Eclipse-Online-Hilfe aus DocBook-Dateien erzeugen

Dein Freund und Helfer

VON PETER FRIESE

Jede Anwendung sollte über eine Online-Hilfe verfügen. Eclipse bietet hierfür eine ausgereifte Plattform an – das Plug-in *org.eclipse.help*. Doch die Erstellung der Hilfetexte ist und bleibt ein mühsames Geschäft. Der Einsatz von DocBook kann die Erstellung eines Hilfe-Plug-ins deutlich vereinfachen.

Wer für seine Eclipse-Plug-ins eine Online-Hilfe zur Verfügung stellen möchte, muss den Inhalt der Hilfe in Form von HTML-Dateien ablegen. Der Inhalt dieser Dateien wird auf der rechten Seite des Eclipse-Hilfefensters angezeigt, üblicherweise vom Standardbrowser des Systems (unter Windows ist das der Internet Explorer, auf Macs kommt Safari zum Einsatz und unter Linux wird die Mozilla Engine genutzt). Die Struktur der Hilfe legt der Autor mittels so genannter TOC-Dateien an (TOC = Table of Contents, Inhaltsverzeichnis). Listing 1 zeigt eine solche Datei für eine einfache Inhaltsstruktur. Wie zu erkennen ist, wird das Inhaltsverzeichnis durch <*topic*>-Elemente strukturiert, die beliebig tief geschachtelt werden können. Abbildung 1 zeigt das aus dieser Datei entstandene Inhaltsverzeichnis.

Jedem Eintrag wird im Inhaltsverzeichnis eine eigene HTML-Datei zugeordnet, die dann den Hilfetext enthält. Für eine kleine Anzahl von Hilfeseiten ist dies noch handhabbar, allerdings wird die Verwaltung des Inhaltsverzeichnisses mit wachsender Anzahl der Hilfeseiten zunehmend aufwendiger und fehleranfälliger. Auch die Pflege von Querverweisen zwischen den einzelnen Hilfeseiten ist ab einer gewissen Anzahl von Seiten nicht mehr zumutbar. Eine Möglichkeit zur Automatisierung muss her!

Tools

Zur Erstellung von Online-Hilfen werden üblicherweise Tools wie RoboHelp [1], DocToHelp [2] oder ähnliche eingesetzt. Die meisten dieser Tools sind jedoch kommerziell und daher für die meisten Open-Source-Projekte unerschwinglich. Hier soll ein Ansatz beschrieben werden, der vollständig auf frei verfügbaren Tools basiert. Kern des beschriebenen Tooling sind DocBook [3] sowie die DocBook XSL Stylesheets [4]. Die Hilfetexte werden gemäß der DocBook DTD verfasst, was zu mehreren Vorteilen führt:

- Die Hilfetexte können mit einem herkömmlichen XML-Editor verfasst werden, z.B. dem in WTP enthaltenen.
- Die Hilfetexte können durch den XML-Editor bzw. durch den XML-Parser validiert werden.
- Es stehen XSL-Transformationen für unterschiedliche Ausgabeformate zur Verfügung: Eclipse Help, HTML, JavaHelp, HTMLHelp, Windows Help sowie PDF.

