x-magic Plugin for DITA
Open Toolkit 実演サイト
DITAのことならmagicalにおまかせください

まじかるテクノロジー有限会社
PDF制作日： 2014-05-19

おしらせnews

更新時に皆様にお知らせしたい事項を記述しています

WEBサイトをリニューアルしました。今回は変換エンジン【x-magic】を独自に開発してDITAで制作しました！（トップページも含めて全てDITA-OTで変換しています）

第一弾は、wordpressを使用していた旧サイトのメニューや機能をDITA流に再現してみました。今後、モバイル版やPDF版の公開も予定しています。

DITA関連の連載も本格化させます。質問やリクエストなど気軽にメールください。

DITAとは？what's dita?

DITAと当社の関わり合いです

Darwin Information Typing Architecture (DITA)は、技術情報を制作・発行・配布するためのXMLに基づいたアーキテクチャです。DITAは、OASIS（構造化情報標準促進協会）の支援の下にIBMが開発し、コミュニティに寄贈されました。

当社は、DITA Open Toolkit(DITA-OT)をはじめ、Saxon XSLT, XPath, XSL-FO, Apache FOP, AH Formatter, XHTML/CSS, Apache Ant, Java, eclipse, oXygen XML Editor(Auther)といった、DITAを導入する上で必要不可欠な関連技術を研究し、メーカー向けに変換エンジンの開発と技術支援を行っています。

DITAは大規模なプロジェクトに対応するものですが、小規模なプロジェクトでも使えます。大規模な環境では相応のCCMSを導入する必要がありますが、小規模であればオープンソース中心に環境を構築すれば初期コストも殆どかかりません。現に、このサイトはDITA文書のオーサリングにoXygen XML Editorを使用した以外はソースのリビジョン管理も含めオープンソースのみを使い、スタッフ2名で他業務の合間に制作しています。

目次

第 1 章 会社概要

まじかるテクノロジーについて

会社概要

名称	まじかるテクノロジー有限会社
代表者	井上 智勇 (Toshio Inoue)
事業内容	1. XSLTスタイルシート開発 2. WEBシステム開発 3. 各種プログラム開発 4. ドキュメント制作 5. DITA導入支援 6. XML技術支援
主要取引先	*NEC*マネジメントパートナー株式会社
DITAシステム導入事例	• NEC全社システム DITA-XHTML変換モジュール
本社所在地	〒252-0254 神奈川県相模原市中央区下九沢65-4
問い合わせ先	MAGICAL DITA Team: dita@magical-tec.co.jp

会社沿革

1998年	NECドキュメンテクス(現NECマネジメントパートナー)と取引開始
1999年	まじかるテクノロジー有限会社設立
2012年～	NECグループ向けDITAシステム開発案件の受託を開始

情報セキュリティ基本方針

当社は、社会の発展に寄与するため、お取引先様からお預かりした情報資産を守ることが責務と考え、情報セキュリティ基本方針を定め、実践することを宣言します。

- 情報資産に対する不正な侵入、漏えい、改ざん、紛失・盗難、破壊、利用妨害などが発生しないよう努めます。
- 万一情報資産にセキュリティ上の問題が発生した場合、速やかに報告するとともに、原因を迅速に究明し、被害を最小限に止め、再発防止に努めます。
- 情報セキュリティに関係する法令、国が定める指針、取引パートナーが定めるガイドライン、その他の社会的規範を遵守します。
- 以上の活動を継続的に見直し、随時改善に努めていきます。

第 2 章 DITAの使い方 magical step

DITAの導入やカスタマイズの実施を検討されている方を対象としたトピックです。

2.1 まず変換してみよう！

実際にDITA-OTを使って、試してみましょう。（2012年12月06日公開）

ドキュメント制作の構造化手法

ドキュメントを効率よく制作・メンテし、多言語対応も簡単に出来る手法として、世界ではDITAというXMLを使った構造化言語が技術マニュアルを中心に使われています。

特に、構造化に適した技術文書を軸に欧米ではかなり広まっています。しかし、日本では、技術文書と言えどもレイアウトデザインや、冗長な言葉使いが壁となって、なかなか広まっていない現実があります。

私たちは、ワンソースマルチユースという一つのソースから、複数の出力（PDF、XHTML、CHM等のヘルプ形式、EPUB）が出来るDITAソースをDITA-OT（DITA-OT Open Toolkit）というフリーな変換システムを使って実践してみます。

まずは・・・どんな出力が出来るかを実際にDITA-OTを使って、試してみましょう。

DITA-OT Open Toolkitのインストール

まずは、核となるDITA-OT Open Toolkitをインストールします。

ダウンロードサイトは次のURLです。

http://sourceforge.net/projects/dita-ot/files/

ここで迷うのは、どのバージョンを使用するか？です。すぐに、DITAを業務で使うなら、一番安定していると思われるDITA-OT1.5.4の安定版をお勧めします。上記サイトから、[DITA-OT Stable Release]をクリックし、次に[DITA Open Toolkit 1.5.4]をクリック、全部が入った[DITA-OT1.5.4_full_easy_install_bin.zip]をダウンロードすると良いでしょう。

今回は、実験として使いますので、11/26のDITA OT 1.7.M4最新版で開発中のものを使ってみます。上記サイトから、[DITA-OT Latest Test Build]-[DITA OT 1.7.M4]-[DITA-OT1.7.M4_full_easy_install_bin.zip]をダウンロードします。解凍して出来たフォルダ「DITA-OT1.7.M4」をルート直下で使います。まず、ドキュメントを見ましょう。C:\DITA-OT1.7.M4\doc\userguide.pdfを開きます。これと同じものが、インターネット上でも見れます。

http://dita-ot.sourceforge.net/dev/

開発中ですので、バージョンM4ですが、ドキュメントはまだ「M2」ですね。

すぐに、サンプルDITAファイルを使ってPDF化したいところですが、待ってください。他にも環境設定が必要です。

ちなみに、「full_easy_install」パッケージにより次のライブラリーがすでに無償でインストールされています。

- Apache Ant, version 1.8.4
- Apache Catalog Resolver, version 1.1
- Apache Commons Codec, version 1.4
- Apache FOP, version 1.0
- ICU for Java, version 49.1
- Apache Xerces, version 2.11.0
- Saxon, version 9.1

変換するのに必要な他のライブラリーはJAVAです。

下記サイトからダウンロードして、インストールしてください。

JRE or JDK, version 6 or later

http://www.oracle.com/technetwork/java/javase/overview/index.html

JDK (Java Development Kit)のインストールを推奨します。

HTMLヘルプを生成するのであれば、合わせて次のものもインストールしておいてください。

Microsoftヘルプワークショップ

http://msdn.microsoft.com/en-us/library/windows/desktop/ms669985%28v=vs.85%29.aspx

