COMP 6302 Internet Application/Java Programming,
The Final Project Report
Ivan Rabak

The Web Community
For
Students of Geophysics

December 7, 2004
Introduction

It took more than two months to conceive, propose and complete this project. In this report I wanted to discuss mainly the problems that still remained half-solved and how I would eventually go about to solve them and the additional Java and networking chapters I have learned about but now is the time to write up what I have done to go on line with the Web Community and provide some information on its structure, feel & look and explanation of the backstage processes of the project. Bells & whistles have been promised but just few of the features delivered. On the other hand the basic functionality of a message board type of application is running backed up with a broad range of what-if-user-enters-this-instead-of-that exceptions handling routines.

Importance of the Project

I found myself many times looking for broad topics in geophysics on the Internet focusing on student needs but unless the (similar) information was posted on a major (geophysical) meeting web site (like AGU) or a university it was almost impossible to find any additional link or useful pointers how to continue my search. By introducing students of geophysics to a Web community of their own I hope to show them a way to better communicate their ideas to anyone interested but mainly other students. Since many of them are members of other more general or specific communities I am certain that they will recognize the opportunity and start contributing material regarding their research, evaluating other students’ work or said simply share needed information. Giving and receiving feedback from other students may be less stressful than doing so exclusively with their professors.

Basic Strategy

The Community is a client server application. It supports up to 500 simultaneous connections. This number is arbitrary and can be reasonably expanded having in mind possible infrastructure restrictions. The server runs Apache Tomcat 5.28 providing an
environment to run Java servlets in addition to a mySQL database. The clients communicate requests over a standard Web browser (Netscape is preferred though: discussed later). This architecture allows focusing exclusively on the server side development since HTML provides enough functionality for a message board type of applications at the client side.

Request – Response, Who & What

Two main types of users are supported: anyone and the registered user. There are also the administrator privileges but they are provided through a direct access to the database, thus not explicitly given by the Community program. Anyone can see the entire content of the site but only the registered user can post content. The content is made out of a top hierarchy element: the thread that further may contain users’ replies. I refer to both as posts. Threads are chaptered by seven (7) different topics, the other top hierarchy elements. The topics are somewhat arbitrary chosen research topics in Geophysics: GPS, Wave propagation, Earthquake seismology, Applied Geophysics, Inverse problems, Hazards and Rest (Figure 1).

Figure 1. The Web Community home page. Listed are some basic subcategories in geophysics. The use of frames helped out in designing the homepage layout.
When registering, a user is asked to fill out form as shown in Figure 2. The form is accompanied by a suite of exception handling procedures since the user must fill out all the fields, user name and password must be unique, reentered password must equal the password field (see Figure 3).

![Figure 2. The user registration form. Initially Invite me check box was supposed to allow the user to be on the email-me list and be invited by other users when they create a new thread and choose to invite other users. See text for problems related to the dropped option.](image)

Validity of the registration form is entirely done on the server side. This isn’t really a common practice. Had I known JSP earlier I wouldn’t have had servlets generate all the (HTML) response. Including JavaScript code that could handle the form validation before submission via servlet response is extremely hard to troubleshoot & maintain. In addition, JavaScript does not execute the same in different browsers averting me further from the common practice in this case. Still I do have some JavaScript written to achieve pop-ups and button clicks but this is where the Netscape compatibility requirement jumps in.

Figure 4 similarly shows a suite of windows attending the login window.
Figure 3. A suite of screen shots showing a variety of server responses after the registration form gets submitted.

Figure 4. A similar suite of screen shots displayed during (in)correct login for registered users.
Figure 5 shows what anyone sees the content of Wave propagation topic. Circled in red is a placeholder for a “Post reply” button reserved for the registered user.

Clicking Expand button expands the thread into replies. Clicking now Collapse button takes anyone back to the thread view.

