Recent Developments – Activity Report – 2017-08-30

As the manual grows in size, it grows in complexity, of course. I'm talking about under the hood – ensuring sustainability and consistency and reliability of information.

Modules of Content

For maximum sustainability of the manual and maximum consistency and reliability of the information in it, the content must be modularized.

This is so that the modules of content/information can be:

- arranged and rearranged,
- selectively published or unpublished, or
- repeated in appropriate places throughout the manual -- in which case, you need *absolute consistency* among the various appearances of that content. That starts out as difficult, but (normally) becomes impossible as the manual increases in size, complexity, and undergoes edits and updates in the future.
- reused (repeated) in the *printed output* as well as in the online output (That's not as simple as it sounds)

Modules of content might be:

- "Topics" (in the lexicon of the RoboHelp program)
- Pages of content such as exposition that's common (and therefore "re-used") across chapters (or across "books," in the lexicon of the RoboHelp program.)

(I'm not talking about snippets, user variables, or other types of modular, reusable content here. Just topics and pages. That keeps the file structure simpler.)

Topics (pages of content) that are common amongst (appear in) multiple chapters -- or steps in a process that are identical across several different processes -- ought to be absolutely identical. To ensure this, they ought to be copies or mirrors of one another, and any editing done in any one of them ought to automatically be reflected across all appearances of them, throughout the manual.

Merged and Nested TOCs

I have recently solved a logistical/technological issue by employing nested TOCs.

Doing this ensures the following:

- (Resolved:) All duplicate appearances of modular content repeated throughout the manual that ***ought*** to be identical ***will*** be identical. Editing any one of them ***will*** edit all others, automatically. This guarantees consistency. It is a very big deal. I've done this by employing nested TOCs. (This is particularly important when an individual step or steps in a process ought to recur *identically* throughout multiple, *otherwise different*, processes.)

- (Resolved:) Intentionally duplicate content will appear in ***all*** locations where it should throughout the ***printed output*** of the manual – not only the online output. Previously, (intentionally) duplicate content appeared throughout the ***online*** manual as intended, but in the ***printed*** manual, only the *first instance* of the content would appear. The others (technically, placeholders) did not appear in the printed output. This is resolved. This, too, is a big deal. I've done this, too, by employing nested TOCs.

Other Developments

- I've created individual conditional build tags for each separate chapter of the manual. This streamlines RoboHelp interface and production of the manual. It modularizes content under the hood for effective workflow in terms of publishing. When creating publishing presets ("SSL Outputs"), it allows me to toggle individual chapters on or off.

- I've created two individual SSL (Single Source Layout) Outputs: namely, **Development** (Output) and **Production** (Output).

  SSL Outputs are publishing presets that streamline the RoboHelp interface and workflow. It allows for single-sourcing of, essentially, two different products: the working draft of the manual on the Development Server, and the live, published version of the manual on the Production Server.

  The content on the Production Server is a subset of the content on the Development Server. This is done through creating conditional build tags for

each chapter (as well as for more granular content), then employing filters when creating publishing presets (i.e. “SSL Outputs”).

Quality Improvement: I recommend eliminating the Filter function of the user interface

Why?

- Saves a little bit of time (but more importantly, reduces complexity)
- Sheer amount of code and formatting attached to a given page is reduced
- Less code means simpler and more stable code
- Helps unclutter and streamline the user interface for the *RoboHelp* user
- I don’t think anyone will miss it

  Building the Filter functionality of the user interface means using the same kinds of conditional build tags used under the hood to streamline publishing. So, any given page or topic could have multiple conditional build tags attached, making them hard to identify. It increases complexity and opportunities for human error on the part of RoboHelp user. It would be beneficial to reduce the sheer number of conditional build tags in the RoboHelp interface.