これで、準備完了です。

実際に変換してみる

DITA-OT1.7.M4パッケージ付属のサンプルを実際にPDF変換してみましょう。

1205出力サンプル.zip

次の手順で動かします。

1. コマンドプロンプトを開きます。

```
C:\DITA-OT1.7.M4\startcmd.bat
```

2. Ant実行もできますが、ここではJAVAで実行してみます。

   次のように、打ってください。

```
java -jar lib/dost.jar /i:C:\DITA-OT1.7.M4\samples\taskbook.ditamap
/transtype:pdf /outdir:out\pdf
```

3. 最後の方で、次のメッセージが出れば、うまくPDF変換が出来ています。

```
BUILD SUCCESSFUL
Number of Fatals : 0
Number of Errors : 0
Number of Warnings : 0
```

4. C:\DITA-OT1.7.M4\out\pdfフォルダ内のtaskbook.pdfを開いて、うまく出来ていれば完了です。
5. 次に他の変換も行ってみましょう。
   - XHTML変換

```
java -jar lib/dost.jar /i:C:\DITA-OT1.7.M4\samples\taskbook.ditamap /
transtype:xhtml /outdir:out\xhtml
```

   C:\DITA-OT1.7.M4\out\xhtmlフォルダ内のindex.htmlファイルを始めに開いてみてください。

- XHTML+JavaScript変換

```
java -jar lib/dost.jar /i:C:\DITA-OT1.7.M4\samples\taskbook.ditamap /
transtype:tocjs /outdir:out\tocjs
```

C:\DITA-OT1.7.M4\out\tocjsフォルダ内のtaskbook.htmlファイルを開いてみてください。

Microsoftヘルプワークショップをインストールしていれば、次の変換もしてみてください。

- XHTML+JavaScript変換

```
java -jar lib/dost.jar /i:C:\DITA-OT1.7.M4\samples\taskbook.ditamap /
transtype:htmlhelp /outdir:out\htmlhelp
```

C:\DITA-OT1.7.M4\out\htmlhelpフォルダ内のtaskbook.chmファイルを開いてみてください。

いかがでしょうか？

C:\DITA-OT1.7.M4\out\xhtml/tocjs/htmlhelpの各フォルダに変換出力が出来ているでしょうか。

次回は、日本語化に挑戦します。

ご意見・ご質問等があれば、お気軽にお尋ねください。

2.2 日本語変換に挑戦してみよう！

実際にDITA-OTを使って、日本語変換に挑戦してみましょう。(2012年12月19日公開)

日本語変換する前に

前回は、DITA-OT(Open Toolkit)をインストールし、DITA-OT1.7.4.M4付属のサンプルを使って、PDF、XHTML、TOCJS(XHTML+JavaScrip)、HTMLHELPに変換しました。

今回は、日本語変換に挑戦です。

DITAはグローバル対応を考慮した作りになっていますし、DITA-OTによる変換もグローバル対応になっています・・・が、実際に日本語変換してみると、オリジナルのままではうまく変換されません。。。日本語変換をうまくするには、少しDITA-OTをいじる必要があります。

まずは・・・前回のサンプルDITAファイルを日本語化して、どんな出力が出来るかを試してみましょう。前回インストールしたDITA-OT1.7.M4を使います。

まず、実際に日本語変換してみる

DITA-OT1.7.M4パッケージ付属のサンプルの一部を日本語に書き替えてPDF変換してみましょう。

1. サンプル（DITAファイル）の日本語化 日本語化したDITAファイルは次の3ファイルです。オリジナルのDITAファイル名に「-j」を付加しています。 *1219日本語samples.zip*（翻訳はgoogle翻訳を元にしています。）

```
C:\DITA-OT1.7.M4\samples\taskbook-j.ditamap
```

taskbook.ditamapを翻訳したファイルがベースです。ファイル内のトピックファイル名を次の日本語ファイル名に変更します。 installing.dita → installing-j.dita installstorage.dita → installstorage-j.dita

```
C:\DITA-OT1.7.M4\samples\taskbook\installing-j.dita
```

installing.ditaを翻訳したファイルがベースです。

```
C:\DITA-OT1.7.M4\samples\taskbook\installstorage-j.dita
```

installstorage.ditaを翻訳したファイルがベースです。

2. 前回同様にJAVAでPDF変換してみます。C:\DITA-OT1.7.M4\startcmd.batファイルをダブルクリックします。次のように打ってください。

```
java -jar lib/dost.jar /i:C:\DITA-OT1.7.M4\samples\taskbook-j.ditamap /
transtype:pdf /outdir:out\pdf
```

3. 最後の方で、次のメッセージが出れば、うまくPDF変換が出来ています。

```
BUILD SUCCESSFUL
Number of Fatals : 0
Number of Errors : 0
Number of Warnings : 0
```

注）皆さんが作ったファイルでPDFが出来ず、「BUILD FAILED」のメッセージが表示されていたら、作ったファイルの文字コードが「UTF-8」かどうかを確認してください。DITA-OTはグローバル対応が基本のため、スタイルシートを始めDITAソースも「UTF-8」が基本です。知らないうちに、「shift-jis」で記述したりしていませんか？ 日本語を記述する場合は、必ず「UTF-8」で開いているか確認してください。

4. C:\DITA-OT1.7.M4\out\pdfフォルダ内のtaskbook-j.pdfを開いてください。ブックマークの[装着]ページ、その下の階層の[ハードドライブまたは他のストレージをインストール]ページをクリックすると、日本語が「####」になっています。これは、指定されている日本語フォントがないためです。（オリジナルのDITA-OTでは、アドビさんの日本語書体を何故だか指定しています。）*taskbook-j日本語フォント無.pdf*

使っている日本語フォントを指定して、変換してみる

さあ、使っている日本語フォントを指定するために、DITA-OTをいじりますよ。。。修正すべきファイルは次の2ファイルです。

1.
```
C:\DITA-OT1.7.M4\plugins\org.dita.pdf2\cfg\fo\font-mappings.xml
```

もとのfont-mappings.xmlファイルは、font-mappings_org.xmlとかにリネームして保管してください。 KozMinProVI-Regularと書かれた3箇所を次のように修正します。 62行目 MS PGothic 88行目 MS PMincho 114行目 MS Gothic

