Alternatives for Dynamic Web Development Projects

Dennis provides a starting point for developers seeking solutions for their web application development requirements.

by Dennis Roman Gesker

I was recently afforded the opportunity to work with the internet department of a large medical publishing firm located in Philadelphia, Pennsylvania. The department assisted the company's various marketing departments with regard to content destined to be posted on the Internet as well as coordination of scheduling and placement of content on the company's web site. Ultimately the department served as a conduit between various marketing and product management departments at sites in Philadelphia, New York, Chicago and the Baltimore-based IT department that physically housed the company's web servers.

As a result of these activities, there were a great many files finding their way into the department via a number of paths. Most arrived by e-mail, some on diskette via interoffice mail and some were uploaded directly to ftp servers in Baltimore. This created a flurry of activity as the marketing staff and developers in the department attempted to organize and coordinate a variety of content. To add to the confusion, there were also numerous versions of documents that were submitted due to last-minute changes made by the supported departments. My task was to develop an interim solution to reduce the confusion created by content submission activity by organizing all the various content in one central application. I attempted to do this by building a small web application prototype that would serve to track each project's title, manager, department, department code and cost accounting code.

The application needed to be built quickly and cheaply because constraints in the purchasing and tenuous acquisition processes ruled out any new requisitions until the beginning of the next fiscal quarter. It needed to run on standard PC equipment that was readily available in the department. Technical support for the application by experienced system administrators would be limited. The applications needed to be light, fast, easy to use, stable and easily maintained by the department's staff that approached web development from a marketing, as opposed to technical, perspective. Despite the fact that this was the Department of the Internet, the focus of the department's activities was on content coordination. The IT department administered the heavier-duty web servers and telecommunications that were off-site.

Objective

This document will relay the lessons learned from my investigation into the software tools required and available for building a rudimentary, dynamic internet application. Consideration will be given to commercial and free software alike. We'll begin with some definitions and background of the elements of a dynamic web application and then continue on to a tour of some of the available technologies. It is my intention that this article serve as an informative starting point for developers and would-be developers as they begin to seek solutions for their web application development requirements.

What Is a Dynamic Web Site?

Advanced web designers often use a scripting language called JavaScript and a system of naming the parts of the web page--the Document Object Model, or DOM--together with HTML and CSS to create dynamic content on a page. These effects are sometimes called Dynamic HTML or DHTML. I, however, am limiting the scope of this project to building a web site that will generate dynamic content by interacting with a database. I was not interested so much in effects displayed to the user but in focusing on building an intuitive web application. The project scripts will be executed at the server, not at the client as occurs with JavaScript. The tools and methods presented here will not limit the addition of interactive or stylistic features to the application at a later time.

What Is a Database?

Data is defined as a general term meaning the facts, numbers, letters and symbols processed by a computer or communications system to produce information. In a computer system these items are typically stored in files. A collection of related files is a database. Within these files, records of items are organized into rows and columns. In this case, I need something more sophisticated than a simple collection of files.

The project requires a RDBMS, or Relational Database Management System. An RDBMS is a software package that stores data in rows and columns as tables. Various tables can be related to one another in order to answer questions posed by the end user. These questions are known as queries. The RDBMS is a concept first introduced by Codd in an academic paper in 1970 but was not commercially available until the mid 1970s. The RDBMS responds to all queries whether they ask the RDBMS to retrieve, add, update or delete data from the tables.

Sometimes the RDBMS that services a web site is called the back end. The web pages that the end user will see is often called the front end. Questions are posed to the back end using a language called SQL (Structured Query Language). In short, if there is data that needs to be processed, a DBMS of some kind is required. The most popular at this time are Relational DBMS, that respond to questions/commands via SQL.

Choosing a Database

Oracle

One product that has become synonymous with the word database is Oracle. Oracle (http://www.oracle.com/) is largely responsible for the current popularity of the relational database. Over the years, its database server has become justifiably respected for being full of features, fast and reliable. Oracle is also supported on Linux, and it appears that Oracle is committed to Linux as a platform.

However, there were two primary reasons not to use Oracle for this project. First, the hardware requirements were well beyond the capabilities of the machine used to develop and serve this application. As a rule of thumb, 800MB of disk space is needed with 256MB of memory. Second, Oracle would have been too expensive an alternative for an application of this size and temporary nature. Even at the minimal licensing fee of five named users in perpetuity for a single server the cost of using this software would have been ($160/user) $800.

MySQL

There are several open-source database options available to developers. In conjunction with web sites, MySQL (http://www.mysql.org/) appears to be a very popular choice in the Linux community. MySQL is a very fast, multithreaded, multiuser and robust SQL database server. MySQL is now also open source and has recently formed a strategic alliance with VA Linux Systems, a company that sells and supports Linux-based computer systems. MySQL was first released to the public in November 1996 and has always been available with source code. MySQL has proven to be a lightning fast and reliable database solution for a growing number of companies such as SGI, ValueClick, Nortel/Insight, Tucows.com, Cisco and many more.

