



**MIDDLESEX Community College**

**Tools and Technologies for Tech Writers 2024**

**Week 11**

# **Sample Bookmap**

**Zoë Lawson**

# Middlesex Community College 2024

This document was prepared as an assignment for the Middlesex Community College Tools and Technologies for Technical Writers class, Winter semester 2024.

Prepared by Zoë Lawson.

The name is defined by the value of the author key defined in your bookmap. I set them up to be your name.

# Contents

**Topic..... 4**  
    Example Concept File.....4  
    Example Task Topic.....4  
    Example Reference Topic..... 5

**Class Demo Concept..... 6**

**The dreaded task..... 8**

# Topic

This is a sample topic. It is in your homework folder and is not shared.

This is the default, base topic type for DITA. This topic is not specialized in any way. Technically, you can use it any way you'd like.

Here is another paragraph.

Since this topic is not shared (it is not in the common folder, it is in your personal folder, Zoe), any changes you make to it will not affect anyone else's output.

If you make changes to files in common folder, everyone gets the change. I humbly request you don't change the common files.

You can change and add all the files you want in your folder, `mcc_tools_tech/Week11-DITA/Homework/Zoe`.

## Example Concept File

An example DITA topic of type concept. This is a shared topic.

Concepts should describe things. All the background information about the thing.

Concepts are not overly structured.

## Example Task Topic

A DITA topic of type task. This is a shared topic. Task topics are the most structured.

This is the `<prereq>` where you list the prerequisites for this task. This section is optional.

This is the `<context>` where you can provide a bit of background of things you should know before performing the task.

Some people consider this a "mini concept". If you realize you need a bit of lead in, but not enough for a stand alone topic, use the `<context>` in the task topic instead of a stand alone concept topic. This section is optional.

1. This is a `<step>` in a task topic. It is part of the `<steps>` element. You can use `<steps>` for numbered steps or `<steps-unordered>` for a non-numbered (usually bulleted) steps.
2. These are the heart and soul of task topic.

There are many additional special elements that can be used in a step to provide more information.

3. Both the `<steps>` and `<steps-unordered>` elements are optional.

That's correct, you can have a task with no procedure.

This is the `<result>` which lets you explain what happens after you complete the steps. This element is optional.

This is `<tasktroubleshooting>` where you can provide troubleshooting information for the entire task.

Not to be confused with `<steptroubleshooting>` which can be used for troubleshooting information for specific steps.

This element is optional.

This is an `<example>` you can use to provide an example for the entire task. This is not to be confused with the `<stepxmp>` which can provide a quick example for a specific step.

This element is optional.

This is the `<postreq>` where you describe what to do after completing the procedure.

This element is optional.

That's correct, all of these elements are optional. It is possible to have a valid task topic with only a `<title>` and `<shortdesc>`. However, the order of these main elements is not optional. For example, you cannot switch the order of the `<context>` and `<prereq>` without a specialization.

## Example Reference Topic

---

This is a DITA topic of type reference. This is a shared topic.

In general, a reference topic is used for things you look up. If you want to put it in a table, it probably should be a reference topic.

Reference topics can contain:

<code>&lt;section&gt;</code>	Generic text space.
<code>&lt;example&gt;</code>	Generic text but the intention is to hold some sort of example.
<code>&lt;properties&gt;</code>	A special list/table for properties and definitions.
<code>&lt;refsyn&gt;</code>	Syntax reference. Great for APIs.
<code>&lt;simpletable&gt;</code> or <code>&lt;table&gt;</code>	You can have a table.

# Class Demo Concept

Learn how to write good short descriptions.

Use the <p> element for paragraphs.

Here is a second paragraph.

- Unordered lists for bulleted list
- same as html.

same iwht ordred lists.

1. Here is an ordered lish.
2. Item 2.

Here is a paragraohs inside a li.

- Here's a sub bullet
  - a. here's a nested list.

Heading	heading
jdkfal;	djkla
lormen	ipsum
icecream	sundae



Figure 1: This is a figure title


```
Great code examples.  
This preserves  
  white  
    space.
```

Here is a paragraph

There isn't a way to just indent paragraphs.

**Note:** This is a note of something to pay attention to.

**Tip:** This a tip on how to do something.

 **CAUTION:** This is a caution.

This is now a p. There is all sorts of inline formatting. [This is a link to google](#).

Here is a link to [Example Reference Topic](#) on page 5. You can also use cut and paste from the maps manager. [Example Concept File](#) on page 4.

DITA uses semantic tags. **uicontrol** for items you click in a GUI. **wintitle** for dialog box names. `codeph` for code phrases. `filepath` for paths and file names. `msgph` for message phrase.

Here is a p with a Zoë Lawson.

# The dreaded task

This is the Prereq.

1. Do the thing.

This is information you need.

- A
- b
- c

**Option**

**color**

**Description**

Set the color.

For example, enter Yes.

The blah dialog box opens.

You may need to click the option twice.

2. Do the thing.

- a) Do this.
- b) And this
- c) then this