A WEB-BASED GRADUATE APPLICATION DATABASE SYSTEM

Lin Han and Xining Li

Department of Computer Science
Lakehead University
Thunder Bay, ON, Canada P7B 5E1

ABSTRACT

This paper discusses the analysis, design, and implementation of a Web based database system capable of handling on-line graduate study applications. The aim of the system is to provide a simple, user-friendly interface and a secure on-line database for submitting, retrieving, and sharing application data through Internet, hence to speed up the application process. We chose to base on the Web for many reasons including user familiarity, broad availability, low distribution cost, and minimal development time. By means of experimenting the system, we describe some design challenges of a Web based database application, such as Authentication, Access control, and Security issues, and how we choose to address these challenges and build the system efficiently and effectively.

1. INTRODUCTION

Most Web applications are built on the Three-Tier Model which facilitates the perceived need to separate business logic from the GUI and the backend database. According to the model, three separate well-defined processes, or modules, run on different layers:

• Client tier: End user application, normally, Web Browser.
• Middle tier: Mediating information servers, that run with Web server and that actually process the data request.
• Resource tier: Information resources that stores and manipulate the data at the backend.

The explosive expansion of the use of Web browser and increasing demand for supplying interactive and dynamic information through the internet rather than the static HTML page made the three-tier applications popular as well as practical. For transaction-oriented applications, such as e-com or on line data processing, middleware is usually required between the network servers and the back-end system to ensure proper interoperability. Considerable security challenges exist within each stage of this model.

CGI (Common Gateway Interface)[1], as the first solution to deploy dynamic Web application, is one of the most popular tools and is supported by almost all Web servers. CGI defines the specification for transferring information between a Web server and information resources. A CGI program accepts parameters from a HTTP request passed by the Web server, then generates and returns a HTML page as if it was a pre-stored one. Although with its simplicity, the biggest drawback of CGI approach is that the Web server needs to throw a separate CGI process for processing each request received. This is time consuming and expensive in terms of server's memory and other system resources.

Java Servlet[2], is a Java program (class) that runs on a Java enabled Web server and resembles a conventional CGI program. However, Servlet is designed to overcome the drawback of CGI and is an increasingly attractive alternative to CGI program. Unlike a CGI program, a Servlet is persistent once it is started. It remains in memory and can therefore be used to handle multiple requests. In general, a Servlet is faster and cleaner than a corresponding CGI script. Although a Servlet runs in the same address space as the Web server does, it is safer than CGI because of the protection mechanism obtained from the Java virtual machine (Digit signature, Sandbox). Servlets can be embedded in many different servers because the servlet API, which programmers use to write servlets, assumes nothing about the server's environment or protocol.

Module Integration is another popular approach developed to ease the program complexity and increase the performance of Web applications. As the first integration project and representation of this technology, Mod_perl[3] brings together the full power of the Perl programming language and the Apache Web server. Mod_perl is composed by several pieces of software which links the Perl runtime library into the server and provides an object- oriented Perl interface to the server's C language API. One part of the software is designed to be compiled and linked together with Apache and Perl. The remainders are Perl code that provide the object-oriented interface to the Perl-enabled web server. The primary advantages of Mod_perl are its power and speed, because programmers have full access to the inner-workings of the web server (authentication, response generation and logging, *etc*) and can intervene at any stage

Thanks to the Natural Science and Engineering Research Council for supporting this research.

of request-processing. There is also very little run-time overhead since the persistent Perl interpreter embedded in the server avoids the overhead of starting an external interpreter.

2. OVERVIEW OF THE GAOL SYSTEM

There are already many on-line systems available aiming for speed up the education application process. The work on GAOL (Graduate Application On Line) system for the Computer Science Department of Lakehead University was motivated by the fact that many of the systems mainly target collecting information for the institute, while supply few feedback or aid for the applicants.

The objectives of the GAOL system are:

• To build an Integrated Application Environment (IAE) for different class of users, including a centralized database, unified GUI and standard processing sequence, hence to efficiently exchange the application data.