So it would seem that MySQL was more than up to the task of the humble application described at the beginning of this document. It is considered very fast with large record sets, and the MySQL Manual reports production systems with upwards of 50,000,000 records. Further, there seems to be a growing relationship between the team that develops MySQL and the team that develops PHP. The increased popularity of this dynamic duo, coupled with boundless enthusiasm from core developers of both technologies, culminated in a meeting of the minds in Israel earlier this year. This resulted in the MySQL library being packaged with the PHP 4.0 distribution, in addition to an agreement to help each other improve the performance quality of product integration whenever the opportunity arises.

However, MySQL does have some shortcomings. One of these shortcomings is in the area of transactions. Tim Kientzle eloquently and succinctly discusses transactions in his July article written for *Dr. Dobb's Journal*:

> A transaction is a set of related changes to a database. The SQL standard specifies that an entire group of updates can be issued to the database and then

either committed or rolled back as a unit. This lets you, for example, transfer money between accounts stored in different database tables by adding the money to one account and then trying to subtract it from another; if the second update fails, you can undo all of the changes at once.

While lack of transaction support will not immediately be an issue in our application, the threat of ``feature creep", even in an application of this nature, compelled me to seek an alternative system that does support transactions. Further, MySQL has only limited support for foreign keys. The foreign key is an important concept of the relational model. It is the way relationships are represented and can be thought of as the glue that holds a set of tables together to form a relational database. Another area where I found MySQL to be wanting was in subqueries that it does not support. A subquery occurs when a developer nests one SQL statement within another SQL statement. Again, while my application is small and likely would not run into too much difficulty with MySQL's support of foreign keys, subqueries and transactions, it was still another needling in the direction of an alternate RDBM.

PostgreSQL

Some of the concerns that I had with MySQL seemed to be addressed by the team developing PostgreSQL (http://www.postgresql.org/). The version of PostgreSQL that ships with Red Hat 6.2 is version 6.5.3. This version already had support for transactions and subqueries, but it did *not* have extensive support foreign keys. However, the PostgreSQL team had recently released version 7.0.2 of the database that does support subqueries.

The PostgreSQL FAQ reports that PostgreSQL has most of the features present in large commercial DBMSs, like transactions, subselects, triggers, views and sophisticated locking. It has some features that other databases do not have, like user-defined types, inheritance, rules and multiversion concurrency control to reduce lock contention. But this functionality seems to be at the expense of the speed afforded MySQL. In comparison to MySQL or leaner database systems, PostgreSQL is slower on inserts and updates because it has transaction overhead.

Similarly to MySQL and in great contrast to Oracle, the minimum system requirements of PostgreSQL are light. The PostgreSQL administrator's guide reports that although the minimum required memory for running PostgreSQL can be as little as 8MB, there are noticeable speed improvements when expanding memory up to 96MB or beyond. The rule is you can never have too much memory.

Check that you have sufficient disk space. You will need about 30MB for the source tree during compilation and about 5MB for the installation directory. An empty database takes about 1MB, otherwise it takes about five times the amount of space that a flat text file with the same data would take. If you run the regression tests you will temporarily need an extra 20MB.

The machine I used easily meets these system requirements. I don't expect the drop in speed to be an issue. I'm satisfied that PostgreSQL addresses my concerns regarding MySQL and Oracle. I decided to use PostgreSQL for this project.

Choosing a Scripting Language

As with database software there are many scripting languages available for the Linux platform. Several are appropriate for use as a server-side scripting language in a dynamic web application. The purpose of the server-side scripting language is to tie together the user interface presented to the user, which here will be written in HTML and accessed via a web browser, and the back end of the application or server system and database used by the application.

ColdFusion

One commercial product considered as an option was ColdFusion from Allaire (http://www.allaire.com). The Allaire product is widely used, and the server portion of the environment is available and fully supported for Linux platforms as well as on Windows NT. Allaire also offers a very nice development environment as part of this package called ColdFusion Studio that requires either the Windows 9x or NT platform to run. Both components are available via download for a period of evaluation or purchase from the Allaire web site.

The components of the ColdFusion package were built to support the ColdFusion Markup Language (CFML). CFML has a syntax that is similar to HTML in that there are opening and closing tags that provide functionality beyond normal HTML. These specialized tags intermingle or are embedded in the HTML application or page. The ColdFusion server works with the web server to intercept these special tags to allow for interaction with the database and server providing the back end of the application.

