A METHOD TO GENERATE USEFUL DESIRED CONTENT IN HYPERTEXTBOOKS

by

Steven Carl Aldrich

A project submitted in partial fulfillment
of the requirements for the degree

of

Master of Science

in

Computer Science

MONTANA STATE UNIVERSITY
Bozeman, Montana

November 2008

TABLE OF CONTENTS

STATEMENT OF PERMISSION TO USE

ACKNOWLEDGEMENTS

I sincerely thank Dr. Rockford Ross, Ph.D., for his help throughout this project.

ABSTRACT

Web-based hypertextbooks require accurate navigational tools to be useful. In this project a Perl program was developed to insert menus and additional necessary and useful content into a hypertextbook website. An additional program was created to number objects within the hypertextbook, and create tables of references to those objects. This paper describes the program structure, methodology, and use.

INTRODUCTION

The purpose of the project was to enhance the *Webworks* hypertextbooks created by professor Rockford Ross. *Webworks* hypertextbooks are intended to be online web-based textbooks that completely replace paper textbooks. Professor Ross created two online html-based textbooks. These textbooks addressed the subjects of biofilms[1] and finite state machines.[2] Both books were used extensively by students and their usefulness was assessed.[3] The need to create menus by which users could navigate the textbooks immediately became apparent. Thus, it was decided that a computer program to generate the menus was desirable. The project described herein meets that need. The scope of the project evolved to include additional features and enhancements, as well.

Project Overview

This project is important to the success of the hypertextbook project because the menu-generating program makes the hypertextbooks useable. The hypertextbooks would be almost useless without a method of navigating through the content. The menus produced by the project Perl program solve that problem.

Creating such menus manually would require many hours of tedious, error prone effort. If, after menus were created manually, so much as one page were added to the hypertextbook,

[1] [1] Cunningham, Alfred, Lennox, John E., and Ross, Rockford J. *Biofilms: The Hypertextbook* (most recent version). http://www.biofilmbook.com.

[2] [2] Ross, Rockford J. *Theory of Computing: The Hypertextbook* (rough prototype) http://www.cs.montana.edu/ross/theory.

[3] [3] Cogliati, Joshua J., Goosey, Frances W., Grinder, Michael T., Pascoe, Bradley A., and Ross, Rockford J. Realizing the Promise of Visualization in the Theory of Computing. *Journal of Educational Resources in Computing*, Volume 5, Issue 2, June 2005.

the menus would become obsolete and have to be re-created. This program solves that problem. Whenever the website content changes, a run of the menu-making program brings the site up to date.

The programming objective was to create a general purpose method to create menus for any hypertextbook, regardless of content or subject matter. After this was accomplished it was decided that the capability of numbering figures throughout the hypertextbook site was also desirable. A table of figures was also desired. After implementation, this numbering feature was expanded to create sequential numbers, and an index page, for any unique content type (video, exercises, interactive displays etc.).

The website upon which the textbook is based must follow the guidelines explained in the **Website Architecture** section, below**.**

Project Details

The Perl programming language was selected because of its powerful text processing features. [4] Developing this program moved the author from a Perl tyro to a highly qualified Perl expert.

The program traverses the directory tree of the hypertextbook website, creating a file of XML that shows the location of every directory and page of HTML in the site. This step was accomplished using the XML:Std library.[5] Familiarity with the Perl CPAN and the installation of Perl modules were developed as a result of using the XML:Std library. All told, three different Perl modules from CPAN were utilized: Getopt::Std, XML::LibXML, and XML::Directory::String.

[4] *Programming Perl*, Larry Wall O'Reilly, ISBN 0-596-00027-8
[5] http://www.cpan.org

PROGRAM METHODOLOGY

Each individual page is searched for its title, and the page title is added to the XML. The page titles that are listed in this XML file are later displayed in the menus.

The XML is used to create a data structure that effectively describes the entire web site. This data structure is implemented via a useful “WebSite” object. The WebSite object is used to easily navigate through the entire website, accessing every file’s path, title, parents and peers.

