KalyanPonnekanti.github.io Documentation

Release latest

Sep 17, 2019

Contents

CHAPTER 1

DITA from FrameMaker: Two Schools of Thought

Traditionally in our product group, we have been using unstructured FrameMaker as the main authoring tool afor our product documentation. In late 2017, we received a requirement that customer requires our documentation source in DITA format with the PDFs too. This requirement posed a challenge. We did not have any experience working with DITA earlier, and we did not know the way forward. We spent a considerable amount of time researching how to convert the sources. This customer had two requirements:

- Needs source in DITA
- Needs product updates for the next two years (that means the DITA conversion is not a one-time job, and the DITA source must be up-to-date always).
- Internal requirement
- Reuse content across product lines

We reached out to other teams within my company to check if they had any experience with DITA/conversion from FrameMaker and any other learning experiences.

1.1 Converting FrameMaker to DITA

Our research on this project led us to the following two schools of thought:

- Maintain our documentation source in FrameMaker and convert the files to DITA each release to the customer.
- Convert the FrameMaker source to DITA, and continue to use the DITA source.

A team in our business group followed method 1. Whereas, we chose to use method 2.

1.2 What the Other Team Did

The other team made it possible by moving to their content from unstructured FrameMaker to structured FrameMaker.

1. Convert unstructured FrameMaker to structure FrameMaker

2. Create mapping tables to convert structured FrameMaker to DITA each release.
3. Run FrameMaker scripts to clean up the DITA sources.

1.3 What We Did

We took the second approach. Our internal content reuse was the main reason we chose the second approach. We used a third-party service (Stilo Migrate) to convert our unstructured FrameMaker source to DITA. This task involved pre-conversion and post-conversion tasks. Continued using DITA as a the main source for all our subsequent work.

You could choose either approaches. There is no correct and incorrect approach. Approach 2 worked better for us our biggest driver was content reuse. Approach 1 works better when there is no content reuse.

1.4 Challenges with our Approach

- The number of objects to manage after conversion to DITA is huge. We might need a CCMS to manage the DITA source.
- We needed to invest in a DITA-aware editor. We continue to use FrameMaker as the XML editor too.
- With a regular repository as Perforce, we could not check out only the files that we need. A CCMS would solve this problem.

More about the pre-conversion and post-conversion tasks, CCMS, and DITA to PDF publishing workflow in future posts.

CHAPTER 2

Customizing Changelist Description in Perforce!

2.1 Customizing Default Changelist Description in Perforce

To improve the traceability of commits to perforce, we were looking to provide a changelist description template to the team. We use Perforce as a repository running on a Windows platform.

Research from the internet suggested that we would have to use P4 triggers. However, our IT team had restricted the usage of triggers on our perforce server. So, my colleague, Krishna (https://krishkasula.github.io/), ended up writing a client-side python script that allowed us to customize the changelist description for Perforce.

Script:

```
#! /usr/bin/env python3
# ------------------------------------------------
# Author:    krishna
# Created:   Wed Jun 12 11:26:39 2019 IST
# USAGE:
#       createNewChange.py
# Description:
#   Creates a new numbered perforce change and shows the Number in a dialogue box
#
# ------------------------------------------------
import ctypes
from P4 import P4

def main():
    '''The Main'''

    # Create a new perforce object and connect
    p4obj = P4()
    p4obj.client = 'user_workspace'
    p4obj.connect()
    if not p4obj.connected():
        ctypes.windll.user32.MessageBoxW(0, "Unable to Connect", "Error", 0)
        return
```

(continues on next page)

(continued from previous page)

```
    # Create a new perforce change with below description
    change = p4obj.fetch_change()
    change._description = """
    Fix for STAR:
    Release (LCA/GA/version):
    Customer Name?
    For RCT?
    Section that is changed in the source:
    Creating PDF?
    PDF Edition Number?
    Technical Review Done?
    G.Review?
    Related HLA:
    """

    # Save and show it on a dialogue box
    ctypes.windll.user32.MessageBoxW(0, *p4obj.save_change(change), "New P4 Change",␣
↪0)

    p4obj.disconnect()


if __name__ == '__main__':
    main()
```

2.1.1 Prerequisites for Running the Script

1. Install Python 3. Ensure that you let the installer to update the environment variables.
2. Install p4python. Use the following command: `pip3 install p4python`

2.1.2 How the Custom/Personalzied Changelist Script Works

1. Run the script:
    1. Save the script to your computer with the .py extension. For example, I saved the script to the file: `create_changelist.py`.
    2. Update the string `user_workspace` with your P4 workspace.
    3. To customize the changelist description, edit the text in the block: `change._description = """ ..... """`
    4. Double-click the python file (`create_changelist.py`) to execute it.

    A new numbered changelist (each changelist other than the *default* changelist has a number) is created. If successful, you must see a message similar to the following:
2. When you open the Perforce client (we use Perforce Helix P4V) and try to check out a file/folder, the changelist created in Step 1 is available in the dropdown.
3. Select the changelist. The changelist is auto-populated with the template description.
4. Fill the description and continue the check out.

