Teaching Web Development with Limited Resources

Ellen L. Walker
Mathematical Sciences Department
Hiram College
Hiram, OH 44234
walkerel@hiram.edu
http://hirame.hiram.edu/~walkerel

Logan Browne
Mathematical Sciences Department
Hiram College
Hiram, OH 44234
brownelc@hiram.edu

Abstract

Computer Science programs are faced with demand from both students and employers for courses in hot topics such as Internet Administration, but resources such as course time and laboratory facilities are often scarce. Another trend facing our programs is the need to increase availability to non-traditional students. This paper describes a course in Internet Administration for both traditional and non-traditional students and how it addressed the issues of limited time, diverse student population, and limited laboratory facilities.

1. Introduction

Along with the growth in popularity of the World-Wide-Web has come increased demand from both students and employers for instruction in the techniques necessary to develop and manage interactive web sites [7]. Web site hosting is becoming a big business, requiring "technical professionals who can handle a variety of servers, platforms and applications" [4].

While students demand courses that will teach them the current "hot topics" for their resume, departments are naturally reluctant to take resources away from the "tried and true" core curriculum. A common compromise, especially for small departments that cannot afford to teach many electives, is to put "hot topics" courses into mini-terms, such as the "January term" in a 4-1-4 curriculum. These courses are allotted only three to five weeks instead of the usual quarter or semester.

A second trend in higher education can further constrain these courses. As the number of traditional, full-time undergraduates declines, colleges are trying to attract more non-traditional students, such as working adults changing careers. These students are also looking for practical courses that can immediately enhance their careers. To make such courses available to non-traditional students, meetings must be further curtailed – limited to late evening and weekend hours. Delivering much of the course over the Internet can significantly increase the content available to such students.

A third issue in providing a web development course is that students need access to and control over their "own" web servers. Neither the generally available campus laboratory facilities nor the campus-wide web server are appropriate for this use. Naturally, there is very little budget for such facilities.

This paper describes the course, Internet Administration, that meets the needs described above. The course was taught in a three-week, weekend format, *i.e.* class met on three Saturdays for six hours of lecture and presentations and two hours of laboratory[1]. Students worked on a final project entirely outside of class.

2. Course design

The primary goal of the course was to familiarize students with the issues, tools, and techniques that a web site administrator should know. These included both server-side and client-side interactive programming techniques, security issues, performance issues, and application areas. The course was explicitly *not* a course in web design, but in the underlying technologies on which a good web site rests.

2.1. Choice of Topics

To a large degree, the topics of the course were similar to those presented by Lim [7]. Because of the short format of the course, however, some material was necessarily curtailed. The main topics that were curtailed or omitted in this course were HTML and Java.

Through their experience in previous courses or on their own, most students entered the Internet Administration course with some background in HTML. Several students had taken a non-majors' introduction to the Internet where they designed their own web pages. Although this course was not a prerequisite, the Internet Administration course description suggested that all students look over the notes for this non-majors' course, and build their own (non-interactive) web page before beginning Internet Administration. Thus, only the HTML commands for

[1] Because of limited accommodations in the laboratory, the traditional students met on Friday for their laboratory session.

developing forms and the JavaScript extensions to HTML were explicitly covered in Internet Administration.

No course for web developers would be complete without a significant programming component. However, the short format of the course precluded a complete in-depth introduction to any programming language. In our course, CGI-scripting with UNIX shell and Perl scripts was covered, as was JavaScript. Both languages were taught by example, rather than from first principles. These choices provided one example each of server-side and client-side interactive content. The Java programming language was omitted because of the perceived amount of mastery necessary to use the language, and also because Java was not as different from the students' experience in other programming (C++) courses as Perl or JavaScript was. Given the limited time in the course, we believed that students would benefit more from limited instruction in the other two technologies than they would from Java.

While the basics were covered, the more advanced and specialized topics were covered only in student presentations. This allowed students to study one topic in depth, while hearing about all chosen topics.

