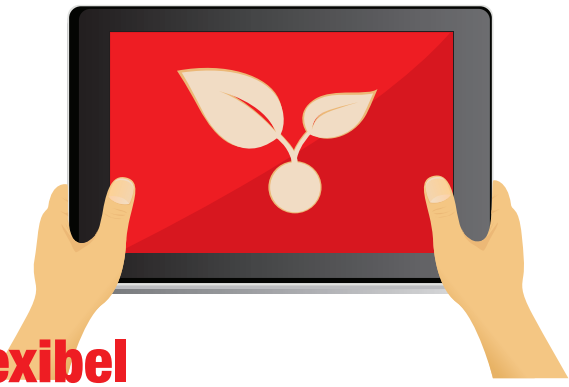
4 194156 606509 04

Datenträger enthält Info- und Lehrprogramme gemäß §14 JuSchG

iPad, Mobility & Multi-Touch

Die neuen Disziplinen für Entwickler

- Multi-Touch
- Online-Updates
- E-Book-Formate
- Modularisierung



Dynamisch & flexibel

Modular mit OSGi

Oracle polieren

Ein Wegweiser für sichere Oracle-Datenbank-Migrationen

Gefahr im Code

Die gefährlichsten Schwachstellen Verstehen. Erkennen. Vermeiden.

EPUB-Format in Theorie und Praxis

Einfach publizieren und benutzen

Spätestens seit der Veröffentlichung des Apple iPad haben elektronische Bücher (E-Books) einen hohen Bekanntheitsgrad erreicht. Neben dem reinen Konsumieren digitaler Werke wird auch ihre technische Umsetzung zunehmend interessant. Insbesondere das offene, breit unterstützte und relativ einfach zu produzierende EPUB-Format bietet sich an.

von Thomas Meinike

Die Bezeichnung EPUB bzw. die davon abgeleitete Dateierweiterung *.epub* steht zunächst für Electronic Publication. Üblicherweise repräsentiert eine solche Datei ein vollständiges E-Book oder zumindest einen Artikel mit sich an die konkrete Darstellungsumgebung anpassenden Inhalten (*reflowable content*). Technisch handelt es sich um ein ZIP-Archiv mit einer speziellen im Folgenden näher betrachteten Datenstruktur. Insofern kann der erste Kontakt durch das Umbenennen einer **.epub*-Datei in **.zip* und anschließendes Entpacken hergestellt werden. Testexemplare lassen sich schnell über das mehr als 30 000 Bücher umfassende Project Gutenberg [1] auffinden, z. B. die Max-und-Moritz-Geschichte von Wilhelm Busch [2].

Anschauen

In den letzten Jahren erschien eine breite Palette an Hardware-Readern (Abb. 1), die mit „elektronischem Papier" (E-Paper) auf Basis leitfähiger Polymere arbeiten. Unterschiedlich gefärbte Pigmentkügelchen fungieren in einer im elektrischen Feld befindlichen Emulsion als eine Art elektronische Tinte (E-Ink). Je nach Ladung werden die Partikel ausgerichtet und bilden somit die Texte und Grafikobjekte ab. Bisherige Graustufendisplays erreichen eine Auflösung um 160 dpi bei einem Raster von 600 x 800 Pixeln. Da lediglich beim Umblättern zum Erzeugen der neuen Seiten Energie benötigt wird, reicht eine Akkuladung je nach Verwendung mehrere Tage bis Wochen. Nicht alle Reader beherrschen das EPUB-Format. Von den gezeigten Geräten kann das ältere iLiad-Modell lediglich die Formate Mobipocket und PDF darstellen. Neuere Geräte und auch das iPad unterstützen neben diesen und weiteren Formaten vor allem EPUB.

Zum Lesen ist jedoch nicht unbedingt ein physisches Lesegerät in Form eines E-Book-Readers erforderlich. Mit Adobe Digital Editions (ADE) [3] steht eine plattformübergreifende Software zur Anzeige und Verwaltung von EPUB-Dokumenten zur Verfügung. Abbildung 2 zeigt ein vom Autor umgesetztes E-Book in der ADE-Bildschirmansicht, während Abbildung 3 die EPUB-Darstellung mit der kostenlosen iPhone-Software Stanza [4] veranschaulicht. Letztere wird übrigens auch für die Wiedergabe der im iTunes-Store erhältlichen E-Books vom Verlag O'Reilly verwendet. In den erhältlichen Buchanwendungen (Apps) ist neben der Anzeigesoftware jeweils ein EPUB-Dokument verpackt [5].

