2003-09

An open-source and Java-technologies approach to Web applications

Siripala, Seksit.

Monterey, California. Naval Postgraduate School

NAVAL
POSTGRADUATE
SCHOOL

MONTEREY, CALIFORNIA

THESIS

AN OPEN–SOURCE AND JAVA-TECHNOLOGIES APPROACH TO WEB APPLICATIONS

by

Seksit Siripala

September 2003

Thesis Advisor:	Neil C. Rowe
Second Reader:	Gary L. Kreeger

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2003	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: An Open-source and Java-Technologies Approach to Web Applications.			**5. FUNDING NUMBERS**
6. AUTHOR(S) Seksit Siripala			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT ***(maximum 200 words)***

Web applications have become a critical component of the global information infrastructure. In government organizations, proprietary software is currently being replaced by open-source. This thesis explores using open-source and Java technologies to implement Web applications. A prototype of the framework was implemented for a military information site. Implementation was straightforward and performance of the prototype was excellent, demonstrating advantages in terms of reliability, portability, maintainability, and economy.

14. SUBJECT TERMS web application, open-source, Java servlets, Java Server Pages (JSPs), Linux, Web server, Structured Query Language (SQL), Java Database Connectivity (JDBC)			**15. NUMBER OF PAGES** 141
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

AN OPEN-SOURCE AND JAVA-TECHNOLOGIES APPROACH TO WEB APPLICATIONS

Seksit Siripala
Captain, Royal Thai Army
B.S., Chulachomklao Royal Military Academy, Thailand, 1998

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 2003

Author: Seksit Siripala

Approved by: Neil C. Rowe
Thesis Advisor

Gary L. Kreeger
Second Reader

Peter J. Denning
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Web applications have become a critical component of the global information infrastructure. In government organizations, proprietary software is currently being replaced by open-source. This thesis explores using open-source and Java technologies to implement Web applications. A prototype of the framework was implemented for a military information site. Implementation was straightforward and performance of the prototype was excellent, demonstrating advantages in terms of reliability, portability, maintainability, and economy.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to dedicate this thesis, the result of two years graduate study at the Naval Postgraduate School, to my parents who have encouraged me through the process. I also appreciate Professor Neil C. Rowe for his great advice. Finally, I want to thank my senior officers: Chaiporn Dechjarorn, Phuwadol Udomsilp, and Thoetsak Jaiaree for their assistance, and suggestions while preparing the research project.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

Web technologies, programs using the World Wide Web "http" protocol on the Internet, have become an important part of information communication. Not only business but also government organizations can benefit from Web technologies. A user can use web browsers to search the Internet for data, send e-mail, or make purchases electronically. In military organizations, Web technologies manage information for many purposes. Today a key technology that allows developers to make Web applications is server-side programming to generate a dynamic Web page.

Currently most government organizations use Web technologies based on proprietary software, necessitating a big budget for software licenses and support. The alternative is integrating non-proprietary software including open-source software. Open-source and Java technologies have grown rapidly in popularity during the past decade. The performance of some open-source software meets or betters the performance of much proprietary software.

B. OBJECTIVES

The goal of this thesis is to design, implement and analyze a prototype Web-based Personal Information Management System for the Supreme Command Headquarters of Thailand using an open-source and Java-technologies approach. The research will address the following questions.

- Which open-source technologies can best implement World Wide Web applications?
- What Web application systems can be implemented using open-source technologies?
- What are the performances of Web application systems using open-source technologies?
- How can the performances of Web applications be improved?

C. SCOPE OF THIS THESIS

The scope of this thesis is the design and implementation of a prototype for a Web-based application in the open-source approach. This thesis will propose specific open-source software and methods that can be used to implement on Web-based application. The application will focus on personal-information management for military organizations. The prototype will be an entire Web application system, including an operating system using Linux, a Web server using Apache Tomcat, a database server using MySQL, and server-side programming using Java Servlets and Java Server Pages. The prototype will manage simulated personal information in the Thai military organization.