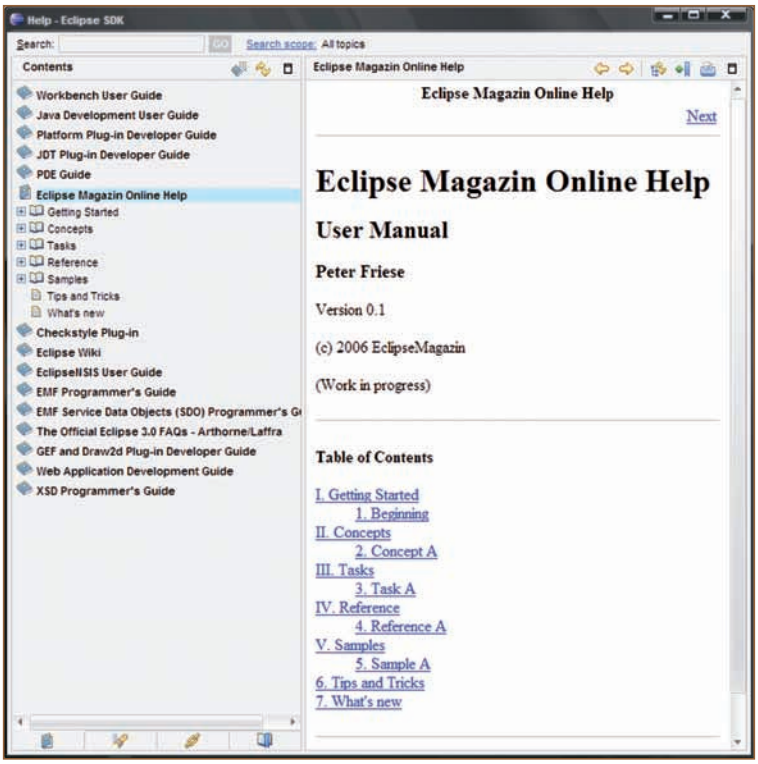

Abb. 1: Erste Fassung der Online-Hilfe

In diesem Artikel werden wir zunächst eine Umgebung schaffen, in der die Hilfetexte komfortabel editiert und transformiert werden können. Dazu werden wir uns WTP und Ant zunutze machen. Unter [5] wird eine ähnliche Umgebung beschrieben, die allerdings auf Ant verzichtet und somit nicht im Rahmen eines automatisierten Softwareentwicklungsprozess eingesetzt werden kann. Die hier beschriebene Lösung kann dagegen ohne Probleme als Bestandteil einer kontinuierlichen Integration (z.B. mit CruiseControl) eingesetzt werden – gerade in agilen Projekten ein nicht zu unterschätzender Vorteil.

Installation der Umgebung

Für die Erzeugung von Eclipse-Help-Plug-ins mithilfe von DocBook sind folgende Tools erforderlich:

- DocBook DTD, Version 4.3 (www.oasis-open.org/docbook/xml/4.3/)
- DocBook XSL Stylesheets, Version 1.69.1 (docbook.sourceforge.net)
- Ant (ant.apache.org), Version 1.6.x
- Saxon-XSL-Prozessor, Version 6.5.5 (saxon.sourceforge.net)

Darüber hinaus sind folgende Tools empfehlenswert, aber nicht zwingend erforderlich:

- Eclipse 3.2
- WTP 1.5

Alternativ kann auch jeder andere XML-Editor eingesetzt werden.

Falls es nicht sowieso bereits installiert ist, sollten Sie zunächst Ant sowie Eclipse und WTP installieren. Wenn Sie ausschließlich innerhalb Eclipse arbeiten, können Sie sich die Installation von Ant sparen, da Eclipse bereits eine Ant-Integration enthält. Die anderen Bestandteile unserer DocBook-Umgebung sollten Sie in das Verzeichnis entpacken,

Listing 1

Inhaltsverzeichnis der Online-Hilfe (*toc.xml*)

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<toc label="Android User Guide" topic="index.html">
 <topic label="Getting Started" href="pt01.html">
  <topic label="Beginning" href="ch01.html"/>
 </topic>
 <topic label="Concepts" href="pt02.html">
  <topic label="Concept A" href="ch02.html"/>
 </topic>
 <topic label="Tasks" href="pt03.html">
  <topic label="Task A" href="ch03.html"/>
 </topic>
 <topic label="Reference" href="pt04.html">
  <topic label="Reference A" href="ch04.html"/>
 </topic>
 <topic label="Samples" href="pt05.html">
  <topic label="Sample A" href="ch05.html"/>
 </topic>
 <topic label="Tips and Tricks" href="ch06.html"/>
 <topic label="What's new" href="ch07.html"/>
</toc>
```

Listing 2

Build-Skript für Transformation DocBook zu Eclipse Help