Anatomie

EPUB 2.0 wurde im September 2007 vom *International Digital Publishing Forum (IDPF)* standardisiert und vereint drei separate Spezifikationen [6]:

- OPS 2.0 – Open Publication Structure
- OPF 2.0 – Open Packaging Format
- OCF 1.0 – Open Container Format

Abb. 1: E-Book-Reader, vlnr: iRex iLiad, SONY PRS-600, BeBook Mini

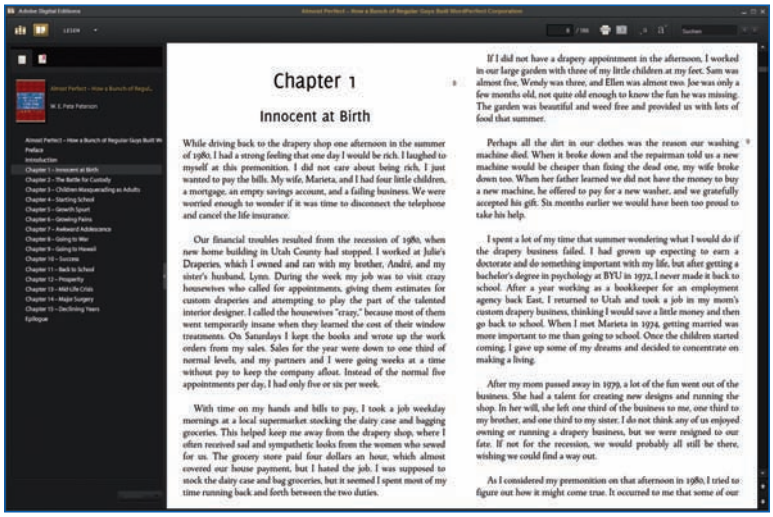


Abb. 2: Adobe Digital Editions

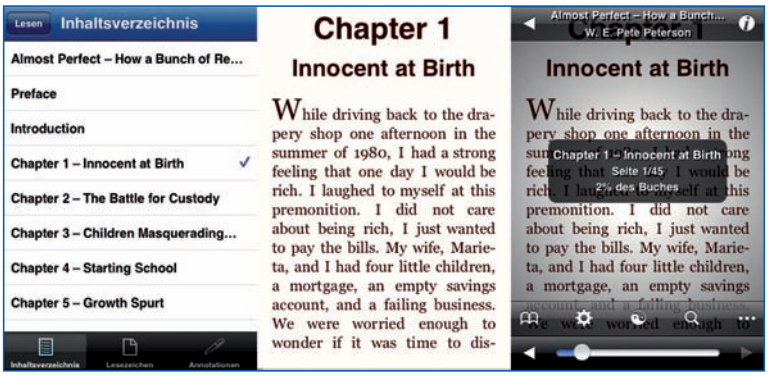


Abb. 3: iPhone-Anwendung Stanza

OCF beschreibt die nötige Verzeichnisstruktur und legt für *.epub* die Auslieferung als ZIP-Archiv fest. OPS definiert das inhaltliche Vokabular, insbesondere die Verwendung von XHTML, seltener DTBook und CSS zur Formatierung. OPF widmet sich schließlich den erforderlichen bzw. optionalen Metadaten, dem Aufbau des Inhaltsverzeichnisses sowie der späteren Leseabfolge.

Verwendet werden grundsätzliche offene Formate und Technologien: HTML, XML, CSS, Raster- und Vektorgrafiken, Schriftarten und MIME-Typen. Zusätzlich können Anbieter Digitales Rechtemanagement (DRM) verwenden und somit ausgelieferte E-Books an ihre Lesegeräte oder Software binden. Dazu dienen optionale – hier nicht weiter betrachtete – Dokumente wie *encryption.xml*, *rights.xml* und *signatures.xml*.

Abbildung 4 zeigt eine typische EPUB-Struktur. In der auf Wurzelebene abgelegten Datei *mimetype* ist lediglich eine Zeile mit dem Inhalt *application/epub+zip* ohne Umbruch hinterlegt. Im ebenfalls obligatorischen Verzeichnis *META-INF* liegt mindestens das XML-Dokument *container.xml* mit dieser Grundstruktur:

```
<?xml version="1.0" encoding="UTF-8"?>
<container version="1.0"
 xmlns="urn:oasis:names:tc:opendocument:xmlns:
                                    container">
 <rootfiles>
  <rootfile full-path="OPS/content.opf"
   media-type="application/oebps-package+xml"/>
 </rootfiles>
</container>
```

