

Mit DITA und `xsl:result-document` zur Onlinehilfe

# Fließbandproduktion

Die im Januar 2007 verabschiedeten Spezifikationen zu XSLT 2.0 und XPath 2.0 enthalten gegenüber den acht Jahre alten 1.0-Versionen umfangreiche Erweiterungen, Verbesserungen und Vereinfachungen [1–3]. In diesem Beitrag wird über ein Projekt zur Umsetzung von Onlinehilfen im HTML-Help-Format auf der Basis von DITA-Dokumenten berichtet. Dabei spielt das Element `xsl:result-document` eine wesentliche Rolle und einige weitere neue Techniken kommen ebenfalls zur Anwendung.

## von Thomas Meinike

Rein quantitativ sind die Erweiterungen in XSLT und XPath beachtlich: 49 (bisher 35) XSLT-Elemente, 19 (bisher 9) XSLT-Funktionen sowie 111 (bisher 27) XPath-Funktionen aus dem *fn*-Namensraum stehen zur Verfügung. Je nach Sichtweise werden weitere Schema-typisierte Konstruktorfunktionen wie `xs:date('2007-10-17')` in die Zählung einbezogen [4, 5]. Da bereits an anderer Stelle über die neuen Techniken mit ausführlichen Beispielen berichtet wurde [6], soll hier im Wesentlichen ein neues Element im Blickpunkt stehen. Es handelt sich um `xsl:result-document`, das innerhalb einer Transformationsvorlage beliebig viele Ergebnisdokumente erzeugen kann. Bisher war die Ausgabe ohne fremde Hilfe durch Erweiterungsfunktionen auf genau ein HTML-, XML- oder Text-Dokument beschränkt. Über die XSLT-Funktion `document()`

konnte aber bereits auf mehrere XML-Dokumente auf der Eingabeseite zugegriffen werden. Im Bereich der XML-basierten Erstellung von Onlinehilfen ist es besonders praktikabel, die strukturierten Inhalte zu den einzelnen Themen jeweils in einem XML-Dokument abzulegen und daraus passend aufbereitete (X)HTML-Dokumente für die spätere Zusammenstellung der Hilfe abzuleiten. Hier soll das von Microsoft eingeführte HTML-Help-Format (Compiled HTML mit der Dateiendung *chm*) betrachtet werden. Die Auszeichnung der Inhalte erfolgt auf Basis von DITA. DITA steht für Darwin Information Typing Architecture. Es handelt sich um einen offenen Standard, der ursprünglich von IBM entwickelt wurde und mittlerweile in Version 1.1 vom OASIS-Konsortium herausgegeben und weitergeführt wird. DITA definiert insbesondere Topics (Themen), die eine möglichst abgeschlossene Informationseinheit

bilden. Ein Topic kann beispielsweise die für einen Hilfeintrag bestimmte HTML-Seite repräsentieren. In XML-Form werden jedoch nur die konkreten Inhalte wie Überschriften, Absätze, Listen, Tabellen, Bilder oder Links ohne Zusatzinformationen bezüglich ihrer Darstellung und Formatierung strukturiert. Das DITA-Konzept ermöglicht auch die Beschreibung von Task-, Concept- und Reference-Dokumenten. Zu Details sei auf die OASIS-Dokumente [7] und aus konzeptioneller Sicht auf das im Heft 5.2007 auf Seite 15 besprochene Buch [8] verwiesen. Wichtig ist an dieser Stelle noch eine weitere Komponente, die DITA-Map. Ein Map-Dokument fasst einen Satz an Topics zusammen und bildet eine Struktur im Sinne des Inhaltsverzeichnisses eines Buches oder der Navigationsstruktur innerhalb einer Onlinehilfe ab. Gerade aus der Sicht von Onlinehilfen kommt das DITA-Konzept wie gerufen, und das Gespann aus XSLT 2.0

und XPath 2.0 ermöglicht eine besonders praktikable Umsetzung.

