Offene eBooks mit EPUB erstellen

Benedict Reuschling
bcr@FreeBSD.org

Grazer Linuxtage
9. April 2011

Überblick

Überblick

Motivation

- eBooks werden immer beliebter

Motivation

- eBooks werden immer beliebter
- Grossflächige Verwendung von mobilen Geräten
  - Unterschiedliche Bildschirmgrössen
  - Horizontales scrollen ist schlecht . . .
  - genauso wie ständiges zoomen
  - Ziel: Inhalt soll sich jeder Bildschirmgrösse anpassen

Motivation

- eBooks werden immer beliebter
- Grossflächige Verwendung von mobilen Geräten
  - Unterschiedliche Bildschirmgrössen
  - Horizontales scrollen ist schlecht . . .
  - genauso wie ständiges zoomen
  - Ziel: Inhalt soll sich jeder Bildschirmgrösse anpassen
- Offenes Format von Vorteil

Überblick

Was ist EPUB?

. . . und warum wir nicht immer PDF verwenden sollten

Aus der EPUB Spezifikation[1]:

"EPUB enables the creation and transport of reflowable digital books and other types of content as single-file digital publications that are interoperable between disparate EPUB-compliant reading devices and applications."

[1] idpf.org

Was ist EPUB?

. . . und warum wir nicht immer PDF verwenden sollten

Aus der EPUB Spezifikation[1]:

”EPUB enables the creation and transport of reflowable digital books and other types of content as single-file digital publications that are interoperable between disparate EPUB-compliant reading devices and applications.”

”.epub” allows publishers to produce and send a single digital publication file through distribution and offers consumers interoperability between software/hardware for unencrypted reflowable digital books and other publications.

[1] idpf.org

EPUB Standards

EPUB besteht aus den folgenden drei Standards des IDPF:

OPF: Open Packaging Format
Wie sieht jede Seite aus?

EPUB Standards

EPUB besteht aus den folgenden drei Standards des IDPF:

OPF: Open Packaging Format
Wie sieht jede Seite aus?

OPS: Open Publication Structure
Was ist der Inhalt des Buches?

EPUB Standards

EPUB besteht aus den folgenden drei Standards des IDPF:

OPF: Open Packaging Format
Wie sieht jede Seite aus?

OPS: Open Publication Structure
Was ist der Inhalt des Buches?

OCF: Open Container Format
Was hält das ganze zusammen?

Schauen wir uns das mal näher an . . .

. . . und machen uns dabei ein bisschen die Hände schmutzig.

```
$ file example.epub
```

Schauen wir uns das mal näher an . . .

. . . und machen uns dabei ein bisschen die Hände schmutzig.

```
$ file example.epub
example.epub: Zip archive data, at least v1.0 to extract
```

Schauen wir uns das mal näher an . . .

. . . und machen uns dabei ein bisschen die Hände schmutzig.

```
$ file example.epub
example.epub: Zip archive data, at least v1.0 to extract
```

Könnte es wirklich so einfach sein?

Schauen wir uns das mal näher an . . .

. . . und machen uns dabei ein bisschen die Hände schmutzig.

```
$ file example.epub
example.epub: Zip archive data, at least v1.0 to extract
```

Könnte es wirklich so einfach sein?
Antwort: Ja, zumindest dieser Teil.

eBooks: So einfach zu öffnen wie gedruckte Bücher

```
$ unzip example.epub
```

eBooks: So einfach zu öffnen wie gedruckte Bücher

```
$ unzip example.epub
Archive:  example.epub
 extracting: mimetype
  inflating: META-INF/container.xml
  inflating: OPS/chapter1.xml
  inflating: OPS/content.ncx
  inflating: OPS/content.opf
  inflating: OPS/image.png
  inflating: OPS/stil.css
```

Apropos tote Bäume . . .

```
.epub
├── mimetype
├── META-INF/
│   └── container.xml
└── OPS/
    ├── chapter1.xml
    ├── content.ncx
    ├── content.opf
    ├── image.png
    └── stil.css
```

Die Struktur ist bis zu `container.xml` definiert in OPF, der Rest kann beliebig gewählt werden.