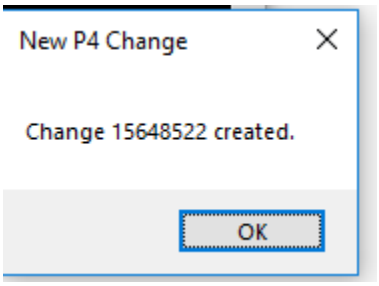

Fig. 1: Numbered Changelist is created

CHAPTER 3

Using IPXACT to Generate Documentation

Since last year, we have seen an increasing interest from customers regarding IPXACT. Customers specifically request the external interfaces and register descriptions in the IPXACT. Recently, we had a customer who requested DITA sources as well as IPXACT. We gave early access to our DITA sources to our customers. The initial feedback on the DITA sources was disappointing. The customer wanted modifications to our register layout description to match her output. This request put us in a quandary. This customer-specific change will impact all customer documentation. And, it would also require additional verification effort. To resolve this issue, we scheduled a call with the customer to understand their justification for this change. The outcome of this call is the following use case.

3.1 Use Case: Using IPXACT to Generate Documentation

Actor: Integration Engineer/Automation

Industry: Semiconductor

Use Case Overview: Generate external interfaces and register documentation from IPXACT.

Basic Flow

1. Vendor provides external interface and register information in IPXACT format.
2. Integration engineer integrates the IPXACT into their documentation flow.
3. The documentation generation flow generates DITA from IPXACT.
4. The documentation publishing flow generates PDFs/HTML from DITA.

3.2 What is IPXACT?

"**IP-XACT** is an XML format that defines and describes individual, re-usable electronic circuit designs (individual pieces of intellectual property, or IPs) to facilitate their use in creating integrated circuits (i.e. *microchips*). IP-XACT was created by the SPIRIT Consortium as a standard to enable automated configuration and integration through tools."

https://en.wikipedia.org/wiki/IP-XACT

3.3 What is DITA?

"The **Darwin Information Typing Architecture** or **Document Information Typing Architecture** (**DITA**) is an XML data model for authoring and publishing. It is an open standard[1] that is defined and maintained by the OASIS DITA Technical Committee."

https://en.wikipedia.org/wiki/Darwin_Information_Typing_Architecture

3.4 How did We Resolve the Customer Request

As the customer requested IPXACT and DITA. We managed to convince the customer to generate the register descriptions from the IPXACT than reuse our DITA. However, this meant, that we had to restructure the document (containing register information) to consolidate the register descriptions in one Part so that customer could easily replace it with the output from IPXACT.

CHAPTER 4

Unexpected or Corrupt Characters in Headings in a PDF Generated from Word

4.1 Problem

When you convert a Word document to PDF, Heading 1 displays unexpected characters. Interestingly enough, the PDF Bookmarks do not display these unexpected characters. Following are some of the symptons:

- Text appears to melt or characters overlap
- Text includes seemingly garbage characters

Yesterday, we had a product release. We received some last minute changes to the Word source. After processing the new content, we published the Word document to PDF. Then, things started going wrong. The chapter headings in the PDF had corrupt characters. They looked like the following:

Fig. 1: Corrupt or Unexpected Characters in Headings in PDF

4.2 Workaround

The first thing we did was to check the Word source. The Headings looked proper in the source. We felt the encoding went wrong when converting to a PDF. So, we copied the corrupt characters and pasted them in the browser. Lo, and behold, the text was rendered properly and it matched the Word source. Now, we were stumped on how to resolve the issue. We realized this is a recurring issue in Word sources that are edited by multiple authors (with Word language settings set to any of the European languages). Finally, we figured the following workarounds to this issue:

- Save the document as `.doc` instead of `.docx`, and publish to PDF.
- Change the font of the headings from **Helvetica** to **Arial**.

CHAPTER 5

Retain Formatting After Updating a Cross Reference

5.1 Problem

You insert a cross reference to a section. And, you apply a character style to display the link in blue. However, when you update the cross reference, the character styling is lost. For example, if you have applied a style to display the link in blue, the formatting is lost when you update the cross reference. The link does not appear in blue anymore.

Tool: Microsoft Word

5.2 Workaround

To retain the character styling of a cross reference after updating it, do the following:

1. Right-click the cross reference, and select **Toggle Field Code**.

   The field code is displayed. The field code is similar to the following:

   ```
   {REF _Ref485728411 \h}
   ```

2. Append the following string to the field code:

   ```
   \*Charformat
   ```

   So, the cross reference field code must be similar to the following:

   ```
   {REF _Ref485728411 \*Charformat \h}
   ```

3. Right-click the cross reference, and select **Toggle Field Code**.
4. Select the cross reference and apply the character style that you want to use. For example, apply blue font color to the cross reference.
5. To test the solution, right-click the cross reference and select **Update Field**. The blue color must be retained after the update.