

DESIGN OF A TWO-COURSE SEQUENCE IN WEB PROGRAMMING AND E-COMMERCE

John Phillips, Joo Tan, Matthew Phillips, Nicholas Andre
Computer Information Science
Mansfield University of Pennsylvania
Mansfield, PA, 16933
(570) 662-4704
jphillip@mansfield.edu

ABSTRACT

Web programming and e-commerce are exciting topics to explore in the classroom. Needing a replacement for an aging COBOL course sequence, we undertook the design and implementation of a web programming course sequence. At the most general level we wanted our students to be able to competently design and implement n-tier web-based business applications. Our solution involved the use of popular open source software including Linux, Apache, Perl, PHP, JavaScript, and MySQL. To support the teaching of this software, we set up our own Internet accessible Linux server, becoming a small-scale web hosting operation. Each student was provided with their own Linux account complete with a private home directory, private database account, and public web page area. This paper will discuss details of these courses. In addition, we look at some of the configuration and system administration issues involved in supporting web programming courses.

1. INTRODUCTION

Our computer science department has long offered a dual programming language path for students to follow. As freshmen, students learn to program by taking a traditional two-course sequence in C++ using the Microsoft Windows environment. As sophomores, they take a two-course sequence in a secondary language along with a third course covering data structures using C++. Prior to the fall of 2000, the secondary language was COBOL taught in the Microsoft Windows environment. At that time we decided to update our secondary course sequence and replace COBOL with an e-commerce and/or Internet-programming course sequence.

Our first attempt included a course covering the Perl programming language, the Hypertext Markup Language (HTML), the Common Gateway Interface (CGI), and the Structured Query Language (SQL), all taught in the Linux environment followed by a course covering Java taught in the Windows environment. The Java course turned out to be a disaster. Over a two-year period, three instructors taught four different sections. Each instructor was surprised at the difficulty our students had in learning Java. Together we decided that we could not get our students to the level where they were doing interesting Java programming in a single semester.

The department debated the merits of covering Java in two semesters versus building on the very successful Perl course. Although a two-course sequence in Java

would have better reinforced the object-oriented programming concepts our students seemed to struggle with, we felt that teaching Java would be somewhat redundant with the material covered in the C++ course sequence. A decision was made to go the Perl route as this seemed to opened up many new learning opportunities.

Thus, the old COBOL-based Business Programming Concepts 1 and Business Programming Concepts 2 courses have been replaced with courses that will be referred to here as Web Programming 1 (WP1) and Web Programming 2 (WP2). This paper will discuss the design and implementation of these courses and the Linux server we set up to support them.

2. COURSE DETAILS

Having a secondary web programming language sequence in a Linux environment contrasts nicely with the primary C++ language sequence taught in the Windows environment. Scripting languages such as Perl tend to be small-program oriented and allow the programmer to do quick (and often dirty) programming. C++ and Java on the other hand, tend to be large-program oriented. It is often easier to write a quick Perl solution to a problem than it is to solve the same problem in C++ or Java. It is possible, therefore, to have students really hone their problem solving skills with Perl by having them write many small programs. Furthermore, students who already had at least one C++ course can master Perl in less than a semester. This leaves time to explore other essential topics such as HTML, CGI, and SQL databases.

The other great contrast we can explore in this course sequence is programming in the Linux environment. We believe that exposure to this operating system is perhaps more important than the choice of programming language. Students learn to use the Vi text editor that is standard across all Linux and UNIX platforms. They also learn the basic Linux shell commands. Finally, they become more self-reliant by learning to get help using the on-line Linux and Perl manuals, Linux How-To documents, and by using Internet search engines such as Google. [Redhat02, Ramirez03]

The essential components of web programming are covered in the first course. By adding a second course that builds on the first course, students become more proficient in this material and are able to construct very sophisticated e-commerce solutions. In addition, we are able to fill in a few gaps by including coverage of Cascading Style Sheets (CSS), JavaScript, PHP: Hypertext Preprocessor (PHP), plus a bit of Extensible Markup Language (XML) and Java as well.

