Dynamic Merge for WebWorks Help 5.0

Introduction

The Problem

While the "Out of Memory" error of earlier versions has largely been eliminated in ePublisher 9.3, very large projects are still seeing their share of problems. In nearly all cases, system memory becomes an issue in the Page pipeline, when ePublisher is generating all of the HTML files required to complete a format conversion. The number of HTML files created depends on the total number of input documents, and—more specifically—the total number of page splits within those documents. There is no exact limit, but it seems like projects with 50+ documents and several hundred page splits are at risk of exhausting the available RAM.

The problem isn't really the number of files being created, though. The current understanding is that memory more likely is drained by the processes of tracking all of the files associated with a project, complete with lengthy path names, complex attributes, and multiple dependencies. During the conversion, ePublisher maintains all of this information for each project group in a file in the Data directory called *files.info*.

As the number of groups and documents increases, so does the size of each *files.info*. Then the conversion starts. WIF, XML, PostScript, images, CSS, HTML…every file created as part of the ePublisher conversion process is logged in the Data file and stored in the system memory. In very large projects with many page splits, the size of *files.info* skyrockets in the Page pipeline. Eventually, ePublisher uses up all of the available memory and the system (no matter how much RAM) throws a warning. In ePublisher 9.2.2, this warning signaled the end of the conversion with the fatal "Out of memory exception" error.

We worked around this issue with ePublisher 9.3 by building in some intelligence to look for that error before it shuts down the conversion. When the system warns that it is out of memory, ePublisher stops what it is doing and restarts the current process from the beginning. It may jump back to the beginning of the Page pipeline or the beginning of the group that it was processing when the limit was reached. The idea was that it could clear up the memory being wasted and make more progress after the restart. What seems to be happening is that progress (or enough progress) is not made. You'll often see projects stuck in the Page pipeline after many hours of processing. The application has not frozen, but it is forced to keep restarting and attempting the same process over and over without success. It might make a few extra steps each time, and it may *eventually*

finish, but the Platform loses its usefulness (its quickness, flexibility, and efficiency) when it functions like this. It's just not the way ePublisher was intended to perform.

So, while our engineers work on a built-in solution to this problem for the next release, we offer this external workaround to help ease the pressure felt by many users with very large projects.

The Solution

A script was developed to allow the merge of multiple WebWorks Help 5.0 output directories into a single helpset. In a typical project with multiple groups, we create an individual helpset for each group, and then we run a final process to combine the groups into a single merged helpset with a combined TOC and Index. That's really what we'll be doing here. We will take several multi-volume helpsets, and we will run a script that will allow a user to view them all as a combined, cohesive unit. You will have the ability to define the sort order of the TOC entries, and you can even modify the file system organization.

This should help to resolve the memory issue by allowing you to create several smaller projects that each generate output without hitting the limits of your system's memory. Then, with this post-process script, you can combine the output of each of those projects in a single directory, as if they had all been generated simultaneously. It requires some work to get things configured and working correctly, but it should give you the results you need to deliver your complete project output.

Using Dynamic Merge

How It Works

Creating merged WebWorks Help 5.0 output does not require the creation of many new files and directories. Rather, it reconfigures existing files in a directory so that each knows the others exist. It also defines the relationships between the child groups so that the features of the merged helpset reflect the contents of all of those groups. For example, a new TOC is not created; but, if the top-level files know that the child groups exist, the tables of contents of those groups can be combined on-the-fly to display a merged TOC to the user at runtime.

Set it up

If your current project is stalling in the Page pipeline or takes an unreasonable amount of time to finish converting output, you'll want to start by breaking it into smaller, more manageable projects. Start with your stationery, and create several new ePublisher Express publication projects, each with a subset of the total number of groups and documents. Generate output for each of the projects, ensuring that they all complete the process successfully. If one of the smaller projects stalls, then break it down into smaller

parts. If necessary, you can go as far as to place a single group in each project. There is no limit to the number of groups or projects you can merge.

It is important to note, though, that content will appear in the merged helpset exactly as it does in the output from the individual projects. Specifically, the main drawback of this strategy is that **cross-references between groups that have been broken into separate projects will not be resolved in the merged output**. For example, if a document in Group A of Project 1 links to a document in Group C of Project 1, the link will be preserved. If that document references a file in Group F of Project 2, that link will *not* be resolved in the output.

Collect the Output

When the output for all of the projects has been generated, you will need to copy all of the group-level folders into a single empty directory. In each project folder, you typically have an Output folder, a target-level folder within that, and then a top-level index file (if there are multiple groups) alongside a separate folder for each of the project's groups. You want to gather each of these group folders and place them all together in a common directory of your choosing (as if they had all been generated from the same project).

Next, copy the top-level `index.html` file and `wwhelp` directory from the Output folder of one of the projects (it doesn't matter which).