Die `mimetype` Datei

Viel gibts hier nicht zu sehen:

```
echo "application/epub+zip" > mimetype
```

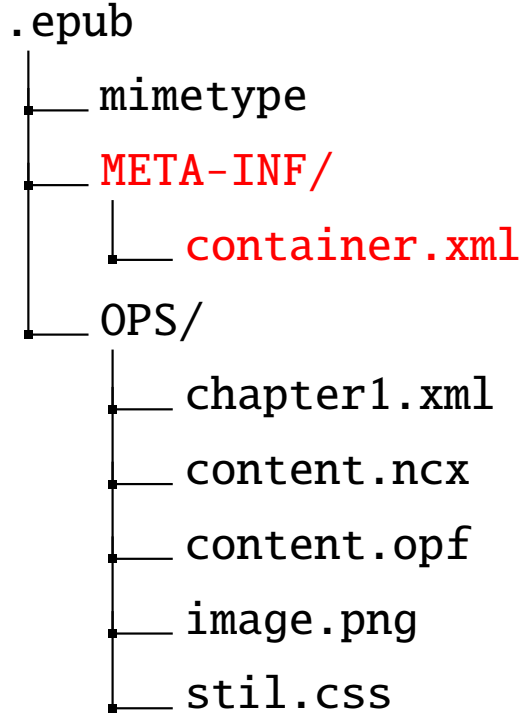
Darf nicht komprimiert oder verschlüsselt sein

```
.epub
├── mimetype
├── META-INF/
│   └── container.xml
└── OPS/
    ├── chapter1.xml
    ├── content.ncx
    ├── content.opf
    ├── image.png
    └── stil.css
```

Das `META-INF` Verzeichnis

- Benötigte Datei: `container.xml` (unverschlüsselt)
- Beschreibt den MIME-Type und das OPS/OPF rootfile zu der Publikation
- Weitere optionale Dateien:

  `signatures.xml`: digitale Signaturen
  `encryption.xml`: Schlüssel, Algorithmen
  `rights.xml`: DRM (zukünftige Versionen)



Beispielinhalt der `container.xml` Datei

Beispielinhalt

```
<?xml version="1.0"?>
<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>
```

Wichtig: der `full-path` ist *nicht* relativ zum `META-INF` Verzeichnis

Text, Bilder und Stil - der Inhalt des Buchs

- Inhalt mit wenigen Ausnahmen XHTML 1.1
- CSS mit eingebetteten Fonts für das Layout
- Externe `img`es sind problematisch
- Externe Links funktionieren, innere Verweise nur bedingt
  - zur Erinnerung: ein *reflowable*-Format
  - Verschiedene EPUB-Reader können Seiten anders umbrechen
  - Inhaltsverzeichnis so umbauen, dass Leser direkt zu bestimmten Abschnitten gelangen
- Kein `script`, EPUB-Reader brauchen es nicht auszuführen
- Grosse Bücher besser mittels Unterverzeichnissen strukturieren

```
.epub
├── mimetype
├── META-INF/
│   └── container.xml
└── OPS/
    ├── chapter1.xml
    ├── content.ncx
    ├── content.opf
    ├── image.png
    └── stil.css
```

Das Inhaltsverzeichnis in `content.ncx` definieren

NCX: Navigation Center eXtended

- Verwendet Spezifikation des DAISY-Konsortiums
- Format, um Personen mit Leseproblemen das Navigieren im Buch zu erleichtern
- Gibt die Lesereihenfolge über mehrere `navPoints` innerhalb einer `NavMap` an
- Kapitel, Abschnitte und Unterabschnitte des Inhaltsverzeichnisses gehören hier hin

```
.epub
├── mimetype
├── META-INF/
│   └── container.xml
└── OPS/
    ├── chapter1.xml
    ├── content.ncx
    ├── content.opf
    ├── image.png
    └── stil.css
```

Beispiel einer content.ncx Datei

```
<?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/"
   version="2005-1" xml:lang="en-US">
```

Beispiel einer content.ncx Datei

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE ...
<ncx>
```

```
<head>
  <meta name="dc:Title" content="FreeBSD on Laptops"/>
  <meta name="dtb:uid"
   content="http://www.freebsd.org/doc/en_US.ISO8859-1/articles/
   laptop/index.html"/>
  <meta name="dtb:depth" content="1"/>
  <meta name="dtb:totalPageCount" content="0"/>
  <meta name="dtb:maxPageNumber" content="0"/>
</head>
```

Beispiel einer content.ncx Datei

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE ...
<ncx>
  <head> ... </head>
```

```
<docTitle>
  <text>FreeBSD on Laptops</text>
</docTitle>
```

