Dr Farshad Ghassemi Toosi

Features of DITA

1. DITA Topics
2. DITA Maps
3. Linking
4. Conditional Processing
5. Metadata

DITA Topic Overview

- As stated previously a topic is "A unit of information with a title and content, short enough to be specific to a single subject or answer to a single question, but long enough to make sense on its own and be authored as a unit".
- Although a book model (e.g., DocBook) works well for novels books, it is not well suited for technical information that's delivered within a product.
- A user of a motorcycle doesn’t want to read a novel when it is the time to adjust the valves of her/his motorcycle.

What a topic is going to tell us?

A topic typically answers one of the following questions:

- *How do I do it?*
- *What is it?*
- *What is the process?*
- *How do I fix it?*

A topic needs a home

Although a topic should be self-contained, it should not live alone. For information delivered with most technical products, services or technologies a topic needs a **home** in a larger organised collection of topics.

That collection can then be packaged and delivered in an output format such as HTML Web pages, online help or a PDF manual.

DITA is a topic-based language

DITA is specifically designed to support topic based writing. With its **modular architecture**, semantic **XML elements** and powerful linking features, DITA can help you create and maintain topic based or componentized technical information.

- *Separating information into discrete topics helps you to:*
  - Design new information more easily and consistently.
  - Eliminate un-important or redundant information easier with less risk.
  - Identify common or reusable topics or pieces of content in topics
- *All topics regardless of their purpose, have the following characteristics:*
  - Meaningful titles.
  - Ability to stand alone from other content.
  - Logical Organisation.
  - Links to other topics that contain related information.

Topic Types

There are three core topic types based on a generic topic types which are described as part of this section.

- **Concept** -Contains information for describing a concept, defining a term or describing why a user should perform a task.
- **Task** -Contains step-by-step procedural information for performing a task.
- **Reference** -Contains information about items such as commands programming keywords and error messages.

How topics help

- By separating content by type, you prevent users from wading through information that they don't need.

- For example when you want to know how to install a new audio system in your car for the very first time, you do not need to know about all the details of its control panel (reference information). Rather you just want to install instructions and connect the wires or plugs to the correct sockets.

- By separating the reference information from the task information users can more quickly install their system.

Task Topic

- We can simulate task topics as worker bees in a technical information.
- They are what distinguish technical documentation from a novel or essay.

- Usually task topics need other supporting topics such as concept and reference topics; however user usually starts looking at the task topics and use the concept and reference topics if needed.

How to write a Task Topic?

Things to consider when writing a task topic:

1. *Make sure you are separating task information from reference and concept information.*
   - *Not too much details. Short enough. Retrievable. Reusable. Too much details and conceptual info makes users frustrated.*
2. *One procedure per topic.*
   - *Having one procedure per task topic makes it easier to to manage, organize and reuse the topic in future and makes it easier for user to find a specific task.*
3. *In the case of a long procedure, create subtasks.*
   - *Some procedures are very long. Instead of having a long procedure in a single topic, you can break it down into multiple smaller task topics. You can later, assemble those task topics and create a logical order so that helps user to finish the procedure.*

How to write a Task Topic?

Structure of a task topic:

- A task usually starts with a brief introduction.
- Include prerequisites if required.
- List all the steps to complete the task.

- A small amount of conceptual or reference information is OK if they are directly directed to the task topic.

Task Elements

- Use task topics to describe the steps of a particular task.

- Task topic has sections that describes the context, prerequisites, actual steps, expected results, example and expected next steps for a task.

Task Elements

- Title (see some details here)
  1. The title should imply that this is a task topic. Do this by using gerund or imperative verb phrase or “How to ” phrase in the task title. Use gerund.

1. Bake chocolate cakes (incorrect)
2. Baking chocolate cakes (correct)

Task Elements

- Title (see some details here)
  1. The title should imply that this is a task topic. Do this by using gerund or imperative verb phrase or “How to ” phrase in the task title.
  2. Try to use plural heading unless the context makes only sense in singular.