```
<?xml version="1.0"?>
<project name="Eclipse Online Help"
  default="build.doc.html">
  <property file="build.properties" />

  <path id="saxon.classpath">
    <pathelement location="${saxon.lib.path}" />
  </path>

  <target name="build.doc.html">
    <java fork="true" dir="${output.dir}"
      failonerror="true"
      classname="com.icl.saxon.StyleSheet">
      <classpath refid="saxon.classpath" />
      <arg value="${xdocs.dir}/index.xml" />
      <arg value="${stylesheet}" />
      <!-- customize -->
    </java>
  </target>

</project>
```

Listing 3

Properties für das Build-Skript

```
xdocs.dir=${basedir}/xdocs
output.dir=../de.eclipsemagazin.help
stylesheet=${docbook-xsl.eclipse-xsl.file}

docbook.dir=D:/develop/libraries/docbook/
                                    docbook-xml-4.3
docbook-xsl.dir=D:/develop/libraries/docbook/
                                    docbook-xsl-1.69.1
docbook-xsl.eclipse-xsl.file=${docbook-xsl.dir}/
                                    eclipse/eclipse.xsl
docbook-xsl.html-xsl.file=${docbook-xsl.dir}/html/
                                    chunk.xsl

saxon.lib.dir=D:/develop/libraries/saxon/saxon6.5.5-bin
saxon.lib.path=${saxon.lib.dir}/saxon.jar
```

in dem Sie auch andere Bibliotheken ablegen.

Das Build-Skript

Das Zusammenspiel aller Bestandteile wird durch ein recht überschaubares Ant-Skript gesteuert, das die Kompilierung unserer Hilfetexte steuert (Listing 2). Das Skript ist so generisch gehalten, dass es komplett über eine externe Properties-Datei konfiguriert werden kann (Listing 3).

Bei genauer Betrachtung fällt auf, dass der XSL-Prozessor nicht über den seit einiger Zeit verfügbaren Ant-Task *style* bzw. *xslt* aufgerufen wird, sondern mittels des *java*-Tasks. Um zu verstehen, warum dieses Vorgehen gewählt wurde, hier einige Hintergrundinformationen: Eclipse setzt voraus, dass die Datei *plugin.xml* wohl formatiert ist. Insbesondere ist es essenziell, dass jedes Element auf einer neuen Zeile beginnt. Die DocBook XSL Stylesheets unterstützen dies, indem die meisten Stylesheets einen Parameter *indent* anbieten, der den Zeilenumbruch sowie eine automatische Einrückung aktiviert. Der im JDK enthaltene XSL-Prozessor unterstützt dieses Feature nicht, ganz im Gegensatz zu Saxon. Es war jedoch nicht möglich, Saxon zuverlässig in den Klassenpfad von Ant zu integrieren, was für die Integration in den *style/xslt*-Task notwendig wäre. Der hier gewählte Weg (Aufruf über den *java*-Task und Setzen des Klassenpfads innerhalb des Build-Skripts funktioniert zuverlässig und ist reproduzierbar, ohne dass man in die Ant-Distribution eingreifen muss.

Der Hilfetext

Wenden wir uns nun der Erstellung des Hilfetexts an sich zu. DocBook unterstützt diverse Dokumenttypen, unter anderem *book* und *article*. Obwohl das von uns verwendete Stylesheet beide Dokumenttypen verarbeiten kann, werden wir für unser Beispiel den Dokumententyp *book* verwenden. So können wir den Hilfetext später sehr gut in ein PDF-Dokument umwandeln und aus ein und derselben Quelle sowohl die Online-Hilfe als auch das Benutzerhandbuch generieren.

Zunächst verfassen wir den Hilfetext in einer einzigen Datei (Listing 4). Dies hat den Vorteil, dass wir uns erstmal keine Gedanken über eine Aufteilung auf einzelne Dateien und die Zusammenführung dieser Dateien machen müssen. Später werden wir den Hilfetext auf separate Dateien verteilen, um die Zusammenarbeit von mehreren Hilfeautoren zu ver-

Listing 4

Gerüst für eine Online-Hilfe