2.2. Textbooks and Source Material

One of the difficulties in developing the Internet Administration course was finding a reasonable textbook. There were a great number of books on introductory HTML and beginning Web page design, and many books devoted to specific tools. No textbook contained both an overview of the issues involved in administering a web site, as well as sufficient specific examples to enable the reader to learn to use the tools. The closest book was [11], which was the only book with a truly server-oriented point of view, and did have some examples of both server-side and client-side scripting. In general, the students were satisfied with this textbook.

To provide background on the tools, students were required to purchase a reference book, [10], which contained material on all the tools that were used in the class. This book was chosen because of its relatively broad coverage at low cost. The students were dissatisfied, however, at the *reference* rather than *tutorial* style presentation. Some students chose to purchase the deluxe edition of [10], which included (on CD-ROM) textbooks on the major tools from the same publisher. The students who used the CD-ROM material found it extremely helpful; in retrospect, this edition should have been required, or at least more highly recommended.

Since neither textbook provided tutorial material on the languages that students would be using, it was necessary to supplement the textbooks with additional material. Freely available tutorials were found on the web for HTML forms [9], beginning CGI programming with Perl [5], and JavaScript Programming [6]. Students used these along with locally-written material in the laboratory assignments.

3. Course Delivery

In designing the course, the extreme limitation on contact hours, the diversity of the students, and the shortened format of the course all needed to be taken into consideration. The final format of the course included lectures, presentations, laboratory assignments, and a final project. Three short quizzes tested students' understanding of the reading material, but there was no final exam. Given the limited amount of time available for the course, it was thought that time spent on laboratories and the final project would be more valuable in the long run than time spent studying for an exam.

All assignments for the course (both handouts and "hand-ins") were placed on the web. This made it easier for students to access and submit assignments from home, their office, or a college laboratory, and helped to accommodate the diversity of students.

3.1. Lectures

Lectures were drawn from material in [11], including history of the Internet and web, introduction to standards such as HTTP and CGI, performance issues, security, and web server setup. Interactive demonstrations of appropriate web pages and scripts were also conducted during the lecture portion of the course.

3.2. Presentations

In addition to the second and third weeks' lectures, students presented summaries of their own research on tools and applications of web technology. Student topics were chosen from a list of suggestions provided by the instructor. Topics included data mining, intranets, firewalls, and multimedia over the Internet. Groups consisted of five to seven students, and each student had approximately five minutes of material to present. These presentations gave everyone in the class a brief overview of many important topics that we didn't have time to cover in depth, although each student did in-depth research on one of the topics. Most of the students included reference material for their presentation topic on their course web page.

3.3. Laboratory Exercises

Lectures and presentations took up six hours per week of class. The remaining two hours per week was devoted to structured lab activities. Each laboratory exercise covered a particular technology. There were two exercises on CGI scripting with Perl, two on JavaScript, and one on password protection.

Because of the severe time constraints on the course, laboratories were geared more towards reading and modifying code than toward writing code from scratch, consistent with the desirable attributes of closed lab activities [8]. Each laboratory assignment contained a statement of the goals of the assignment, followed by specific exercises. Most operated on a single example program: first exercise the program, then make simple modifications, and finally do a more significant extension to the example. Where available, pointers were given to available libraries, such as the Perl modules already installed on the server.

Perl examples included a Pig-Latin translator, a simple spell-checker, and a simple flat-file database. The

JavaScript example was a math test generator using cookies to keep score. We attempted to make useful as well as simple examples. The complete laboratory exercises are available on the course web site (http://cs224.hiram.edu/).

Students were required to produce a laboratory writeup (web page) for each laboratory assignment. These writeups included descriptions of and pointers to the students' solutions to the problems in the assignment.

The two-hour laboratory period was too short for most students to finish the exercises. Therefore, students completed their work outside of class, using their own Internet connections if they weren't on campus. Students communicated with the instructor by e-mail and sometimes telephone when they had difficulty with their assignments. Interactive debugging of bad scripts was simplified by the fact that all students' work was on a single machine (to which the instructor and TA had root access). For many students, especially those with little UNIX experience, more time in the laboratory would have been desirable.

