Category: ARTICLE
Subject: **"Web Apps" 101 - Part 3**
Name: Creces, Gus

"Web Apps" 101 - Part 3

Server-Side Scripting With PHP

PHP is an acronym standing for "Personal Home Pages". Since that name was originally applied, PHP has come a long way from being strictly for "Personal" use. Most of the things said about ASP in *Part 2* of this series of articles would apply to PHP. While ASP can be considered the gold standard for browser-based web application development in the IIS (Microsoft *I*nternet *I*nformation Server) world, PHP is easily the gold standard for browser-based web applications in the Linux/Apache world. PHP versions, nonetheless, are also available for use with MS Windows IIS.

PHP was designed to run as a plug-in for existing Web server software such as IIS, Apache, Sambar or OmniHTTPD. To test web pages with PHP, you need to equip your own computer with Web Server Software, so that PHP has something to plug into. As an interesting side note, in advance of extensive coverage of this topic in later parts of this series or articles, *CHT Web Servers do not support PHP or ASP*.

Why? Because you don't really need to use yet another a server-side scripting language in an open-server environment like CHT where all the work required of *any or all* of these server-side scripting systems can be handled in the Clarion language alone. But we are getting ahead of ourselves.

Just like ASP, PHP is also a *server-side scripting language*. Unlike ASP which is proprietary to Microsoft, PHP is open-source and readily available in binary or source code format. When a visitor to your website types a URL in the Address box or clicks a link to your website on any Web page, they're asking a Web server on a computer somewhere to send a file to the client Web browser on your computer. If that file is a normal HTML file, it looks exactly the same when your Web browser receives it as it did before the Web server sent it. That's why it's called a static page. After receiving the file, your Web browser renders its contents as a combination of text, images and sounds.

In the case of a page containing PHP scripts, the process is similar, except there's an extra processing step that takes place just before the Web server sends the file. Before the Web server sends the PHP page to the Web browser, it executes all *server-side scripts* embedded in the page. Some of these scripts might be made to display the current date, time, or to perform data table access and wrap resulting information in HTML and Javascript. Others scripts may process and store information the user has just typed into a form. To distinguish them from normal HTML pages, PHP pages are given the ".PHP" extension.

It's important to note too that to use PHP, as with ASP you really need to get intimately familiar with the language to get anything done. Unlike ASP, however, the PHP language is far from elegant. While it's *very* functional and does everything intended, it's not going to win any modern design awards. It resembles to some extent the older BASICs. Commodore 64 BASIC comes to mind, perhaps only because I spent a lot of time in the early 80's with that language. If you've ever hacked your way through a Clarion template-language file, initially reading PHP scripts stirs the same kind of vague discomfort until you get used to the look of it. Computer language purists, keep away!

That having been said, there's an interesting side-use for PHP writing desktop applications. PHP is probably not the very best language to create a desktop application with a graphical user interface, but for developers who know PHP well, and would like to use some advanced PHP features in their client-side applications, they can also use PHP-GTK to write them. Programs written this way are cross-platform, since there are PHP-GTK compilers for both the Windows and Linux operating systems. PHP-GTK is an extension to PHP, not available in the main distribution.

Clarion PHP Templates

Clarion's PHP templates let you design a generic browse-form application inside the familiar Clarion IDE and use your application Dictionary to help overcome some of the problems of unfamiliarity with PHP and those other requisite browser-interface-building tools: HTML, Javascript and CSS (Cascading Style Sheets). It's important to note here the word "generic". The starting point for a web browse or web form produced by these templates is always a *generic* HTML page and a *generic* PHP script, which the developer is free to change via ordinary embedding inside embed points provided by the Clarion PHP templates. Of course, what the developer is embedding now isn't Clarion code. It's HTML, Javascript, CSS and PHP. If the developer hasn't done the personal professional development required to learn these things they're pretty much stuck with *generic*. If they discover bugs, design limitations or even simple design irritants in any of the generic stuff, they're dead in the water until they learn HTML, Javascript, CSS and PHP. As with generated Clarion programs, Clarion's PHP templates can customize your PHP server-side scripts to accommodate many variations of style and function while still keeping within the familiar Clarion *browse-form paradigm*.

Generated PHP pages, are placed in a special directory on the server machine to be sent to the client browser when someone types the PHP page name into their browser's address area, or links to the PHP page from another web page.

As with ASP pages, most ISP's are happy to allow them on their web servers. PHP is ubiquitous, especially in the Apache server world. If you're going to go the server-side scripting route to build web applications and you don't want to limit your server options to IIS installations, PHP is the server-side scripting tool of choice. Of course, if you have a set of company data tables like your order processing system sitting at company headquarters, you're either going to have to replicate those data tables to the ISP, **or** give the ISP's hardware remote access to your data tables via a separate TCP/IP connection so that it can pick up raw data from your SQL server. Unless you or your customer are prepared to allow this you're better off hosting the web pages yourself via a server running locally to the data tables. Many PHP developers make MySQL their SQL tool of choice, since it too evolved in the open-source world where PHP had it beginnings.