Configure the Merge Files

In the `wwhdata` folder of any group's output, you'll find a file called *info.txt*. In most cases, the file is empty. But, you can use this file to configure the order and structure of your final merged table of contents—much as you would with the GUI's Merge Settings. These settings are optional, but an `info.txt` file is required for each group to be merged properly.

The first two lines of the text file are used for this purpose. The first line may contain a number representing its relative position among all of the merged groups. For example, if the i`nfo.txt` file for Group A shows the number 20, and the number for Group B is 15, then the topics and TOC books for Group B will be positioned before those of Group A in the merged helpset.

The second line can be used optionally to define additional parent books in the TOC. By default, a top-level book will be created in the TOC representing each group folder found by the script. By providing a path in line 2 of `info.txt`, you can add books to the TOC that will contain the active group's book. That is, if you add "New/Child" to the `info.txt` file of Group B, then that group's main TOC book will be nested within a book called "Child," which in turn will be shown below a top-level book called "New."

Using Markers

After generating and collecting the output for each of your groups, you may configure the `info.txt` sort order setting manually for each group. However, this can become tedious if you have a lot of groups (which is likely the case if you are experiencing memory issues). Each time you generate output, though, a new, blank info.txt file will be created, and you don't want to have to modify the files each time you update your content.

In ePublisher 9.3, we added two new marker behavior settings to make this process easier. You can now use custom markers in your input documents to specify the sort order and group hierarchy. To use this feature, just add one instance of each marker (or at least one with a sort order integer) to each group. If Group A contains 2 FrameMaker books representing 16 documents, you need just one sort order marker in one of those documents to define the `info.txt` contents for that group. If Group A is to be nested within new books in the output TOC, then you'll need one additional marker in one of the group's documents to record that setting as well.

Like most of the markers used by ePublisher, you can give the markers any name you like. Just be sure to add the marker name to the Master Project's Style Designer, and choose the appropriate Marker Type value from the list provided. For example, you might create a marker called SortOrder. In context, the SortOrder marker for Group A might contain the number 20. On the Style Designer's Marker Styles tab, you will scan your document or manually add the SortOrder marker, and then choose "*WebWorks Help Merge Order*" from the Marker Type value list. You might also add a new marker called "MergeFolders" to represent any top-level parent folders under which to nest some of your merged groups. For Group A, the marker text might be "New/Child." If you add the MergeFolders marker to the Style Designer and specify the "*WebWorks Help Merge Grouping*" type, then the `info.txt` file for Group A will automatically be populated with a "20" on the first line, and "New/Child" on the second line.

Execute the Script

With all of the group output folders collected in a common directory alongside the `index.html` file and `wwhhelp` folder from one of them, and an `info.txt` file configured for each of the groups, you are ready to merge. You'll want to execute the script from the Windows Command Line Interface. Click Start, choose Run…, type "cmd" and click OK to open the CLI. From the command line, navigate to (or just type) the path to your saved copy of *wwh5merge.vbs*. You will execute the script with a single argument: the path to the directory containing your collection of group output folders. The script will scan the directory for constituent groups and their `info.txt` files. It will then reconfigure the files in the common `wwhelp` directory so that the top-level `index.html` file will launch a combined WebWorks Help 5.0 helpset representing all of the individual groups, complete with a merged TOC and Index. A message will appear each time a folder is found and added to the merged helpset. The process shouldn't take much time at all, since no new files are created, and none of the individual HTML topic files needs to be processed. When the script completes its actions, the

directory will contain everything you need to deliver a complete merged helpset with all of the output from your separate projects.

Automate the Merge

Once you understand the basics of the Dynamic Merge behavior, you can merge more efficiently by using AutoMap to generate output, and by configuring a batch file to collect the output files and execute the merge script for you.

The Dynamic Merge example here includes a batch file (*run.bat*) representing one option for this kind of automation. It contains a list of commands that use AutoMap to generate output for each of the small Express projects. The output for each one is deployed to a common "*output*" folder. The `index.html` file and `wwhelp` folder from the first output are copied to the root level of the `output` folder. Then, the merge script is executed. It scans the collective `output` folder, finds the individual groups with their marker-configured `info.txt` files, and merges everything into a single cohesive helpset, accessible via the top-level `index.html` file. When the batch is complete, the contents of the new `output` folder are ready for distribution as a merged WebWorks Help 5.0 helpset.

Conclusion

The memory allocation error is a problem that is high on the priority list of issues to be addressed in a future ePublisher release. The Dynamic Merge add-on for WebWorks Help 5.0 represents a stop-gap solution for users who must generate WWH5 output for very large projects prior to the public availability of that fix. The loss of inter-group cross-references is acknowledged as a significant disadvantage, and the script is only available for WebWorks Help 5.0 projects; however, if you need to generate output for a large project that won't work otherwise, this workaround will allow you to move forward with your project ahead of your deadlines.