DITA_Syntax_Markup

Contents

Chapter 1. Introduction to DITA

The Darwin Information Typing Architecture (DITA) is an XML-based architecture for authoring, producing, and delivering topic-oriented, information-typed content that can be reused and single-sourced in a variety of ways. While DITA historically has been driven by the requirements of large-scale technical documentation authoring, management, and delivery, it is a standard that is applicable to any kind of publication or information that might be presented to readers, including interactive training and educational materials, standards, reports, business documents, trade books, travel and nature guides, and more.

DITA is designed for creating new document types and describing new information domains based on existing types and domains. The process for creating new types and domains is called specialization. Specialization enables the creation of specific, targeted XML grammars that can still use tools and design rules that were developed for more general types and domains; this is similar to how classes in an object-oriented system can inherit the methods of ancestor classes.

Because DITA topics are conforming XML documents, they can be readily viewed, edited, and validated using standard XML tools, although realizing the full potential of DITA requires using DITA-aware tools.

Chapter 2. Organisation of DITA elements

Elements in DITA are grouped together into a number of categories for organizational and comprehension purposes.

DITA elements can be broadly categorized as follows:

- Topic elements
- Map elements
- Body elements
- Prolog elements
- Domain elements

In fact, this is how the DITA Language Reference groups DITA elements. This categorization of the various elements is partly to aid understanding, and partly to make it technically easier for the schema files (DTD and XSD) to be managed.

Elements used within topics can also be differently categorized as block or phrase elements.

Topic Elements

The topic elements are the basic structural building blocks of all information types.

The topic elements are the main structural elements of topics. Some topic elements are generic (that's inherited from the topic proto information type), while others are specific to the concept, task or reference information types.

Examples of topic elements include:

- topic (and concept, task, reference)
- titlealts
- shortdesc
- body (and refbody, conbody, taskbody)
- section
- example
- related-links

Map Elements

The map elements are the elements used in ditamaps and bookmaps.

The map elements are a small set of elements, some of which have been specialised into other elements for use in bookmaps.

The map elements include:

- map
- topicref
- topicmeta
- topicgroup
- topichead
- reltable

Body Elements

The simple block structures within the body of topics are categorized as the body elements.

Body elements are the most common content authoring block elements, and include:

- paragraph
- list
- phrase
- figure

Prolog Elements

A topic's metadata is stored in a range of prolog elements.

The DITAprolog elements contain the main metadata for a topic or collection.

The types of information recorded in the prolog include:

- author
- copyright information
- critical tracking dates
- permissions for use/management of the content
- extensive metadata about the content of the document

Domain Elements

The domain elements are comprised of a number of separate sets of elements that relate to specific documentation fields.

Remembering that DITA started life within IBM as a tool for creating software and hardware documentation, it shouldn't be a surprise to discover that DITA's base elements reflect that background.

Elements that relate to a particular field (such as software) are called domain elements. The domain elements within DITA are grouped into:

typographical elements

generic word-processor like elements used to highlight text

programming elements

terms and structures related to programming environments

software elements

terms and structures related to the operation of a software program

table elements

elements that relate to table structures

user interface elements

terms and structures related to a software user interface

utilities elements

elements that don't fit anywhere else!

If you are writing a programmer's reference, you will mainly use elements in theprogramming domain.

If you are writing a mobile phone user guide, you should avoid using programming domain elements, and mainly use user interface domain elements.

The typographical domain elements are designed to be used only when **no semantically- appropriate elements are available** and a formatting effect is required. These elements should therefore only be used as a last resort.

Programming domain elements

The elements in the programming domain each have a specific semantic purpose.

Note: Use <dl> to describe each of the following element and provide an example.

Element Name	Semantic Purpose
apiname	API name
codeblock	code block

Element Name	Semantic Purpose
codeph	code phrase
option	one of a set of options
parmname	parameter or argument
parml	parameter list
plentry	parameter list entry (within parml)
pt	parameter term (within plentry)
pd	parameter definition (within plentry)
synph	syntax phrase
syntaxdiagram	syntax diagram
groupseq	group of syntactic units (used only for syntax diagram)
groupchoice	choice of a group of syntactic units (used only for syntax diagrams)
groupcomp	group of composite syntactic units (used only for syntax diagrams)
fragment	fragment of syntax (used only for syntax diagrams
fragref	cross-reference to a fragment of syntax
synblk	block of small pieces of syntax
synnote	footnote within syntax (syntax note)
synnoteref	cross-reference to a syntax note
kwd	syntax keyword (used only for syntax diagrams)
var	variable that a user must supply (used only for syntax diagrams)
oper	operator character (such as +, -, and =) within syntax
delim	delimiter character (such as /, |, and ;) within syntax
sep	separator character within syntax
resep	separator character for repeated syntax elements

User Interface domain elements

The elements in the user interface domain each have a specific semantic purpose.

Note: Use <dl> to describe each of the following element and provide an example.

Element Name	Semantic Purpose
uicontrol	user interface control
wintitle	window title
menucascade	menu cascade
shortcut	shortcut
screen	character (text only) screen

Software domain elements

The elements in the software domain each have a specific semantic purpose.

Note: Use <dl> to describe each of the following element and provide an example.

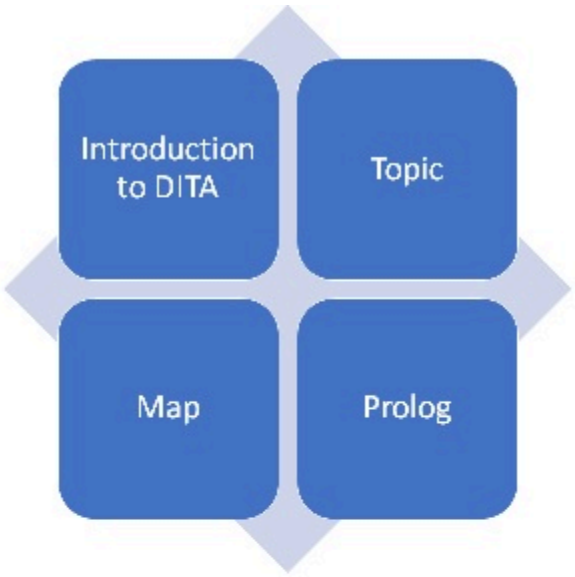
Element Name	Semantic Purpose
msgph	message phrase
msgblock	message block
msgnum	message number
cmdname	command name
varname	variable (to be provided by user) name
filepath	file name or path, or URI
userinput	user input
systemoutput	system output

Utilities domain elements

The elements in the utilities domain each have a specific purpose in defining image map properties.

Note: Use <dl> to describe each of the following element and provide an example.

Element Name	Semantic Purpose
imagemap	client-side image map
area	hotspot area within an image map
coords	co-ordinates of a hotspot area within an image map
shape	shape of a hotspot area within an image map



1.
2. Introduction to DITA *(on page)*

Utilities domain elements

The elements in the utilities domain each have a specific purpose in defining image map properties.

Note: Use <dl> to describe each of the following element and provide an example.

Element Name	Semantic Purpose
imagemap	client-side image map
area	hotspot area within an image map
coords	co-ordinates of a hotspot area within an image map
shape	shape of a hotspot area within an image map