Basic DITA

Contents

What is DITA?

- DITA background
- DITA as a markup language
- DITA topics

DITA stands for the DITA Information Typing Architecture. It is an open standard originally created by IBM. In 2005, IBM donated DITA to *OASIS*.

DITA is an XML-based, tool-independent way to create, organize, and manage content. DITA is built on:

- Topic-based authoring
- Separating content from formatting
- Minimalism
- Structured authoring concepts

The fact that DITA is an open, XML-based standard means that it is relatively easy to move from one tool to another as business needs change.

Markup languages

The most commonly used markup language is HTML. XML is similar to HTML; it also uses angle brackets to indicate elements (like <p>). Many of the tags in DITA XML are also found in HTML.

In XML, you must be careful to open and close all tags properly. HTML is forgiving when you make mistakes. XML requires you to provide a file that is tagged correctly; if you make any error in the encoding, parsing will stop. This is known as draconian error handling, and it is a controversial part of the XML specification.

DITA is a markup language and is a flavor of XML. DITA allows you to mark up content semantically; the tag markup tells the author about the meaning of the markup.

Video: *Overview of HTML markup versus DITA markup*

DITA is structured authoring

DITA enforces structure and standard for an authoring environment. Typically, an information architecture will set up the overall environment.

Content has implicit structure, even in a word processor. DITA models and enforces that structure explicitly. For example, a task topic must have at least one step. The result is that all task topics are structured consistently.

What is a DITA topic?

A DITA topic is a container in which you create content. Topics can be:

- Constrained. More limited than what the DITA standard allows by default.
- Specialized. Include additional elements beyond what’s provided in the standard.

Content and topics have elements. Elements are the tags for the content. The tags available to a specific author depend on how that author’s environment is set up by the information architect. For example, if you are not writing in a software environment and don’t need code samples, your information architect could constrain your DITA environemnt to eliminate the code-related element.

Elements have attributes. An attribute provides additional information about the element. For example, a topic might have an attribute called author, in which you specify who created a particular topic.

Assessments

To be valid, a topic must have the following:

- a title element
- a title element and an id attribute
- an href attribute

A topic is a piece of content that addresses one question or idea

- True
- False

  A topic is a unit of content. Concept topics describe one idea; reference topics provide information about one item, and task topics provide step-by-step instructions for one procedure.

HTML and DITA are very different. Knowing one doesn’t help you read the other.

- True

  HTML and DITA use many of the same tags, such as title, p, ul, and li.
- False

The only way to do structured authoring is to use the DITA standard.

- True

  There are other standards, such as DocBook, S1000D, NLM, and more. You can also create your own custom structure from scratch.
- False

DITA is an open standard, which means (choose one):

- Anyone can use it without paying for it.
- When updates are released, you must upgrade and pay or lose access to your information.
- Updates are controlled by the ISO organization.

An element is (choose all that apply):

- a tag for content.
- inserted when you generate output to control formatting.
- identified with angle brackets (such as <p>).
- a DITA term for indexing content.

Specialization means

- Adding new elements based on existing elements
- Removing unused elements

Constraining means

- Adding new elements based on existing elements
- Removing unused elements

In DITA, content is created in:

- specialized attribute containers
- topics
- in bulleted lists that are verified when you create output
- with XML markup that the reader must learn to understand

Because DITA is an open standard (choose all that apply):

- No one can charge for tools.
- None of these answers are correct.
- You can only use a text editor.
- Anyone can use the standard.

DITA topics

- What kind of topics exist in DITA
- What information goes in different types of topics
- What are constraints and specialization?

The DITA standards includes basic topic type (<topic>), along with the following:

- Concept
- Task
- Reference
- Glossary entry

All topics must have at least a title element and an id attribute for the root topic. That is, the following is a valid topic:

```
<topic id="sample">
<title>Topic title goes here</title>
</topic>
```

However, a *useful* topic will have additional content.