Unterhalb von *rootfiles* verweist ein *rootfile*-Element auf die weiterführenden Inhalte. Im Beispiel wird das Dokument *content.opf* referenziert, welches im relativ von der Dateiwurzel aus adressierten Verzeichnis *OPS* liegt. Der Verzeichnisname ist frei wählbar. Übliche Namen sind *OPS* oder *OEBPS (Open EBook Publication Structure)* und auch die Dateibezeichnung *content.opf* ist lediglich eine Konvention. Es handelt sich ebenfalls um ein XML-Dokument zur Referenzierung der eigentlichen Inhalte mit diesem formalen Aufbau:

```
<?xml version="1.0" encoding="UTF-8"?>
<package xmlns="http://www.idpf.org/2007/opf"
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 unique-identifier="BookId" version="2.0">

 <metadata>
  <!-- ... -->
 </metadata>

 <manifest>
  <!-- ... -->
 </manifest>

 <spine toc="toc">
  <!-- ... -->
 </spine>

 <tours>
  <!-- ... -->
 </tours>

 <guide>
  <!-- ... -->
 </guide>

</package>
```

Innerhalb des Elements *package* sind zunächst die einzelnen Sektionen ersichtlich. Beschreibende Metadaten bilden den Inhalt von *metadata*. Hauptsächlich wird die Dublin-Core-Notation mit dem Namensraum-Präfix *dc* verwendet. Als Kindelemente stehen zur Verfügung:

Anzeige

- contributor
- coverage
- creator
- date
- description
- format
- identifier
- language
- publisher
- relation
- rights
- source
- subject
- title
- type

Diese sind bis auf die drei Pflichtelemente *dc:identifier, dc:language* und *dc:title* optional:

```
<dc:title>Titel</dc:title>
<dc:language>de</dc:language>
<dc:identifier id="BookId">http://epub.example.net
                                    </dc:identifier>
```

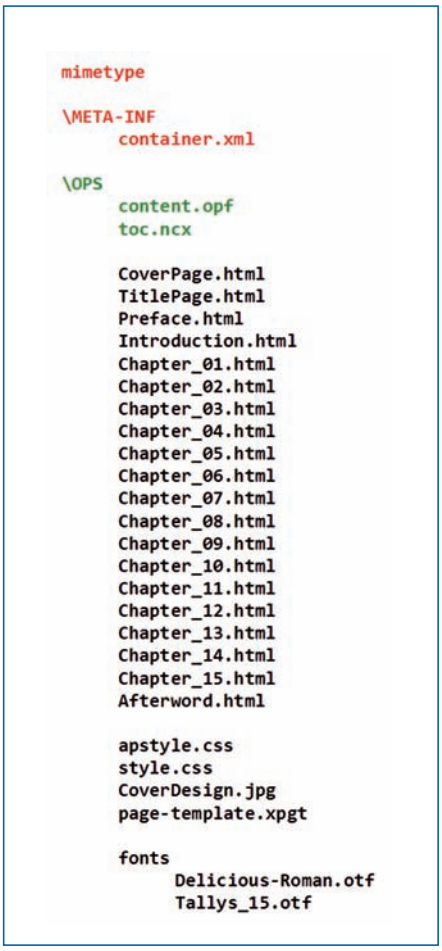


Abb. 4: EPUB-Dateistruktur

Zur Identifizierung eigener Werke bietet sich die Angabe eines URI an, während bei Verlagsveröffentlichungen die ISBN in der Form *urn:isbn:?????????????* verwendet wird. Der *id*-Attributwert (hier *BookId*) muss mit dem beim *package*-Element angegebenen Attributwert von *unique-identifier* übereinstimmen. Zusätzlich können weitere Metaelemente deklariert werden, etwa mit *x*-Präfix. Kein Bestandteil von EPUB, jedoch breit in Verwendung, ist die Angabe eines speziellen *meta*-Elements für das Buchcover:

