Introduction to Database-Backed Web Programming
CSE 491, Fall '08
Syllabus 8/26/08

Lecture - Tuesdays 12:40 - 2pm, Egr 1202
Lab - Thursdays 12:40 - 2pm, Egr 3353

Instructor: C. Titus Brown, ctb@msu.edu; Egr 3137.

Course Web site: http://ged.msu.edu/courses/2008-fall-cse-491/

Objectives: In this course, you will implement a functional database-backed Web site in the Python programming language. The goal of the course is to introduce you to technologies, techniques, and thought patterns underlying database-backed Web programming, while exposing you to modern software development practices including version control, automated testing, defensive programming, good code hygiene, and whatever else I think you need to learn.

Materials: There are no required books. Lecture notes will be provided in outline form, as will certain demonstration code. If you must have a book, I suggest "Philip and Alex's Guide to Web Publishing". It's available online for free, and you can also buy paper copies quite cheaply. But it's only indirectly relevant to the course.

Attendance: You are responsible for learning the material presented in class whether or not you attend class. This should not in general be impossible, but it will be easier if you show up for lecture... I will provide an outline of the material covered in each lecture, in any case.

Office hours: TBD. There is a course mailing list & wiki, and you are encouraged to ask and answer questions there. Please contact me via e-mail if you want to set up a private meeting. For e-mail, please make sure that 'titus@idyll.org' and 'cse491-fall-2008@lists.idyll.org' make it through your spam filters! (If you use gmail, you may have problems.)

Homework & grading: Weekly programming assignments will make up 65% of your grade. The programming assignments will generally be progressive, so you will need to fix whatever bugs or mistakes were present in the previous week's work. I plan to hand out assignments on Tuesdays in lecture and they will be due the following Tuesday at the beginning of lecture. Assignments will be graded on depth of understanding (do you know what you're doing and why?), correctness (do they work?) and English (spelling matters, folks). **The bottom two assignment scores will be dropped**. The remaining 35% of your grade will be determined by two ~5page essays, one due at midterm and one due at the end of the course (topics and exact due dates to be determined).

You are welcome to work in groups, and you are welcome to use code from anyone and anywhere for your projects - this explicitly includes the Internet and the Python Cookbook. (You must cite your sources for any substantial borrowed pieces of code, of course.) However, you must turn in your assignments individually and you will be individually responsible for the quality of your work. You may also be asked to explain your work to me in private or in public in front of the class. **The primary goal is for you to do the work and to understand what you have done and why.**

All assignments must work in Python 2.5, which is freely available and cross-platform. Your work will be tested under Linux; details to follow.

Late or incomplete homework will not be graded and will receive a score of 0 (note, you can drop your two lowest scores). However, because programming assignments will build upon previous work, you will have to do it anyway...

Commercialization (my notes) and intellectual property (your code): My notes will be licensed under a Creative Commons license, attribution required (http://creativecommons.org/licenses/by/2.5/). This means you can do whatever you want with them as long as you properly attribute their origin to me. Copyright to your code will be retained by you, but in order to take the course you must give me and all other students in the course a license to share and adapt your work for non-commercial purposes (Creative Commons, Attribution-Noncommercial-Share Alike; http://creativecommons.org/licenses/by-nc-sa/3.0/). Please see me privately *before drop day* if you want to discuss this.

Course project: Over the next 15 or so weeks, you will implement a complete dynamic Web site. There are several options for what site to implement: we could write a personal course scheduling system for MSU courses; a Web site recommendation engine a la Digg or Reddit; or a blogging/bulletin board framework with commenting. Let me know if you have strong preferences one way or another.

Course Outline (Approximate)

1. Overview, class structure, How The Web Works
1b. Python, types, constructs, introspection, etc.
HW: fib series with different constructs (for, while, list comp, generators, classes, recursion)

2. Networking, client/server/peer, protocols, name service, etc.
2b. Mechanics, opening sockets, processes/threads/async/etc.
HW: Make a TCP echo server.

3. HTTP protocol, statelessness, GET/POST, query strings, cookies, etc. Markup intro.
3b. HTML, parsing headers, etc.
HW: Make a simple Web server.

4. Computational hygiene and software carpentry: version control, code layout, testing
4b. Project layout & imports; subversion; twill testing.
HW: Reorganize development environment appropriately, provide automated tests.

5. Abstractions, architecture, patterns, classes, etc.
5b. Designing a general request single-process/thread handler/HTTP server.
HW: Implement and write a simple single-process/thread in-memory application.

6. Maintaining persistent state, databases, etc.
6b. Sessions, pickles, etc.
HW: Implement login and sessions.

7. Relational theory, SQL databases, ACID
7b. Using sqlite
HW: Build a SQL database for persistent sessions, login, expiry.

8. More SQL, relational theory, client/server databases.
8b. Using PostgreSQL
HW: TBD by class topic.

First paper due, topic TBD. Probably something to do with Web architecture, transactions, and statelessness.

9. Revisiting code hygiene and architecture; the MVC arch
9b. Code reviews, refactoring, etc.
HW: Do a code review of three other student's code & switch to another student's code base.

10. Concurrency models: processes, threads, and cooperative multitasking.
10b. Working with concurrency.
HW: Implement "shared nothing" backend; make multithreaded, multiprocess safe. Implement simple coop multitask server?

10. Templating, security, Unicode; the Semantic Web, REST/RPC
10b. More HTML, CSS, etc.
HW: Take template/ideas from OSWD, apply consistently to your site

11. Client-side scripting, JavaScript, and asynchronous queries
11b. JavaScript, jQuery.
HW: Adding some simple JS.

12. The DOM, JS sandbox, JS limitations. Alternatives.
12b. More JavaScript, jQuery.
HW: JS library scavenger hunt

Second paper topic due; one paragraph proposal regarding final paper topic.

13. Advanced Web testing: GUI and AJAX elements, code coverage, cont integration.
(proxy recording; peekaboo; manhole; wsgi_intercept; buildbot?)
13b. Recording and running Selenium tests.
HW: Complete test coverage of server.

14. Object databases, O/R mapping, impedance mismatches, etc.
14b. Using Durus, SQLAlchemy.
HW: Reimplement with Durus *or* wrap with SQLAlchemy.

15. Deployment; wrap-up and other topic slippage.
15b. Deployment demo, general Q&A.
HW: Final project

Second paper due.

Potential "bonus" (optional) lectures, if sufficient interest:
- software licensing, and open source;
- starting your own business & "the long tail";
- using HTML and JS for (scientific) data presentation and data set exploration;