DITA

Writing, compiling, and maintaining documentation is a necessary evil. While moving to DITA might not improve the quality of your documentation, it can streamline the process of creating and managing those documents.

What Is DITA?

The Darwin Information Typing Architecture (DITA) is an XML-based method for writing and delivering information in a variety of forms

DITA consists of a Document Type Definition (DTD), which specifies how the elements and that make up a DITA document can be defined. There's also set of XSLT stylesheets that control the look and feel of the documents that are output. Writers use the stylesheets in conjunction with an XML processor to convert a DITA document to more usable formats, such as HTML or PDF.

Like HTML or other variants of XML, DITA consists of a set of tags. There are around 200 tags in the DITA specification. The tags are easy to understand, and many of them are similar to HTML tags. For example, <p> denotes a paragraph, <b> is for making text bold, and <table> creates a table. Other tags are as easily understood.

Topics

FrameMaker (a standard documentation tool) and other XML-based solutions such as DocBook use the traditional, fairly rigid book-chapter-section metaphor—the book is a container that contains chapters, which in turn contains individual sections describing an idea or a task. DITA, on the other hand, is based on topics. Topics are small pieces of information on a single subject. But a topic is just a container that can hold the following elements (called information types in DITA lingo):

- Concepts are background or conceptual information.
- Tasks are procedures that explain how to do something.
- References are more specialized information, or information that's intended for a highly technical audience; for example, code documentation for software developers.

Following is an example of a simple DITA topic:

```
<title>A DITA topic</title>
<titlealts>
 <searchtitle>This is a DITA topic</searchtitle>
</titlealts>
<shortdesc>This is a brief description of a DITA
topic.</shortdesc>
<prolog>
```

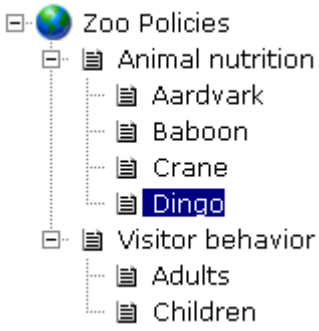
```
 <author>Scott Nesbitt</author>
</prolog>
<body>
 <p>A topic must have both a title and body.</p>
</body>
<topic id="main">
 <title>A really short topic</title>
<body>
<p>This is a really short topic. It only contains this text, along
with a title.</p>
</body>
</topic>
</topic>
```

Topics can consist of entire sections of a chapter, a single paragraph, or just a list. They're chunks of information that can be taken and fit together as needed, as if they were Lego blocks. This arrangement gives writers almost unlimited flexibility with what they write and how they combine topics.

Reuse with map files

DITA provides two major reuse mechanisms: maps and references. DITA maps provide a list of links, in a particular sequence and hierarchy, that describe the content , another definition :Topics are combined using topic maps. A topic map is like a table of contents in a book. It spells out which topics are included in a particular document, and how those topics are organized

Example:




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

<!DOCTYPE map PUBLIC "-//OASIS//DTD DITA Map//EN" "map.dtd">

<map title="Zoo Policies">

 <topicref href="Animal_nutrition.xml">

  <topicref href="Aardvark.xml"/>

  <topicref href="Baboon.xml"/>

  <topicref href="Crane.xml"/>
```

```
  <topicref href="Dingo.xml"/>

 </topicref>

 <topicref href="Visitor_behavior.xml">

  <topicref href="Adults.xml"/>

  <topicref href="Children.xml"/>

 </topicref>

</map>
```

Map files are similar to FrameMaker book files, but offer more flexibility—a map file can contain references to other map files, and topics can be nested more than one level.

Typically, you would create a map file to support each major deliverable. Thus, components of the *Zoo Policies* shown in the preceding example could be reused in an *Animal Care Guide.* The information about animal nutrition might be provided in both documents, but the discussion of visitor policies would appear only the policy manual. Multiple map files can reference the same topic as necessary.

Reuse with content references

For reuse inside a topic, DITA provides a content referencing mechanism. A bit of content that will be reused carries a unique ID. In this example, the <note> element is set up with an ID for reuse:

```
<topic>

 <title>Aardvark</title>

 <body>

  <p>Aardvarks eat mostly termites.</p>

  <note type="danger" id="nofeeding">Do not feed snacks, scraps, or
  people food to the animals.</note>