The impressions that I carry of this product and environment are very positive. It seems to be a very capable platform that is easy to set up and use. There also appears to be a large group of web-site developers using this program. I believe that casual interactions with colleagues are often very helpful when becoming familiar with a new language or software product. The similarities of the CFML to HTML would also make it easy for a novice to be up and running quickly. Additionally, ColdFusion works with Apache, another check in its plus column.

Drawbacks are, however, that the system requirement for the Linux version of the ColdFusion server, 512MB of RAM, is beyond the capabilities of the system that I have available for this application. The July 17, 2000 price list posted on the Allaire site reports a price tag of $1295 US for the ColdFusion Server 4.5 Professional for Linux. The Windows-based development environment will run another $495.00. If the budget for your application can absorb the licensing cost of ColdFusion, it deserves consideration.

Perl

In the Linux community Perl carries about as much respect as any programming language deserves to expect. It seems that beyond its use as a web-scripting language there is little that Perl cannot handle. Written by Larry Wall, open-source Perl was first released in 1987. Perl's latest version released this past March is version 5.6 and is available from http://www.perl.org/.

Perl is a very mature and viable alternative as a web-scripting language. There is a large developer community and a great deal of support available. It can be tightly integrated with the Apache web server and is available with most, if not all, Linux distributions. However, what I was hoping to find was a simpler solution for attaching my web pages to a database back end. Just because Perl is the most obvious solution didn't necessarily mean that is was the right solution. The simplicity criteria weighed heavily in my decision to examine ColdFusion and partially ruled out Perl in this case. I wanted to get my application up and running quickly with as small a learning curve as possible.

PHP

PHP (http://www.php.net/) is an open-source, HTML-embedded scripting language. Unlike Perl, which was born as a tool to assist in system administration, PHP was designed from the ground up to work with web pages. It seems to borrow from many programming languages that have preceded PHP, including Perl and C. The PHP FAQ discusses the differences between PHP and Perl:

> The biggest advantage of PHP over Perl is that PHP was designed for scripting for the Web, where Perl was designed to do a lot more, and because of this, can get very complicated. The flexibility/complexity of Perl makes it easier to write code that another author/coder has a hard time reading. PHP has a less confusing and stricter format without losing flexibility. PHP is easier to integrate into existing HTML than Perl. PHP has pretty much all the ``good" functionality of Perl: constructs, syntax and so on, without making it as complicated as Perl can be. Perl is a very tried and true language; it has been around since the late eighties. But PHP is maturing very quickly (http://www.php.net/FAQ.php#9.4).

From this discussion it would seem that PHP would be a good fit for the application I was building. There has also been some discussion as to how PHP compares with ColdFusion. The PHP FAQ gives a good summary but also points to a comparison of the two packages laid out by Mike Sheldon. Mike's comparison (http://marc.theaimsgroup.com/) appears even handed. He points out some strengths and functionality of ColdFusion that I had not considered. One of the most interesting is an abstraction layer to database interaction. In ColdFusion, the developer uses the ColdFusion administrator to set up a data source. Whether the source is Oracle or an ODBC connection is inconsequential with regards to the CMFL tag used to interact with the source. In PHP, each database that we may want to use has a set of functions that are specific to the chosen database. Mike also points out that ColdFusion is bundled with a search engine called Verity. PHP has no bundled search engine and does not have an integrated development environment.

Two areas I was surprised to hear that PHP had weaknesses in were that of error handling and the ability to handle dates. Because of the simple nature of this application, I don't see either of these weaknesses becoming a problem. I only intend to use one database, and there is a function set for each database under consideration for this application. I don't see the data abstraction offered by ColdFusion as being a bonus. In the final analysis, I selected four primary components for this application: GNU/Linux operating system, PostgreSQL RDBMS, PHP server-side web-scripting language facilities and the Apache web server.

Conclusion

There are a myriad of software options available for the Linux operating system. The alternatives all have strengths and weakness that arise from the software's efforts to serve a particular primary need. I would recommend that those interested in deploying one of these software packages use a reasoned approach in determining which package best suits their needs. The value of developers and consultants is only partially derived from their mastery of a particular technology. The understanding of the particular processes by which they hope to influence their deployment of an information-technology solution should be of primary concern. Because there are many high-quality alternatives available to address a given information-technology challenge, it is the technologist's ability to grasp and deal with the key issues of the project in question that will ultimately determine of its success or failure.

For my project, the combination of Linux, Apache, PostgreSQL and PHP was appropriate. This combination will likely not be appropriate for all projects. However, I hope that you find this article an informative first step in your investigation of alternatives for your next project.

Resources

Dennis Roman Gesker

(drgst30@katz.pitt.edu) is currently a candidate for the degrees of MBA and MS-MIS in the dual-degree program at the Katz Graduate School of Business, University of Pittsburgh. Upon graduation he will be joining the management team of Alamon Telco, Inc. in Kalispell, Montana.