• To allow an applicant tracing his/her own application status according to a pre-defined processing sequences.

The GAOL project is composed of two major components: a dedicated library that encapsulates the database access details of user and application data, and a collection of programs that drive the generation of output of HTML pages.

In a typical session, the client's browser sends a request to the Web server, which passes it to the processing program. For those pages that include dynamic content, such as a registered user's account, the processing program calls the appropriate library for accessing data from the database and delivers formatted data to the HTML page generator. Finally, the HTML page generator assembles the complete HTML page, and ships it to the client browser. The architecture of GAOL is given in Fig. 1.

From user's perspective, the GAOL is designed for four categories of users:

An applicant can submit application information on line by filling in a pre-designed form. Then the collected data will be stored into the centralized database. In addition, an applicant can check his/her latest application status periodically.

A supporting staff is responsible for the routine work of handling applications, such as sending out application packages, updating applicant's status, responding to special queries, and generating statistics and reports.

A faculty member is mainly interested in the data provided by applicants. By reviewing the applicant's information, such as previous education record or academic background, department graduate committee can make a quick admission decision before the paper-based documents circulate to their desk.

The system administrator will focus on manipulation user accounts, such as creating new user account, modifying access control level, resetting user password, or deleting useless database records, *etc*.

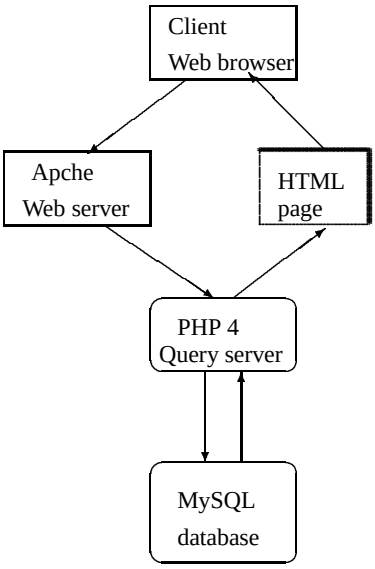

Fig. 1. GAOL Architecture

3. DESIGN AND IMPLEMENTATION

For the system to be effective, the development must be configured to implement certain policies and guidelines. Some of these are common and applicable to any project, whereas some differ from project to project. In this section, we are going to discuss some interesting issues in the design and implementation of the GAOL system.

3.1. Development Platform and Tools

GAOL was mainly developed on MySQL [4] and PHP [5]. Both of them are open source projects, of course freeware, distributed under GNU general public license. Together with the most popular Web server Apche, PHP and MySQL become more and more widely accepted by Web application developers and the population of Web sites based on PHP has already overweight Microsoft ASP & IIS.

MySQL is an efficient, multi-threaded, multi-user, and robust SQL database server. Features supplied by MySQL are far more enough for manipulating our centralized application dataset.

As a server-side HTML-embedded scripting language, PHP differs from CGI in the sense that a CGI script usually involves using other programming languages, such as C or Perl, to generate and output HTML scripts, whereas a PHP script is embedded inside of an HTML page. In other words, a piece of PHP code is enclosed in a pair of special

tags - *start* and *end* - that allows control to jump into and out of PHP mode. Therefore, a programmer can configure a Web server to process all HTML files incorporated with a PHP server for handling PHP code and generating dynamic content. Normally, the PHP server will be installed as a Web server module (like Mod_perl) for the reason of performance efficiency. Comparing with other scripting languages, such as Perl, PHP is specially designed for Web scripting with less confusing and stricter grammar which is perfect for a programmer without Perl background. In addition, the most significant feature of PHP which makes it so popular is its power of supporting a wide range of databases.

3.2. User Interface and Interactivity

User Interface is a crucial part of any Web-based application. It enforces special effect on the degree of the interactivity a Web-based application could bring. We use several different technologies, such as Java script and Cascading Style Sheets supported by current Web browsers, in an attempt to break the constraint brought by the limited capability of HTML.