1. Using external keyboard with laptop (incorrect)
2. Using external keyboards with laptop (correct)

Task Elements

- Title (see some details here)
  1. The title should imply that this is a task topic. Do this by using gerund or imperative verb phrase or “How to ” phrase in the task title.
  2. Try to use plural heading unless the context makes only sense in singular.
  3. It should focus on user’s goal and not the tool.

1. Using the spell checker in Microsoft word (incorrect)
2. Correcting documents using spell checker (correct)

Task Elements

- Title (see some details here)
  1. The title should imply that this is a task topic. Do this by using gerund or imperative verb phrase or “How to ” phrase in the task title.
  2. Try to use plural heading unless the context makes only sense in singular.
  3. It should focus on user’s goal and not the tool.
  4. Heading with specific topic heading.

1. Installing antiviruses on Windows machines to protect them against viruses (incorrect)
2. Installing antiviruses (correct)

Task Elements

- shortdesc
- The most important element in the topic.
    1. This element is seen in links, search engine results and it acts as the first paragraph of your topic.
    2. It has the following details:
        1. An overview of the procedure
        2. The benefits or importance of the task
        3. Limitations or requirements
        4. Brief conceptual information.

Task Elements

- **<taskbody>**
  - The <**taskbody**> element is the main body-level element inside a task topic.
  - A task body has a very specific structure, with the following elements in this order: <*prereq*>, <*context*>, <*steps*>, <*result*>, </*result*>, <*example*> and <*postreq*>.
  - Each of the body sections are optional.

Task Elements

- **<context>**
  - The <context> element is used to provide a background information for the task.
  - It helps the user to understand what the purpose of the task is and what they will gain by completing the task.
  - Note: This section should be short enough and does not recreate a concept topic on the same subject, although the context section may include some conceptual information.
  - Note: If the task is short or relatively simple and you provide an effective introduction in the <shortdesc> element, omit the <context> element.
  - You can also include a step introduction in the <context> or <stepsection>element to make it easier for users to quickly locate the procedure.

Steps

- The <*step*> element represents an action that a user must follow to accomplish a task. Each step in a task must contain a command <*cmd*> element which describes the particular action the user must do to accomplish the overall task.
- Beginning with DITA 1.2, it is possible to place a <*note*> element before the command in order to notify the user of dangers or other important information about the step. <hazardstatement> is anther element that can be used within step.
- The <step> element can also contain additional optional information about the step, such as substeps, a list of choices, or result information. (we will review them shortly)

Steps

- Steps can be either **ordered** or **un-ordered**. This section describes how to write procedures using these types of steps.
- In DITA, you can describe two types of steps in a task topic:
  - **Procedures with ordered steps** - ordered steps use the **<steps>** element, which is the outermost container for your steps. You should use this element for most of your tasks.
  - **Procedures with un-ordered steps** – un-ordered steps use **<steps-unordered>.** Depends on the context of the procedure, you can decide to use either ordered or un-ordered steps.

Steps can be optional or required. You can use an attribute in step element to provide this information.

```
<steps>
        <step importance="optional">
            <cmd>Add some ginger powder </cmd>
        </step>
        <step importance="required">
            <cmd>Add sugar</cmd>
        </step>
        <step importance="required">
            <cmd>Add flour</cmd>
        </step>
    </steps>
```

1. **Optional:** Add some ginger powder
2. **Required:** Add sugar
3. **Required:** Add flour

Steps (More details)

- Each step can be clarified more by adding more elements:
  - <info>
  - <choicetable>
  - <choices>
  - <stepresult>
  - <stepexp>
  - <substeps>

Steps (More details)

- <info>
  - Provides additional information about a step.

```
<steps>
    <step importance="optional">
        <cmd>Open the remote control box</cmd>
        <info>
            The remote control is in a blue box inside of the main television box.
        </info>
    </step>
</steps>
```

Steps (More details)

- Choice elements
  - Note: There are two elements that sound similar to each other:
    - <choices>
    - <choicetable>

Steps (More details)

- <choices>
  - Use <choices> when there is a simple one-part items. It renders output as bullet points.

```
<steps>
    <step>
        <cmd >Mix all the ingredients</cmd>
        <info>Make sure to mix them properly</info>
        <choices>
            <choice>
                If you want to bake darker chocolate cake, use 120g dark chocolate
            </choice>
            <choice>
            </choice> If you want to bake normal chocolate cake, use 60g dark chocolat
        </choices>
    </step>
</steps>
```