Concept topics

A concept topic answers the question “Why?” It provides background information about a subject that the reader needs to know.

Concepts often include overview graphics and other big picture information needed to understand the ideas behind a particular subject.

The concept topic has a required title. Other common elements used in a concept topic include:

- shortdesc (short description)
- conbody (the body of the concept topic)
- section
- ul
- ol
- fig
- image

Task topics

- Understand the purpose of the DITA task type
- Identify key components of a task
- Create a DITA task

A task topic answers the question “How do I?” It includes step-by-step instructions to complete a procedure. DITA also allows step results, graphics, notes, and substeps, but not subsubsteps.

Technical content is often heavy on tasks.

DITA 1.2 provides two task types:

- Strict task
- General task

Strict task was the only task provided in DITA 1.1, so the “strict task” in DITA 1.2 is equivalent to the “task” in DITA 1.1. The strict task requires you to include steps. The “general task,” also known as the “loose task” is more forgiving. General tasks are useful for process overviews. Strict tasks are appropriate for items that require step-by-step instructions.

The task topic has a required title. Other common elements used in a strict topic include:

- shortdesc (short description)
- taskbody (the body of the task topic)
- steps (the sequence of actions)
- step (each individual action-result pair)
- cmd (
- stepresult (what happens after you perform an action)
- stepxmp (an example of how to do the step)
- fig
- image

The general task uses stepinformal.

Video: *Code overview of DITA task topic*

Reference topics

A reference topic answers the question “What is it?” Reference topics typically contain descriptive facts, such as a table identifying items on a software screen or an alphabetical list of commands available in a programming language.

Reference topics do not include steps or concepts. Reference topics are similar to dictionary entries and provide facts only.

The reference topic has a required title. Other common elements used in a reference topic include:

- shortdesc (short description)
- refbody (the body of the reference topic)
- properties
- refsyn

Video: *Code overview of the DITA reference topic*

Glossary entry topics

A glossary topic answers the question “What does this word or phrase mean?” Glossary topics typically contain one term, along with one or more definitions.

Common elements used in the glossary topic are:

- glossentry (identifies this topic as a glossary entry)
- glossterm (the word or phrase)
- glossdef (the definition of the glossary term)

Video: *Code overview of the DITA glossary topic*

Constraints

In DITA, a constraint lets you eliminate elements that you do not need or want to use. For example, if you are not documenting software code, you could eliminate <codeblock> and <codephrase>. Constraining reduces the number of elements presented to the authors, which makes their lives a little easier.

Specialization

Specialization lets you create elements and attributes that fit your organization better than what is provided in default DITA.

Organization often specialize to support their unique requirements, such as:

- Creating a customer attribute to identify customer-specific information
- Creating codeblock elements that identify the language used in each codeblock. For example, to distinguish between Java code examples and PHP code examples, you might create:
    - codeblock-java
    - codeblock-php

  You could also use attributes in this example; something like:

    - <codeblock type="java">
    - <codeblock type="php">

Caution:

Be careful with specialization. When you specialize, you make tags more specific, but specialization adds to the cost of implementation. You must balance the value gained from specialization against the cost of implementing and maintaining specializations.

Assessments

For each element, identify the topic type it belongs with:

refbody	reference topic
glossdef	glossary topic
steps	task topic
conbody	concept topic
property	reference topic
stepresult	task topic
glossentry	glossary topic

For each topic type, identify the question it answers:

What is it?	reference topic
What does this term or phrase mean?	glossary topic
How do I?	task topic
Why?	concept topic

When creating a concept topic, you will often:

- Avoid introducing new terms or phrases

  Concepts are very often where you introduce new words or phrases.
- Describe in detail how to accomplish a task

  Step-by-step instructions belong in task topics.
- Provide a graphic

Which of the following are topic types in the DITA standard (choose all that apply)?

- Reference
- Task
- Procedure

  DITA does have support for procedures, but the topic type is called a task.
- Cross-reference

  You can create cross-references in DITA, but they are created inside topics.
- Concept
- Glossary entry

What kind of topic uses a glossterm element?