3.4. Final Project

The culmination of the course was an individualized final project, which consisted of a web page with interactive content that each student developed using some of the tools that we had covered in class. Many of these projects required significant additional research. Example projects include: an archeological walk-through of a house using VRML technology, a Concentration game developed in JavaScript, several forms-based database front ends, an interactive computer system configuration tool, and a computer system administrator's tool (giving graphical access to process and load information on a Unix machine).

Draft final projects were made available during the third week of class, and students provided feedback on each others' projects to the instructor via a form. After removing identifying information, this feedback was provided to the authors so that they could modify their projects before final submission. "Submission" of a project was simply an email to the instructor that a project was ready for grading.

A complete project was a single web page with pointers to the student's original proposal, their interactive web page project, all source files, and a writeup including a technical description of the page itself, a description of the development process, and a self-evaluation of the page.

4. Computing Facilities

In order to run the Internet Administration course as described, it was necessary for each student to have access to a web server, including the ability to create and install CGI scripts. Neither the general-purpose student computing laboratory nor the campus-wide public web server were appropriate for this use, given that the students in the course were likely to render their servers useless for at least part of the time as they experimented. Logistically, the easiest way to administer the course was to have a single machine devoted to the course, including web servers for the faculty, the TA, and each student. With 23 students in the class, the machine would need to support approximately 25 web servers, as well as a peak load of 25 concurrent users. This section describes how such a web server was constructed and administered at very little cost.

4.1. Hardware Acquisition

At the time the course was planned, the "best" computer available for the course was a workstation running Ultrix under an 8-user license. Already, there was a plan to construct a computer for the Computer Science Program primarily from spare parts donated by the computer science majors. The inadequacy of existing facilities for the Internet Administration course provided the impetus for us to assemble those parts into a working server.

Donations included a 100MHz Intel 486 and motherboard, an old VGA monitor, an 850-megabyte hard drive, a 64-bit video card, and a keyboard. This left us with several pieces to purchase: a network interface card, a case, and 64 megabytes of RAM. This system had a total cost of about $250 – a very inexpensive server by any standard.

4.2. Operating System Choice and Configuration

Given the components of our server, it was no computing powerhouse, especially considering the peak load of 25 concurrent users. The chosen operating system had to allow for remote access the machine via the Internet, since many of the students would not be on campus while doing their projects. It also needed a file system that allowed for security based on the user. The system had to handle a separate web server for each of the 25 students, and to do all this on an old Intel 486. Finally, the operating system could not add significantly to the cost of the server. Linux was one of the few operating systems that met our needs.

Linux is a free, Unix-like operating system that was developed by thousands of programmers around the world. Many groups have developed packages of software and the operating system that are called Linux distributions. For the purposes of our course, we desired a distribution that was simple to set up with an easily navigable system of organization. Red Hat Linux 5.0 met these needs.

Out of the box, Red Hat Linux 5.0 had all of the features we needed (web, telnet and ftp servers, basic security, and many powerful software development tools). The entire installation process took only fifteen minutes.

After installing Linux with the default configuration, it was clear that many security features were in place (shadow passwords, detailed system logs, and strong file protections); however, there were several changes that needed to be made before we could let the students loose on the system. These changes improved the security of the system or made it easier for our users.

By default, the console had certain root privileges; however, the location of our server in a public computer lab made these permissions inappropriate. In particular, the console user normally has the ability to immediately shut down the system with a [Ctrl]-[Alt]-[Del] key sequence. Since the OS catches these keys and initiates a graceful shutdown, it was possible for us to print a warning on the console and log the action, instead of shutting down. Otherwise, a curious individual in the computer lab would

have had the power to crash the machine out from under the remote users![1]

Before creating any user accounts on the system, we made a template for the home directories in the directory /etc/skel. Files and directories located there are automatically copied to a new user's home directory at account creation. Each user was given a log directory where the log files specific to their server would be kept. A symbolic link, web, was also created in each user's directory that pointed to the content of their web server.

Other than these simple changes, the default Red Hat configuration was quite usable for our purposes. We were able to provide a customized server to the students on the first day of class, after only about 2 days of configuration.