## DITA: Topics und Map

Ein einfaches Topic-Dokument hat diesen Aufbau:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE topic PUBLIC "-//OASIS//DTD DITA Topic//EN"
    "dtd/topic.dtd">
<topic id="unter12">
  <title>Unterpunkt 1.2</title>
  <shortdesc>Hier folgen die Inhalte für den Unterpunkt
    1.2.</shortdesc>
  <prolog>
    <author>Dr. Thomas Meinike</author>
  </prolog>
  <body>
    <section>
      <title>Überschrift ...</title>
      <p>Hier wird noch fleißig gearbeitet.</p>
    </section>
  </body>
</topic>
```

Das Wurzelement lautet *topic* und ist mit dem Pflichtattribut *id* versehen. Anschließend folgen die Elemente *title* für den Titel und *shortdesc* für eine kurze Beschreibung dieses Topics. Innerhalb des optionalen Prologs lassen sich Angaben zum Autor (*author*-Element) und weitere Metadaten wie Stichwörter und Ähnliches unterbringen. Die eigentlichen Nutzdaten stehen im *body*-Element. Dessen Inhalte muten anfangs wie eine Mischung aus HTML und DocBook an und tatsächlich findet man sich schnell zurecht, sofern entsprechende Grundkenntnisse in diesen Auszeichnungssprachen vorliegen. Es gibt natürlich auch eine Reihe von neuen Elementen, die sich Schritt für Schritt erschließen lassen. Die zur Validierung benötigten Dokumenttyp-Definitionen können unter [7] bezogen werden und sind auch auf der Heft-CD im Rahmen des Beispielprojekts enthalten. XML-Editoren wie XMLSpy bringen die DTDs ebenfalls mit und ermöglichen somit das komfortable Einfügen der Elemente und Attribute im jeweiligen Kontext. In den Beispieldokumenten werden folgende DITA-Elemente für die Topic-Inhalte verwendet:

- *section* für Abschnitte
- *title* für Überschriften

- *p* für Absätze
- *ol*, *ul*, *li* für Listen
- *image* für Bilder
- *xref* für Hyperlinks
- *table*, *tgroup*, *thead*, *tbody*, *row*, *entry* für Tabellen
- *indexterm* bzw. *keyword* für Stichwörter

Im Beispielprojekt ist neben acht Topics ein Dokument als Task zur genauen Beschreibung von Handlungsanweisungen ausgezeichnet. Statt des *body*-Elements wird *taskbody* verwendet und die einzelnen Schritte einer Handlungskette (*steps*) werden innerhalb von *step*-Elementen notiert (Listing 1).

Im optionalen Element *stepresult* kann das jeweils beabsichtigte Teilergebnis des aktuellen Handlungsschritts vermerkt werden. Nach *steps* kann zusätzlich ein *result*-Element folgen, welches das Ergebnis der gesamten Handlung dokumentiert. Zur Beschreibung der globalen Dokumentstruktur dient eine DITA-Map mit diesem Grundgerüst:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE map PUBLIC "-//OASIS//DTD DITA Map//EN"
    "dtd/map.dtd">
<map title="Demo-Hilfe" id="demohilfe">
  <topicref navtitle="Startseite" href="start_topic.xml"
    type="topic"/>
  <!-- weitere topicref-Elemente -->
  <topicref navtitle="Zusammenfassung" href="schluss_
    topic.xml" type="topic"/>
</map>
```

Nach dem Wurzelement *map* mit den Attributen *title* und *id* erscheinen die einzelnen innerhalb eines Inhaltsverzeichnisses oder einer Navigation zu platzierenden Einträge jeweils als *topicref*-Element. Den Text des Eintrages nimmt das Attribut *navtitle* auf und der Verweis auf das konkrete Dokument steht im *href*-Attribut. Zusätzlich macht das *type*-Attribut auf die Art des DITA-Quelldokuments aufmerksam (*concept*, *reference*, *task* oder *topic*). Im obigen Beispielcode sind die Einträge für die Startseite und die als Zusammenfassung deklarierte letzte Seite der später zu produzierenden Hilfe zu sehen. Interessant für die Darstellung von Unterpunkten ist die Möglichkeit der Verschachtelung von *topicref*-Elementen. Für

den ersten Hauptpunkt und dessen Unterpunkte wurde dieses Konstrukt gewählt:

```
<topicref navtitle="Erster Hauptpunkt" href="topic1.xml"
  type="topic">
  <topicref navtitle="Unterpunkt 1.1" href="topic11.xml"
    type="topic"/>
  <topicref navtitle="Unterpunkt 1.2" href="topic12.xml"
    type="topic"/>
</topicref>
```

Auf diese Weise lassen sich beliebige tief verschachtelte Strukturen aufbauen. Abbildung 1 zeigt die vollständige Map und ein Topic sowie die Struktur von Quell- und Zielverzeichnis.