 </body>

</topic>
```

You then create a reference to the element you want to reuse by specifying the file name and the ID in a conref attribute:

```
<topic>

 <title>Baboon</title>

 <body>
```

```
  <p>Baboons eat mostly fruit.</p>

  <note type="danger" conref="aardvark.xml nofeeding"/>

 </body>

</topic>
```

One difficulty with conrefs is that they are quite tedious to create and manage manually. If you plan to use conrefs, consider how well your potential authoring tools support content creation and management.

Another aspect of DITA is the domain. A domain defines the elements and tags that enable writers to mark up documentation on a specific subject. For example, there are domains for the elements that are used to mark up the descriptions of user interfaces and software, or examples of programming code.

Conditional content

The DITA architecture provides support for attribute-based versioning. That is, if you have some content that is intended only for some deliverables, you label the content with the relevant attributes:

```
<topic audience = "internal">

 <title>Secret Settings in our Software</title>

</topic>

<topic>

 <title>Setting up Your Project</title>

</topic>

<topic audience="external">

 <title>Controlling Your Files</title>

</topic>
```

When you publish content, you use a settings file called ditaval to specify which information to exclude from the final output. In the preceding example, a deliverable for internal users does not require a ditaval file because internal users get all of the information (internal, external, and all audiences). For external users, you need to suppress the internal-only information, so you set up a ditaval file as follows:

```
<val>

 <prop att="audience" val="internal" action="exclude"/>
```

```
</val>
```

You can also set up more complex conditional processing with intersecting attributes. For instance, you could produce the Linux-specific version of a document for external users by excluding the internal and Windows-specific information, as shown here:

```
<val>

 <prop att="audience" val="internal" action="exclude"/>

 <prop att="platform" val="win" action="exclude"/>

</val>
```

You will need a ditaval file for each combination of conditions, or you can create some additional processing to create a custom ditaval file when you are ready to publish your content.

Specialization

One of DITA's unique features is specialization. At its simplest, specialization is the process that enables writers to extend existing information types or to define new ones. Specialization allows technical writers to extend DITA to meet their needs and the needs of their organizations

a specialization can't just be created out of thin air. It has to be based on an existing DITA element. For example, if a writer creates a three-column table with a title spanning the top row, that table needs to be based on the standard DITA table. Each time a specialization is created, the writer must add information to the DITA file to identify the element on which the specialization is based.

DITA's Advantages

DITA offers a number of advantages over tools and languages such as FrameMaker and DocBook:

- DITA uses about half as many tags as DocBook does. The tags are easy to understand, and you'll quickly find that you'll use only a small percentage of them regularly.
- Because DITA is based on XML, the source files are plain text. They can be opened in any editor and used on any operating system without a loss of formatting.
- DITA is a good choice for single sourcing documents. Single sourcing involves identifying all the types of documents you create, and what content is shared among those documents. From there, you combine the information as needed and output it into various formats as needed.

- DITA enables technical writers to manage and combine their content easily. Writers can quickly extract and combine the information they need, when they need it.
- DITA enables writers to reuse standard content quickly and easily.

The Tools

Although DITA is only just exploding on to the technical writing scene, a number of tools are already available for authoring with DITA, and for outputting documents into various formats. And more are on the way.

Because DITA is based on XML, any text or XML editor can be used to author DITA documents. One popular application is the free XEmacs text editor. When combined with psgmlx (an add-on to XEmacs that makes working with XML easier), XEmacs becomes a powerful DITA editor.

Challenges for Technical Writers

The biggest challenge for any technical writer who is moving to DITA is to change the way in which he or she looks at documentation. Because DITA topics are written in isolation, it can be difficult to see how they fit together. Writers have to move from viewing manuals and help files as monolithic documents with a beginning, a middle, and an end to visualizing documentation as structured chunks of information.

Of course, the advantages of DITA need to be weighed against any productivity lost while learning and adapting to DITA and the DITA way of doing things. However, the benefits of streamlining and efficiency that using DITA brings your organization will outweigh those costs in the long run.

Conclusion

DITA offers an entirely new way of working with documentation. It gives technical writers more flexibility not only in how they write documentation, but how they assemble it for delivery to users.

Making the switch to DITA is a big step. It will require writers to switch tools, and to change the ways in which they plan and write documentation. Moving to DITA might not improve the quality of a company’s documentation, but it can streamline the process of creating and managing those documents.