Pedagogical benefits were realized during the development of the program, as an exhaustive set of Perl data types were used, including arrays, hashes, arrays of hashes, and references to each data type.[6] The final object is a useful tool that recursively builds a dynamic data structure that represents the entire website. This data structure can be used to systematically visit and modify every page in a website, adding links to a page’s neighbors, parents and children. The existence of this object in the toolbox greatly simplified programming the figure-numbering module.

[6] *Intermediate Perl*. Randal L. Schwartz et al. O’Reilly ISBN 10: 0-596-10206-2

WEBSITE OBJECTS

Xmlrootref	Reference to the XML object created by reading the XML representing all the directories and files in the website
UnixRootDir	“/home/username/www/biofilmbook”
WebRootDir	“/biofilmbook”
SiteRootDirName	“biofilmbook”
Directory array	Dynamically built path of directories from root to current element. (Full Unix path name to file)
Web directory array	Same as the Directory array, except the path is relative to ‘www’ directory.
Tree data hash	Hash of references to various parts of the tree
Page array	Array of fully qualified path names of every page in the website. Used to process every page in the file for figure element numbering.
‘root element’	Tree Element that holds the Unix and web paths to the hypertextbook website files.
‘homepage’	Reference to a tree element that represents only the ‘homepage.html’ file. Parent element is ‘root element’.
‘contents’	Reference to an array of tree elements that represents the elements below the ‘contents’ directory of the website. (directories ‘help,’ ‘appendices,’ ‘chapters.’) Any file or directory can be reached from this element. This element’s parent is ‘root element’. Each subordinate element holds a references to its parent, and a reference to an array of its subordinate elements.

Table 1 List of Website Objects

Tree elements for the ‘homepage.html’ file and the ‘contents’ directory are built individually. Then, the entire tree is searched recursively to capture all required data from the rest of the website.

TREE ELEMENTS

Type	‘directory’ or ‘page’
Name:	“chapter002”
Unix path:	/home/prof_jones/www/biofilmbook/contents/help/help001/page003.html
Webpath:	/biofilmbook/contents/help/help001/page003.html
Title:	“Using a biofilm wisely”
Filename:	Page003.html
Patent:	Reference to the parent tree element

Table 2 List of Tree Elements

Utility functions were created, which were used to add many kinds of useful text to the HTML files. Marker text was inserted in the HTML files that indicated where the menus and other file modifications were to be inserted.

For example, when links to the previous and next pages in a specific directory were desired, the string “<!-- @previous - ->” was added to the file in which the links were to appear. The program found the incidences of this text in each page. The WebSite object was used to determine what page was previous to the current page. Thus, a link to the previous page could be inserted into the current page. This process was repeated for every page in the site.

Another standard marker, “<!-- @endPrevious - - >”, shows where the text should be left untouched.

When a link is inserted into a page, any previously existing text between the *begin* and *end* markers is removed. Thus, a new page can be added anywhere in the site, and when the program is re-run, all links will be correctly updated, and old invalid links are removed. Text outside of the *begin* and *end* markers is unmodified.

Similarly, markers are used to indicate where to insert the menu text into the HTML. The "<!-- @menu - ->" marker also contains parameters that tell the program which menus to insert. Thus, the author of a page of HTML can determine which menus should appear on that page. This is an outstanding feature of the program, as hypertextbook authors may choose to insert a menu to help files, appendices, student exercises, other chapters, or other sections, in any page. Other pages may hold a completely different set of menus.

USERS GUIDE

How to Use the Menu-Maker and Figure-Numbering Programs

Website Architecture: The Menu-Maker Program

The website to be used as a hypertextbook must contain a directory named "contents."

All the website's pages must be subordinate to this directory. Files and directories that are not below the "contents" directory are ignored by the menu-maker and figure-numbering programs.

Directories named 'help,' 'appendices,' and 'chapters' must be below the contents directory.