## Crashkurs HTML Help

Das mit Windows 98 eingeführte HTML-Help-Format stellt technisch ein komprimiertes Archiv mit der Dateierweiterung *chm* dar. Darin befinden sich alle HTML-Dokumente sowie die verknüpften Bilder und sonstige referenzierte Dateien wie Cascading Stylesheets (CSS) sowie eingebundene Medien wie z. B. Flashfilme oder Videos. Zur Einbindung letzterer Objekte ist üblicherweise etwas „Feintuning“ nötig, worauf an dieser Stelle aber nicht weiter eingegangen werden kann. Zumindest Bilder und Stylesheets werden beim Kompilervorgang direkt einbezogen. Benötigt wird letztlich nur der mit dem kostenlosen HTML Help Workshop [9] ausgelieferte Help-Compiler *hhc.exe*. Dieses Programm kann

### Listing 1

```
<taskbody>
  <steps>
    <step>
      <cmd>Öffnen Sie das Konsolenfenster über Start >
        Ausführen > cmd > OK!</cmd>
      <stepresult>Das Fenster ist für Eingaben bereit.</stepresult>
    </step>
    <step>
      <cmd>Wechseln Sie in das gewünschte Verzeichnis
        mit dem cd-Kommando!</cmd>
      <stepresult>Die Eingabeaufforderung wird neu
        positioniert.</stepresult>
    <!-- weitere step-Elemente ... -->
  </steps>
  </steps>
  <result>...</result>
</taskbody>
```



Abb. 1: Struktur der Quell- und Zieldaten mit DITA-Map und -Topic

entweder aus dem Workshop heraus aufgerufen werden oder von der Kommandozeile aus: `hbc projekt.hhp`. In einer `hhp`-Datei werden die grundlegenden Projekt-Informationen für die spätere Verarbeitung abgelegt. Der Aufbau ist am unter Windows verbreiteten INI-Format orientiert:

#### [OPTIONS]

```
Compatibility=1.1 or later
Display compile progress=No
Full-text search=Yes
Language=0x407 Deutsch (Deutschland)
Compiled file=demohelp.chm
Contents file=demohelp.hhc
Index file=demohelp.hhk
Default topic=start_topic.html
Title=Demo-Hilfe
```

#### [FILES]

```
start_topic.html
...
schluss_topic.html
```

```
[MERGE FILES]
default.css
```

Unter `[OPTIONS]` werden insbesondere die benötigten Komponenten für die Inhaltsstruktur und den Index sowie einige Compiler-spezifische Angaben erwartet. Diese lassen sich leicht durch das Erstellen eines neuen Projekts mit dem Workshop ermitteln. Die Rubrik `[FILES]` nimmt zeilenweise die zum Projekt gehörenden HTML-Dateien auf, wobei sich hier auch weitere Formate angeben lassen. Zusatzdokumente können auch bei den `[MERGE FILES]` angegeben werden. Hier wurde nach gelegentlichen schlechten Erfahrungen sicherheitshalber die Standard-CSS-Datei einbezogen, die eigentlich direkt aus dem im HTML-Code enthaltenen `link`-Element übernommen werden sollte. Weiterhin muss die Inhaltsstruktur (*Table of Contents*) in einer `hbc`-Datei aufbereitet werden. Wiederum unter Nutzung des

Workshops lässt sich diese leicht testweise „zusammenklicken“ und inhaltlich studieren. Es handelt sich um einen HTML-Code mit dem in Listing 2 gezeigten Aufbau:

Die Navigationspunkte werden durch verschachtelte `ul`-Listen mit `object`-Elementen innerhalb von `li`-Elementen abgebildet. Über die Attribute `Name` und `Local` der `param`-Elemente erfolgt die Festlegung der Texte und Verweise. Der bei *Window Styles* angegebene Wert wird nach der Auswahl mehrerer Optionen im Workshop unter `TABLE OF CONTENTS PROPERTIES | STYLES` in die erzeugte `hbc`-Datei geschrieben. Dieser beschreibt das Aussehen und die Funktionalität der Baumstruktur in der fertigen `chm`-Datei. Anzumerken ist hier noch, dass die Elementnamen vom Workshop groß geschrieben werden und zwischen `<LI>` und `<OBJECT>` noch ein Leerzeichen eingefügt wird. In der Praxis spielen diese Details jedoch keine Rolle. Allerdings ist die Positionierung beider Elemente innerhalb einer Zeile von Bedeutung. Zeilenumbrüche toleriert der Compiler dazwischen nicht. Schließlich ist noch das Format der `hbc`-Indexdatei von Interesse, welches Analogien zur `hbc`-Inhaltsdatei aufweist und ebenfalls eine Sitemap darstellt:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>
<head>
<title>Index-File</title>
</head>
<body>
```

## Listing 2

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>
<head>
<title>TOC-File</title>
<!-- Sitemap 1.0 -->
</head>
<body>
<object type="text/site properties">
<param name="ImageType" value="Toc">
<param name="Window Styles" value="0x801427">
</object>
<ul><li><object type="text/sitemap">
<param name="Name" value="Startseite">
<param name="Local" value="start_topic.html">
</object>
<!-- weitere Inhalte -->
</li>
</ul>
</body>
</html>
```

## Listing 3

```
<xsl:template match="map">
<xsl:variable name="map_title" select="@title"/>
<xsl:variable name="map_hrefs" select="//topicref/@href"/>
<!-- Help-TOC-File (.hhc) erstellen -->
<xsl:call-template name="toc_out"/>
<!-- Help-Index-File (.hhk) erstellen -->
<xsl:call-template name="index_out">
<xsl:with-param name="map_hrefs" select=
"$map_hrefs"/>
</xsl:call-template>
<!-- Help-Project-File (.hhp) erstellen -->
<xsl:call-template name="hhp_out">
<xsl:with-param name="default_topic" select="fn:
replace(topicref[1]/@href,'\.xml','html')"/>
<xsl:with-param name="chm_title" select=
"$map_title"/>
<xsl:with-param name="map_hrefs" select=
"$map_hrefs"/>
</xsl:call-template>
</xsl:template>
```

```
<!-- Help-TOC-File (.hhc) erstellen -->
<xsl:call-template name="toc_out"/>
<!-- Help-Index-File (.hhk) erstellen -->
<xsl:call-template name="index_out">
<xsl:with-param name="map_hrefs" select=
"$map_hrefs"/>
</xsl:call-template>
<!-- pro Topic-File eine HTML-Datei erzeugen -->
<xsl:call-template name="html_out">
<xsl:with-param name="map_hrefs" select=
"$map_hrefs"/>
</xsl:call-template>
</xsl:template>
```

```
<ul><li><object type="text/sitemap">
<param name="Name" value="Willkommen">
<param name="Local" value="start_topic.html">
</object></li>
<!-- weitere Inhalte -->
</body>
</html>
```

Die Attribute *Name* und *Local* beschreiben das jeweilige Stichwort bzw. einen Suchtext und den Verweis auf die passende Hilfeseite. Weiterführende Informationen zum Aufbau der genannten Formate und zur Nutzung der Help-Tools von Microsoft sind unter [10] und [11] zu finden.

### Praktische Umsetzung

Mit den bisherigen Ausführungen sind die formalen technischen Voraussetzungen gelegt, um die genannten Projektdateien und die benötigten (X)HTML-Dokumente aus den DITA-Topics über die Map auszugeben und anschließend zu kompilieren. Hier kommt endlich XSLT und speziell *xsl:re-*

*sult-document* zum Einsatz. Da das Stylesheet mehr als 400 Codezeilen einnimmt, sollen im Folgenden nur die prinzipiellen Abläufe dargestellt werden. Das Haupt-Template greift auf das Wurzelement der DITA-Map zu und steuert den Aufruf der benannten Templates zur Generierung der benötigten Dokumente (Listing 3).

Zuvor wurden bereits einige Parameter definiert und mit Inhalten belegt, u. a. die Namen von Quell- und Zielverzeichnis, der CSS-Dateiname und das Bildverzeichnis. Die Ausgabeformate werden durch mehrere *xsl:output*-Elemente vorgegeben, ebenfalls eine Neuerung in XSLT 2.0 (Listing 4).