Steps (More details)

- < choicetable >
  - Use < choicetable > when there is two-part items such as an ingredient and its amount. Choicetable creates a table-like structure to accommodate data.

```
<step importance="required">
        <cmd>Choclate Cake with White Choclate</cmd>
        <choicetable>
          <chhead>
            <choptionhd>Ingredients</choptionhd>
            <chdeschd>Amount</chdeschd>
          </chhead>
          <chrow>
            <choption>Cream</choption>
            <chdesc>70g</chdesc>
          </chrow>
          <chrow>
            <choption>Sugar</choption>
            <chdesc>50g</chdesc>
          </chrow>
          <chrow>
            <choption>Milk</choption>
            <chdesc>40g</chdesc>
          </chrow>

        </choicetable>
</step>
```

See StepsChoiceTable.dita example

Steps (More details)

- < stepxmp >
  - Use <stepxmp> when there is a need to illustrate a step of a task. It can be simple just a couple of words or larger amount of text e.g., paragraph.

```
<steps>
        <step>
           <cmd>Add your favourite flavour</cmd>
           <stepxmp>For example mint and mango</stepxmp>
        </step>
        <step>
            <cmd></cmd>
        </step>
 </steps>
```

Steps (More details)

- < stepresult >
  - Use < stepresult > when you want to provide some expected outcome of a step. It is a good idea to use step-result so that user knows that they are on track. Obviously you shouldn't use it for every single step otherwise it becomes complicated.

```
<steps>
    <step>
      <cmd>Add some water into the mixture </cmd>
      <stepxmp>The mixture should now become softer</stepxmp>
 </step>
```

Steps (More details)

- < substeps >
  - Use < substeps > when you need to break a step down into a series of separate actions, and should be used only if necessary.
  - Try to describe the steps of a task in a single level of steps. If you need to use more than one level of substep nesting, you should probably rewrite the task to simplify it.

See taskStepSubSteps.dita example

Unordered Steps

< steps-unordered>

- Use < steps-unordered> when the ordering is not important. The output will be a bulleted list instead of numbered steps.

```
<steps-unordered>
        <step>
           <cmd>Prepare a mixture of choclate chips and cream</cmd>
        </step>
        <step>
           <cmd>Mix eggs with Flour </cmd>
        </step>
 </steps-unordered>
```

Prerequisites

- Prerequisite is any *requirement* that a user must provide, any *task* that user must perform or any *authority* or *privilege* that must be set up before the user can do the actual task.
- Note: Not all tasks have prerequisite.
- Task topic has an element to describe the prerequisite. <prereq>
- <prereq> can contain any element in it such as ordered or unordered steps, paragraph.
- Remember to not to put too much info or procedure in a <prereq>.
- If there is a long list of steps for <prereq>, you can create another task topic and have that list in a separate topic.

Prerequisites

- Prerequisites come before the first step.
- Some examples:
  - You should have an oven with timer
  - You should have an oven cake
- Sometimes your prerequisites are explained in another task then you need to link to that task topic.

```
<prereq>
        You should have an oven with timer
    </prereq>
    <steps>
      <step>
        <cmd>Put the mixture into the oven</cmd>
      </step>
      <step>
        <cmd>Wait for 40 minutes</cmd>
      </step>
  </steps>
```

Prerequisites

- Sometimes your prerequisites are explained in another task then you need to link to that task topic.
- When you link to a prerequisite task, do not create an inline link that is part of a sentence.
- Linking will be discussed more in details later …

```
<prereq>
        Here is what
        <xref href="prerequiste.dita"></xref>
        you need to prepare.
</prereq>
```

```
<prereq>
        Here is
        <xref href="prerequiste.dita"> what </xref>
        you need to prepare.
</prereq>
```

```
<prereq>
        You need to prepare some ingredients first
        <xref href="prerequiste.dita">See the list of ingredients</xref>
</prereq>
```