- Glossary
- Task
- Concept

Task topics include the following components (choose all that apply)?

- Optional substeps
- Overview content

  You can provide contextual information with the context element, but detailed overview information should go in a concept topic.
- Optional step results
- Step-by-step information

Match the glossary elements to their meaning:

<glossterm>	Word or phrase
<glossdef>	Glossary definition
<glossentry>	Container for a glossary entry

Metadata

- What is metadata?
- How do you use metadata?
- What metadata is required

summary here

What is metadata?

Metadata provides information about information. For example, word processing applications often have document properties, which tell you who created the file and on what date it was last modified. The author and modification date are not part of the text that is displayed; they are metadata about the document itself.

Metadata provides different ways to classify information. For example, you can:

- Label information with a document type, like “white paper” or “proposal”
- Associate information with a specific product number
- Indicate a user level, like “beginner,” “intermediate,” or “expert”
- Provide status information for a topic, like “draft,” “review,” or “final”
- Provide an expiration date when information is no longer valid

Once you have metadata assigned to your content, you (or your reader) can use it to:

- Find information efficiently
- Support governance effort (control when information is made public or removed from a web site)
- Personalize information (your reader could request only beginner information)
- Filter information for different delivery variants
- Reuse information efficiently
- Manage project status

In DITA and other XML formats, metadata is encoded inside your topics, either at the root (topic) level or on specific elements. You can manipulate DITA content using those metadata values.

DITA requires one piece of metadata. Every topic must have an id attribute. These IDs are often assigned automatically by authoring tools or content management systems. Here is an example of a topic with an id attribute:

```
<topic id="xyz">
<title>Your title here</title>
<body>
....
</body>
</topic>
```

Your organization or your DITA setup may require additional metadata. For example, it’s common to include the language of the topic with the xml:lang attribute.

```
<topic id="xyz" xml:lang="en-us">
```

```
<topic id="abc" xml:lang="de-de">
```

As a general rule, your readers do not see the metadata; they see the results of filtering or use the metadata in their content search.

How do you use metadata?

You can assign metadata to DITA content in several different locations:

- At the topic level
- At the element level
- At the map file level

At the topic level, DITA provides a <prolog> element in which you can store metadata for the entire topic. Here is an example of basic topic metadata:

```
<topic id="xyz">
<title>Metadata example</title>
<prolog>
<author>Sarah O’Keefe, Scriptorium</author>
<critdates>
<created date="2015-05-01"/>
</critdates>
</prolog>
<body>
<p>Body content goes here</p>
</body>
</topic>
```

The author is specified in the <author> element, and <critdates> provides a spot for <created> and <revised> date elements.

Some useful prolog elements are:

- author. The content author.
- critdates. Critical dates, such as <created> and <revised>
- copyright. Copyright year (<copyryear>) and holder (<copyrholder>)
- vrm. The content version

Important: Use the <prolog> metadata only for system information, such as the author and created/revised dates. By default, the DITA Open Toolkit does not support using <prolog> metadata to filter topics. For that, you need to provide metadata in the map file.

At the element level, you use attributes in elements to specify metadata. Here is an example:

```
<step>
<cmd audience= "novice" >Consult the side of the duckling mash box to determine how much mash your ducklings need.</cmd>
</step>
<step>
<cmd>Measure out the mash for your ducks. </cmd>
</step>
<step>
<cmd> Pour in the blender .</cmd>
</step>
<step>
<cmd>Put in feeding pan. </cmd>
</step>
```

Only novices need to be reminded that the box provides measurements. When you generate your output, you could suppress the step with the audience = "novice" metadata for an expert-level audience.

At the map level, you can specify metadata in your topic references. This allows you to suppress entire topics when you generate output. Here is an example:

<topicref href="abc.dita>
<topicref href="def.dita **audience="novice"**>

Keep in mind that you must use the map file for topic-level filtering.

