Structured Dynamic Content Roundtable

Peter D Bethke

Overview

It is an unavoidable truth of web application development that the more complex a site is, the more difficult it becomes to implement site-wide changes. In the "early" days of web application development, when sites were primarily static html documents, individual pages were coded one at a time, and often pages were added by cloning and altering a previous page. While this was effective in creating a site with only a few pages that seldom changed, it became increasingly inefficient in handling sites where information had to shared between pages, and more importantly changed and updated consistently between pages. Many an html coder has (hopefully in the past) experienced the tedium of adding the same information over and over again to multiple web pages, or relying on "search and replace" operations while praying that the search pattern has remained constant throughout all the pages.

With the advent of "dynamic" web applications, developers began to overcome these issues by taking advantage of two fundemental tags that are common to Lasso and virtually all other scripting languages - variables and included files (most often called "includes"). An included file in LDML is called, using the [include] tag, by another file, and the contents of that file is then essentially "folded" into the body of the first file at the point in the script (proceeding in a linear fashion from top to bottom) where the [include] tag appears. Most importantly, if there are variables on the "included" file that are global in scope (in other words cabable of being recongnized as proper variables in the body of the "calling" page), they too are "folded" into the body of the calling page, and can be used on that page for a number of purposes. The simplest of these is of course declaring and calling simple string variables. For example, if the developer had a file, "A", which included the variable "foo" with value "bar", and this file was called via the [include] tag into the body of another file, "B", the developer could call (at any point below the "point of insertion", ie the place where the [include] tag was declared) the "foo" variable and get the value "bar". At any point after that, changing the value of "foo" in file "A" would result in the value instantly changing in file "B".

These very simple concepts lead to the use of included files to provide, for example, "header" and "footer" files for web sites. The "header" file might contain a common logo, and the "footer" might contain contact information. If these files were called as dynamic includes by any number of pages, essentially "shared" by the whole site, it greatly reduced the developer's effort when it came time to change any of the information contained therein. Building further on this concept, it became advantageus to create a "site configuration" file, or "siteconfig" as it is sometimes called, in which common variables were placed. This siteconfig file was then included by all pages on the site, and these pages could call any of the global variables declared in the siteconfig at any time, and similarly those variables could be changed and their values would update instantly across the site.

One important function of the [include] tag is that it allows "nesting" of files. Included files are not limited to one "level" of inclusion -- much like a Russian Doll that has many dolls contained within, multiple files may be "chained" or "nested" together to produce complex structures. Even

more important to the creation of dynamic systems, the global variables contained in included files become available to files that are part of the nested structure (though always in a "top-down" fashion, as discussed earlier).

The combination of the "nesting" function of included files and the fact that the file path specified by the [include] tag can itself be a variable, enabled the creation of Structural Methodologies such as the Corral Method. Corral had the distinction of being the first Lasso-based structural method proposed, but it was by no means perfect. Rather, it served the purpose of "opening a dialogue" on the roll of structural methods at a time when a number of Lasso developers were pursuing similar ideas. It was eclipsed soon after by systems like [Framework:] (now called "Pageblocks") and other much more robust systems ported from other languages (like Fusefox). A quick distinction should also be made - Corral was a Structural Method, ie a set of ideas that produced a result, and open to broad interpretation and reinvention. This stands in contrast to a Structural System, sometimes called an API, which is much more devoted to defining a specific system and set of guidelines to follow. Pageblocks is a terrific example of a very mature Structural System, as is Fusebox. Much more effort has gone into these systems to provide developers with specific tools to enable rapid application development.

In the end, however, neither Corral or Pageblocks or Fusebox or any other of the multitude of Structural Methods and Systems could function without the [include] or [variable] tag (or their equivalents in other languages) and the way that they nest together to enable complex and dynamic systems.