Listing 1

Navigation toc.ncx

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ncx PUBLIC "-//NISO//DTD ncx 2005-1//EN"
"http://www.daisy.org/z3986/2005/ncx-2005-1.dtd">
<ncx xmlns="http://www.daisy.org/z3986/2005/ncx/"
                              xml:lang="de" version="2005-1">
 <head>
  <meta name="dtb:uid" content=
                                 "http://epub.example.net"/>
  <meta name="dtb:depth" content="1"/>
  <meta name="dtb:totalPageCount" content="0"/>
  <meta name="dtb:maxPageNumber" content="0"/>
 </head>
 <docTitle>
  <text>Der Blindtext</text>
 </docTitle>
 <navMap>
  <navPoint id="navPoint-1" playOrder="1">
   <navLabel>
    <text>Table of Contents</text>
   </navLabel>
   <content src="TableOfContents.html"/>
  </navPoint>
  <navPoint id="navPoint-2" playOrder="2">
   <navLabel>
    <text>Kapitel 1</text>
   </navLabel>
   <content src="kap1.html"/>
  </navPoint>
  <navPoint id="navPoint-3" playOrder="3">
   <navLabel>
    <text>Kapitel 2</text>
   </navLabel>
   <content src="kap2.html"/>
  </navPoint>
  <navPoint id="navPoint-4" playOrder="4">
   <navLabel>
    <text>Kapitel 3</text>
   </navLabel>
   <content src="kap3.html"/>
  </navPoint>
 </navMap>
</ncx>
```

```
<meta name="cover" content="CoverDesign"/>
```

CoverDesign repräsentiert in diesem Fall den Inhalt des *id*-Attributs einer im Manifest zugeordneten Bilddatei. Ein Lesegerät oder eine Software kann somit direkt auf das Titelbild zugreifen und es beispielsweise in einer Vorschauansicht darstellen.

Im Bereich *manifest* werden alle Inhaltsdateien aufgelistet. Dazu gehören hauptsächlich die verwendeten XHTML-Dokumente, verlinkte Stylesheets und Bilder sowie eingebundene Schriftarten. Der erste Eintrag bezieht sich auf eine zusätzliche Strukturdatei (hier *toc.ncx*) für das Inhaltsverzeichnis und die Aufrufabfolge der einzelnen Dokumente (Kapitel). Deklariert wird jeweils ein *item*-Element mit den Attributen *id* (Identifizierung), *href* (zu *content.opf* relativer Dateilink, ggf. in weitere Unterverzeichnisse innerhalb von *OPS* verweisend) und *media-type* (Formatbeschreibung als MIME-Type):

```
<item id="toc" href="toc.ncx"
 media-type="application/x-dtbncx+xml"/>

<item id="CoverDesign" href="CoverDesign.jpg"
 media-type="image/jpeg"/>

<item id="CoverPage" href="CoverPage.html"
 media-type="application/xhtml+xml"/>

<item id="style" href="style.css"
 media-type="text/css"/>

<item id="epub.embedded.font1" href=
                              "fonts/Delicious-Roman.otf"
 media-type="font/opentype"/>

<item id="kap1" href="kap1.html"
 media-type="application/xhtml+xml"/>

<!-- weitere item-Elemente ... -->
```

Der sich anschließende *spine*-Block gibt an, welche im Manifest vermerkten Dateien die einzelnen Archivteile darstellen und legt ihre lineare Abfolge in der Ausrichtung von oben nach unten fest:

```
<spine toc="toc">
 <itemref idref="CoverPage" linear="no"/>
 <itemref idref="kap1"/>
```

Anzeige

```
<itemref idref="kap2"/>
<itemref idref="kap3"/>
</spine>
```

Das *toc*-Attribut von *spine* stellt eine Referenz zum im Manifest aufgeführten Dokument *toc.ncx* her. Der Attributwert *toc* bezieht sich also auf den *id*-Wert des zugehörigen *item*-Elements. Die *idref*-Attribute der *itemref*-Elemente zeigen wiederum auf die *id*-Angaben im *manifest*-Bereich. Anfangs- oder Schlussseiten, die nicht zur eigentlichen Lesefolge gehören sollen, werden mit dem Attribut *linear*="*no*" versehen. Lesegeräte sollten das Buch beim ersten Auffinden eines *itemref*-Elements ohne explizite Angabe des Wertes "*no*" (also implizit ohne *linear*-Attribut oder explizit gesetztem *linear*="*yes*") öffnen.

Am Schluss der Paketstruktur befinden sich die optionalen Elemente *tours* und *guide*. Mittels *tours* kann eine Vorschau besonders interessanter Stellen im Buch angeboten werden (mehrere *tour*-Kindelemente sind möglich):

```
<tours>
 <tour id="tourID" title="Vorschau">
  <site href="kap1.html#fragment_id" title="..."/>
  <site href="kap2.html#fragment_id" title="..."/>
 </tour>
</tours>
```

Das *guide*-Element ist zwar ebenfalls optional, wird jedoch im Gegensatz zu *tours* meistens verwendet. Es vermittelt insbesondere die Bedeutung der einzelnen Abschnitte:

```
<reference type="cover" title="Titelseite"
 href="CoverPage.html"/>
<reference type="text" title="Kapitel 1"
 href="kap1.html"/>
<reference type="text" title="Kapitel 2"
 href="kap2.html"/>
<reference type="text" title="Kapitel 3"
 href="kap3.html"/>
```

Bei jedem *reference*-Element wird der anzuzeigende Titel (*title*) und das zugehörige Inhaltsdokument (*href*) angegeben. Zusätzlich beschreibt das *type*-Attribut die Funktion des jeweiligen Dokuments. Möglich sind hier u. a. diese weitgehend selbsterklärenden Schlüsselbegriffe:

- acknowledgements
- bibliography
- colophon
- copyright-page
- cover
- dedication
- epigraph
- foreword
- glossary
- index
- loi (list of illustrations)
- lot (list of tables)
- notes
- preface
- text
- title-page
- toc (table of contents)

Listing 2

XHTML-1.1-Kapiteldokument

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
 "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
                                    xml:lang="de">
 <head>
  <meta http-equiv="Content-Type"
   content="application/xhtml+xml; charset=UTF-8" />
  <meta name="description"
                          content="EPUB-Beispiel" />
  <link rel="stylesheet" type="text/css"
                               href="formate.css" />
  <title>Kapitel 1</title>
 </head>
 <body>
  <h1>Kapitel 1</h1>
  <h2>Hinter den Wortbergen</h2>
  <p class="first">Weit hinten,
                        hinter den Wortbergen, ...</p>
  <p>Eines Tages aber ...</p>
  <p>Wehmütig lief ihm ...</p>
 </body>
</html>
```

Listing 3

Cascading Stylesheet formate.css

```
body { margin: 1.5em 1em; padding: 0; }
h1,h2 { text-align: center; }
p { text-indent: 1em; text-align: justify; }
p.first, p.intro { text-indent: 0; }
p.first:first-letter { font-size: 2em; }

/* Fonts by Jos Buivenga (exljbris) ->
                                  www.exljbris.com */
@font-face { font-family: "Tallys"; font-style: normal;
     font-weight: normal; src: url("fonts/Tallys_15.otf")
                                  format("opentype"); }
@font-face { font-family: "Delicious"; font-style:
                        normal; font-weight: normal; src:
url("fonts/Delicious-Roman.otf") format("opentype"); }

body { font-family: "Tallys", serif; }
h1,h2 { font-family: "Delicious", sans-serif; }
```

Navigation

Der erste *item*-Eintrag im Manifest verweist auf *toc.ncx*, eine eigenständige Struktur zur Ablage des Inhaltsverzeichnisses und der so genannten Navigations-Map. Ein Beispiel ist in Listing 1 hinterlegt. Wie der Dokumententyp-Deklaration zu entnehmen ist, stammt *NCX (Navigation Center eXtended)* aus dem separaten Standard *DAISY (Digital Accessible Information SYstem)* [7]. In einem *head*-Element sind wiederum Metadaten enthalten. Das *name*-Element mit dem Attribut *dtb:uid* entspricht der *dc:identifier*-Angabe in *content.opf*. Als Verschachtelungstiefe (≥ 1) ist bei *dtb:depth* der Wert *1* angegeben und die Einträge *dtb:totalPageCount* bzw. *dtb:maxPageNumber* erhalten jeweils den Wert *0*, da diese nur für gedruckte Werke relevant sind. E-Books besitzen keine festen Seitenzahlen.

Auch der Inhalt des *docTitle*-Elements korrespondiert mit *dc:title* in der OPF-Struktur. Von besonderer Bedeutung ist die folgende *navMap*-Sektion mit mehreren *navPoint*-Kindelementen. Hier werden die einzelnen Kapitel oder Leseabschnitte referenziert (*src*-Attribut von *content*), bezeichnet (*text*-Element unter *navLabel*) und für die Wiedergabefolge vorbereitet (*playOrder*-Attribut von *navPoint*). *navPoint*-Elemente können weiter ineinander verschachtelt werden, um Verzweigungen abzubilden.

Mithilfe dieser Informationen kann ein Lesegerät oder eine Software die Generierung eines eigenständigen navigierbaren Inhaltsverzeichnisses vornehmen, unabhängig von einem zusätzlich in XHTML-Form vorliegenden.