By default, DITA provides you with three attributes that support filtering or conditional processing. They are:

- audience
- product
- platform

If you need additional or different filtering attributes, your information architect will need to specialize to create additional metadata. Some common requirements are:

- customer, for customer-specific information
- region, for information that applies only to specific geographic areas
- product-family, for information that applies to a group of products

Assessments

Where would you find an <author> element?

- DITA does not allow you to use an author element; you have to use an author attribute.

  DITA does not provide an author attribute.
- At the end of the topic body.

  <related-links> is after the topic body.
- In the <prolog> element.

Metadata is information about content.

- True
- False

  Metadata is often described as “information about information.” It provides additional context for your content.

With metadata and output filtering, you can create a content experience that is personalized to a reader.

- True
- False

  If your content includes metadata attributes, such as platform or product, you can filter based on the metadata values to create output variants.

During the publishing process, you can use element metadata to include or exclude content.

- True
- False

  Filtering is one of the primary purposes of metadata.

Be careful when you create metadata—your readers always see these tags.

- True

  It is possible to expose metadata to your readers but by default, they do not see it.
- False

Match the metadata to the meaning:

<vrm> element in prolog	The version of the content
xml:lang attribute on root (for example, <topic> or <concept>	The language of the content
audience attribute on an element	The target audience of the content
platform attribute on an element	The target platform of the content
<author> element in prolog	The person who created the content

The metadata stored in the prolog element is information about the entire topic.

- True
- False

  The prolog is at the topic level. For element-level metadata, use the element’s attributes. Remember that you cannot easily filter on prolog metadata.

If you assign metadata in the topic prolog, you can’t assign additional metadata to pieces of content in the topic.

- True

  The topic prolog contains the topic-level metadata. You can also assign metadata at the element level using attributes.
- False

Metadata is useful because it allows you to (choose all that apply):

- None of these.
- Create conditional outputs
- Reuse content

  You can reuse content that has metadata applied to it, but you can also reuse content without metadata applied.
- Find content
- Translate content

  You can translate content that has metadata applied to it, but you can also translate content without metadata applied.

The following metadata is required for a topic (choose all that apply):

- <title id="xxx">

  The id attribute is required for the topic but not the topic title.
- <topic author="xxx">

  Author information is not required. When you do provide it, use the <author> element in the <prolog>.
- <topic id="xxx">

All content requires specialization.

- True

  You can create unique content that uses the default DITA structure.
- False

Creating relationships among topics

- Understand the purpose and use of a map file
- Understand the purpose and use of a content reference (conref)
- Understand the purpose and use of cross-references (xref) and related-links
- Understand the purpose and use of a relationship table (reltable)

Topics stand on their own, but content in one topic often has relationships to other topics. A map file, for example, describes the sequence and hierarchy of topics for a deliverable.

In addition to map files, you can create relationships among topics in several different ways:

- conrefs
- Cross-references
- Related links
- Relationship tables

Map files

Map files are how you organize content for delivery. They are like a table of contents: they create sequence and hierarchy among topics. When you generate a PDF file or a help system from a map file, your reader will see the topics in the order and hierarchy established by the map file.

You generally do not add all available topics to a map file—just the ones you want included in a deliverable. Also, you can include the same topic in multiple maps files, which is another example of reuse in DITA.

Video: *Example of a DITA map file*

Map files are made up mainly of the following components:

- topicref elements, which provide a link to a specific topic
- mapref elemetns, which provide a link to another map

In a map file, you put topicrefs in order from top to bottom to indicate sequence. To indicate hierarchy, you need the topics. Consider the following example:

```
<map>
<title>My first map</title>
<topicref href="ducks.dita">
<topicref href="range.dita"/>
<topicref href="size.dita"/>
<topicref href="nests.dita"/>
</topicref>
</map>
```

For convenience, the code is indented. But what matters is that the first topicref (ducks) encloses the other three. range.dita, size.dita, and nests.dita are all subtopics of ducks.dita. The result in a table of contents would be something like this:

- Ducks
  - Range
  - Size
  - Nests

Referencing map files in map files

In addition to link to topics, you can reference map files inside map files. In this approach, the subordinate maps (submaps) are usually a collection of related content. For example, you can create a chapter-level map file for each chapter in a book, and then reference those chapter-level map files inside your main book-level map file.