注）指定する日本語フォントは、皆さんの環境にあるフォントにしてしてください。

もし、日本語フォントが無ければ、独立行政法人情報処理推進機構が提供するオープンソースのIPAフォントなどを使うこともできます。 *http://ossipedia.ipa.go.jp/ipafont/index.html* DL後、フォントが通常置かれるフォントフォルダにインストールすれば、自動的に検出されます。

2.
```
C:\DITA-OT1.7.M4\lib\configuration.properties
```

3行目 default.language = enをdefault.language = jaに修正します。ここで各DITAファイルのデフォルト言語を指定しています。

3. 再度、PDF変換してみます。開いたままのコマンドプロンプト画面で、次のように打ってください。

```
java -jar lib/dost.jar /i:C:\DITA-OT1.7.M4\samples\taskbook-j.ditamap /
transtype:pdf /outdir:out\pdf
```

4. C:\DITA-OT1.7.M4\out\pdfフォルダ内のtaskbook-j.pdfを開いてください。ブックマークの[装着]ページ、その下の階層の[ハードドライブまたは他のストレージをインストール]ページをクリックして、日本語がうまく表示されるようになっていれば完成です。 *taskbook-j日本語フォント有.pdf* 注）「BUILD SUCCESSFUL」のメッセージが表示されたのに、4項で変換されたPDFと同じ結果になるという場合は、まず、変換前に前のPDFであるtaskbook-j.pdfファイルを閉じていたか確認してください。DITA-OTでは、上書きするPDFファ

イルが開いていても、PDF変換が正常終了と判断されてしまいます。C:\DITA-OT1.7.M4\out\pdf\taskbook-j_pdf.logファイルを見ると、最後の方で次のメッセージが出ています。

```
[fop] Caused by: java.io.FileNotFoundException: C:\DITA-OT1.7.M4\out
\pdf\taskbook-j.pdf （プロセスはファイルにアクセスできません。別のプロセスが使用中です。）
```

taskbook-j.pdfファイルを閉じているのに変換がうまく行かない場合は、taskbook-j_pdf.logファイルをよく見てみてください。フォントが見つからないなどの、何らかのメッセージが残されていると思います。ヘルプが必要な場合は、是非お問い合わせください。

5. 次に他の変換も行ってみましょう。

- XHTML変換

```
java -jar lib/dost.jar /i:C:\DITA-OT1.7.M4\samples\taskbook-
j.ditamap /transtype:xhtml /outdir:out\xhtml-j
```

C:\DITA-OT1.7.M4\out\xhtml-jフォルダ内のindex.htmlファイルを始めに開いてみてください。

- XHTML+JavaScript変換

```
java -jar lib/dost.jar /i:C:\DITA-OT1.7.M4\samples\taskbook-
j.ditamap /transtype:tocjs /outdir:out\tocjs-j
```

C:\DITA-OT1.7.M4\out\tocjs-jフォルダ内のtaskbook.htmlファイルを開いてみてください。うまく行きましたか？ XHTML変換、XHTML+JavaScript変換ともに、UTF-8で出力されるため、DITA－OTをいじらなくても文字化けしません。Microsoftヘルプワークショップをインストールしていれば、次の変換もしてみてください。

- HTMLHELP変換

```
java -jar lib/dost.jar /i:C:\DITA-OT1.7.M4\samples\taskbook-
j.ditamap /transtype:htmlhelp /outdir:out\htmlhelp-j
```

C:\DITA-OT1.7.M4\out\htmlhelp-jフォルダ内のtaskbook.chmファイルを開いてみてください。これは・・・文字化けしていますね。C:\DITA-OT1.7.M4\samples\taskbook-j.ditamapファイル内のbookmapタグにxml:lang属性を追加します。 <bookmap id="taskbook"> → <bookmap id="taskbook" xml:lang="ja-jp"> 再度、HTMLHELP変換してみてください。今度は、うまく日本語が表示されていると思います。

いかがでしょうか？

C:\DITA-OT1.7.M4\out\xhtml-j/tocjs-j/htmlhelp-jの各フォルダに変換出力が出来ているでしょうか。

1219出力サンプル.zip

次回は、DITAソースを新規に書いてみます。

ご意見・ご質問等があれば、お気軽にお尋ねください。

2.3 DITAソースを書いてみよう！

実際に日本語DITAソースを書き、DITA-OTを使って、PDF化に挑戦してみましょう。(2013年2月1日公開)

DITAソースを書く前に

前回は、DDITA-OT付属のサンプルDITAファイルを一部日本語にして、PDF、XHTML、TOCJS(XHTML+JavaScrip)、HTMLHELPに変換しました。

今回は、新規に日本語のDITAソースを書いてPDFにしてみます。

DITAは、マップ（map）ファイルが一つと、トピック（topic）ファイルが複数集まったソースです。まずトピックを作りましょう。汎用トピックを頂点として、機能を制限したタイプとしてconceptとtask、reference、glosarryがあります。もちろん、自分で特殊化を行えば様々なタイプを作ることが出来ますし、実際、学習関係とかいろいろなタイプが出てきています。特殊化を行うには十分な知識と工数が掛かりますので、ここでは取り扱いません。

さて、皆さんはどのタイプを使えばよいのでしょうか？

- 一般的に、概要・概念などを記述する場合はconceptタイプ
- 手順などの段階的な事項を扱う場合はtaskタイプ
- 参照、ライブラリィ、用語集などにはreference、glosarrタイプ

というふうに推奨されています。

ここでは、第一回目の「まず変換してみよう！」のページの一部をDITA化してみましょう。どのタイプでしょうか・・・そうです。上記タイプで言えば、conceptタイプとtaskタイプの両方ですね。前半がconceptタイプ、後半がtaskタイプになるでしょう。そして、取りまとめるマップと呼ばれる目次のようなものを作る必要があります。

今回は、まず前半のconceptタイプを作って、うまくPDF変換が出来るかを試してみましょう。今回もインストール済みのDITA-OT1.7.M4を使います。注）1/2に安定版がリリースされましたが、1/14にDITA-OT1.7.1、1/28にDITA-OT1.7.2がリリースされています。。DITA-OTは使い勝手が良くなるように、常に進化しているのですが。。。もう少し落ち着くまで、DITA-OT1.7.M4を使います。