4.3. Web Server Choice and Configuration

Our choice of web server was Apache [1]. Apache is one of the most commonly used web servers on the market today, not only because it is free, but because it is simple to configure, stable, and is now available for many computing platforms. Even the Apache 1.3 beta for Win32 is considered by many webmasters to be more stable that the current version of Microsoft's Internet Information Server (IIS).

By default, Apache is well configured to operate as a single site web server, but we needed something more for our course. We didn't feel that the students would have the same experience if they shared a server, so we gave each one a separate web server on this single machine. Each student was given a port (above 8000), with its own server, log files, and configuration [2]. Thus, the students had administrative privilege to their individual server without letting them roam free on the global server.

Besides having their own web servers, the student servers needed the power to execute CGI scripts. Therefore, we enabled CGI execution in each directory [3]. Although this was a very insecure practice and dangerous to the server, it was necessary for student experimentation. Given the security risks, it was important to instruct the students at the first class in how to write safe CGI scripts, and how to keep scripts from being abused by malicious web surfers. In fact, the risk on our machine was lowered by the fact that the machine was newly connected to the Internet with a name that was difficult to guess.

With only a few changes to the basic configuration to improve compatibility with Windows systems, we began the course with 25 individual servers, each with its own powerful administrator. The simple and well-documented configuration files of the Apache server made the entire process fairly painless.

4.4. Problems and Future Changes

There were several problems that we discovered with our server during the three-week run of this course. Many of the problems that we encountered before the beginning of this course have already been listed, with their solutions, in the previous sections; however, there are some problems for which we found no solution. These are areas where the procedure will be changed in future courses.

One problem with the configuration of the server as described above is that the students did not have their own set of configuration files. All of the students use a system set of httpd configuration files that can only be modified by the root user. To give the students even more control over their individual servers, we would like to give each student server its own separate configuration files. This would allow all of the students to get the full experience of administering an Apache web server, and train them better to be real Internet administrators.

Giving each student a set of configuration files would also remedy the security problem of using non-script-aliased CGI scripts. Each student could define a secured script-aliased **cgi-bin** directory that would contain all of their trusted scripts.

During the three weeks of this course, we logged about 50 hours worth of system maintenance time in repairing various problems with the server. Most of this time was spent repairing simple configuration oversights and altering file permissions. More serious problems that developed led to the changes described above.

5. Addressing Constraints

This section summarizes the constraints that made the environment for the Internet Administration course less than ideal, and how those constraints were addressed. These constraints include resource constraints, time constraints, and the diversity of the students in the class.

5.1. Resource Constraints

Of the three types of constraints, our approach to dealing with resource constraints was most successful. The constraints on available hardware and support personnel were dealt with by building our own machine as described in section 4, and in staffing the course with an undergraduate (the second author of this paper) who served both as teaching assistant and as system administrator. This student's association with this course dovetailed with an independent-study course on Computer Security, so he received both pay and credit for his effort.

5.2. Time Constraints

The time constraint was evident both in the number of contact hours for the course (24, including laboratory time) and the total "real time" that the students could use to complete their work (approximately 4 weeks). The first step in addressing this constraint was to pare the content of the course. As discussed in Section 2.1, both HTML and Java were entirely eliminated from the course.

Because of limitations on lecture time, Perl and JavaScript weren't taught in lecture from first principles, but introduced directly in the lab. As mentioned in Section 3.3, the emphasis was on students' reading and modification of code. Although this is a more "real-world" way of learning a language, some students complained about having to teach themselves more of the tools to

[1] Unfortunately, a crash was caused in this manner before we made the configuration change. We removed the console keyboard for additional safety.

complete the project. We hope that these students will realize the value of this experience in time.

A more significant issue was the lack of total time from the beginning to the end of the course. Within the four weeks from the first day of class to the last (one weekend was skipped), it was difficult for the working students to find enough time to complete all the coursework. Some of the students took a day off work to finish their assignments. This issue of time was less of a problem for traditional students, who were able to devote full time to the course, but still, there was less time for new concepts to sink in.