Some other elements in Task topic

- <steptroubleshooting>
- <postreq>
- <result>
- <prolog>
- <fig>
- <table>

Some other elements in Task topic

- <steptroubleshooting>
  - Use as step trouble shooting if there is possibility of happeing a

```
<steps>
        <step>
          <cmd>Put the mixture into the oven</cmd>
        </step>
        <step>
          <cmd>Wait for 40 minutes</cmd>
        </step>
        <step>
          <cmd>Slice the cake using knife</cmd>
          <steptroubleshooting>
            If you could not cut the cake by knife, add some water.
          </steptroubleshooting>
        </step>
</steps>
```

Some other elements in Task topic

- <postreq>
  - The <postreq> element describes steps or tasks that the user should do after the successful completion of the current task. It is often supported by links

```
<steps>
   <step>
      <cmd>Mix sugar with flour</cmd>
   </step>
   <step>
      <cmd>Add dark chocolate</cmd>
   </step>
   <step>
      <cmd>Bake the cake in oven</cmd>
   </step>
</steps>
<postreq>
   Cake icing
</postreq>
```

Some other elements in Task topic

- <result>
  - The <result> element describes the expected outcome for the task as a whole.

```
<steps>
   <step>
      <cmd>Mix sugar with flour</cmd>
   </step>
   <step>
      <cmd>Add dark chocolate</cmd>
   </step>
   <step>
      <cmd>Bake the cake in oven for 40 minutes</cmd>
   </step>
   <step>
      <cmd>Wait 40 minutes and ta</cmd>
   </step>
</steps>
<result>
   The cake is now properly baked and ready to be served.
</result>
```

Some other elements in Task topic

- < related-links>
  - The related information links of a topic (<related-links> element) are stored in a special section following the body of the topic. After a topic is processed into its final output form, the related links are usually displayed at the end of the topic, although some Web-based help systems might display them in a separate navigation frame.

```
<related-links scope="external" format="html">
    <link href="http://www.something1.org">
        <linktext>Bake my cake</linktext>
    </link>
    <link href="http://www.something2.com">
        <linktext>I love choclate cake</linktext>
    </link>
</related-links>
```

Some other elements in Task topic

- < prolog > (Before shortdesc after title)
  - The <prolog> element contains information about the topic as an whole (for example, author information or subject category)
  - Much of the metadata inside the <prolog> will not be displayed with the topic when the topic is rendered, but it might be used by processes that generate search indexes or customize navigation.

```
<prolog>
    <author>Farshad </author>
    <copyright>
      <copyryear year="2019"/>
      <copyrholder>CIT</copyrholder>
    </copyright>
  </prolog>
```

Some other elements in Task topic

- < fig >
- The <fig> element is a figure with an optional title for a wide variety of content.
- A figure commonly contains an image or artwork, but it can contain several kinds of text objects as well.
- A title is placed inside the figure to provide a caption that describes the content.

```
<fig>
    <title>A choclate cake</title>
    <image href="cake.jpg" placement="break"></image>
</fig>
```

Some other elements in Task topic

- < table >
- The <table> element is one of the very useful elements that can help us to create tables with any size.
- Use colsep and rowsep to draw borders between columns and rows.

```
<table>
  <tgroup cols="3">
    <colspec colname="name" ></colspec>
    <colspec colname="surname" ></colspec>
    <colspec colname="age"></colspec>
    <thead>
      <row>
        <entry>Name</entry>
        <entry>Surname</entry>
        <entry>Age</entry>
      </row>
    </thead>
    <tbody>
      <row>
        <entry>Farshad</entry>
        <entry>Toosi</entry>
        <entry>98</entry>

      </row>
      <row>
        <entry>Andrew</entry>
        <entry>Jack</entry>
        <entry>99</entry>

      </row>
    </tbody>
  </tgroup>
</table>
```

Formatting elements

- <p> Indicates that text should be in a paragraph,
- <b> Indicates that text should be bold,
- <i> Indicates that text should be italic,
- <u> Indicates that text should be underlined,
- <sub> Indicates that text should be subscripted,
- <tt> Element is used to apply monospaced highlighting to the content of the element. This element is part of the DITA highlighting domain. Use this element only when a more semantically appropriate element is not available.


<dita>