DITAソースを書いてみよう1（conceptタイプのトピックを作ってみる）

まずは、第一回目の「まず変換してみよう！」のページのどこまでを1トピックにするか決めます。手順が始まる『実際に変換してみる』の前までが、conceptタイプのトピックになりますが、ここでは、『DITA-OT Open Toolkitのインストール』の前でも切って二つのトピックにします。

conceptタイプのトピックを作るにあたってはお決まりのパターンがあります。UTF-8文字コードが使えるエディタで、次のように記述してみましょう。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE concept PUBLIC "-//OASIS//DTD DITA Concept//EN" "concept.dtd">
<concept id="ユニークなID（ファイル名で良い）" xml:lang="ja-jp">
 <title>タイトルを入れる</title>
 <shortdesc></shortdesc>
 <conbody>
  <p>本文です。</p>
 </conbody>
</concept>
```

- m1-1.ditaファイル

  さて、準備が整ったところで、実際に『DITA-OT Open Toolkitのインストール』の前までを記述していきます。ファイル名をm1-1.dita、タイトルを「ドキュメント制作の構造化手法」、各段落を<p>タグで囲みます。ブラウザからテキストをコピーして、タグ等を付加してください。、次のようになりましたか。注）ファイル名（m1-1.dita）の先頭文字に数字や特殊文字を使わないでください。「文字」か「_」を使用します。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE concept PUBLIC "-//OASIS//DTD DITA Concept//EN"
 "concept.dtd">
<concept id="m1-1" xml:lang="ja-jp">
 <title>ドキュメント制作の構造化手法</title>
 <shortdesc></shortdesc>
 <conbody>
<p>ドキュメントを効率よく制作・メンテし、多言語対応も簡単に出来る手法として、世界ではDITAというXMLを使った構造化言語が技術マニュアルを中心に使われています。
</p><p>特に、構造化に適した技術文書を軸に欧米ではかなり広まっています。しかし、日本では、技術文書と言えどもレイアウトデザインや、冗長な言葉使いが壁となって、なかなか広まっていない現実があります。
```

```
</p><p>私たちは、ワンソースマルチユースという一つのソースから、複数の出力
（PDF、XHTML、CHM等のヘルプ形式、EPUB）が出来るDITAソースをDITA-OT（DITA-OT
 Open Toolkit）というフリーな変換システムを使って実践してみます。
</p><p>まずは・・・どんな出力が出来るかを実際にDITA-OTを使って、試してみましょう。
</p>
 </conbody>
</concept>
```

さぁ、PDF変換してみましょう。m1-1.ditaファイルがC:\Magicalフォルダにあるとします。

```
java -jar lib/dost.jar /i:C:\Magical\m1-1.dita /transtype:pdf /outdir:C:
\Magical\out
```

いかがでしょうか？

C:\Magical\outフォルダに、m1-1.dita.pdfファイルが出来ているでしょうか。

m1-1.dita出力サンプル.pdf

DITAソースを書いてみよう2（6個のタグを追加する）

続いて残りの『DITA-OT Open Toolkitのインストール』から、『実際に変換してみる』の前までを同じように作ります。

- m1-2.ditaファイル

  ファイル名をm1-2.dita、タイトルを「DITA-OT Open Toolkitのインストール」にします。ここでは、見栄えを少し良くするためもあり、更に次の6個のタグを追加します。

  - <filepath></filepath>：URLなどのファイルパスに使用します。
  - <uicontrol></uicontrol>：[ ]で括られた単語に使用します。
  - <menucascade></menucascade>：[ ]-[ ]と連続した場合、一番外側に使用します。その際、間のハイフン（-）は削除します。
  - <codeph></codeph>：「 」で括られた単語に使用します。
  - ：中黒（・）を使用している文章全体の一番外側に使用します。
  - ：中黒（・）を使用している各文章に使用します。

  出来上がりは次のようになります。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE concept PUBLIC "-//OASIS//DTD DITA Concept//EN"
 "concept.dtd">
<concept id="m1-2" xml:lang="ja-jp">
 <title>DITA-OT Open Toolkitのインストール</title>
 <shortdesc></shortdesc>
 <conbody>
<p>まずは、核となるDITA-OT Open Toolkitをインストールします。
</p><p>ダウンロードサイトは次のURLです。
</p><p><filepath>http://sourceforge.net/projects/dita-ot/files/</
filepath>
</p><p>ここで迷うのは、どのバージョンを使用するか？です。
</p><p>すぐに、DITAを業務で使うなら、一番安定していると思われるDITA-OT1.5.4の安定
版をお勧めします。
</p><p>上記サイトから、<uicontrol>[DITA-OT Stable Release]</uicontrol>を
クリックし、次に<uicontrol>[DITA Open Toolkit 1.5.4]</uicontrol>をクリッ
ク、全部が入った<uicontrol>[DITA-OT1.5.4_full_easy_install_bin.zip]</
uicontrol>をダウンロードすると良いでしょう。
</p><p>今回は、実験として使いますので、11/26のDITA OT 1.7.M4最新版で開発中のもの
を使ってみます。
```

```
</p><p>上記サイトから、<menucascade><uicontrol>[DITA-OT Latest
 Test Build]</uicontrol><uicontrol>[DITA OT 1.7.M4]</
uicontrol><uicontrol>[DITA-OT1.7.M4_full_easy_install_bin.zip]</
uicontrol></menucascade>をダウンロードします。
</p><p>解凍して出来たフォルダ<codeph>「DITA-OT1.7.M4」</codeph>をルート直下で
使います。
</p><p>まず、ドキュメントを見ましょう。
</p><p><filepath>C:\DITA-OT1.7.M4\doc\userguide.pdf</filepath>を開きま
す。
</p><p>これと同じものが、インターネット上でも見れます。
</p><p><filepath>http://dita-ot.sourceforge.net/dev/</filepath>
</p><p>開発中ですので、バージョンM4ですが、ドキュメントはまだ<codeph>「M2」</
codeph>ですね。
</p><p>すぐに、サンプルDITAファイルを使ってPDF化したいところですが、待ってくださ
い。他にも環境設定が必要です。
</p><p>ちなみに、<codeph>「full_easy_install」</codeph>パッケージにより次のラ
イブラリーがすでに無償でインストールされています。
</p>
<ul><li>Apache Ant, version 1.8.4
</li><li>Apache Catalog Resolver, version 1.1
</li><li>Apache Commons Codec, version 1.4
</li><li>Apache FOP, version 1.0
</li><li>ICU for Java, version 49.1
</li><li>Apache Xerces, version 2.11.0
</li><li>Saxon, version 9.1
</li></ul>
<p>変換するのに必要な他のライブラリーはJAVAです。
</p><p>下記サイトからダウンロードして、インストールしてください。
</p>
<ul><li>JRE or JDK, version 6 or later
 <p><filepath>http://www.oracle.com/technetwork/java/javase/overview/
index.html</filepath>
 </p><p>JDK (Java Development Kit)のインストールを推奨します。
 </p>
</li></ul>
 <p>HTMLヘルプを生成するのであれば、合わせて次のものもインストールしておいてくださ
い。
 </p>
<ul><li>Microsoftヘルプワークショップ
 <p><filepath>http://msdn.microsoft.com/en-us/library/windows/desktop/
ms669985%28v=vs.85%29.aspx</filepath>
 </p>
</li></ul>
<p>これで、準備完了です。
</p>
 </conbody>
</concept>
```

さぁ、PDF変換してみましょう。m1-2.ditaファイルがC:\Magicalフォルダにあるとします。