2.1 Web Programming 1 (WP1)

The first course in our two-course sequence has a reputation among our students as being difficult but also one of the best courses in the department. The reason that many students find it difficult is that there is so much material to cover in a very short amount of time. The student must learn to work in Linux, which is a new environment for most. The student must also learn Vi, a rather unfriendly but very powerful text editor found on most Linux and UNIX machines. The student then learns the Perl programming language, HTML, CGI, SQL, and database normalization. Finally, the student must show that they can put all of these pieces together by completing a final course project – typically an e-commerce project such as an online store.

Perhaps the biggest problem many of our students have is that they are very weak problem solvers. Hence, the biggest emphasis of the course is not e-commerce or Linux or Perl; rather we stress problem solving by having the students write lots of

short programs. Each week the students are given a new problem set. Each problem set contains between 5 and 18 problems that the student must code.

We cover one or two chapters a week from our Perl textbook. To help students keep up on the reading and working hard on their problem sets, a weekly quiz is given. The quiz typically has a couple of complex programs where the student has to trace through the code and figure out what the output will be. Often there will be a short program that the student must write as well.

Linux basics, HTML, and the Perl textbook are covered in just 10 weeks. This is followed by two weeks coverage of the basics of SQL and database design. Two weeks are left to work with the students on their final projects. Each student demonstrates his or her project to the class the final week of the course.

It is interesting that students will complain a great deal about the amount of work they are doing during the semester. However, after the final project is completed, they seem to forget about the amount of the work they did and instead have a great deal of pride in their project as well as a high level of satisfaction in what they were able to accomplish in the course.

2.2 Web Programming 2 (WP2)

The new version of WP2 builds on the material from WP1. A primary goal is to move the students away from the realm of short textbook style exercises and programs into the realm of real-world applications development. Therefore, WP2 is taught in a very different style. There are no quizzes or exams. The students are required to complete three team projects over the semester, one individual project, and a portfolio. The first project is Perl based. The second project is PHP based. The third project is JavaScript and CSS based. The final project requires using either Perl or PHP on the server-side with a normalized MySQL database, and both JavaScript and CSS to be used on the client-side.

To prepare for the first project the instructor walks the students through the development and coding of a small e-commerce application in Perl. The student teams then take the instructor's code and modify it to solve a slightly different problem—an example is the development of an online classroom reservation system complete with documentation and working demo site. This was an actual problem that our department needed solved giving the project a bit more credibility to the students.

The second project is similar to the first except the students learn how to program in PHP. This language is so similar to Perl that it is easy for students to pick up. This time the instructor walks the students through the creation of a simple web-based discussion board. Small student teams are given the task of creating a business scenario for a similar application and then told to develop a working program complete with detailed documentation. For example, one team developed a music discussion board; another developed an online dating service; and another a help discussion board for a software company with an advanced threaded message system, and so on. Many of the student applications are very impressive and go far beyond the simple code provided by the instructor.

The third team project is to develop a web site that uses CSS and JavaScript. The project web site is to be about CSS and/or JavaScript as well. For example, the site could offer a JavaScript tutorial with dynamic online quizzes, or it could be a Dynamic Hypertext Markup Language (DHTML) reference site that shows how to combine CSS, JavaScript, and the Document Object Model (DOM) in order to create interesting client-side effects.

Finally, the individual project is to be the two-course sequence capstone. The student must develop an application that combines a server-side language with a MySQL database backend as well as a CSS and JavaScript front end, creating an n-tier application. The exact problem to be solved is left to the student.

To pull all of this work together, a large part of the student's grade derives from a web site portfolio consisting of the three team projects and the individual project. The documentation includes project write-ups along with programmer and user guides. A primary goal for this course is to have each student put together an impressive web site complete with documentation and working programs that he or she could later show to prospective employers.

2.3 Course Materials

The text for the WP1 course is *Sams Teach Yourself Perl in 21 Days*. [Lemay02] This book has been used for three years and seems to work very well when supplemented with course notes. Throughout the course the student is referred to many other online sources of information such as perldoc.com for a web-based source of the Perl documentation.

Often lectures are presented where the instructor Telnets into the server and interactively writes a program for the students. The students constantly see how the instructor uses Linux shell commands and the Vi text editor to write programs. Hence, little instruction is needed on these topics beyond the demonstrations given in class. For additional information, students are referred to the many web-based tutorials covering the Vi text editor, the Linux shell commands, and HTML coding.

Database design and SQL take a bit more time to cover. These topics are new and unfamiliar to most of the students. We use our own set of notes to cover this material. In addition, we like to assign the websites sqlcourse.com and sqlcourse2.com as homework. These sites do a good job in covering the basics of SQL in a hands-on environment.