Über den Inhalt des Attributes *name* wird bei der folgenden Verarbeitung mit *xsl:result-document* Bezug auf den jeweiligen Ausgabebetyp genommen. Das dort verwendete Attribut *href* wird z. B. auf den Wert *html\_file* gesetzt. Die beiden anderen Formate sind für die *hbp*-Projektdatei sowie die Sitemap-Strukturen der *bhc*- bzw. *bhk*-Dateien zuständig. Der Zusatz *use-*

### Listing 4

```
<xsl:output method="text" name="project"
encoding="ISO-8859-1"/>
<xsl:output method="html" name="sitemap"
encoding="ISO-8859-1" indent="yes"
use-character-maps="li_ob"
doctype-public="-//IETF//DTD HTML//EN"/>
<xsl:output method="xhtml" name="html_file"
encoding="ISO-8859-1" indent="yes" omit-xml-
declaration="yes"
doctype-public="-//W3C//DTD XHTML 1.0 Strict//EN"
doctype-system="http://www.w3.org/TR/xhtml1/DTD/
xhtml1-strict.dtd"/>

<xsl:template name="html_out">
<xsl:param name="map_hrefs"/>
<xsl:for-each select="$map_hrefs">
<xsl:variable name="akt_topic_file" select="if
(fn:doc-available(.)) then fn:doc(.) else()"/>
<xsl:variable name="akt_html_file" select="fn:
replace(.,'\.xml';html)"/>
<xsl:result-document format="html_file"
href="{ $outdir }/{ $akt_html_file }">
<html xmlns="http://www.w3.org/1999/xhtml" xml:
lang="de" lang="de">
<head>
<title><xsl:value-of select="fn:
substring-before($akt_html_file,'html')"/></title>
<link rel="stylesheet" href="{ $cssfile }" type="text/
css"/>
<meta name="generator" content="dita2chm.xsl by
TM 2007"/>
</head>
<body>
<xsl:choose>
<xsl:when test="$akt_topic_file">
<xsl:apply-templates select="$akt_topic_file/topic |
$akt_topic_file/task"/>
</xsl:when>
<xsl:otherwise>
<h1>Kein DITA-Topic-File vorhanden</h1>
<p><em>Platzhalter-HTML-Dokument wurde
verwendet.</em></p>
</xsl:otherwise>
</xsl:choose>
</body>
</html>
</xsl:result-document>
</xsl:for-each>
</xsl:template>
```

### Anzeige

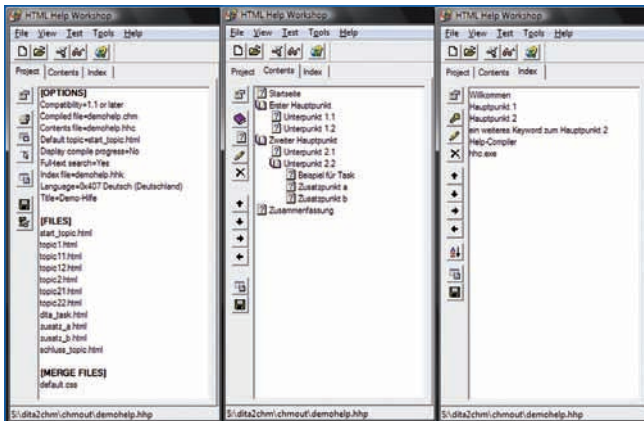


Abb. 2: Erzeugte Projektdateien im HTML Help Workshop

*character-maps* regelt in Verbindung mit *xsl:character-map*-Zuweisungen die Ersetzung von Sonderzeichen wie spitzen Klammern zur effektiven zeilenweisen Behandlung von *li/object* in den Sitemap-Dateien. Im oben angegebenen Start-Template wird die Variable *\$map\_hrefs* als Sequenz mit allen in der Map verfügbaren Verknüpfungen belegt. In einem *xsl:for-each*-Block können alle in der Sequenz enthaltenen Dateinamen abgearbeitet werden (Listing 5).

Die Funktionsweise der nicht näher ausgeführten Templates sollte sich aus dem auf der Heft-CD und online verfügbaren Code von *dita2chm.xsl* erschließen lassen [12]. Das gilt ebenfalls für die verwendeten neuen XPath-Funktionen *fn:doc()*, *fn:replace()* und *fn:tokenize()*. Nicht betrachtet wurden die Teil-Templates zur Umsetzung der DITA-Inhalte in HTML-Bestandteile. Allerdings sind diese schon fast trivial aufgebaut, denn wie eingangs bereits erwähnt, haben diverse DITA-Elemente eine direkte HTML-Entsprechung (u. a. Absätze und Listen). Andere lassen