```
java -jar lib/dost.jar /i:C:\Magical\m1-2.dita /transtype:pdf /outdir:C:
\Magical\out
```

C:\Magical\outフォルダに、m1-2.dita.pdfファイルが出来ているでしょうか。*m1-2.dita出力サンプル.pdf*

次回は、この二つのトピックを使って、1ファイルのPDF出力となるようにマップ（map）を作ってみます。

ご意見・ご質問等があれば、お気軽にお尋ねください。

2.4 DITAマップ（ditamap）を書いてみよう！

二つのDITAソースをDITAマップ（ditamap）を書いて、一つのPDFにしてみましょう。（2013年3月4日公開）

DITAソース（マップ）を書く前に

前回は、日本語のDITAソース（トピック）を二つ書いてPDFに変換しました。

今回は、これをつなげて、一つのPDFにしてみます。

一つのPDFにするために、まとめ役のファイルが必要です。これがDITAでは、マップファイルとよばれるものになり、拡張子は.ditamapとなります。このマップファイルに、トピック（topic）ファイルを複数関連付けるわけです。

マップには、2種類あります。汎用的なmapと、本の情報を記述できるように特化したbookmapです。

複数のトピックを束ねて一つのPDFにする場合、表紙やタイトルに第一部、第一章、付録とかの項目が付いた方が見やすいでしょうから、ここでは、bookmapを使用します。

今回は、前回作った二つのトピックを関連付けしたbookmapを作って、うまく1ファイルにPDF変換出来るかを試してみましょう。今回もDITA-OT1.7.M4を引き続き使います。注）2/24にメンテナンス版の3版（DITA-OT 1.7.3）がリリースされました。と同時に、DITA-OT1.8.M1もリリースされたようです。頃合いを見て、バージョンアップしますが、ここでは、引き続きすでにインストール済みのDITA-OT1.7.M4を使います。

DITAマップ（ditamap）を書いてみよう！

前回作ったトピックファイル名は、次の通りです。

- m1-1.dita
- m1-2.dita

圧縮したファイルを置いておきます。*m1-1_2入力ditaサンプル.zip*

bookmapを作るにあたってはお決まりのパターンがあります。UTF-8文字コードが使えるエディタで、次のように記述してみましょう。今回はid値を「magical-1」として、上述2トピックファイルを含めています。

- magical-1.ditamapファイル

  さて、準備が整ったところで、実際に『DITA-OT Open Toolkitのインストール』の前までを記述していきます。ファイル名をm1-1.dita、タイトルを「ドキュメント制作の構造化手法」、各段落を<p>タグで囲みます。ブラウザからテキストをコピーして、タグ等を付加してください。、次のようになりましたか。注）ファイル名（m1-1.dita）の先頭文字に数字や特殊文字を使わないでください。「文字」か「_」を使用します。

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE bookmap PUBLIC "-//OASIS//DTD DITA BookMap//EN"
 "bookmap.dtd">
<bookmap id="magical-1" xml:lang="ja-jp">
<booktitle>
 <booklibrary>DITA-OT編</booklibrary>
 <mainbooktitle>DITAの使い方</mainbooktitle>
 <booktitlealt>DITAソースを書いてみよう！</booktitlealt>
</booktitle>
<bookmeta>
</bookmeta>
<frontmatter>
<booklists>
<toc></toc>
</booklists>
</frontmatter>
<chapter>
 <topicref href="m1-1.dita"></topicref>
 <topicref href="m1-2.dita"></topicref>
</chapter>
```

```
<appendix>
</appendix>
<backmatter>
</backmatter>
</bookmap>
```

さぁ、PDF変換してみましょう。magical-1.ditamap,m1-1.dita,m1-2.ditaファイルがC:\Magicalフォルダにあるとします。C:\DITA-OT1.7.M4\startcmd.batをダブルクリックします。次のように打ってください。

```
java -jar lib/dost.jar /i:C:\Magical\magical-1.ditamap /transtype:pdf /
outdir:C:\Magical\out
```

いかがでしょうか？

最後の方で、「BUILD SUCCESSFUL」のメッセージが出ましたか。

C:\Magical\outフォルダに、magical-1.pdfファイルが出来ているでしょうか。

前回作ったトピック単体と比較して如何でしょうか？ 次の差異が見つけられると思います。

1. 二つのトピックが一つのPDFファイルに入りました。
2. 表紙がmagical-1.ditamapファイル内のbooktitleタグ内の情報を表示しています。
3. 2ページ目が「はじめに」のページとして、空白ページですが、挿入されています。
4. 各トピックの始まりが奇数ページからになっています（ページ右上に表示されるタイトルやノンブルも付きません）。そのため、空白ページ（4,6ページ）が調整用として挿入されています。
5. ブックマークに目次だけでなく、各トピックのタイトルが含まれています。

magical-1出力サンプル.pdf

DITAマップ（bookmap）を少しいじってみよう！

magical-1.pdfでは、確かに一つのPDFになりましたが、単にくっつけただけの感じで、少し物足りない気分になったでしょうか。そこで、もう少し本らしさを見せるために、挿入するトピックは変わりませんが、階層化にしたり、各章・付録のタイトル専用のページも作ってみました。id値を「magical-2」として、次のように記述してみましょう。

- magical-2.ditamapファイル

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE bookmap PUBLIC "-//OASIS//DTD DITA BookMap//EN"
 "bookmap.dtd">
<bookmap id="magical-2" xml:lang="ja-jp">
<booktitle>
 <booklibrary>DITA-OT編</booklibrary>
 <mainbooktitle>DITAの使い方</mainbooktitle>
 <booktitlealt>DITAソースを書いてみよう１！</booktitlealt>
 <booktitlealt>DITAソースを書いてみよう２！</booktitlealt>
 <booktitlealt>DITAソースを書いてみよう３！</booktitlealt>
 <booktitlealt>DITAソースを書いてみよう４！</booktitlealt>
</booktitle>
<frontmatter>
<booklists>
<toc></toc>
</booklists>
</frontmatter>
<chapter navtitle="DITAソースを書いてみよう１！（navtitle属性）">
 <topicref href="m1-1.dita"></topicref>
 <topicref href="m1-2.dita"></topicref>
</chapter>
<chapter>
 <topicmeta><navtitle>DITAソースを書いてみよう２！（navtitle要素）</
navtitle></topicmeta>
```