Numbers in the top left field of each post are generated chronologically, when inserted into the database. The yellow and orange color scheme helps to separate threads from corresponding replies. Note the awkward date-time stamp format. In general Java allows many complex tasks to be dealt with an ease using numerous APIs but I find in to much work when a simple type conversion or string formatting is required in comparison to other programming languages more popular when solving numerical problems (i.e. C, Fortran, MatLab).
Figure 6 shows that in addition to anyone privileges, the registered user can create a new thread and/or reply to an existing thread/reply. New replies are sequentially placed at the bottom of the corresponding thread. New threads are placed at the bottom of the corresponding topic.

When posting a reply/thread (Figure 7) a part of the subject is automatically generated based on the chosen post. The user can entirely rewrite the subject line. The user receives a confirmation window if the post is successfully added to the database.

There is a number of servlets processing user’s requests. Some servlets are multifunctional since they can process a different number of parameters. HTML code on the client side is almost entirely generated via servlets.
Setting up the Infrastructure

The complexity of the project required an early start so while conceiving and writing the proposal I did not now how to solve many anticipated major problems such as: concurrent database access and session tracking among others. Since I knew bits and pieces of mySQL and Java I thought that just a few simple and basic features of a typical message board application would not suffice for a class project. Very soon, though it proved that there were many smaller unanticipated “major problems” too in addition to
that I later realized I had to handle various exceptions thrown by passing “wrong” data from the user.

Downloading and installing a web server and learning early how to run servlets just to get a feel of what they could do paired with mySQL was another thing.

The MySQL Database

The relationship between the data types is quite simple but it is still important to design an appropriate model as shown in Figure 8.

![Figure 8. The entities of the database. Underlined attributes are (multiattribute) primary/foreign keys. For simplicity, Post table is not normalized.](image-url)
Post table should be normalized but for simplicity I decided to keep it as shown. This simple & awkward design simplifies and reduces the total number of database query calls. Important: mySQL 4.x does not support nested queries.

Figure 9 shows the database (webdb) and User table using a publicly available GUI.

![Figure 9. The database (webdb). Here, the administrator has the full access to the database. Some attributes are not used (the screen shot is from an early stage of development so includes some later not allowed values such as ‘0’ for InviteMe attribute and zero for userName/password).](image)

In Java, the appropriate mySQL driver is simply loaded by:

```java
Class.forName("com.mysql.jdbc.Driver").newInstance();
```

On Missing (Proposed) Functionality

So, you cannot invite other users to see your new thread. I don’t have a decent excuse for this. One of the homework involved writing Java code to use Telnet to send emails so basically I just had to set it in the project code and loop it for all the “interested” users. Still, it would require handling exceptions (i.e. failed connection attempts, unsent emails, transfer interruptions …) and I would have to use the UofM postoffice as the mail server but for testing purposes only.
All the content web pages are created dynamically (via servlets querying the database) so pointing Goggle, for example, to search users’ posts would be futile. Using queries to match a string for content in the database is not difficult but organizing the result seems quite so. Perhaps there is some Java code out on the Web but I don’t know how easy it would fit in the project code.

Pool of Connections

I think the most interesting thing is using the idea of a pool of database connections that considerably improves the concurrency aspect of the project since creating a connection takes approx. 1-2 sec. I found the information on the web and substantially reworked the very good examples offered on the web page


There is a single instance class, the manager that is referenced form all the servlets’ init method. The manager takes care of the connections in a form of vector. Servlets (within doPost/Get methods) check connections in/out from the vector. If a connection to be checked out (manager class) is no longer valid a new connection is created. A connection is checked in as soon as the servlet using it is done querying the database.

Final Note

I enjoyed very much working on the project. In total, over the course of the class I learned so much but above all one unsettling fact about Java (or is object programming?). I only learn one thing through one/more examples at a time. Number and possibility of APIs and reusable classes is too vast for an earth scientist. I guess many complex things can be solved fast & easy (like session tracking here where I just had to place a user as an attribute in a session) but this isn’t like simple C or Fortran programming I am accustomed to where you tell the computer what to do next and where a few examples can give you a pretty good idea of what you can do.
References

Marty Hall. Core Web Programming (especially including the web page where the updated classes are).

David Kroenke. Database Processing, Fundamentals, Design and Implementation

Ideas @: http://philip.greenspun.com/seia/

Java Reference @: http://java.sun.com/