The GAOL GUI mainly depends on the HTML Form to provide a set of standard interactive components which a Web browser is responsible for presenting in order to share a same or similar UI look and feel cross different platforms. To improve our control on the way how a Web browser presents the HTML page cross platforms, we resort to W3C CSS (Cascading Style Sheets) specification to guarantee that the HTML page is exactly provided in the way we want by different browser vendors.

To decrease the server-side computation workload and improve the UI responsiveness, we deploy client-side script language, Javascript, for checking the common input availability, or popping up confirming or warning windows to prevent side effects from unintentional operations.

Although we could use PHP script to generate all the HTML pages within the GAOL integrated application environment, we specifically avoid doing so because we believe, regarding to some mostly static content (such as User manual), it would be better to provide them in a plain HTML page for easier maintaining work.

3.3. Database and Performance

The analysis of data sets is fundamental to this on line database system. According to the user requirement analysis, we specify eight possible stages during an application sequence:

Submitted: Applicant has registered into GAOL and supplied the on-line information.

Received: Application has been received and further contact will continue.

Completed: All necessary documents have received in the department.

Under-review: Application is under review by the department graduate study committee.

Admitted: Applicant has been admitted and a formal admission has been send out.

Rejected: Application has been rejected.

Deferred: Application has been deferred to next academic year.

Declined: Application has been cancelled or admission has been rejected by the applicant.

For each registered applicant, there is an associated status history table which shows the application processing steps and timestamps. By tracking this table on line, an applicant can have an update picture about his/her application progress, and hence to avoid most of the common queries and replies between the applicant and department.

Bandwidth is important issue for a high performance Web-based application. Meanwhile considering the fact that applicants connect to our department from many places of the world, we can not always assume a campus kind of fast internet connection. In order to improve the data exchange efficiency, our solution is to break down and group a big chunk of data into several small pieces, the user can optionally choose exchange an entire group or only parts of them at each operation. For example, the application form has been divided into seven categories, such as personal information, contact address and academic background, *etc*. Rather than reflex this division in the database splitting, we put the responsibility onto the application software aiming for the database simplicity. A set of checkboxes have been added to the main application page where each corresponds to a set of information. By click one or several checkboxes, we leave users the flexibility to choose retrieving or updating data according to their Internet access speed and performance.

3.4. Authentication and Access Control

Authentication is the process that establishes the identity of a user. Authentication is required to access the system Web pages, except those static ones for general instruction purpose. To register a user account, the user must supply at least an available email address and an unoccupied username/password pair. By clicking the submission, a confirmation email will be automatically send out, and the user account will be activated. Designing in this way is aiming to hinder the visitor who may not already for submitting application to the department. Before being able to access

the protected services, a user must go through a login interface, and after successful password validation, the user will be directed to the group homepage with respect to the user's access level. The following PHP code illustrates the authentication process.

```
if (isset($actionflag) && $actionflag=="login" ) {
 if ( user_login($entered_login,$entered_password)) {
 // successful username/password validation
   $loginlevel = user_getuserlevel();
   if ($loginlevel == 1) { // Access level 1 - applicant
    // redirect to applicant homepage
    header("Location:http://$site_url/$site_dir/$ApplicantPage");
   }
   elseif ($loginlevel == 2 ) { // Access level 2 - faculty
    header("Location:http://$site_url/$site_dir/$FacultyPage");
   }
   elseif ($loginlevel == 3 ) { // Access level 3 - staff
    header("Location:http://$site_url/$site_dir/$StaffPage");
   }
   elseif ($loginlevel == 4) { // Access level 4 - sysadm
    header("Location:http://$site_url/$site_dir/$SysadmPage");
   }
   else {
    // otherwise, send user back to system main page
    header("Location:http://$site_url/$site_dir/$cfgIndexPage");
   }
 }
}
```

Access control is a process by which use of system resources is regulated according to a security policy and is permitted by only authorized entities. Traditional access control models are broadly categorized as Discretionary Access Control(DAC) and Mandatory Access Control(MAC) models. New models such as Role-based Access Control (RBAC) or Task-based Access Control (TBAC) models have been proposed to address the security requirement of a wider range applications [7].