The text for the WP2 course is *Internet & World Wide Web How To Program*. [Deitel02] This text covers a great many topics including CSS, JavaScript, Perl, PHP, MySQL, and XML. We tend to use this book as a reference and lecture by developing the project applications described in section 2.2 above. In addition, we present our own notes that we have developed that are based on many different resources.

In the WP2 course the student continues to be referred to many of the excellent web-based resources found on the Internet. As mentioned before, perldoc.com is a good starting point for Perl documentation. The PHP manual found at php.net is made even better by the running commentary of postings from users giving help and useful code examples for each command and function. [PHP03] Likewise, the MySQL manual found at www.mysql.com is very thorough and includes a nice tutorial on SQL. [Widenius03] This documentation is written for professional programmers. As such, we believe it is very important for our students to become comfortable with this material rather than relying exclusively on a textbook.

2.4 Student Satisfaction Survey

An anonymous survey was given late in the spring 2003 semester to students enrolled in the WP2 course. There were 25 respondents to the survey representing every student enrolled in the course. A variety of questions were asked about the WP1 and WP2 course sequence. The survey results indicate that the majority of students approve of the course content and of the way the courses are taught. Quite a few commented on the large amount of work created by the weekly quizzes and the large

number of programming problems assigned in WP1. Several others commented that their grades were suffering in other classes due to the extra time they were putting into their WP2 programming projects. For the most part, students indicated that they would not change anything major in either course.

Two questions on the survey asked the students to rate each web-programming course on a scale of 1 to 10. A one would be equivalent to the worst course the student had taken at the university. A 10 would represent the best course the student had taken. The average score for both web-programming courses was 8 indicating that the students highly approve of the courses. For the WP1 course, three students indicated that it was the best course they had ever taken. For the WP2 course, two students indicated that it was the best course they had ever taken.

2.5 Language Choices

Other languages could be used to teach a web-programming course sequence. Perl seemed like a good choice given its strength in text processing and the ease in which students pick it up. Perl has been a favorite of web-programmers developing CGI applications for many years. In addition to web programming, Perl is an essential programming language for system administration work. It is freely available for Unix and Windows platforms and comes standard with most Linux distributions.

For the second course, Advanced Perl and PHP seemed to make a good combination of languages for server-side programming. PHP is very similar to Perl except it is tailored to web programming, whereas, Perl is a more general-purpose language. JavaScript and CSS are used for client-side programming and compliment the Perl or PHP code running on the server-side.

It is interesting to see that the March 2003 TIOBE Programming Community index ranked Perl as number 4 after Java, C, and C++ respectively. [Tiobe03] The languages HTML, PHP, JavaScript, and SQL took positions 6 through 9 respectively, following Visual Basic. Although the exact meaning and validity of this index might be questioned, it is nonetheless reassuring that these two courses are covering languages that seem to be popular in the industry.

2.6 Spin-off Courses

Operating our own Linux server and offering this popular two-course web programming sequence has created a demand by the students for additional related courses. In the past, when WP2 had been Java based, we offered an advanced course on PHP, one on JavaScript, one on Advanced Java, and one on Linux System Administration. These courses were all highly successful and had high enrollments with the exception of the Advanced Java course. Now that PHP and JavaScript have been pulled into WP2 we continue to offer Linux System Administration and plan to offer a new course based on Macromedia Flash with ActionScript programming in spring of 2004. [Macromedia03]

3. THE COMPUTING ENVIRONMENT

Most of the computers in our department run Microsoft Windows. Our IT department was not anxious to support anything else. Therefore, it fell on the department's faculty to set up a Linux server that all of our students could access from the Windows machines. As Linux is a multi-user operating system, it is not difficult to set up a Linux server on the network that supports all of the languages and the database that the courses required. Each student is given his or her own private Linux account, home page, and a private SQL database on the server.