Subordinate to the help and appendices directories are any number of subdirectories named help001, help002 … or appendix001, appendix002, appendix003… .

Below the individual help and appendix directories there must a directory of pages. The best name for these directories is 'pages.'

In any directory of pages the pages must be named page001.html, page002.html page003.html etc. This rule also applies to help, appendix and contents pages.

Individual pages of HTML within the site must be created with the *Webworks* templates, as designed by Professor Ross. For individual viewable pages, these templates will contain marker text such as:

<!--@previous-->

<!--@endPrevious -->

<!--@next-->

<!--@endNext-->

and

<!--@menus contents chapters sections pages appendices help -->

<!--@endMenus-->

The menu-maker program searches for these tags, and inserts appropriate text between them. Pages that do not contain these markers are affected by the menu-maker program.

A file named “homepage.html” must be at the same level as the contents directory (the root directory of the website). The menu-maker program will search for this file and insert a menu with a link to page ‘contents/contents/html.’ Thus, ‘homepage.html’ must contain this text:

!--@menus contents help –

A file named “webroot/contents/contents.html” must also exist in the site.

File ‘contents.html’ must contain this text:

!--@menus chapters appendices help –

The directory named ‘chapters’ lies below ‘contents.’ Within the ‘chapters’ directory are directories for any number of chapters, named ‘chapter001,’ ‘chapter002,’ ‘chapter003’ etc. Within each individual chapter menu there must be a file named ‘chapter.html.’ This file name is the same in each chapter directory. ‘chapter001’ holds file ‘chapter.html’ and chapter002 directory hold file ‘chapter.html’ as well.

Each ‘chapter.html’ file must contain a <title> tag. The program will search each of these files, and find the title tag. The title will be added to the XML representation of the website, and then that title will be stored in the ‘WebSite’ object. Chapter.html files contain this text:

```
!--@menus contents chapters sections sections appendices help –
```

Below the chapter directories are numbered section directories, named similarly to the chapter directory. Thus, a site may contain any number of section directories named section001, section002, section003 etc.

Each section directory must contain a file named ‘section.html.’ This same file name is used in every section directory.

Below the sections are directories named “black,” “blue,” and “green.” These directories represent the “difficulty level” of the content in that section. The difficulty levels correlate with ski slope difficulty levels, such that black is the most difficult, green is easiest, and blue represents material somewhere between green and black.

Within the color/difficulty level directories are any number of individual pages, named page001.html, page002.html, page003.html etc.

Pages must be created using the templates as designed by Professor Ross.

Individual pages will contain this text:

```
<!--@menus contents chapters sections pages appendices help –>
```

as well as the @previous, @next, and other text content markers.

The menu-maker finds the @menu marker, reads the remainder of the text, thus determining the desired menus for this page, and those menus are inserted, with contextually correct content. For example, if the page is in Chapter 2, Section 3, Black, then the page menu will display the other pages in the Black level of Section 3, Chapter 2. The Section menu will display all the sections within Chapter 2. The Chapter menu will display all the chapters within the hypertextbook.

Figure Re-Numbering

The figure numbering utility searches the website chapter by chapter, and section by section within each chapter. Within each section, each page file is searched for this type of div:

```
<div class=imageCaption>
```

When this text is found, text corresponding to this pattern:

Figure 1 – 1

is inserted into the imageCaption div. The first number represents the chapter number, and the second number is the figure number within that chapter.
Any other text within the div is captured and re-inserted into the page.

Captions for objects that will be numbered must be inserted according to the example below:

```
<div class= “imageCaption”>You may insert text here
additional text on a separate line is allowed
use as many lines as you want
caption text may go here too</div> <!—end image caption div -->
```

While the text between the <div> and </div> tags may be on as many lines as desired, the text will not appear on separate lines on the html page unless </br> tags are inserted in the caption text.

The text will appear on separate lines within the HTML file itself, that is to say, the separate lines will be retained in the “source code” HTML file.