Inhalte

Wie bereits ersichtlich, werden die EPUB-Inhalte mit konventionellen Webtechnologien umgesetzt, insbesondere mit

XHTML, CSS und Grafikformaten (GIF, JPEG, PNG, SVG). Der EPUB-Standard sieht explizit XHTML 1.1 vor und erwartet dafür den MIME-Type *application/xhtml+xml*. Auch XHTML-1.0-Dokumente haben im Test beispielsweise Lesegeräte von Sony problemlos verarbeitet. Dennoch sollte die Version 1.1 hinsichtlich breiter Kompatibilität der erzeugten E-Books bevorzugt werden.

Es können die üblichen Elemente zur Textauszeichnung wie *hx*-Überschriften, *p*-Absätze mit weiteren Auszeichnungen wie *em*, *strong* usw. verwendet werden. Listen, Bilder, (interne) Hyperlinks, Objekte (*object/param*) und Tabellen stehen ebenfalls zur Verfügung. Externe CSS-Vorlagen lassen sich mittels *link*-Element einbinden. JavaScript-Code wird zumindest von den Hardware-Lesegeräten nicht ausgeführt. Auf der Heft-CD befindet sich ein auf das Wesentliche beschränktes EPUB-Beispielprojekt. Das fertige E-Book besteht lediglich aus drei kurzen Kapiteln und enthält nur etwas Blindtext. Dieser wurde [8] entnommen und ist durchaus lesenswert. In Listing 2 ist ein verkürztes XHTML-1.1-Beispieldokument vom ersten Kapitel ersichtlich. Zusätzlich enthält Listing 3 das verwendete Stylesheet mit zwei über *@font-face*-Regeln eingebundenen OpenType-Schriftarten [9].

Einpacken

Alle zu einem EPUB-Projekt gehörenden Dateien brauchen nun nur noch mit einem geeigneten Packprogramm (etwa *Info-ZIP* oder *7-Zip*) in das ZIP-Format überführt werden, und nach dem Umbenennen der Dateierweiterung in *.epub* sollte sich das frisch erstellte E-Book mittels Software wie ADE oder nach dem Übertragen auf ein Lesegerät via USB-Kabel anzeigen und darin navigieren lassen. Gefordert wird, dass sich die Datei *mimetype* als erste in unkomprimierter Form im Archiv befindet.

Komfortabler lassen sich EPUB-Archive mit speziellen Anwendungen wie eCub umsetzen (Abb. 5). Die Software steht plattformübergreifend zur Verfügung [10] und erleichtert Autoren insbesondere die Eingabe der Metadaten und die formale Strukturierung. Es lassen sich