Beispiel einer content.ncx Datei

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE ...
<ncx>
  <head> ... </head>
  <docTitle> ... </docTitle>
```

```
<navMap>
  <navPoint playOrder="1" id="id_Chapter_01">
    <navLabel>
      <text>FreeBSD on Laptops</text>
    </navLabel>
    <content src="chapter1.xml"/>
  </navPoint>
</navMap>
```

Beispiel einer `content.ncx` Datei

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE ...
<ncx>
  <head> ... </head>
  <docTitle> ... </docTitle>
  <navMap> ... </navMap>
</ncx>
```

Ob `content.ncx` oder `content.opf` zuerst vom EPUB-Readern geparst wird ist unterschiedlich → Um sicher zu gehen, sollte man beide Dateien angeben

Gebunden wird alles mit `content.opf`

XML-Datei, referenziert von `container.xml`

Zweck

- definiert, wo sich die Inhalte des Buchs befinden
- referenziert die NCX-Inhaltsverzeichnisdatei

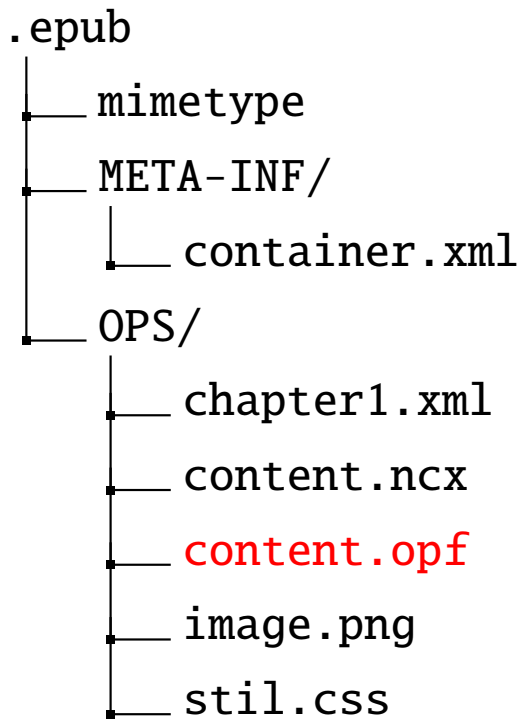
Grundlegende Struktur

`<package>`: Header mit Namensraum und unique-identifier

`<metadata>`: Metadata wie Titel, Autor, URI

`<manifest>`: die Liste der Dateien, die den Inhalt ausmacht

`<spine>`: definiert die lineare Lesereihenfolge



Beispiel für eine `content.opf` Datei

```
<?xml version="1.0" encoding="UTF-8" ?>
```

```
<package
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:opf="http://www.idpf.org/2007/opf"
  xmlns="http://www.idpf.org/2007/opf"
  version="2.0" unique-identifier="BookId">
```

Beispiel für eine content.opf Datei

```
<?xml version="1.0" encoding="UTF-8" ?>
<package>
```

```
<metadata>
  <dc:language xsi:type="dcterms:RFC3066">en-US</dc:language>
  <dc:title>FreeBSD on Laptops</dc:title>
  <dc:identifier id="BookId">
    http://www.freebsd.org/doc/en_US.ISO8859-1/articles/
    laptop/index.html
  </dc:identifier>
</metadata>
```

Beispiel für eine content.opf Datei

```
<?xml version="1.0" encoding="UTF-8" ?>
<package>
  <metadata> ... </metadata>
```

```
<manifest>
  <item id="cover" href="cover.html"
    media-type="application/xhtml+xml"/>
  <item id="ncx" href="content.ncx"
    media-type="application/x-dtbncx+xml"/>
  <item id="Chapter_1" href="chapter1.xml"
    media-type="application/xhtml+xml"/>
  <item id="css" href="stil.css" media-type="text/css"/>
</manifest>
```

Beispiel für eine `content.opf` Datei

```
<?xml version="1.0" encoding="UTF-8" ?>
<package>
  <metadata> ... </metadata>
  <manifest> ... </manifest>
```

```
<spine toc="ncx">
  <itemref idref="Chapter_1"/>
</spine>
```

Beispiel für eine `content.opf` Datei

```
<?xml version="1.0" encoding="UTF-8" ?>
<package>
  <metadata> ... </metadata>
  <manifest> ... </manifest>
  <spine> ... </spine>
</package>
```

- Es gibt einen optionalen `<guide>`-Abschnitt für semantische informationen
- Definiert Dinge wie das Cover, Inhaltsverzeichnisse und Anhänge

If you judge a book by the cover . . .