3.1 Configuring Linux

In the spring of 2003 we are using Red Hat Linux 8.0 running on an older Pentium III computer with 512 MB of RAM and an 80 GB IDE hard disk drive. It is relatively easy to setup and operate Linux once you have been through the process. However, good instructions are essential if you are new to Linux system administration. [Redhat02] We chose a server install from the standard Red Hat Linux 8.0 installation menu and selected the components discussed below. Step-by-step details on how we setup and configured our server have been documented and can be obtained from the authors on request. [Phillips03]

We chose to install X-Windows and GNOME to provide a graphical user interface in order to ease some of the system administration tasks. These are not absolutely necessary but we have found them useful for a variety of tasks. Rather than use automated tools, we do most of our configuration directly on the text-based configuration files. We have a better understanding of how the server works using this method.

We chose to install the Perl, PHP, and Python programming languages among others. The popular MySQL relational database software was selected for the database server and Apache's HyperText Transfer Protocol (HTTP) software was selected for the web server. [Apache03] We installed a File Transfer Protocol (FTP) server to allow students to easily transfer files between their computer and the server. We installed both the Secure Shell (SSH) server and the Telnet server to allow students to login to their Linux accounts from any Internet-based computer including all of our computer lab's Windows computers.

We installed VanDyke's SecureCRT SSH/Telnet client on the Microsoft Windows computers in our computer lab. [VanDyke03] The SecureCRT application greatly improves on Windows' own built-in Telnet application. In addition, IPswitch's WS-FTP LE program was installed on the Windows machines to allow for simple file transfers to and from the Linux server. [Ipswitch03] This gives the students an easy way to back up all of their work.

3.2 Automating User Accounts

One key to making this system successful is to reduce any system administration burdens the instructor may face. The largest burden is the setup and management of student accounts. These accounts include both the Linux login account and the SQL database account.

We use a two-step process to automate student account creation. At our institution we are provided with a web interface to our class rosters. To create the student accounts, the instructor simply copies the online class roster and pastes it into a web form. Upon submitting the form, a PHP program parses the roster and produces a nicely formatted list of the essential student data. The format we use is the student's school assigned email name followed by a tab and then the student's name. The email name is unique and is used as the student's account name. This is saved as a text file. The second step is to feed this text file into a Perl program that automatically creates the Linux and MySQL accounts. The entire process to add a class of students takes about a minute.

As each student account is created, the contents of the `/etc/skel` directory are automatically copied into the student's home directory. In the home directory, a `public_html` directory is created to store files that will be made available to the web. A few sample programs are placed in the student's `public_html` directory. Finally, we find it useful to place some common alias commands in the student's `.bashrc` file. For

example, the following alias command is useful for reviewing the Apache error log file: `alias log='tail -50 /var/log/httpd/error_log'`. Now the student can just type the word `log` and then see the last 50 lines of the error log without typing in the full command. The `/etc/skel` directory allows these and other similar tasks to be setup ahead of time and be automatically applied to new accounts as they are created.

3.3 System Administration

After the system is configured and the student accounts are created, the Linux system is typically trouble free. The administrator should occasionally check the log files and the disk space. A plan for backups should be devised. Our plan is simply not to have any backups. Each student is responsible for maintaining one or more copies of their work on disk. If the server crashes, then we will simply reinstall it and the students will be required to ftp their work back into their accounts.

One administration problem we ran into is that we made the `/var` partition too small on the server. When students were busy heavily debugging their Perl projects, the Apache error log file grew so large that it filled the entire one Gigabyte partition. Since we had an 80 Gigabyte hard drive we realized that we should have allocated a few more Gigabytes of storage space to this partition.

Another interesting system administration problem we encountered was the abuse of the Linux `wall` command by a couple of our students. The `wall` command stands for write all and will let the user broadcast a message to all of the other users logged into the server. Several enterprising students thought it was fun to log in as a student who neglected to change his password from the default setting and then to write a Perl script that broadcast thousands of messages as if they were coming from this student. Unfortunately for the perpetrators, the `wall` command makes an entry in the system log. By locating the machine that the messages were broadcast from and by identifying the students who were sitting at that computer at that time, we were able to get a confession. The penalty was for the perpetrators to make a public apology to the class for wasting everyone's time and a promise that nothing like that would ever happen again. In addition, the class got a lesson on computer ethics with a look at some of the penalties for computer crime. Last but not least, we learned that it would be better to disable the `wall` command and the similar `wwrite` command so as not to tempt our students.

3.4 System Security

System security is of concern to us. However, we have done little to protect ourselves beyond the basic security provided by the Red Hat installation. Our server is located behind a campus network firewall and so far we have had no problems other than the minor mischief from our own students described above. One fear that we had was our server might be turned into an email spam machine. Therefore, we were careful to limit email so that students can only send email to other students on the Linux server; no email can be sent outside the server.