Abb. 5: Software eCub

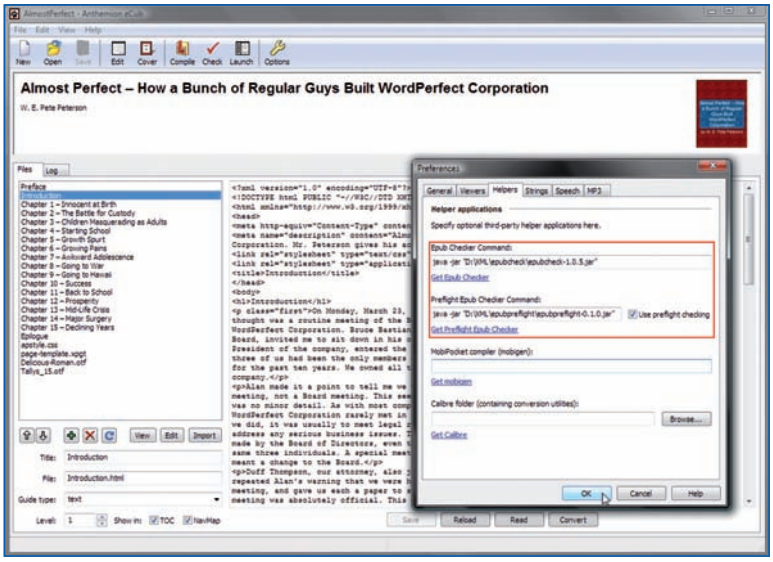


Abb. 6: iPad – Inhaltsverzeichnis

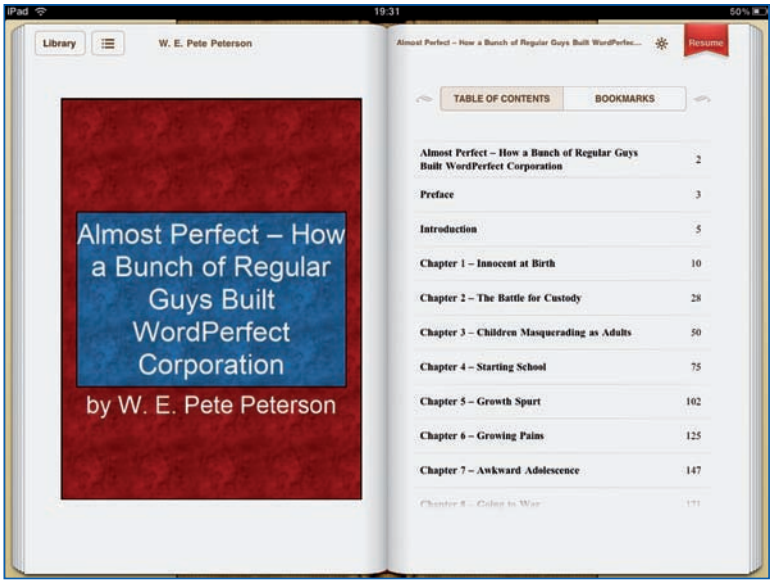
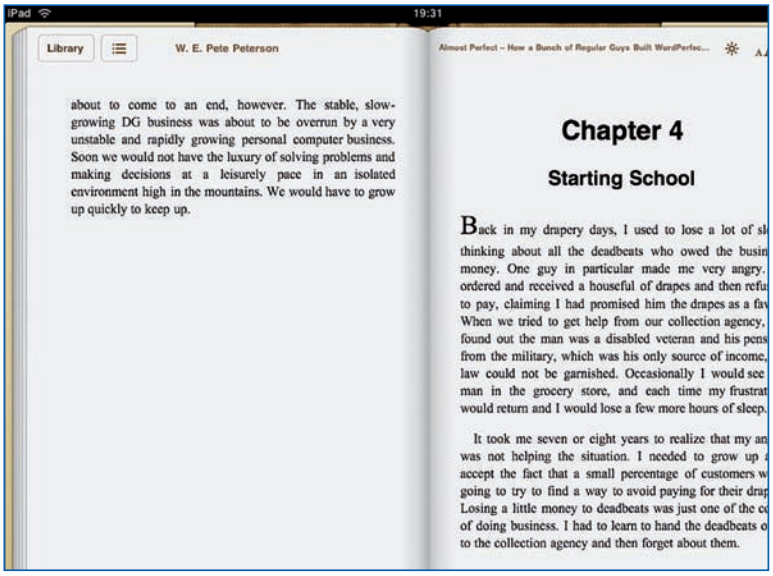


Abb. 7: iPad – Kapiteltextansicht



auch die geladenen XHTML- und CSS-Dateien im Textmodus bearbeiten, und ein einfacher Cover-Designer ist bei der Gestaltung des „Deckblatts“ behilflich. Sinnvoll ist jedoch die Vorbehandlung aller Inhalte im gewohnten HTML-Editor. eCub erzeugt beim „Kompilieren“ des Projekts jeweils ein *build*-Verzeichnis mit der zu packenden Struktur. Über die genannten Steuerdateien können manuelle Anpassungen auch nachträglich vorgenommen werden. So unterstützte die aktuelle verwendete Version 1.10 noch keine Font-Einbindung. Die Dateien lassen sich zwar dem Paket hinzufügen, es fehlen allerdings die zugehörigen *item*-Elemente im Manifest. Nach dem Einfügen dieser Informationen konnte der *build*-Inhalt erfolgreich neu gepackt werden.

eCub erlaubt über die Menühierarchie View | Preferences... | Helpers die Einbindung der in Java geschriebenen Konsolenwerkzeuge *EpubCheck* und *EpubPreflight* [11]. Diese testen daraufhin auf Knopfdruck die Integrität und Standardkonformität des aktuell bearbeiten EPUB-Dokuments. Leere oder zu große Inhalts- und Bilddateien (> 300 KB bzw. 10 MB) werden von *EpubPreflight* reklamiert. Der direkte Konsolenaufruf von *EpubCheck* ist in dieser Weise möglich (analog mit *epubpreflight-0.1.0.jar*):

```
java -jar X:\Pfad_zu\epubcheck-1.0.5.jar name.epub
```

Es steht eine Reihe weiterer Tools zur Verfügung. Noch in der Entwicklung befindet sich das bereits ansatzweise nutzbare Open-Source-Programm Sigil [12]. Auch neuere Versionen von Adobe InDesign verfügen über eine EPUB-Exportmöglichkeit [13]. Die Software calibre [14] bietet sich zur Konvertierung von PDF nach EPUB an. Speziellere EPUB-Umsetzungen innerhalb von XML-Workflows lassen sich unter Nutzung der DocBook-XSLT-Stylesheets [15] oder über die Transformation von DITA-Strukturen mit dem XMLmind DITA Converter [16] realisieren.