```
<?xml version='1.0'?>
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML
        V4.3//EN" "file:/D:/develop/libraries/docbook/
                    docbook-xml-4.3/docbookx.dtd" >
<book>
  <bookinfo>
    <title>Eclipse Magazin Online Help</title>
    <subtitle>User Manual</subtitle>
    <releaseinfo>Version 0.1</releaseinfo>
    <pubdate>(Work in progress)</pubdate>
    <authorgroup>
      <author>
        <firstname>Peter</firstname>
        <surname>Friese</surname>
      </author>
    </authorgroup>
    <legalnotice>(c) 2006 EclipseMagazin</legalnotice>
  </bookinfo>
  <toc />
  <part>
    <title>Getting Started</title>
    <chapter>
      <title>Beginning</title>
      <para>
      </para>
    </chapter>
  </part>
  <part>
    <title>Concepts</title>
    <chapter>
      <title>Concept A</title>
      <para>
        Describe concept a.
      </para>
    </chapter>
  </part>
  <part>
    <title>Tasks</title>
    <chapter>
      <title>Task A</title>
      <para>
      </para>
    </chapter>
  </part>
  <part>
    <title>Reference</title>
    <chapter>
      <title>Reference A</title>
      <para>
      </para>
    </chapter>
  </part>
  <part>
    <title>Samples</title>
    <chapter>
      <title>Sample A</title>
      <para>
      </para>
    </chapter>
  </part>
  <chapter>
    <title>Tips and Tricks</title>
    <para></para>
  </chapter>
  <chapter>
    <title>What's new</title>
    <para>What's new in EclipseMagazin?</para>
  </chapter>
</book>
```

Automatisierung

Um das Build-Skript automatisch beim Speichern der Hilfetexte aufzurufen, kann man sich die Eclipse-Build-Architektur zunutze machen. Neben der bereits in vorangegangen Artikeln beschriebenen Möglichkeit, eigene Builder als Plug-in zu schreiben, kann man Builder auch ganz einfach über den Properties-Dialog eines Projekts definieren:

- Öffnen Sie den Properties-Dialog des Projekts *de.eclipsemagazin.help.source*.
- Wählen Sie auf der linken Seite den Eintrag Builders.
- Klicken Sie rechts auf New und
- wählen Sie im nun erscheinen Dialog Ant Build.
- Eclipse öffnet einen *Launch Configuration*-Dialog, in dem das Build-Skript sowie das auszuführende Target anzugeben sind.
- Legen Sie zunächst einen sinnvollen Namen für den neuen Builder fest, z.B. *DocBook*.
- Wählen Sie also mit dem Button Browse Workspace das Build-Skript aus und
- wechseln Sie dann auf den Reiter Targets.
- Im Abschnitt *Auto Build* klicken Sie auf den Button Set Targets und
- wählen Sie aus der Liste der Targets den Eintrag *build.doc.html*.

Fertig! Immer dann, wenn Sie eines der DocBook-XML-Dokumente speichern oder Eclipse aus irgendeinem anderen Grund den Workspace baut, wird das Build-Skript ausgeführt und Ihre Hilfetexte werden kompiliert.

einfachen. Eine kurze Übersicht über die verwendeten DocBook-Elemente ist dem Kasten „Wichtige DocBook-Elemente“ zu entnehmen, eine komplette Referenz ist online unter [8] einzusehen.

Hilfe kompilieren

Legen Sie zwei Projekte an, *de.eclipsemagazin.help* und *de.eclipsemagazin.help.source*. Wählen Sie als Projekttyp bitte jeweils *Simple Project*. Im Projekt *de.eclipsemagazin.help.source* legen Sie einen Unterordner *xdocs* an und erzeugen dort eine Datei *index.xml* mit dem Inhalt aus Listing 4. Erstellen Sie dann im Wurzelordner dieses Projekts die Dateien *build.xml* und *build.properties* mit dem jeweiligen Inhalt aus den Listings 2 und 3.