```
 <topicref href="m1-1.dita"></topicref>
 <topicref href="m1-2.dita"></topicref>
</chapter>
<chapter>
 <topicmeta><navtitle>DITAソースを書いてみよう３！（navtitle要素+階層化）</
navtitle></topicmeta>
 <topicref href="m1-1.dita">
  <topicref href="m1-2.dita"></topicref>
 </topicref>
</chapter>
<appendix>
 <topicmeta><navtitle>DITAソースを書いてみよう４！（navtitle要素+階層化）</
navtitle></topicmeta>
 <topicref href="m1-1.dita">
  <topicref href="m1-2.dita"></topicref>
 </topicref>
</appendix>
<backmatter>
</backmatter>
</bookmap>
```

さぁ、PDF変換してみましょう。magical-2.ditamapファイルがC:\Magicalフォルダにあるとします。

```
java -jar lib/dost.jar /i:C:\Magical\magical-2.ditamap /transtype:pdf /
outdir:C:\Magical\out
```

C:\Magical\outフォルダに、magical-2.pdfファイルが出来ているでしょうか。*magical-2出力サンプル.pdf*

いかがですか。

magical-2.pdfを開けてみて、少しは本らしい構成になったでしょうか。次のような特徴が見つけられると思います。

1. 表紙がmagical-1.ditamapファイル内のbooktitleタグ内の情報を表示しています。
2. 2ページ目が「はじめに」のページとして、空白ページですが、挿入されています。
3. 「DITAソースを書いてみようx！」のページが出来上がり、すべて奇数ページ始まりです（ページ右上に表示されるタイトルやノンブルも付きません）。そのため、空白ページ（4,6ページ）が調整用として挿入されています。また、自動的に「第一章」や「付録A」などの番号が付与されます。ミニ目次として、次の階層のタイトルがリンク付きで表示されています。
4. ブックマークに目次だけでなく、各トピックのタイトルが含まれています。
5. タイトルトピックが最上位のタイトルではなくなったため、magical-1.pdfと比べて、フォントの大きさとレイアウトが異なります。階層化した場合は、2階層目と3階層目のタイトルでも、フォントの大きさとレイアウトが異なります。

注）「DITAソースを書いてみよう1！（navtitle属性）」とDITAソースを書いてみよう2！（navtitle要素）ではditamapの書き方は異なりますが、同じ構造になっています。この場合、DITA1.2からはtopicmeta要素中のnavtitle要素を使うことが好ましいとされています。

次回は、今回のditamapによるpdfを使って、ミニ目次や空白の削除、ページサイズの変更など、dita-otのオプションや初期値などを少しいじってみましょう。

ご意見・ご質問等があれば、お気軽にお尋ねください。

2.5 PDFの見栄えを変更してみよう！

DITAマップとソースを使って、PDFの見栄えを変更してみましょう。(2013年4月8日公開)

PDFの見栄えを変更する前に

今回は、前回の日本語DITAマップとソースを使って、PDFの見栄えを変更してみます。

PDFの見栄えを変更するには、PDF用のスタイルシートを修正します。PDFの見栄えに関するファイルは、C:\DITA-OT1.7.M4\plugins\org.dita.pdf2フォルダ内にあります。この中のスタイルシートを修正していけば良いわけですが・・・今回は、オリジナルファイルをいじらなくて済む形で行います。

次のように、初期値や属性値を上書きするcustom.xslファイルと、本体部分を上書きするcustom.xslファイルを使います。同じファイル名なので、上書きする時は間違えないようにしてください。

- 初期値や属性値用スタイルシート　C:\DITA-OT1.7.M4\plugins\org.dita.pdf2\cfg\fo\attrs\custom.xsl
- 本体部分用スタイルシート　C:\DITA-OT1.7.M4\plugins\org.dita.pdf2\cfg\fo\xsl\custom.xsl

変更する項目は次の通りです。

1)ページサイズの変更

2)ヘッダ/フッタの位置変更(奇数偶数とも右側だったのを、偶数は左側に変更)

3)空白ページの削除

4-1)codephタグ内の文字列を緑色

4-2)codephタグ内の文字列の接頭接尾に『』を付加

5-1)文字列を赤色

5-2)章項節番号付加(文字列を赤色)

6-1)文字列を青色

6-2)filepathタグ内の'http://'から始まる文字列を青色

7)ミニ目次の削除

今回もDITA-OT1.7.M4を引き続き使います。

DITAマップを少し変更し、magical-3.ditamapファイルを作ってみよう！

前回作ったmagical-2.ditamapファイルでは少しページ数が不足なので、「DITAソースを書いてみよう3！(navtitle要素+階層化)」のtopicrefを丸ごとコピーしてページ数を増やします。次のような感じです。

```
<chapter>
 <topicmeta><navtitle>DITAソースを書いてみよう３！（navtitle要素+階層化）</
navtitle></topicmeta>
 <topicref href="m1-1.dita">
 <topicref href="m1-2.dita"></topicref>
 </topicref>
<!-- add. 2013-04 MAGICAL 上記ファイル構成をコピー -->
 <topicref href="m1-1.dita">
 <topicref href="m1-2.dita"></topicref>
 </topicref>
</chapter>
```

出来たものを置いておきます。*magical-3.ditamap.zip*

とりあず、PDF変換してみましょう。magical-3.ditamap,m1-1.dita,m1-2.ditaファイルがC:\Magicalフォルダにあるとします。C:\DITA-OT1.7.M4\startcmd.batをダブルクリックします。次のように打ってください。

```
java -jar lib/dost.jar /i:C:\Magical\magical-3.ditamap /transtype:pdf /
outdir:C:\Magical\out-0
```

C:\Magical\out-0フォルダに、magical-3.pdfファイルが出来ているでしょうか。*magical-3修正前版.pdf*

初期値や属性値を変更してみよう！

各項目ごとに説明します。C:\DITA-OT1.7.M4\plugins\org.dita.pdf2\cfg\fo\attrs\custom.xslファイル内に記述します。*attrs_custom.xsl.zip*

1. まず初期値を変更します。

   どれもbasic-settings.xsl内にある値をコピーして、変更します。

   1）ページサイズの変更

   USレターサイズ値になっていたのをA4の値に変更します。**<xsl:variable name="page-width">210mm</xsl:variable> <xsl:variable name="page-height">297mm</xsl:variable>**

   2）ヘッダ/フッタの位置変更

   元は、select="false()"だったのをselect="true()"に変更します。**<xsl:variable name="mirror-page-margins" select="true()"/>**

2. 次に属性値を変更します。

   3）空白ページの削除

   commons-attr.xsl内にある属性名"__force__page__count"の値をコピーして、select値を'even'から'auto'に変更します。