Video: *Reusing a map file in a map file*

Transforming map files into outputs

Topics are organized into map files, from which you create deliverables (outputs). Creating those outputs is called transformation.

Transforms are generally provided to you; they are not something DITA authors generally work on. Stylesheet developers create and modify the transforms.

Transforms are independent of map files. You can apply multiple transforms to a map file. For example, you could run the transforms for a PDF file and an online help system on the same map file to get the two outputs.

conrefs

In DITA, you use a conref to reuse pieces of content. Frequently, this could be notes, cautions, and warnings, boilerplate text, such as your company address, and more. A product's description should probably be the same across all your documents.

If you are familiar with other authoring tools, conrefs are roughly equivalent to:

- Flare snippets
- RoboHelp embedded topics
- FrameMaker import text by reference
- HTML server-side includes

Video: *DITA conref code example*

Cross-references

Cross-reference let you link from one topic to another, or from text to a figure or table caption. You typically link to the figure or table title.

Video: *Code examples of DITA cross-references*

You can use the <xref> element to link to resources outside your DITA topics. For external references, you supply a @scope attribute with a value of external and a @format attribute, as shown here:

```
<xref href="http://www.scriptorium.com" scope="external" format="html"/>
```

Note: For links to a PDF File, use @format = "pdf".

Technically, it is possible to use <xref> to link from one topic to another. This inline linking is a Bad Idea™ because you have to create and maintain the link manually. You have to specify what you are linking to. When you set up the cross-reference, you have the source and target topics in your map file. But if you reuse the source topic in another map that does not include the target topic, you will get a broken link in the output generated from that map, and you probably will not be notified about the problem.

Related links

related links

difference between this and relationship tables

creating external references

Relationship tables

Relationship tables, or reltables, allow us to describe topic relationships that are not sequential or hierarchical. To do so, you create a table. Each row of the table describes a collection of related topics. The reltable is created in its own topic. You must then include that topic in the map file.

The relationships described in the reltable used when you generate output to create a collection of related topics links. By default, each topic with reltable entries gets a Related Topics section at the end of the topic output.

The relationships you capture in the reltable are *not* typically shown when you are authoring topics.

Video: *Overview of DITA relationship table (reltable)*

Start simple with reltables. They can get very complex.

Reltables are preferred over related links or xrefs because of the following factors:

- The topicrefs in the reltable are evaluated against the current map file. If you point to a file that is not included in the map file, that link is not generated in the output. This prevents the broken link problem that can occur with related links and xrefs.
- reltables are easier to maintain then embedded related links. Each row in a reltable can contain multiple topics and captures their interrelationships. So, if you have eight related topics, you can create a single row in a reltable that lists those eight topics, or you can create eight slightly different related-links lists in your eight topics. If you need to remove one topic from the list, the reltable change is done once rather than eight times in the files.

Assessments

To create a link to a figure or table inside the current topic, you should use the <xref> element.

- True
- False

  <xref> elements are dangerous for linking across topics because of the possibility of broken links. Inside a topic, however, you can safely use them.

Creating relationships among topics in DITA requires you to set up a list and manually track which topics are included in the output.

- True

  You have to keep track of cross-references and related links, but if you use a reltable, DITA will suppress links that point outside the scope of the map file.
- False

You can create a link to a web page. The best way to do this in DITA is to use a conref.

- True

  conrefs let you include content from elsewhere at the current location. For web links, use an <xref> element with a @scope = "external and a @format = "html".

