

## **Project #2 (due December 20)**

In the second project, you have to create a web-based user interface for your blogging software database schema designed in the first project. In particular, authors should be able to register, to create new weblogs, and to create new postings for their weblogs. Visitors should be able to view postings in a weblog in chronological order, search for postings containing particular keywords, or list the names of all weblogs (with hyperlink pointers to the weblogs themselves). It would be desirable if a visitor could bookmark a particular weblog or posting, so think about this issue – but how to allow this may depend on the system setup you are using. For extra credit, visitors should be allowed to post comments, and advertisers might be allowed to post ads that will be displayed on the weblogs as suggested in the first part.

Authors and visitors should be able to perform all operations via a standard web browser. This should be implemented by writing a program that is called by a web server, connects to your Oracle database, then calls appropriate stored procedures that you have defined in Oracle (or maybe sends queries), and finally returns the results as a web page. You can implement the interface in several different ways:

- You can use the existing web server setup on `pdclab` or on `utopia` to connect to the Oracle installation on `pdclab`, or you could also use your own Oracle installation and web server. You may also use other database systems such as SQL Server or Sybase that you have access to, but please ask first. We can only give real support for the Oracle setup at Poly, of course.
- Additional technical information will be available at <http://cis.poly.edu/cs308/projects/>. You can use several different languages and tools to make your application web-accessible. Please contact Hao Yan (email: [hyan@cis.poly.edu](mailto:hyan@cis.poly.edu)) for technical questions.
- One option is to use Java and JDBC to connect to Oracle. If you know Java, this may be the easiest solution. See the sample program on the above mentioned web page.
- Another option is to use the proC precompiler for Oracle Embedded SQL. This is essentially C/C++ with additional SQL calls. If you know C/C++ well, then this is a reasonable option. Note however that proC has some pitfalls and takes some efforts to use, so this may not be the easiest approach. The Oracle8 book by Sunderraman contains material on proC, and also the Online Guide at Stanford and other pages pointed to from the course page. (The Oracle9i version of the Sunderraman book does not contain the chapter but you can download it from <http://tinman.cs.gsu.edu/~raj/books/oracle9-primer.html>.)
- Another good option is to use PHP. If you do not know Java and do not want to deal with proC, then this might be the best approach. See the sample program on the above page. You should be able to pick up PHP fairly easily. If you know Perl or Python, this might also be a good approach.
- We cannot support ASP, JSP, or ODBC on campus, but you may use your own installation for this. For C++ programmers, the Oracle installation on campus is set up for ODBC, but you would need to install your own windows-based ODBC client software and this would only work while your client is within the Poly network. So maybe better to use your own database setup in this case.

Every student will be expected to demo his project to one of the TAs at the end of the semester. If you use your own installation, make sure you can access this during the demo. One popular choice is to use Oracle and a local web server and a browser on your laptop, in which case you bring your laptop to the demo. Grading will be done on the entire project based on what features are supported, how convenient the system is for the user, your project description and documentation, and the appropriateness of your design in terms of overall architecture and use of the database system.

Describe and document your design. Log some sessions with your system. Bring your description (documentation), the logs, and the source code in hardcopy to the demo. The documentation should consist of 10-20 pages of carefully written text describing and justifying your design and the decisions you made during the implementation, and describing how a user should use your system. Note that your documentation and other materials should cover both Projects 1 and 2, so you should modify and extend your materials from the first project appropriately.