```
<xsl:attribute-set name="__force__page__count">
        <xsl:attribute name="force-page-count">
            <xsl:choose>
                <xsl:when test="name(/*) = 'bookmap'">
                        <!--xsl:value-of select="'even'"/-->
                        <xsl:value-of select="'auto'"/>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:value-of select="'auto'"/>
                </xsl:otherwise>
            </xsl:choose>
        </xsl:attribute>
    </xsl:attribute-set>
```

   4-1）codephタグ内の文字列を緑色

   pr-domain-attr.xsl内にあるcodephタグ用属性名"codeph"に、"color"値=greenを追加します。

```
<xsl:attribute-set name="codeph" use-attribute-sets="base-font">
      <xsl:attribute name="font-family">monospace</xsl:attribute>
        <xsl:attribute name="color">green</xsl:attribute>
    </xsl:attribute-set>
```

   5-1）文字列を赤色

   新規属性名"test_attr"を作り、"color"値=redを記述します。

```
<xsl:attribute-set name="test_attr">
        <xsl:attribute name="color">red</xsl:attribute>
    </xsl:attribute-set>
```

6-1）文字列を青色

sw-domain-attr.xsl内にあるfilepathタグ用属性名"filepath"をコピーして、"color"値=blueを追加し"filepathB"とします。

```
<xsl:attribute-set name="filepathB" use-attribute-sets="base-font">
        <xsl:attribute name="font-family">monospace</xsl:attribute>
        <xsl:attribute name="color">blue</xsl:attribute>
    </xsl:attribute-set>
```

さぁ、PDF変換してみましょう。次のように打ってください。

```
java -jar lib/dost.jar /i:C:\Magical\magical-3.ditamap /transtype:pdf /
outdir:C:\Magical\out-1
```

いかがでしょうか？

最後の方で、「BUILD SUCCESSFUL」のメッセージが出ましたか。C:\Magical\out-1フォルダに、magical-3.pdfファイルが出来ているでしょうか。先ほど変換したout-0\magical-3.pdfと比較して如何でしょうか？ 次の差異が見つけられると思います。

1）A4サイズ

2）偶数ページのヘッダが左側

3）空白ページが無い

4-1）codephタグの文字列が緑色

注）5-1）と6-1）は、次の本文中のスタイル変更時に使われるので、ここでは表示されません。*magical-3スタイル修正版1.pdf*

本文中のスタイルをいじってみよう！

各項目ごとに説明します。C:\DITA-OT1.7.M4\plugins\org.dita.pdf2\cfg\fo\xsl\custom.xslファイル内に記述します。*xsl_custom.xsl.zip*

1. テンプレートモジュールを変更します。

   4-2）codephタグ内の文字列の接頭接尾に『』を付加

   \xsl\fo\pr-domain.xsl内にあるcodephタグを含むテンプレートに『』を付加します。

```
<xsl:template match="*[contains(@class,' pr-d/codeph ')]">
        <fo:inline xsl:use-attribute-sets="codeph">
            <xsl:call-template name="commonattributes"/>
            <xsl:text>『</xsl:text>
            <xsl:apply-templates/>
            <xsl:text>』</xsl:text>
        </fo:inline>
    </xsl:template>
```

   5-2）章項節番号付加（文字列を赤色）

   \xsl\fo\commons.xsl内にある章の下の階層にあるトピックタイトル（'topicSimpleS'と設定）の先頭に番号を付加し、5-1）で作った赤色属性"test_attr"を付けます。

```
<xsl:template match="*" mode="getTitle">
        <xsl:variable name="topicType">
            <xsl:call-template name="determineTopicType"/>
        </xsl:variable>
        <xsl:choose>
```

```
            <xsl:when test="@spectitle">
                <xsl:value-of select="@spectitle"/>
            </xsl:when>
            <xsl:when test="$topicType='topicSimpleS'">
                <fo:inline xsl:use-attribute-sets="test_attr">
                    <xsl:number level="multiple"
 count="*[contains(@class, ' topic/topic ')][child::*[contains(@class,
 ' topic/title ')]]"
                     format="1.1.1 "/>
                    <xsl:value-of select="' '"/>
                </fo:inline>
                <xsl:apply-templates/>
               </xsl:when>
            <xsl:otherwise>
                <xsl:apply-templates/>
            </xsl:otherwise>
        </xsl:choose>
    </xsl:template>
    <xsl:template match="*[ancestor::*[contains(@class, ' bookmap/
chapter ')]]" mode="determineTopicType">
        <xsl:text>topicSimpleS</xsl:text>
    </xsl:template>
```

6-2）filepathタグ内の'http://'から始まる文字列を青色

\xsl\fo\sw-domain.xsl内にあるfilepathタグを含むテンプレートに filepathタグ内で'http://'から始まる文字列には、7-1）で作ったを青色属性"filepathB"を付けます。

```
<xsl:template match="*[contains(@class,' sw-d/filepath ')]">
   <xsl:choose>
    <xsl:when test="starts-with(., 'http://')">
      <fo:inline xsl:use-attribute-sets="filepathB">
       <xsl:call-template name="commonattributes"/>
       <xsl:apply-templates/>
      </fo:inline>
    </xsl:when>
    <xsl:otherwise>
     <fo:inline xsl:use-attribute-sets="filepath">
      <xsl:call-template name="commonattributes"/>
      <xsl:apply-templates/>
     </fo:inline>
   </xsl:otherwise>
   </xsl:choose>
  </xsl:template>
```

さぁ、PDF変換してみましょう。m1-2.ditaファイルがC:\Magicalフォルダにあるとします。

```
java -jar lib/dost.jar /i:C:\Magical\magical-3.ditamap /transtype:pdf /
outdir:C:\Magical\out-2
```

いかがでしょうか？

C:\Magical\out-2フォルダに、magical-3.pdfファイルが出来ているでしょうか。先ほど変換したout-1\magical-3.pdfと比較して如何でしょうか？ 更に次の差異が見つけられると思います。4-2）codephタグの文字列の接頭接尾に『』が付加 5-1）と5-2）目次と本文中の各章内タイトルに赤色の章項節番号が付加、6-1）と6-2）filepathタグの'http://'から始まる文字列が青色 *magical-3スタイル修正版2.pdf*

スタイルシートをいじることによりを、見栄えを自由に変更することがご理解頂けたでしょうか。

2. antコマンドのオプション設定で変更します。

7）ミニ目次の削除 これは、antコマンドのオプション設定で変更してみましょう。

次のようなコマンドでPDF変換してみましょう。