Führen Sie nun das Ant-Skript aus. Im Projekt *de.eclipsemagazin.help* entsteht nun eine ganze Reihe von Dateien: Die Datei *plugin.xml* enthält die Deklaration des Hilfe-Plug-ins, die Datei *toc.xml* die Struktur des Inhaltsverzeichnis und die **.html*-Dateien den Inhalt der Hilfe. Der Kasten „Automatisierung“ enthält eine Zusammenfassung dieser Schritte.

Einstellung	Bedeutung
html.stylesheet	Name der CSS-Datei, die zur Formatierung der HTML-Dateien verwendet wird
chunk.first.sections	Erste Sektion ebenfalls in separate Datei schreiben
chunk.section.depth	Bis zu welcher Gliederungstiefe sollen Einzeldateien ausgegeben werden
base.dir	Verzeichnis, in dem die HTML-Dateien abgelegt werden sollen
use.id.as.filename	IDs der DocBook-Strukturelemente zur Bildung der Dateinamen benutzen
suppress.navigation	Navigationslinks (oben und unten) ausblenden
chapter.autolabel	Arabische Nummerierung der Kapitel (1, 2, 3 ...)
generate.section.toc.level	Bis zu welcher Gliederungsebene sollen Kapitelverzeichnisse erzeugt werden?
table.borders.with.css	Formatierung von Tabellen mithilfe von CSS Styles
table.cell.border.thickness	Legt Rahmenbreite für Tabellenrahmen fest
html.cellspacing	Analog zum Attribut *cellspacing* für HTML-Tabellen
html.cellpadding	Analog zum Attribut *cellpadding* für HTML-Tabellen
html.cleanup	HTML-Code formatieren
generate.toc	Legt fest, für welche Elemente ein Inhaltsverzeichnis angelegt wird
Weitere Parameter sind unter [7] dokumentiert.	

Tabelle 1: Stylesheet-Anpassungen

Wichtige DocBook-Elemente

DocBook verfügt über eine Vielzahl von XML-Tags, die zur Strukturierung Ihrer Texte genutzt werden können. Eine erschöpfende Liste ist unter [6] zu finden. Für einen ersten Start sind in der folgenden Tabelle die wichtigsten Tags kurz zusammengefasst:

Element	ist Subelement von	Funktion
article	N/A	Wurzelelement für Artikel
book	N/A	Wurzelelement für Bücher
title	*article, book, part, chapter, section*	Titel des Elements
subtitle	*article, book*	Untertitel
articleinfo	*article*	Gruppiert Metainformationen zum Artikel
bookinfo	*book*	Gruppiert Metainformationen zum Buch
releaseinfo	*articleinfo, bookinfo*	Version des Dokuments
pubdate	*articleinfo, bookinfo*	Veröffentlichungsdatum des Dokuments
authorgroup	*articleinfo, bookinfo*	Gruppiert die Autoreninformationen
author	*authorgroup*	Gruppiert Informationen zu einem Autoren
firstname	*author*	Vorname des Autors
surname	*author*	Nachname des Autors
email	*author*	E-Mail-Adresse des Autors
toc	*article, book*	Inhaltsverzeichnis
part	*book*	Teil eines Buchs
chapter	*Part*	Kapitel
section	*article, chapter, section*	Abschnitt, kann beliebig tief geschachtelt werden
para	*section*	Absatz
xref	*Para*	Querverweis

Hilfe starten

Nun ist der Moment der Wahrheit gekommen: Legen Sie eine neue Eclipse Launch Configuration an und stellen Sie sicher, dass das neue Hilfe-Plug-in aktiviert ist. Am einfachsten geht das, indem Sie auf der Registerseite *Plug-ins* die Option *Launch with all workspace and enabled external plug-ins* markieren. Das Ergebnis ist in Abbildung 1 zu sehen. Diese erste Fassung der Online-Hilfe enthält noch einige kleine Unschönheiten:

- Die Angaben in der Datei *plugin.xml* (Plug-in-Provider, Plug-in-Name, Plug-in-ID) sind noch nicht passend.
- Die Hilfetexte sind noch nicht mit dem Eclipse CSS Stylesheet formatiert.
- Die Hilfetexte enthalten noch unnötige Navigationslinks am oberen und unteren Bildrand.

Finetuning

Im Folgenden werden wir diese Kleinigkeiten korrigieren. Das Wichtigste ist, die Informationen in der Datei *plugin.xml* zu korrigieren. Dazu übergeben wir dem Stylesheet-Prozessor die zusätzlichen Parameter mithilfe des *arg*-Elements (Listing 5).

Listing 5