- False

Using conrefs allows you to:

- Link to frequently used content
- Be lazy. A good writer varies the content to keep the reader from getting bored.

  In fiction, variety is a good thing. In technical content, consistency is important.
- Plagiarize from other sources.

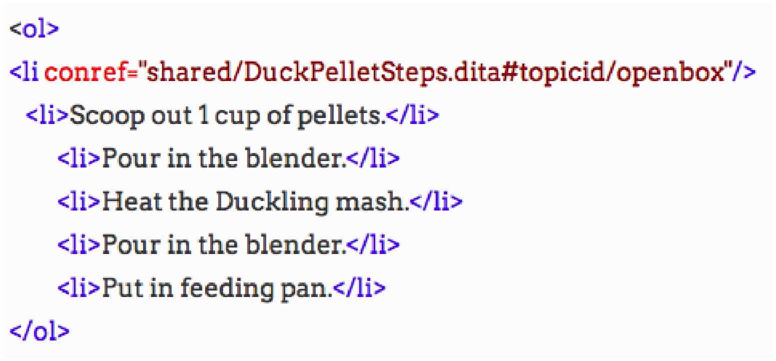
  Reuse in a corporate environment is not plagiarism.

When cross-referencing text in another topic, the best approach is to create an inline link with the <xref> element.

- True

  DITA best practices discourage use of inline links because of maintenance problems. A better approach is a relationship table.
- False

Read the following code. What does this code show?



```
<ol>
<li conref="shared/DuckPelletSteps.dita#topicid/openbox"/>
  <li>Scoop out 1 cup of pellets.</li>
     <li>Pour in the blender.</li>
     <li>Heat the Duckling mash.</li>
     <li>Pour in the blender.</li>
     <li>Put in feeding pan.</li>
</ol>
```

- A list of numbered steps with a conref step.
- A concept topic about feeding ducks.

  This fragment shows only ol and li lists, so you cannot determine its topic type.
- A bulleted list named DuckPelletSteps.dita.

  The filename reference is part of a conref, not a label for this list.

To create a link to a web page, use the following syntax:

- <xref href="http://www.scriptorium.com" scope="external" format="html"/>
- <xref href="http://www.scriptorium.com"/>

  You need the scope and format attributes.
- <xref link="http://www.scriptorium.com scope="external" format="html"/>

  That link attribute should be an href attribute.

reltables describe hierarchical relationships among topics.

- False
- True

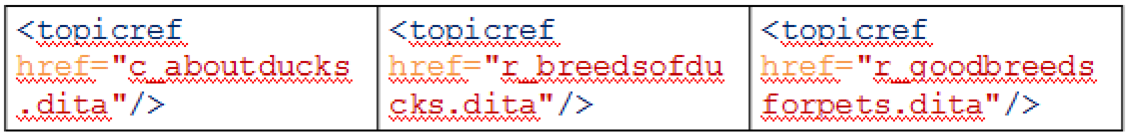
  Map files describe hierarchical relationships. reltables describe peer-to-peer related topics links.

To create related topics links, the best practice is to create a list at the end of each topic.

- False
- True

  reltables are much more efficient and eaiser to maintain than creating links in each topic.

Assume that the following code snippet is in the reltable for the current map file and all of the referenced topics are also in the map file. The c_ prefix indicates concept topics and the r_ prefix indicates reference topics. What will be the result in the related topics sections?



`<topicref href="c_aboutducks.dita"/>`	`<topicref href="r_breedsofducks.dita"/>`	`<topicref href="r_goodbreedsforpets.dita"/>`

- A Related References section at the end of each of the topics.

  What about the concept topic?
- For the reference topics, a link to the other reference topics and a link to the concept topic. For the concept topic, a link to both reference topics.
- No more than one Related Concepts link and no more than one Related References link.

  The concept topic will include references to both reference topics.
- Many inline links.

  reltables let you avoid inline links.
- One related concepts section in these topics.

  What about the reference topics?

