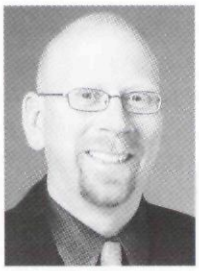
Learning Server-Side Scripting

"AS ALWAYS IN SMALL LIBRARIES, YOUR EFFORTS TO LEARN TECHNIQUES WILL YIELD KNOWLEDGE FOR YOU AND GREATER SERVICE POSSIBILITIES FOR YOUR PATRONS."

by GARY ROBERTS



My colleague has a placard over her desk that reads, "Oh no, not another learning experience." It's a great sign because it really captures the frustration of having to keep up with the incessant pace of library technology. Even the most ambitious librarian can grow dizzy from the centrifugal force of the library technology learning curve. It is impossible to learn every new technology. I have recently come to the conclusion that, as a worker at a small library, I must be strategic and learn only those technologies that will yield the greatest benefit to my facility and patrons.

Of all the technologies that I have learned during the last 10 years, the one that has been the most beneficial to my patrons and the most empowering for me has been server-side Web scripting. If you are a regular reader of *CIL*, you know what I'm talking about. Look at past *CIL* tables of contents: It is clear that librarians are implementing and writing about online knowledgebases, intranet/content management systems, and customizable library portals. And at the core of every one of these systems is some type of server-side scripting technology.

And yet, not every library has been able to utilize server-side scripting. Small libraries, in particular, have been hard-pressed to implement this useful (dare I say "essential") technology. What is it about server-side scripting that makes it so difficult to implement? It isn't the hardware or software that makes it so inaccessible—after all, you probably already have a Web server that could be configured to run server-side scripts with a click or two on a mouse. Yet once that server is enabled, a language barrier remains. (Great—an opportunity for yet another learning experience!) In this column, I'll compare and contrast the most popular scripting languages that are used to create truly dynamic service-oriented Web sites, building a conceptual framework that you can use as a starting point for your specific server-side library project.

To fully utilize server-side scripting, you must have two components: a script-enabled server and virtual or physical access to it. Nearly all Web servers, including Apache and Microsoft's IIS (Internet Information Server), come with a pre-installed server-side scripting engine. Enabling these services is usually a trivial task that requires just a few mouse clicks. I'll ignore the politics of getting access to server-side technologies except to say that, in some organizations, IT managers have been able to thwart server access in the name of security. These managers are partially correct when they underscore the importance of limiting access to high-profile computing resources. However, taken to its ultimate conclusion, this level of caution would have us all unplug our computers from the network and bunk in our offices. As an enlightened, service-oriented information professional, your goal is to impress upon your IT department that security must be balanced against

»

the productivity and service concerns of access and information sharing. Furthermore, there are successful IT management strategies that limit the risk associated with server access and still enable librarians and staff to fully utilize the functionality of server-side computing. Assuming that you have access to a script-enabled server, let's look at the models.

LAMP Lights the Way

Because Linux is such a magnificent example of computing efficiency, the LAMP (Linux, Apache, MySQL, and Perl/PHP) model allows smaller libraries to turn derelict hardware into working servers. Try this as an example: Take a standard 3-year-old PC, max out the RAM, and install Linux. You just scored yourself a top-shelf server.

TAKE A STANDARD 3-YEAR-OLD PC, MAX OUT THE RAM, AND INSTALL LINUX. YOU JUST SCORED YOURSELF A TOP-SHELF SERVER.

Another big advantage of Linux is the passionate and growing open source community, which is very willing to share the source code to some impressive library-specific applications. Now you have a powerful server platform that runs on old hardware and the growing body of library-specific software—you really couldn't have asked for a better model. The LAMP model does come with a cost, though. If you read my May 2005 column, "Linux Adventures on a Laptop," you know that the computing power of Linux requires a significant investment in knowledge acquisition. The same can be said about Perl and PHP, the standard Web-scripting technologies in the LAMP model. Built by programmers and loosely based on the structures found in more complex languages, Perl and PHP are not intuitive for newbies. Even to this day (despite a significant background in Web scripting), I still have a little bit of trouble reading and writing Perl and PHP. One thing is for certain: A little bit of extra learning really enables you to take advantage of an impressive server platform.

ASP, ASP.NET, and Windows

If you have a Microsoft IIS, Active Server Pages (ASP) may be the natural place to begin a Web-scripting project. If you are already running IIS, configuring ASP is a simple task. And that is how many libraries start with ASP scripting—the path of least resistance is compelling. But there are other reasons to use ASP (which is a relatively easy-to-learn language). It also interfaces nicely with such products as Microsoft Access and Excel, two familiar technologies. ASP is a mature scripting language (it's nearly 10 years old), and, judging by its popularity, it isn't going anywhere soon.

If you have confused ASP with ASP.NET, you aren't the first to do so. ASP.NET, despite the similar name, is quite different from its mature sibling. ASP.NET was developed to help remedy some of the issues that are associated with ASP. Like many Web scripting languages (such as ColdFusion and PHP), ASP mixes the formatting with programming syntax. If applications were finite and static organisms—programmed once and then never changed—the mixing of syntax and formatting would be a completely acceptable practice. However, the second you finish an application, invariably someone will say, "It would be really cool if your application could do this." Three months later, you will add that functionality. And so on and so forth, until your application resembles a hodgepodge—certainly functional, but not elegant.