Any desired type of object may be numbered, such as videos, tables, etc.

To number an object in the hyptertextbook pages, pass the object name to the figure-numbering utility with the “–o” switch.

Example:

```
Perl figures –o image,movie,exercise (…other program parameters)
```

In the example case, there would be any number of divs within the pages named thus:

```
<div class=”imageCaption”>  … </div>
<div class=”movieCaption”>  … </div>
<div class=”exerciseCaption”>  … </div>
```

Tables for each type, named “imagetable.html,” “movietable.html,” and “exercisetable.html” will be generated and placed in the root directory of the website.

If the figure numbering program is re-run against the same website, the ‘Figure M – N’ text is replaced. Any existing numbering text is always replaced with the correct number in the sequence; thus an author can insert a figure, (image, movie, etc) anywhere on any page, and then simply re-run the figure numbering program. New, correct numbers, and the corresponding “objectTypetable.html” file, will be generated.

<u>Using the Final Product</u>

To run the menu-maker:

1.) You must have Perl installed on the machine where the website is located.

2.) The website must be created using *Webworks* templates, and the necessary files must exist, and be located in the correct directory structure. (See the **Website Architecture** section, above.)

3.) To run the menu-maker, the menu-maker driver program must be in the executable path, and files Menus.pl, ErrorMsg.pl, MakeXML.pl, WebSite.pl, BuildPath.pl, Files.pl and TreeElement.pl must be in Perl’s search path. The easiest way to do this is to simply keep all these files together in the same directory

4.) Use command:

```
perl driver -d /home/projects/webworks/www/projects/biofilmbook
-w /webworks/projects/biofilmbook -f test.xml
```

To Run the Program

The –d switch indicates the Unix directory where the site is located.

The –w switch indicates the root of the website, relative to the 'www' environment.

The –f switch names the XML file that will be produced by the program. If the –f switch is left out, the file will be named 'dirtree.xml.' This file will be created in the Unix directory where the site is located (as indicated with the –d option).

To Run the Figure-Numbering Program

1.) Steps 1 through 3 above are the same when running the figure number generator, except that you must also have files named figures, Figures.pl and FigureElement.pl in same directory with ErrorMsg.pl, MakeXML.pl, WebSite.pl, BuildPath.pl, Files.pl and TreeElement.pl.

2.) The –d, -w and –f command line arguments are the same as for the menu-maker:

–d is the Unix path to the website.

-w is the www-based webpath.

–f is the file name of the XML file created by the program.

The renumbering utility requires an additional parameter:

-o is the list of object to be numbered, comma separated without space between the entries.

Example: -o image,video,exercise

Both programs generate a new copy of the XML file every time that they run. Thus, either program may be independently, at any time. The XML file will have the name that was passed with the –f option. The default XML file name is ‘dirtree.xml’.

If a file name of an existing XML file is reused (with or without –f option), after a previous run of either program, that XML file will be overwritten.

The XML file may be examined or edited for any reason. Program users may retain or delete the file for any purpose. Modifying the XML file will have no effect on the menu-maker or figure numbering programs, as a new XML file for the current state of the website is always generated when either program is run.

CONCLUSION

The menu maker program and object numbering utilities constitute two useful utilities that will advance the hypertextbook concept and implementation.

LIST OF TABLES

REFERENCES

Cunningham, Alfred, and Ross, Rockford J. *Biofilms: The Hypertextbook* (Version 1).

Cunningham, Alfred, Lennox, John E., and Ross, Rockford J. *Biofilms: The Hypertextbook* (most recent version). http://www.biofilmbook.com.

Cunningham, Alfred, and Ross, Rockford J. *Biofilms: The Hypertextbook* (Version 1).

Ross, Rockford J. *Theory of Computing: The Hypertextbook* (rough prototype)

http://www.cs.montana.edu/ross/theory.

Ross, Rockford J. *Hypertextbooks and a Hypertextbook Authoring Environment. Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education*, 2008. Madrid, Spain. Pages 133-137.