Project #3 - Dependency-Based CodePublisher

Version 2.0, Revised: 03/21/2017 12:50:24
Due Date: Tuesday April 4th

Purpose:

A Code Repository is a Program responsible for managing source code resources, e.g., files and documents. A fully developed Repository will support file persistance, managment of versions, and the acquisition and publication of source and document files. This project focuses on just the publishing functionality of a repository.

In this project we will develop means to display source code files as web pages with embedded child links. Each link refers to a code file that the displayed code file depends on.

There are several things you need to know in order to complete this project:

- Each file to be published is a C++ source file. Our publisher will generate, for each of these, an HTML file, with most of the contents drawn from the code file.
- The pages we will generate have only static content, with the exception of some embedded JavaScript and styling, so we won't need a web server.
- We will need to preserve the white space structure of the displayed source code. That can be done embedding all the code between the tags <pre> and </pre> or by using the CSS white-space property with value "pre" to style a div with all the code in its contents.
- Any markup characters in the code text will have to be escaped, e.g., replace < with < and > with >.
- File dependencies are displayed in the web page with embedded links, which are implemented in HTML5 with anchor elements:

  source code file name
- For each class, we will, optionally, implement outlining, similar to the visual studio outlining feature. To do that we will use the CSS display property, with values: normal or none, to control whether the contents of a div are visible or not.

The Code Publisher will be embedded in a mock Repository with almost no functionality except to support publishing of source code as web pages. Specifically you are **not** expected to provide support for:

- package checkin or checkout
- versioning

You **are** expected to support:

- Dependency analysis of the C++ source code files you will publish, using the analyzer you developed in Project #2.
- The ability to specify, on the command line, files to be published, by providing command line arguments for path and file patterns.
- The ability to display any file cited on the command line, by starting a process that runs a browser of your choice, naming the specification of the file you want to display.

Note that the CodePublisher project creates a code generator. Its inputs are C++ code and its outputs are HTML code.

Requirements:

Your CodePublisher Project:

1. (**1**) **Shall** use Visual Studio 2015 and its C++ Windows console projects, as provided in the ECS computer labs.

2. (**2**) **Shall** use the C++ standard library's streams for all console I/O and new and delete for all heap-based memory management[1].

3. (**4**) **Shall** provide a Publisher program that provides for creation of web pages each of which captures the content of a single C++ source code file, e.g., *.h or *.cpp.

4. (**10**) **Shall**, optionally[2] provide the facility to expand or collapse class bodies, methods, and global functions using JavaScript and CSS properties.

5. (**2**) **Shall** provide a CSS style sheet that the Publisher uses to style its generated pages and (if you are implementing the previous optional requirement) a JavaScript file that provides functionality to hide and unhide sections of code for outlining, using mouse clicks.

6. (**2**) **Shall** embed in each web page's <head> section links to the style sheet and JavaScript file.

7. (**4**) **Shall** embedd HTML5 links to dependent files with a label, at the top of the web page. Publisher **shall** use functionality from your Project #2 to discover package dependencies within the published set of source files.

8. (**2**) **Shall** develop command line processing to define the files to publish by specifying path and file patterns.

T N P B

9. (**3**) **Shall** demonstrate the CodePublisher functionality by publishing all the important packages in your Project #3.
10. (**5**) **Shall** include an automated unit test suite that demonstrates you meet all the requirements of this project[2].

---

1. That means that you are not allowed to use any of the C language I/0, e.g., printf, scanf, etc, nor the C memory management, e.g., calloc, malloc, or free.
2. This optional requirement will take a significant amount of work to complete successfully. You should get everything else working before attempting this additional effort.
3. This is in addition to the construction tests you include as part of every package you submit.

What you need to know:

In order to successfully meet these requirements you will need to know:

1. Details of the C++ language: http://CppReference.com.
2. HTML5, CSS, and JavaScript basics.
   I will provide a demo that shows you how to do most of the things web things required for this project.
3. All those things you learned while developing code for Projects #1, and #2.

Back Back