3.5 Automating Instructional Tasks - Tracking Homework

In both WP1 and WP2, all of the student coursework including programs and documentation is kept on the Linux server and is available at all times to the instructor. In WP1 the students create a large number of small Perl programs. To help track these programs and each student's progress, we developed a small Perl program that will scan each student's directory and produce a report showing what programs have been completed so far. We then do spot checks on certain problems to make sure the students are really doing them and that the quality is acceptable.

Likewise in WP2, students are required to organize their work according to a predetermined directory structure and file naming system. However in this case, the work is all web-based and the instructor can easily browse the project documentation and program code starting from the student's home page.

4. CONCLUSION

Developing this ambitious web-programming course sequence involved a good deal of work. The material covered included in-depth coverage of Perl, PHP, SQL, HTML, CSS, and JavaScript. We immersed the students in the Linux programming environment for two semesters. In addition, we offered a very brief look at XML and Java. We could have eliminated PHP from the mix. However, PHP was so easy to learn after learning Perl that we felt it was appropriate to include this popular alternative to Perl.

We are very pleased with this course sequence. WP2 may need some tweaking to ensure all of the students are participating fully in the team projects. Yet, the projects and the portfolios have been impressive. The students now seem to be very comfortable working in the Linux environment and very comfortable doing web programming.

Setting up our own Linux server had a steep learning curve. However, through this experience we learned a great deal so that Linux now seems much easier to work with. Running our own server naturally lead to the development of a Linux System Administration course allowing us to share much of our newfound knowledge with our students.

We believe that our students have greatly benefited by learning to program in the Linux environment as well as the Microsoft Windows environment. Our students have experienced the contrast of lightweight scripting languages versus a heavily structured large-scale object-oriented programming language. This course sequence has given our students a portfolio of work that they can share with prospective employers not to mention all of the additional acronyms they can add to their resume. We currently have no plans to change the overall layout of these courses and look forward to offering them again next year.

REFERENCES

- [Apache03] The Apache Software Foundation. "Apache HTTP server version 2.0 documentation". Accessed March 28, 2003. Available at: <http://httpd.apache.org/docs-2.0/>.
- [Deitel02] Dietel, H. M., Deitel, P. J., & Nieto, T. R. "Internet & world wide web how to program", Prentice Hall, 2002.
- [Ipswitch03] Ipswitch, Inc. WS_FTP LE download web site. Accessed March 28, 2003. Available at: <http://ipswitch.com/downloads/index.html>.
- [Lemay02] Lemay, L. "Sams teach yourself Perl in 21 days, 2nd ed.", Sams, 2002.
- [Macromedia03] Macromedia, Inc. Macromedia Flash MX web site. Accessed March 28, 2003. Available at: <http://www.macromedia.com/software/flash/>.
- [Phillips03] Phillips, J., Andre, N., Tan, J., Phillips, M. "Administering a Linux-based server for student programmers", Spring 2003 PACISE Conference Proceedings. Accessed June 22, 2003. Available in Adobe PDF format at: <http://www.mansfield.edu/~jphillip/documents/linuxadmin0403.pdf> (1.4MB).

[PHP03] The PHP Group. "PHP manual". 2003. Accessed March 22, 2003. Available at: <http://www.php.net/manual/en/>.

[Ramirez03] Ramirez, C., "Perl Documentation Website". Accessed March 26, 2003. Available at: <http://perldoc.com/>.

[Redhat02] Red Hat, Inc. "Red Hat Linux manuals". 2002. Accessed March 28, 2003. Available at: <http://www.redhat.com/docs/manuals/linux/>.

[Tiobe03] Tiobe Software, "Tiobe software programming community index for March 2003". Accessed March 25, 2003. Available at: <http://www.tiobe.com/tpci.htm>.

[VanDyke03] VanDyke Software, Inc. SecureCRT web site. Accessed March 28, 2003. Available at: <http://www.vandyke.com/products/securecrt/index.html>.

[Widenius03] Widenius, M., Lentz, A., & DuBois, P. "MySQL reference manual." Accessed March 22, 2003. Available at: <http://www.mysql.com/documentation/mysql/bychapter/index.html>.