Programmers use a somewhat different metaphor. They call this type of ad hoc programming "spaghetti code," since it is difficult to read and almost impossible to amend. By separating formatting (HTML) from programming syntax, ASP.NET eliminates much of the potential for spaghetti code. ASP .NET also differentiates itself from ASP in that ASP.NET is somewhat more difficult to learn. So if you could choose either one, which language would you pick? It depends on your ambition. If you just want to throw together some service forms or a couple of small database applications, you may need to learn ASP. However, if you are planning a sizable project (or if you anticipate that your coding project may need modifications and additions in the future), it may well be worth some extra time to learn ASP.NET.

Coding with ColdFusion

My personal favorite scripting language at this moment is ColdFusion, an easy-to-learn, tag-based language that is not unlike HTML itself. ColdFusion enables the developer to create fairly sophisticated applications without learning a lot of esoteric programming structures. Here at Herrick Library, we have chosen ColdFusion (switching from ASP) because it allows us to utilize the coding skills of relatively novice student Web developers. One advantage, as an academic institution, is that we have a sizable population of computer-literate students. In a relatively short period of time, we are able to train these talented students to create sophisticated ColdFusion Web applications.

Unfortunately, this easy-to-use scripting language does come at a price. ColdFusion is sold as a server add-on to IIS and Apache and runs on Linux, UNIX, and Windows platforms. This server add-on sells for nearly $1,000 (with an educational discount). Fortunately,

we have found that we can recoup that initial investment in a short period of time by utilizing inexpensive student labor. Furthermore, this student training supports the educational mission of Alfred University by providing a worthwhile extracurricular learning experience.

JUDGING BY ITS POPULARITY, ASP ISN'T GOING ANYWHERE SOON.

You may be thinking that it is disingenuous to talk about a $1,000 product in the context of a "small libraries" column that is all about doing things inexpensively. However, if you can get past the initial sticker shock, you may see that with ColdFusion, what you expend in initial investment you recoup in labor cost, particularly if you can utilize an enthusiastic (and inexpensive) pool of novice programmers.

Now Where Do You Go?

For many libraries, choosing a server or a particular server technology may not be an option—the decision may have been made long ago by your IT department. The good news is that many server-side scripting technologies run on several types of computing platforms. For example, PHP and Perl, in addition to running on LAMP, also run on Windows machines in the awkwardly named WAMP (Windows, Apache, MySQL, Perl/PHP) model. Of course, in many cases, you will need to have someone from your IT department install the additional components on your server.

This brings me to another point: If you don't like the choices and restrictions that your parent organization or IT department has made, outsource to a Web-hosting company. Since the late 1990s, the Web-hosting industry has really expanded. Today, there are thousands of companies. Even through the dot-com bust, many Web-hosting companies remained both stable and profitable. The difficulty resides in separating the wheat from the chaff in this business. I could easily write an entire column about how to choose one of these companies, but for now I'll just say that if your IT department has denied access to Web scripting, there are many companies that will gladly provide reliable script and database-enabled server space for less than $500 per year.

Now that you know about the most popular scripting technologies and about how to outsource your Web-hosting needs to someone other than your IT department, where do you go next? You could take a course in Web scripting. But don't overlook the obvious and extensive resources that are available through ILL. There are dozens (if not hundreds) of books dedicated to this popular topic. Using Amazon in conjunction with OCLC WorldCat will really show you the extent of available offerings. This method has the advantage of allowing you to educate yourself while reviewing potential books for your collection.

Another useful technique is to bind your learning to a particular product or outcome. Nothing is more motivating than working toward a tangible service or product. I can think of at least two types of projects that could not only provide a gentle introduction to newbies, but could also yield some useful products for your patrons. For example, the novice Web programmer can easily create online service forms. While such forms have become a ubiquitous way of transacting business for larger libraries, many smaller libraries have yet to make this jump to virtual service. A slightly more ambitious project for beginners is to Web-enable a database. In fact, many librarians learn Web scripting for the specific purpose of making a local and unique collection of art, images, historical artifacts, or electronic resources accessible online. In fact, my main reason for learning ASP was to Web-enable a local database of electronic and print serials. (This was before companies such as Serials Solutions and EBSCO began selling similar products.) An informal survey of my colleagues revealed that enabling a database for the Web is perhaps the single most important reason for learning a scripting language. The good news is that this type of project is within the reach of most scripting newbies—if they are willing to stretch a little.

'SPAGHETTI CODE' IS DIFFICULT TO READ AND ALMOST IMPOSSIBLE TO AMEND.

When choosing a server-side scripting technology, your choice will be determined by cost, convenience, and level of cooperation with your IT department. Once the decision has been made, enjoy your ride on the learning curve. As always in small libraries, your efforts to learn techniques will yield knowledge for you and greater service possibilities for your patrons. ■

Gary Roberts is an information systems and reference librarian at Herrick Library, Alfred University in Alfred, N.Y. His responsibilities include overseeing reference, ILL, Web services, and most other technologies in a small library. In 2004, he received the 21st Century New Librarian Award. His e-mail address is roberts@alfred.edu.