Adding more laboratory hours to the course would somewhat alleviate the difficulties, as it would both increase contact hours and decrease the number of outside hours necessary for students to devote to the course.

5.3. Diversity of Students

The majority of the students' complaints could be traced to the diversity of the student population in the course. Although the class was taught in a weekend format to appeal to non-traditional students, it was also popular with traditional students, so the class consisted of approximately half traditional and half non-traditional students. While traditional students could devote full-time effort to the class, non-traditional students had to fit in their classwork around full-time jobs. A second area of diversity was the students' computer science background. Programming background varied from a single course in C++ to a nearly-complete computer science major, and UNIX experience ranged from complete novice to administrator.

Although the makeup of the course was known in advance, the differences with regard to available time and background knowledge were not planned for. The laboratory assignments, in retrospect, were too long for the non-traditional students, and not substantial enough for the residential students. The level of presentation in class was not detailed enough for students with little background, and overly detailed for students with greater background.

The laboratory format helped somewhat in dealing with differences in preparation. With two instructors (professor and TA) in the lab for 12-15 students, individualized attention was possible. In some, but not all, cases, extra time in the laboratory was sufficient to overcome a lack of background.

Because of the limited number of seats in the laboratory, the traditional and weekend students were already separated into two laboratory sessions. In retrospect, this distinction could have been taken advantage of to provide enrichment for the traditional students that had more time to devote to the course.

To avoid the extreme disparity of preparation, more specific prerequisites should have been enforced. Students with no UNIX background should be required to take a short course on UNIX before the Internet Administration course. Similarly, students who need additional training in Internet tools should have that provided outside of class, preferably before the beginning of the regular class.

6. Conclusions

Despite the difficulties discussed here, many of the goals of the course were accomplished. The students learned the material, as evidenced by their performance, especially the final projects. Several of the traditional students obtained web administration jobs, and the feedback from these students was quite positive. One student emailed during the summer, "I am so thankful for the Perl that I picked up in [Internet Administration] because that is pretty much what I spend my time doing..." Another student has become webmaster for the college computer center based on his experience in the course. Students uniformly answered positively to the statement "I learned a lot in this course."

Efforts to overcome the lack of laboratory facilities by constructing a computer were successful, while the time constraints and student population diversity posed more severe problems. Future improvements include increasing laboratory time and strengthening course prerequisites.

7. Acknowledgements

The authors thank the Hiram Women's Club for the money to complete the course web server machine.

8. References

1. Apache Project, *About the Apache HTTP Server Project.* http://www.apache.org/ABOUT_APACHE.html
2. Apache Project, *Apache IP-Based Virtual Host Support.* http://www.apache.org/docs/vhosts/ip–based.html
3. Apache Project, *Frequently Asked Question: How do I enable CGI execution in directories other than the ScriptAlias?* http://www.apache.org/docs/misc/FAQ.html
4. Gerwig, K, Business: the 8th layer: Web site "outsourcing": go with the pros, *netWorker: The Craft of Network Computing*, Vol. 2, No. 3 June 1998), Pages 19-23.
5. Hamilton, J., *Kira's Web Toolbox – CGI Programming 101*. http://lightsphere.com/dev/class/
6. Koch, S., Voodoo's Introduction to JavaScript. http://rummelplatz.uni–mannheim.de/~skoch/js/tutorial.htm
7. Lim, Billy B.L. Teaching Web Development Technologies in CS/IS curricula, in *Proceedings of SIGCSE '98*, ACM, 1998.
8. Parker, B.C. and McGregor, J.D., A Goal-oriented Approach to laboratory Development and Implementation, in *Proceedings of SIGCSE '95*, ACM, 1995.
9. Pero, C.A., *Carlos' Forms Tutorial.* http://robot0.ge.uiuc.edu/~carlosp/cs317/cft.html
10. Spanhour, S. and Quercia., V., *Webmaster In a Nutshell*, O'Reilly Publishers, 1996.
11. Yeager, N.J. and McGrath, R.E., Web Server Technology - The Advanced Guide for World Wide Web Information Providers, Morgan Kauffman Publishers, 1996.