Praxis

Ein vom Autor umgesetztes elektronisches Buch ist unter [17] zu finden. Es basiert auf den Quellen des in den späten 1990er Jahren gedruckt vertriebenen Buches „Almost Perfect“ von W. E. Peterson und behandelt Aufstieg und Fall der einst berühmten Firma WordPerfect Corporation. Die bereits online verfügbaren HTML-Quelldateien der einzelnen Kapitel wurden zunächst nach XHTML 1.1 konvertiert, mit CSS und freien Schriftarten angereichert und schließlich mit eCub in das EPUB-Format überführt (vgl. Abbildungen 6 und 7). Dankenswerterweise hat Mr. Peterson das Ergebnis im März 2010 auf seiner Website veröffentlicht.

Primär für Lehrzwecke entstand das Projekt „epubMinFlow“. Es handelt sich um einen Batch-Job, der zunächst eine EPUB-Grundstruktur anlegt und ein kompakt aufgebautes XML-Buchdokument mit einem XSLT-2.0-Stylesheet transformiert. Über die Ausgabetechnik *xsl:result-document* werden alle relevanten Dateien bis hin zu den eigentlichen XHTML-Inhaltsdokumenten produziert. Anschließend wird der Kompressionsschritt vollzogen und das Ergebnis mit den genannten Testwerkzeugen überprüft. Auf der Heft-CD sowie unter [18] stehen weitere Informationen und der Quellcode zur Verfügung.

Fazit

Die EPUB-Technologie ermöglicht die kostengünstige Entwicklung von E-Books für die Nutzung mit modernen Lesegeräten oder spezieller Software. Zum Einsatz kommen offene Webstandards und es stehen bereits komfortable Werkzeuge zur Verfügung. Eigenen Entwicklungen steht bei Kenntnis und Berücksichtigung der diskutierten Aspekte nichts im Weg. Allerdings kommt man um sauberes Arbeiten und akribisches Testen der erzeugten E-Books nicht herum.

Eine Fülle an weiterführendem Material zum Thema „Digitales Publizieren im EPUB-Kontext“, speziell zu Optimierungen und Software, liefern die von Adobe veröffentlichten Dokumente [19]. Neuere Entwicklungen kündigen sich ebenfalls an. So wurde eine neue Arbeitsgruppe zur Etablierung von EPUB 2.1 ins Leben gerufen, um den neuen inhaltlichen und technischen Anforderungen an moderne elektronische Publikationen gerecht werden zu können [20].

Zum Schluss geht noch ein herzlicher Dank an Alexander Olma (iPhoneBlog.de) für die iPad-Screenshots, die den erhaltenen Ergebnissen eine besondere Note verleihen.

Dr. Thomas Meinike *ist seit 1997 an der Hochschule Merseburg (FH) als Lehrkraft tätig. Seine Arbeitsschwerpunkte sind XML-Anwendungen in der Technischen Dokumentation, Onlinehilfen und Webentwicklung.*
E-Mail-Kontakt: thomas.meinike@hs-merseburg.de.

Links & Literatur

[1] Project Gutenberg: gutenberg.org
[2] Max und Moritz: gutenberg.org/etext/17161
[3] ADE: adobe.com/de/products/digitaleditions
[4] Stanza: lexcycle.com
[5] O'Reilly: oreilly.com/ebooks/oreilly_iphone_tips.csp
[6] IDPF: idpf.org
[7] DAISY Consortium: daisy.org
[8] Blindtextgenerator.de
[9] Buivenga, J.: exljbris.com
[10] eCub: juliansmart.com/ecub
[11] EpubCheck/EpubPreflight: code.google.com/p/epubcheck
[12] Sigil: code.google.com/p/sigil
[13] Adobe: blogs.adobe.com/digitaleditions/indesign-epub.html
[14] calibre: calibre-ebook.com
[15] DocBook-Stylesheets: sourceforge.net/projects/docbook
[16] XMLmind DITA Converter: xmlmind.com/ditac
[17] Peterson, W. E.: wepeterson.com/ap/ap.htm
[18] Meinike, T.: datenverdrahten.de/epubMinFlow
[19] Adobe: adobe.com/devnet/digitalpublishing
[20] IDPF: idpf.org/idpf_groups/IDPF-EPUB-WG-Charter-4-6-2010.html