GAOL adopts a simple access control strategy by supporting four user access levels identifying the privileges associated with various roles in the system.

Access level 4 corresponds to the role of the GAOL system administrator. Individuals in these roles have maintenance privileges, are allowed to create new user accounts as well as modify the account information, such as user access level. We also supply system administrator the functionality to reset a user's password in case the user has difficulty to do it for different reasons.

Access level 3 is used by supporting staff who can review application data, and update their status according to the actual processing procedure. This group of user has the richest functionality set because of their closest role to the application handling progress.

Faculty members are assigned Access level 2 which allow them to review all application data, such as applicants, their academic background, or application status. User accounts of Access level 2 and 3 must be created by the system administrator, while the Access level 4 is the default level when the system is installed.

Finally, applicants belong to Access level 1 which places tight controls on what they may do. User accounts are created automatically when an applicant is registered to the system first time.

3.5. Session Control

The World Wide Web facilitates Web applications on the Internet via its underlying hypertext transport protocol, which carries all interactions between a Web server and a browser. Since the HTTP *stateless* nature, this characteristic makes every server interaction independent of all other interactions, so there is no notion of persistence. A Web server responds to an HTTP request either by returning an HTML page directly or by triggering an external application, such as CGI or server API, to handle such request. Once the request has been satisfied, the transaction is complete and the connection closes. In other words, it does not support continuity for browser-server interaction between successive user visits. Without a concept of session in HTTP, users are strangers to a website every time they access the Web server.

Regardless of the programming complexity involved, there are a couple of ways being invented to get around the HTTP stateless problem by maintaining session information or passing state information back and forth to the client, such as URL query string and HTML Hidden Field. Another solution is the experimental HTTP header called cookie [6] suggested by Netscape but right now supported by most of web browsers. They contain character strings encoding relevant information about the user. Whenever the browser received a HTTP header claiming to set a cookie during a visit to a cookie-enabled website, the Web browser will keep cookie's value into a file or RAM. Thereafter, browser will automatically attach the cookie's value with each client's subsequent request when the user visit the same website, hence get a chance to maintain a persistent state between server-browser communication. Cookies serve many purposes on the Web application, such as configuring user display mode, maintaining shopping cart selection, *etc*.

GAOL uses cookie to store user identification data. After a successful user authentication during the login process, the system will inform the browser to set up a cookie according to user's identification. In the case of GAOL, this cookie includes the username, and an id_hash value for security reason. Meanwhile the cookie's lifetime is set to be three hours as the session length which is far more enough for most of the operation situation.

During subsequent user visits to the GAOL services, this cookie will be automatically attached with each following

HTTP request. The system will acquire the user identification from the cookie, identify the user's access level, then decide if the service can be provided. If the service is rejected, the system will sent the user back to the GAOL main page.

3.6. Security

Last but certainly not least among the implementation issues is the issue of security. Needless to say, security issues are crucial as Internet becomes more prevalent. Security, in the context of Web application, includes protecting a Web site, restricting access to a Web site, and the degree of safety of data transfer between the server and the client, *etc*.

Considering access restriction to the GAOL site, the system provides a Role-based verification mechanism. Basically, separating Web pages supply distinct system services to the user. However, there is no way to prevent a user from typing in a page's URL, and try to enter the page no matter intentionally or not. This is the reason why the system need to verify the user's identification at the beginning of a session. If the user is unauthenticated or not holding a required access level, the service will be denied.

In order to provide a flexible way to let different group of users to share the same service, we defined two variables for this purpose. One is an array variable ($requiredGroup) which include all access level values allowed to access this page, another is a singular value ($minLevel) which represents a minimum access level value on demand. Values above the minimum value will automatically get the access right no matter it belong to the required access group or not. Therefore, each page can set its own access restriction policy by simply assign these two variables. The actual verification work is done by a security checking function with respect to these setting.