```
ant -l C:\Magical\out\magical3ant.log -Dargs.input=C:\Magical
\magical-3.ditamap -Dtranstype=pdf -Doutput.dir=C:\Magical\out-3 -
Dargs.chapter.layout=BASIC
```

最後の「-Dargs.chapter.layout=BASIC」が、ミニ目次削除のオプションです。C:\Magical\out-3フォルダに、magical-3.pdfファイルが出来ているでしょうか。先ほど変換したout-2\magical-3.pdfと比較して、ミニ目次が削除されていると思います。

「-l C:\Magical\out\magical3ant.log」はログファイルとなりますので、エラーとかが出た場合に見てください。
magical-3ミニ目次の削除版.pdf

次回からは、更に深くDITA-OTを使って、様々なことを行ってみましょう。

注）DITA-OT1.8.M2が3/25にリリースされました。V1.8では、ApacheFOPのバージョンが1.0から1.1にアップされました。次回からは、このDITA-OT1.8を使って、構築します。

ご意見・ご質問等があれば、お気軽にお尋ねください。

第 3 章 DITAでWEBサイトを執筆 HTMLオーサリングと決別

特殊化しない標準DITA文書からWEBサイトを自動生成 ディタリストなプログラマーの挑戦

なぜDITAで書くの？ HTMLのCMSではイケナイの？

DITAは文書生産そのものを目的としたXML規格で、従来の文書生産管理の課題を解決するためのものです。DTPとは異なる手法ですが、WEBやPDFの制作経験がある方なら執筆の為の基礎的な技術はすぐに習得出来ると思います。

ひとつの文書データからHTMLとPDFなど異なる形式に変換し易いメリットがよく取り上げられますが、本質は執筆作業の効率化と品質マネジメントのし易さではないかと思います。

レイアウトを持たないの？ 見栄えは悪いの？

構造化文書は出力時に機械的にレイアウトを与えることが出来ますから問題ありません。DTPのような自由度はありませんが、機械的に組版されるということは作業ミスでレイアウトが不揃いになることもありません。

ただ、出力の見栄えを重視するとカスタマイズは必須で、通常は専門家に依頼することになります。

もっと手軽に使えないの？ 開発の予算や工期は？

DITAの執筆と変換そのものはデスクトップ環境でも動かせますから、人数の少ない現場ならこのような簡易なシステムから始めても良いのではないでしょうか。ちなみに、このサイトはDITAで書いていますが、使っているのはオープンソースのみ、スタッフは2名、もちろん通常業務の合間に作業しています。

関連リンク

第 4 章 x-magic紹介 (23ページ)

x-magicの特徴を紹介します

第 4 章 x-magic紹介

x-magicの特徴を紹介します

WEB固有のUIをフルサポート 言わなければ誰もDITAとは気付かない

DITAには目次やリンクをサポートする機能がありますが、WEBサイトとして使うにはUIのカスタマイズが必要と考えます。

左右分割のフレーム表示で目次とトピックを並べる手法が広く使われていますが、それは満足できません。

x-magicは、Wordpress などWEB専用に設計されたシステムと同等のUIを提供します。

グローバルメニュー ダイレクトな章移動をサポート

どのトピックを開いていても、ヘッダー部やフッター部のグローバルメニューをワンクリックするだけで任意の章に移動できます。

パンくず機能 現在位置の表示

パンくずを表示するようにしてありますので、巡回中に現在位置を見失うことはありません。階層の深いトピックから上の階層に戻るときにも便利です。

サイドバー 姉妹リンク対応

サイドバーには、現在表示中のトピックに関連するトピックが一覧表示されます。この親リンク、子リンク、姉妹リンク、関連リンクはトピックの下部にも表示されます。(トピック下部では、関連リンクは更に詳細にグループ分けされます。)

テンプレート編集も殆ど不要 ぜんぶDITAで書けます

章やトピックへのリンクメニューはもちろん、サイト名、著作社名、著作年なども全てDITAソースから自動取得しますので、基本的にはHTML出力用のテンプレートを編集する必要はありません。

ヒント： ヘッダー右上のリンクメニューは、非DITAページなど自動化出来ないものに使うことを想定したものなので、こはHTML変換用のテンプレートで編集するようになっています。

ヒント： WEBサイトのトップページは、PDFには存在しないもので、デザイン性を重視したいケースも多いため、100%DITA化には拘らず、一部HTML変換用テンプレートの直接編集で柔軟に対応出来るようにしています。(メニューやトピック一覧など自動構築させたい部分は自動構築するようになっています。)

関連リンク

第 3 章 *DITAでWEBサイトを執筆 HTMLオーサリングと決別*(22ページ)

特殊化しない標準DITA文書からWEBサイトを自動生成 ディタリストなプログラマーの挑戦

4.1 サブディレクトリのlink@href

DITA-OT1.8既知のバグ

DITA-OT1.8(Windows版のみ)のバグで、transtype=xhtml でサブディレクトリのトピックを変換した際、commonltr.css のリンクhrefがエンコード化けして動作不具合を引き起こすことが分かっています。

x-magicは、上記不具合をプラグインで修正するようになっています。

4.2 shortdescがない場合

x-magicスタイル仕様の特徴

トピックのshortdescは省略可能です。省略時はHTML上でshortdesc表示蘭そのものが消えます。(枠だけ表示されるようなことはないようになっています)

4.3 目次自動生成機能 section/title

x-magicスタイル仕様の特徴

section/titleが含まれている場合は、section/title で目次が自動生成されます。

また、各sectionブロックの末尾には、ページトップへ表示を戻すためのリンクが加えられます。

第 5 章 リンク

リンク集

DITA関連 パートナー企業 / 交流企業 / 加盟団体等（順不同）

- *NEC*マネジメントパートナー株式会社
- *NEC*マネジメントパートナー デザイン・プロモーション事業サイト
- アンテナハウス株式会社
- *Bluestream XML Content Solutions*
- *DITA*コンソーシアムジャパン
- インフォパース株式会社

x-magic Plugin for DITA Open Toolkit 実演サイト

DITAのことならmagicalにおまかせください

まじかるテクノロジー