Fall, 2004 CIS 550

Database and Information Systems
Project groups and choice due via email, 10/19 at 1:30 PM
Course Project due 12/9 at 1:30 PM

1 Overview

The course project will be assigned to teams of 3 students. There are two options for the project: one more applications-oriented (building upon database technology), the other more systems-oriented (working under the covers).

2 PMail

The first option is to build a Web-based email system, along the lines of Microsoft's Hotmail or Google's Gmail. You will manage multiple users' email folders, allow them to send and receive email, and allow them to search for email by keyword. This will involve a number of components:

- Interfacing with Java routines (we suggest using Apache Tomcat, which will be offered on departmental machines) that send and receive email from a POP email account (or set of accounts). We will make use of a free email service provided by hotpop.com as the actual host, so you will not put your normal email account at risk.
- Storing email messages and other information in a relational database, e.g., Oracle or MySQL. The department hosts both Oracle 8i and MySQL.
- When displaying emails, using XML and style sheets to format their appearance.
- Ensuring that appropriate precautions are taken to preserve security of different users' accounts. For instance, it should not be possible to "snoop" on the information sent from the browser to the PMail server; nor should it be possible to access an unauthorized account simply by changing a URL.
- Building an "inverted index" of keywords and message IDs, so a user can query for certain key words and the system will return a list of the appropriate emails. For extra credit, ranking and "stemming" may be added.

Figure 1 shows an overview of the major system components. We describe them individually in more detail in the following sections.

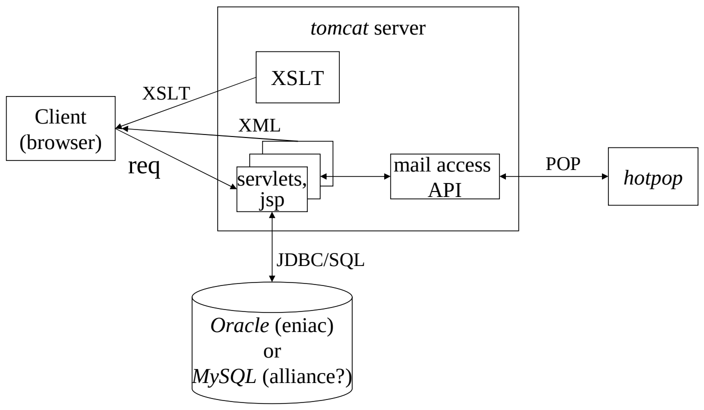

Figure 1: Basic system architecture

2.1 Tomcat

Tomcat is the web server you will use for the system. Tomcat allows you to generate HTML pages dynamically using JSP pages, which are HTML pages containing embedded calls to Java servlets. The servlets are where you will write the code which sends and retrieves messages via hotpop, and stores and manages emails, contacts, etc in the DBMS.

If you wish to install it on a machine of your own, Tomcat can be downloaded from the Apache Jakarta web site at `http://jakarta.apache.org/tomcat`. We will provide information about how to access Tomcat on the departmental servers.

2.2 XML and XSLT

The Web pages that are part of your site should all be in XML, with two or more XSLT stylesheets that convert them into HTML. This means that your JSP pages and servlets should both return XML data, as opposed to HTML data. Every user will choose a default stylesheet that will control the appearance of the Web site, by defining transformations from the XML into HTML.

You can assume that every user will have a Web browser capable of rendering XML with XSLT. This includes recent versions of Internet Explorer, Netscape, Mozilla, and (we believe) Safari. For extra credit, you may consider the integration of Apache Cocoon, a “middleware” that allows you to convert from XML to HTML on the server side (as opposed to client-side).

2.3 HotPop

We have arbitrarily chosen as our standard email server `www.hotpop.com`, because the email accounts are free and can be accessed via the POP3 protocol. Your system must work with HotPop. Each student should go to the HotPop web site and sign up for a free email account to be used for testing their project. For extra credit, you can also have your system work with other email servers, such as SEAS. See extra credit section for more details.

2.4 Javamail

Javamail is a Java library you will use to talk to the HotPop server. Javamail can be downloaded for free from Sun's web site. Since the Javamail API is somewhat cumbersome, we have written a small wrapper library simplifying things for your purposes. The wrapper is called HotPopLite and can be downloaded from the class web site. It consists of two main source files: `MessageLite.java`, which represents message instances, and `HotPopLite.java`, which has methods to send and retrieve mail via hotpop. The distribution also includes two command-line sample applications to demonstrate usage, `GetMail.java` and `SendMail.java`.

2.5 DBMS

You will use either Oracle or MySQL to store messages, users, contacts, folders, and so forth. You will also implement the inverted keyword index using the DBMS (no custom text search libraries or making use of special text indices, please, as that defeats the purpose!). You will need to design the database schema and consider what indices to create. You already have access to Oracle on eniac, and we will supply additional information about connecting to it with JDBC.

2.6 Extra Credit

There are many opportunities to add extra capabilities to your system, such as:

- Word stemming (looking for different forms of a keyword, e.g., run vs. ran vs. running), which may make use of Porter's Algorithm,
- ranking of answers (as with Google),
- shared folders,
- different methods of browsing emails,
- message threads,
- (your idea here).

Up to 20 points of extra credit will be awarded, at the discretion of the graders, for implementing extra features like these.

2.7 Deliverables

For this project, you will ultimately need to submit:

- An 8-10 page *writeup* describing your system. The writeup should explain (briefly) how each of the required aspects of the project was implemented and (in more detail) the database design. Also describe here any features you implemented for extra credit.
- All *source code.*
- A confidential email directly to Zack describing (1) your contributions to the group, (2) other your group members' contributions, (3) your assessment of group dynamics. This will be cross-correlated with our sense of what work everyone did (based on discussions during your demo of the project), as well as the emails from your group partners. The individual contributions *will* be given significant weight, in order to ensure everyone is contributing.

The project should be submitted via email to the class TA, `tjgreen@cis.upenn.edu`. Additionally, as a group you will need to be prepared to interactively *demonstrate* your system to the graders. Your demo will highlight the interesting aspects of the system, and we will ask about details. (The graders will want to test your system themselves, using their own email accounts.) Please propose a time for the demonstration when you submit your writeup and code.

3 P2P Database Synchronization

An alternative project, recommended for students who want to know more about what goes on "under the covers" of a data management system, is to build a peer-to-peer system for "synchronizing" multiple copies of a database.

If a database gets copied to multiple sites, each of which independently makes changes to it, how can we quickly and effectively "sync up" the different databases so they all look the same?

One solution, which doesn't require everyone to sync at the same time, is to use a peer-to-peer system substrate, such as Pastry or Chord, to distribute the data across a network – where each peer gets part of every table. Individuals download a copy of the entire table, make changes, and then send these changes out to the different peers in the P2P network. Each peer synchronizes part of the database and sends its data back to the individual.

This project is much more open-ended than option 1, and the first step would be to learn the Pastry Java APIs (which take care of most details of peer-to-peer distribution), then to build a database interface over Pastry. This database might be relational or it could even be an XML database.