A DITA map file does the following (choose all that apply):

- Describe the sequence of the referenced topics
- Describe the hierarchy of the referenced topics
- Creates a link to web content

  External web links are usually xrefs.
- Provides a way to create inline links in topics

  Generally, you should avoid inline links. Map files list a collection of topics, but not content inside the topics.

Tables

-
-
-

Table occupy an uneasy middle ground between formatting and structure.

There are two types of tables in DITA:

- *Simple tables.* The <simpletable> element supports basic tables with minimal customization
- *Tables.* The <table> element can support complex tables with spanned rows and columns and precise display properties

Most markup languages use CALS tables, which were built as part of an initiative of the United States Department of Defense. HTML tables are also built on CALS, so if you are familiar with HTML table markup, you will recognize similarities in DITA tables. Both <table> and <simpletable> are based on CALS.

Simple tables are ideal for basic representation of data within columns and rows. You can use heading rows, but you cannot use a table title, merged columns or rows, or define column widths.

The <table> element allows for more complex arrangements of tabular data and allow table titles.

DITA tables are always described row by row and not column by column.

The <simpletable> elements

Simple tables (<simpletable>) use the following elements to represent and organize tabular data:

- *<sthead> (0 or 1)*

  A simple table may contain one header row.
- *<strow> (1 or more)*

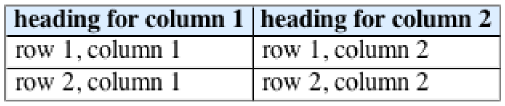
  Contains a single row of data within your table.
- *<stentry> (1 or more)*

  A table cell.

Here is an example of a table with one heading row, two body rows, and two columns.

```
<simpletable>
   <sthead>
      <stentry>heading for column 1</stentry>
      <stentry>heading for column 2</stentry>
   </sthead>
   <strow>
      <stentry>row 1, column 1</stentry>
      <stentry>row 1, column 2</stentry>
   </strow>
   <strow>
      <stentry>row 2, column 1</stentry>
      <stentry>row 2, column 2</stentry>
   </strow>
</simpletable>
```

Visually, the table looks something like this:



heading for column 1	heading for column 2
row 1, column 1	row 1, column 2
row 2, column 1	row 2, column 2

The <table> element

CALS tables (<table>) use the following elements to represent and organize tabular data:

- *<title> (0 or 1)*

  A title is not required. One title is allowed.
- *<tgroup> (1 or more)*

  Every table must contain at least one tgroup, which defines the specifications for columns, rows, headers, and footers. In practice, multiple <tgroup> elements are rare.
- *<colspec> (any number)*

  One per column. Allows you to specify widths, and identify columns for spans.
- *<thead> (0 or 1)*

  One per tgroup. Contains the header row(s) for the table group.
- *<tbody> (1)*

  Contains your table's rows.
- *<row> (1 or more)*

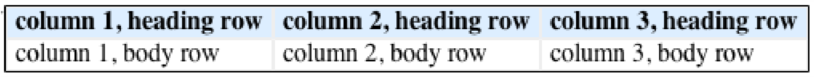
  Contains a single row of data within your table.
- *<entry> (1 or more)*

  A table cell.

Below is an example of a minimal CALS table structure (one heading row, one body row, and three columns). Note that in this case, the <tgroup> specifies only the cols attribute (which is required):

```
<table>
     <tgroup cols="3">
       <thead>
         <row>
           <entry>column 1, heading row</entry>
           <entry>column 2, heading row</entry>
           <entry>column 3, heading row</entry>
         </row>
       </thead>
       <tbody>
         <row>
           <entry>column 1, body row</entry>
           <entry>column 2, body row</entry>
           <entry>column 3, body row</entry>
         </row>
       </tbody>
     </tgroup>
   </table>
```

The result looks like this:



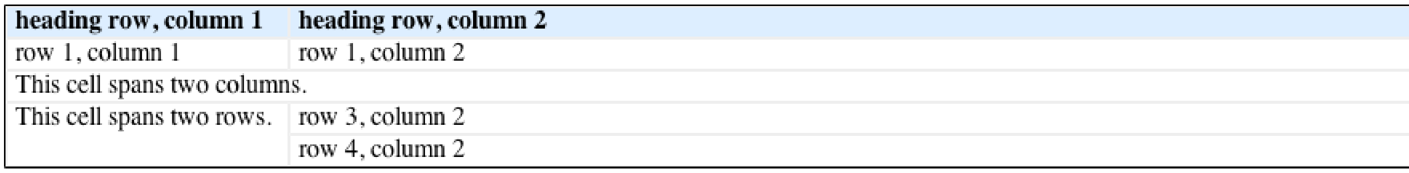
column 1, heading row	column 2, heading row	column 3, heading row
column 1, body row	column 2, body row	column 3, body row

Below is a more complex table:

```
<table>
       <title>My first table</title>
```

```
        <tgroup cols="2">
          <colspec colname="c1" colnum="1" colwidth="1*"/>
          <colspec colname="c2" colnum="2" colwidth="4*"/>
          <thead>
            <row>
              <entry>heading row, column 1</entry>
              <entry>heading row, column 2</entry>
            </row>
          </thead>
          <tbody>
            <row>
              <entry>row 1, column 1</entry>
              <entry>row 1, column 2</entry>
            </row>
            <row>
              <entry namest="c1" nameend="c2">This cell spans two columns.</
entry>
            </row>
            <row>
              <entry morerows="1">This cell spans two rows.</entry>
              <entry>row 3, column 2</entry>
            </row>
            <row>
              <entry>row 4, column 2</entry>
            </row>
          </tbody>
        </tgroup>
      </table>
```

And the visual:



heading row, column 1	**heading row, column 2**
row 1, column 1	row 1, column 2
This cell spans two columns.	
This cell spans two rows.	row 3, column 2
	row 4, column 2

Some notes on the table:

- The <colspec> element gives the columns names (colname), numbers (colnum), and widths (colwidth). In this example, the colwidths are “1*” and “4*”. The result is that the first column gets 20% of the available width and the second column gets 80%. The asterisk indicates that the settings are proportional.
- Each row has a <row> element with <entry> elements for each cell.
- Notice that row 2 has only a single <entry>. That's because the entry spans across both columns, which is specified by the namestart (namest) and nameend (nameend) attributes.
- In row 3, the first entry spans rows 3 and 4, so there you have a morerows attribute.

The easiest way to set up the table code correctly is to use an editor that manages these settings for you. Hand-coding tables is not for the faint-hearted.

Best practices for tables

wrap content in <p> tags

nested tables bad. Legal, but bad.

generally, lots of rows is better than lots of columns because of pagination issues

do not use for visual formatting (indentation of lists, etc.)

localization considerations for tables

graphics in cells? maybe just small icons...

remember that table rendering may be on a small screen

Consider using non-tables and rendering as table on big screen or big paper. example: definition lists, which can be rendered as either a table or as a hanging indent (default)

Assessments

Match the following CALS table elements with their function.

<tgroup>	Defines columns, rows, headers, and footers
<colspec>	Specifies widths and spans
<table>	The primary container for a table
<tbody>	Contains columns and rows
<entry>	A single table cell

Simpletables allow the following elements (choose all that apply):

- <sthead> (header row)
- <stbody> (table body region)

  Only CALS tables define a table body region.
- <strow> (table row)
- <title> (table title)

  Only CALS tables can have titles.
- <stentry> (table cell)

What types of tables does DITA support?

- Tables and Simpletables

  <table> elements are CALS tables.
- CALS tables and Simpletables
- DITA tables and Simpletables

  "DITA table" elements do not exist.
- DITA tables and CALS tables

  "DITA table" elements do not exist.

CALS tables and simpletables both support header rows.

- True
- False

  Both table types within DITA support header rows, but simpletables do not support titles.