```
<target name="build.doc.html">
  <java fork="true" dir="${output.dir}"
    failonerror="true"
    classname="com.icl.saxon.StyleSheet">
    <classpath refid="saxon.classpath" />
    <arg value="${xdocs.dir}/index.xml" />
    <arg value="${stylesheet}" />
    <arg value="eclipse.plugin.id=
                              ${eclipse.plugin.id}" />
    <arg value="eclipse.plugin.name=
                            ${eclipse.plugin.name}" />
    <arg value="eclipse.plugin.provider=
                          ${eclipse.plugin.provider}" />
  </java>
</target>
```

Um die restlichen Einstellungen anzupassen, werden wir einen so genannten Customization Layer einsetzen. Hierbei handelt es sich um eine XSL-Datei, welche zunächst die ursprüngliche Transformation importiert und danach bestimmte Parameter überschreibt bzw. ergänzt. Die in Listing 6 vorgenommenen Einstellungen werden in Tabelle 1 kurz erläutert. Als letzte Anpassung sorgen wir nun noch dafür, dass die CSS-Datei *book.css* an die richtige Stelle kopiert wird:

```
<target name="build.doc.html">
  ...
  <copy file="${css.file}" todir="${output.dir}"/>
</target>
```

With a little Help from my Friends

Wenn die DocBook-Hilfedatei irgendwann zu groß oder zu unübersichtlich geworden ist oder man mit mehreren Autoren gleichzeitig an den Hilfetexten arbeiten möchte, sollte man die Hilfetexte modularisieren. Der einfachste Weg führt über XML Entities. In der Hauptdatei (in unserem Fall *index.xml*) definiert man für jedes Modul eine Entity und reiht diese Entities dann in der gewünschten Reihenfolge aneinander (Listing 7).

Fazit und Ausblick

Durch die Verwendung von DocBook gewinnt die Erstellung von Online-Hilfe und Benutzerhandbuch eine neue Qualität: Aus einer Quelle können die unterschiedlichen Zielformate erzeugt werden. Als positiver Nebeneffekt lässt sich beobachten, dass die Entwickler wieder mehr Lust bekommen, ihre Software zu dokumentieren. Das liegt wahrscheinlich daran, dass das Schreiben einer DocBook-Datei sich so ähnlich anfühlt wie das Schreiben einer Java-Datei und das Endergebnis durch einen Kompilierlauf erzielt wird.

Da mithilfe von XSL nahezu beliebige Transformationen möglich sind, kann man DocBook auch für andere Hilfe-Formate verwenden. Die DocBook-XSL-Distribution enthält bereits Stylesheets für HTMLHelp, JavaHelp und man pages. Aber auch eigene Transformationen lassen sich erstellen, z.B. für Eclipse Cheat Sheets [6].

Listing 6

Customization Layer

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/
                          XSL/Transform" version="1.0">

  <xsl:import href="file:/D:/develop/libraries/
    docbook/docbook-xsl-1.69.1/eclipse/eclipse.xsl" />

  <xsl:param name="html.stylesheet">../book.css
                                    </xsl:param>
  <xsl:param name="chunk.first.sections" select="1" />
  <xsl:param name="chunk.section.depth" select="3" />
  <xsl:param name="base.dir" select="'contents/'">
                                    </xsl:param>
  <xsl:param name="use.id.as.filename" select="0" />
  <xsl:param name="suppress.navigation" select="1" />
  <xsl:param name="chapter.autolabel" select="0" />
  <xsl:param name="generate.section.toc.level"
                              select="0"></xsl:param>

  <xsl:param name="table.borders.with.css" select="0">
                                    </xsl:param>
  <xsl:param name="table.cell.border.thickness"
                             select="'1'"></xsl:param>
  <xsl:param name="html.cellspacing" select="'0'">
                                    </xsl:param>
  <xsl:param name="html.cellpadding" select="'10'">
                                    </xsl:param>
  <xsl:param name="html.cleanup" select="1">
                                    </xsl:param>

  <xsl:param name="generate.toc">
    appendix  toc,title
    article/appendix nop
    article     toc,title
    book        nop
    chapter     nop
    part        nop
    preface     toc,title
    qandadiv  toc
    qandaset  toc
    reference  toc,title
    sect1       toc
    sect2       toc
    sect3       toc
    sect4       toc
    sect5       toc
    section     toc
    set         toc,title
  </xsl:param>

</xsl:stylesheet>
```