Using a cookie to store a user identification token allows a Web site to remember visitors between sessions. However, it is possible that a user might try to modify his/her cookie in order to log in as another user. Generally speaking, when Web servers use cookies to identify users and their status, there are three types of security threats to cookies [8]: network threats, end-system threats, and cookie-harvesting threats.

GAOL is mainly concerning about the end-system threats. Once the cookie is stored in browser's side in the form of plain text, its content can be trivially altered by users and easily copied from one computer to another computer, without notification of the user whose computer the cookie was originally stored. The ability to alter and copy cookies lets attackers easily forge cookies' content and impersonate other users.

Therefore, to deploy cookie for system authentication, we need the concepts of *confidentiality* and *integrity*. Confidentiality is the property that information is not made available or disclosed to unauthorized individuals, whereas integrity is the property that information has not been modified or destroyed in an unauthorized manner.

In the case of GAOL, we use cryptographic technologies to enforce cookies' confidentiality and use an integrity verification function to check the cookie's owner and protects our system against unauthorized modification of the cookie.

To achieve this, the GAOL deploys the secret-key cryptography by using a message digest algorithm - MD5. After a user having successfully logged in, the system generates a message digest form the username and a system secret key, then puts the signature (id_hash) into the cookie together with the character string of user name, referring the following cookie example. When the user makes subsequent visits to the system, the browser sends the secure cookies to the system. GAOL verifies the signature in this secure cookie using the same cookie-issuing policy in the authentication stage. A successful integrity verification means a cookies has not been altered.

An Example of Cookie Content
user_name
jmark
lakeheadu.ca/
0
2575189120
29389530
2178271520
29389505
*
id_hash
3674367e34ac079880885e68b4dc7813
lakeheadu.ca/
0
2575189120
29389530 2178271520
29389505
*

4. CONCLUSION

Applications and queries for graduate study from internet have been increased dramatically in recent years. Rather than the traditional way of communication via tedious individual emails, or non-interactive departmental web page that only provides the basic application guide, we developed a Web-based graduate application system GAOL. The goal of the system is to provide a simple, user-friendly interface and a secure on-line database for submitting, retrieving, and sharing data on internet, hence to speed up the application process.

The GAOL system integrates the World Wide Web and distributed computing technologies to allow users sharing a centralized database, processing applications via standard

Web browsers. Authentication capability is provided by the username and password verification mechanism. Access restriction to the system is enforced by implementing a role-based access control policy. A user session (stateful service) is based on acquiring the cookie value which is protected by secret-key cryptography.

GAOL is currently under its Alpha test and can be accessed at *http://www.cs.lakeheadu.ca/~gads/*. Ongoing improvements include adding a XML message layer to expand GAOL application layer to be independent of the underlying database API.

Based on our experience, we believe that the combination of PHP and MySQL under three-tier model is a good, practical environment for developing multi-user distributed applications that exploit WWW infrastructure.

5. REFERENCES

[1] CGI Specification, NCSA for Supercomputing Applications, Champaign, Illinois. http://hoohoo.ncsa.uiuc.edu/cgi

[2] C. Bloch and S. Bodoff, The JavaTM Tutorial - Servlet, http://java.sun.com/docs/books/tutorial/servlets/

[3] S. Bekman, Mod Perl Developer's Mini Guide, http://apache.gns.com.br/perl/guide/all.html

[4] MySQL Manual Online, http://www.mysql.com/documentation/mysql/

[5] PHP official site. http://www.php.net/manual/

[6] Netscape's cookie specification. Netscape Communications Corporation http://www.netscape.com/newsref/std/cookie_spec.html

[7] J. Joshi, W. Aref, A. Ghafoor, and E. Spafford, Security Models for Web-based Applications, CACM Vol. 44 No. 2, Feb 2001, pp. 38-44

[8] J. Park and R. Sandhu, Secure Cookies on the Web, IEEE Internet Computing, Los Alamitos, CA. July 2000, pp. 36-43,