PHP scripted web applications, whether built by hand from scratch, or built using Clarion's PHP templates to assist, follow the two-piece principle outlined at the beginning of this series of articles. The **client piece** consists of a set of web pages containing HTML, Javascript and CSS code from which the user interface is rendered inside the browser. The **server piece** is provided by the Apache or IIS server which takes care of TCP/IP connectivity and sending pages to the browser as well as executing the embedded PHP, via the PHP plugin server extension. This inserts the interactive data from your data tables into the HTML pages. Be aware that the embedded PHP scripts *never make it to the client browser* which wouldn't know what to do with them if they did. All non-HTML, non-Javascript, non-CSS elements are stripped out and data elements from your tables are hard-coded into your pages in their place. That's the primary purpose of PHP server-side scripts in PHP based web applications, to access your data tables and package the return data in a way that can be rendered in the browser. Data elements are tagged, wrapped and hard-coded into your web pages before they're sent to the visiting browser.

What really *bites* for a lot of Clarion developers who take the plunge with PHP - *or ASP for that matter* - is that they come to realize that with the server-side scripting approach to web apps, the developer ultimately ends up having to write data table access code using the PHP language. If they're already struggling with SQL syntax, and getting familiar with HTML, Javascript and CSS, adding PHP data access code is the final straw.

There is a better way of course. CHT servers eliminate server-side scripting altogether. They let you use Clarion code and templates to write the data table access layer that provides two-way data packaging and storage for the web browser client. You don't even have to use SQL tables unless you want to. With Clarion as the data table access tool, you can use your favorite back end. But again, we're getting ahead of ourselves.

One good thing about the PHP approach to web app building - as opposed to using Clarion's Web Builder - is the fact that the IIS or Apache server can handle multiple client browser connections all without starting your "server" app multiple times the way that Web Builder does. As with ASP there is no separate "server" app in the background except IIS or Apache. With Web Builder your template-doctored application generates the web page ultimately being sent to the browser. In a PHP setting those same pages are generated from generic, pre-built HTML pages containing server-side scripts which are pre-processed, server-side in order to produce the final pages sent to the browser. Clarion's PHP templates help you create those server-side scripts from your Clarion data dictionary.

If there is a downside to PHP it's in the fact that it does not really separate your data from the client-side code that renders it in the browser. So when a page is requested, as I've already stated, your data elements are hard-coded into the page each time it's requested and the whole page is sent to the browser each time as well. If you compare how that works in light of how a desk top app works its equivalent to the data table sending you screen-building code *each time* an new browse or form is requested. That's not how web pages *have* to work and it's certainly not how data tables and SQL servers work. In desktop apps, data elements are sent to the application and the client application's own screen design determines how the data are displayed. Changing your client-side screen design, to move the location of a column or field does not necessarily have any direct impact on the data packaging design. That's because in desktop applications data componentry and rendering code are separate.

For the most part, PHP and other server-side scripting systems, do not work this way. Because like ASP, PHP is designed on a *mail-merge model* (though far more advanced). Pages built with PHP server-side scripts tend to have page data elements intermingled with code and screen rendering elements. It's the nature of the beast. Keep this fact in mind later in this series of articles when we begin to discuss how CHT servers handle data elements and screen rendering scripts entirely separately.

Clarion PHP Templates Summary

Strengths

1. *You can build a rudimentary web interface without knowing much about browsers or PHP*
2. *You can build your PHP browses and forms in the context of the ever-familiar Clarion Template environment*
3. *Very scaleable since the server is IIS or Apache and not an app that must run separately for each connected client browser*
4. *Good ISP support as long as you're willing to have them also host your data tables and/or connect to them over the internet*
5. *You can build on an existing data dictionary*

Weaknesses

1. *To build sophisticated, fully-under-your control web interfaces you have to learn the very things Clarion PHP templates are is supposed to help you avoid having to learn (i.e. HTML, Javascript, CSS and PHP)*
2. *Data is intermingled with screen rendering code slowing page turnaround*
3. *Reliance on IIS or Apache forces you to give access to your data tables or you need learn the care and feeding of an IIS or Apache web server*
4. *Pushing your web app beyond the "generic" forces you into learning PHP file access syntax and even into writing PHP script that writes Javascript that writes HTML and CSS*
5. *Can't build forward from an existing Clarion .APP, only the existing .DCT*

Coming up in "Web" Apps 101 Part 4...
Building Web Clients With ClarioNet

To obtain a PDF version of this document, .

Cheers...
Gus Creces
The Clarion Handy Tools Page

Message Posted From CHT Support Forum Client [V2.01]