sich leicht umformen, etwa *topic/title* zu *h1 section/title* zu *h2*, *row/entry* zu *tr/td*, *xref* zu *a* oder *image* zu *img*. Aus der DITA-Map ergibt sich unmittelbar die Struktur der *hbc*-Datei. Die Stichwörter für die *hbk*-Datei werden aus allen Topics ermittelt und in die *hbp*-Projektdatei müssen neben den genannten statischen Informationen alle beteiligten HTML-Seiten aufgenommen werden. Aufgerufen wird letztlich der Hilfecompiler über das Kommando *hbc demohelp.hhp* oder nach dem Laden der Projektdatei in den Workshop (vgl. Abbildung 2) durch Aktivierung des entsprechenden Menüpunkts oder Buttons. Sofern beim Laden des Projekts bzw. beim Kompilieren keine Fehler auftreten, kann die Aktion als erfolgreich betrachtet werden. Die fertige *chm*-Datei steht zur Anzeige im Help-Viewer bereit (Abbildung 3).

### Fazit

Das beschriebene Projekt zeigt einerseits die gute Eignung des DITA-Konzepts zur

Strukturierung von Onlinehilfen und andererseits die Anwendung neuer XSLT/XPath-Techniken. Mit dem veröffentlichten XSLT-Stylesheet wird übrigens keine vollwertige Konkurrenz zum DITA Open Toolkit [13] angestrebt. Dieses arbeitet zwar noch auf dem Level der 1.0-Versionen, enthält allerdings einen wesentlich ausgefeilteren Satz an Templates für alle relevanten DITA-Elemente und ermöglicht weitere Ausgabeformate wie RTF, XSL-FO oder JavaHelp. Eine automatisierte Produktion von Onlinehilfen ist mit dem vorgestellten und ausbaufähigen Stylesheet dennoch möglich und liefert bei moderatem Aufwand durchaus akzeptable Ergebnisse.



**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. Er verfasst regelmäßig Fachartikel und hält Vorträge zu Themen im XML-Umfeld. E-Mail-Kontakt: [thomas.meinike@hs-merseburg.de](mailto:thomas.meinike@hs-merseburg.de)

### Links & Literatur

- [1] W3C: [www.w3.org/TR/xslt20](http://www.w3.org/TR/xslt20)
- [2] W3C: [www.w3.org/TR/xpath20](http://www.w3.org/TR/xpath20)
- [3] W3C: [www.w3.org/TR/xpath-functions](http://www.w3.org/TR/xpath-functions)
- [4] FunctX XQuery Functions: [www.xqueryfunctions.com](http://www.xqueryfunctions.com)
- [5] XSLT 2.0 Quick reference: [www.dpawson.co.uk/xsl/rev2/xslt2.pdf](http://www.dpawson.co.uk/xsl/rev2/xslt2.pdf)
- [6] Meinike, T.: Wandlungsfähig – Neues in XSLT 2.0 und XPath 2.0; Internet Professionell 5/2007, S. 84-87 (Beispiele: [www.testticker.de/ipro/listings/download/0507xml.zip](http://www.testticker.de/ipro/listings/download/0507xml.zip))
- [7] OASIS, DITA Version 1.1 Specification: [docs.oasis-open.org/dita/v1.1/overview/overview.html](http://docs.oasis-open.org/dita/v1.1/overview/overview.html)
- [8] Closs, S.: Single Source Publishing – Topicorientierte Strukturierung und DITA; entwickler.press 2007
- [9] Microsoft HTML Help Downloads/HTML Help 1.4 SDK: [msdn2.microsoft.com/en-us/library/ms669985.aspx](http://msdn2.microsoft.com/en-us/library/ms669985.aspx)
- [10] Help-Guide, HTML Help: [www.help-guide.de/htmlhelp.htm](http://www.help-guide.de/htmlhelp.htm)
- [11] Unofficial (Preliminary) HTML Help Specification: [www.nongnu.org/chmspec/latest](http://www.nongnu.org/chmspec/latest)
- [12] Meinike, T.: [www.datenverdrahten.de/xslt2/dita2chm.html](http://www.datenverdrahten.de/xslt2/dita2chm.html)
- [13] DITA Open Toolkit: [dita-ot.sourceforge.net](http://dita-ot.sourceforge.net)



Abb. 3: Ansicht der erzeugten CHM-Hilfe