Listing 7

Modularisierung mit XML Entities

```
<?xml version='1.0'?>
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook
   XML V4.3//EN" "file:/D:/develop/libraries/docbook/
                          docbook-xml-4.3/docbookx.dtd" [
  <!ENTITY gettingstarted SYSTEM „gettingstarted.xml">
  <!ENTITY concepts SYSTEM "concepts.xml">
  <!ENTITY tasks SYSTEM "tasks.xml">
  <!ENTITY reference SYSTEM "reference.xml">
  <!ENTITY samples SYSTEM "samples.xml">
  <!ENTITY tipsntricks SYSTEM "tipsntricks.xml">
  <!ENTITY whatsnew SYSTEM "tipsntricks.xml">
]>
<book>
  <bookinfo>
    <title>Eclipse Magazin Online Help</title>
    <subtitle>User Manual</subtitle>
    <releaseinfo>Version 0.1</releaseinfo>
    <pubdate>(Work in progress)</pubdate>
    <authorgroup>
      <author>
        <firstname>Peter</firstname>
        <surname>Friese</surname>
      </author>
    </authorgroup>
    <legalnotice>(c) 2006 EclipseMagazin</legalnotice>
  </bookinfo>
  <toc />
  &gettingstarted;
  &concepts;
  &tasks;
  &reference;
  &samples;
  &tipsntricks;
  &whatsnew;
</book>
```

Peter Friese arbeitet als Softwarearchitekt bei Lufthansa Systems in Hamburg. Er ist Committer für FindBugs und AndroMDA. Derzeit arbeitet Peter an der Entwicklung einer Eclipse-basierten IDE für AndroMDA. Kontakt und weitere Informationen: peter.friese@lhsystems.com, f3.tobject.de.

Links & Literatur

[1] RoboHelp: www.macromedia.com/software/robohelp/

[2] DocToHelp: www.componentone.com

[3] DocBook Technical Committee Document Repository: www.oasis-open.org/docbook/

[4] The DocBook Project: docbook.sourceforge.net

[5] Chris Aniszczyk, Lawrence Mandel: Authoring with Eclipse: www.eclipse.org/articles/Article-Authoring-With-Eclipse/AuthoringWithEclipse.html

[6] Manfred Heiland: Spickzettel: Informationsmanagement à la Eclipse, in *Eclipse Magazin* Vol. 6

[7] DocBook XSL Parameter Reference: docbook.sourceforge.net/release/xsl/current/doc/html/

[8] DocBook: The Definitive Guide: www.docbook.org/tdg/en/html/docbook.html