EPUB Cover Definition

- Separate XHTML-Datei
- Unterstützt sowohl Text als auch GIF, JPG, PNG & SVG
- Definiert im `<guide>`-Abschnitt von `content.opf`
- Erstes Element in `<spine>` mit dem `linear="no"`-Attribut

Erstellen des EPUB-Containers

Zuerst wird die `mimetype` Datei unkomprimiert ohne Attribute hinzugefügt:

```
$ zip -0X myepub.epub mimetype
adding: mimetype (stored 0%)
```

Erstellen des EPUB-Containers

Zuerst wird die `mimetype` Datei unkomprimiert ohne Attribute hinzugefügt:

```
$ zip -0X myepub.epub mimetype
adding: mimetype (stored 0%)
```

dann rekursiv die übrigen Dateien ohne Verzeichniseinträge anhängen:

```
$ zip -9DurX myepub.epub *
adding: META-INF/container.xml (deflated 32%)
adding: OPS/chapter1.xml (deflated 63%)
adding: OPS/content.ncx (deflated 48%)
adding: OPS/content.opf (deflated 54%)
adding: OPS/cover.html (deflated 34%)
adding: OPS/image.png (deflated 4%)
adding: OPS/stil.css (deflated 57%)
```

Fertig!

Validierung

. . . oder validiert ihr eure XML-Dateien etwa nicht? ;-)

- Fehler früh zu finden erspart einem später einiges an Kopfzerbrechen
- `epubcheck` prüft den Container und die Dateistruktur
- Läuft als Terminalanwendung, Java Web Anwendung oder Java Bibliothek
- Verfügbar unter `http://code.google.com/p/epubcheck/`

```
$ java -jar epubcheck.jar myepub.epub
```

Überblick

Konverter für EPUBs

calibre[2] von Kovid Goyal

- eBook-Software zur Verwaltung der eigenen eBibliothek
- Erhältlich für Windows, Linux, Mac OS X
- Enthält einen Konverter mit zahlreichen Optionen für verschiedene Formate
- Bsp.: EPUBs für Amazon Kindle umwandeln
- Ergebnisse abhängig von der Komplexität der Quellen

[2] http://calibre-ebook.com/

Sigil - WYSIWYG-Editor für EPUBs

Sigil[3] von Strahinja Markovic

- Komfortabler Editor zum erstellen von EPUBs
- Erhältlich für Windows, Linux, Mac OS X
- Generiert mehrstufige Inhaltsverzeichnisse
- Unterstützng für über 200 Metadaten
- Verschiedene Ansichten: Code, Buch, Split
- eingebaute Validierung
- HTML Tidy für sauberen Code

[3] http://code.google.com/p/sigil/

Überblick

Zusammenfassung

- Mit wenig Aufwand lassen sich früh Ergebnisse erzielen
- Freie Formate unterstützen die Wiederverwendbarkeit
- Tests auf verschiedenen Geräten angeraten
- Verzweigungen innerhalb der Publikation prüfen
- Coverbilder verleihen eine eigene persönliche Note
- Integration in viele andere Softwarepakete

Danke für Euer Interesse!

Fragen, Anmerkungen, Feedback?

Referenzen und weiterführende Links

Liza Daly, IBM Developerworks
Build a digital book with EPUB - The open XML-based eBook format
http://www.ibm.com/developerworks/xml/tutorials/x-epubtut/
25 Nov 2008

International Digital Publishing Forum
EPUB specifications for OPS, OPF, OCF
http://www.openebook.org/specs.htm
09 Apr 2010

Harrison Ainsworth
Epub Format Construction Guide
http://www.hxa.name/articles/content/epub-guide_hxa7241_2007.html
27 Aug 2010