D. THESIS ORGANIZATION

The following is an outline of this thesis.

- Chapter I. Introduction
- Chapter II. Open-Source Overview
- Chapter III. Introduction to Server-Side Technologies
- Chapter IV. Introduction to Web Application Development
- Chapter V. Database Management
- Chapter VI. Design and Implementation
- Chapter VII. Conclusions

II. OPEN-SOURCE OVERVIEW

A. OPEN-SOURCE SOFTWARE

The term “open-source” has been widely used to describe a software development process that embodies the idea of building software within a cooperating community through the Internet. The basic requirement for this term is the availability of source code, which allows everyone to modify and redistribute it. Beyond this basic requirement, different definitions apply.

1. The Free Software Definition

The Free Software Foundation, founded by Richard M. Stallman, defines "open-source" as free software in term of liberty, not price. To meet the free software definition, the following four freedoms are required [Ref. 1]:

- The freedom to run the program for any purpose (Freedom 0).
- The freedom to study how the program works, and adapt it to individual needs (Freedom 1). Access to the source code is a precondition for this.
- The freedom to redistribute copies so a user can help another user (Freedom 2).
- The freedom to improve the program, and release improvements to the public, benefiting the whole community (Freedom 3).

2. The Open Source Definition

The Open Source Initiative defines “open-source” by the following [Ref.2]:

- Free Redistribution: The license does not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. Moreover the license does not require a royalty or any other fee for such sale.
- A Source Code: The program must include source code and must allow distribution in source code as well as compiled form. Where some form of

a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost, preferably by downloading it via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.

- Derived Works: The license must allow modifications and derived works that allow these to be distributed under the same terms as the license of the original software.
- The Integrity of the Author's Source Code: The license may restrict a source-code from being distributed in modified form **only** if the license allows the distribution of "patch files" with the source code modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.
- No Discrimination Against Persons or Groups: The license must not discriminate against any person or group of persons.
- No Discrimination Against Fields of Endeavor: The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.
- The Distribution of License: The rights attached to the program must apply to all those whom the program is redistributed to without the need of those parties executing an additional license.
- No License Specific to a Product: The rights attached to the program must not depend on the program being part of a particular software distribution. If the program is extracted from that distribution and used or distributed

within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those granted the original software distribution.

- No License Restricts Other Software: The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.
- A Technology-Neutral License: No provision of the license may be predicated on any individual technology or style of interface.

The OSI and FSF disagree on the basic principles (commercialism, licensing etc.) but agree on practical recommendations (availability of source code, ability to modify the code, etc.). In practice, nearly all software meeting one definition also meets the other.

3. Open Source Software Licenses

In the last few years, a number of open-source software licenses have been created. Most of the newer licenses are modified to support a particular business model. Well-known open-source software licenses: GNU General Public License, GNU Lesser General Public License, Apache Software License, BSD License, Mozilla Public License, IBM Public License, Zope Public License, and the Artistic License.

4. Advantages of Open-Source Software

Many items of open-source software are good alternatives to proprietary software. This thesis, which implements the Web-based application prototype, chose open-source software for several reasons.

a. Market Share

The market share of software suggests the future risk. Open-source software has a significant market share in numerous markets. For Web servers, the Apache HTTP Server is currently the #1 web server as of September 2003. For Web serving operating system, GNU/Linux is the #2 web serving operating system on the public Internet, according to a study by Netcraft in June 2001 (Figure 1 and Tables 1 and

2). Further statistics show that open-source software is increasing steadily, especially the Apache HTTP server and the GNU/Linux.

	August 2003	Percent	September 2003	Percent	Change
Apache	27,388,860	63.98	27,836,622	64.52	0.54
Microsoft	10,165,745	23.75	10,156,289	23.54	-0.21
Sun One	1,534,586	3.58	1,501,241	3.48	-0.10
Zeus	746,240	1.74	742,950	1.72	-0.02

Table 1. Market Share for Top Developer Across All Domains, September 2003 (From: Ref.3)

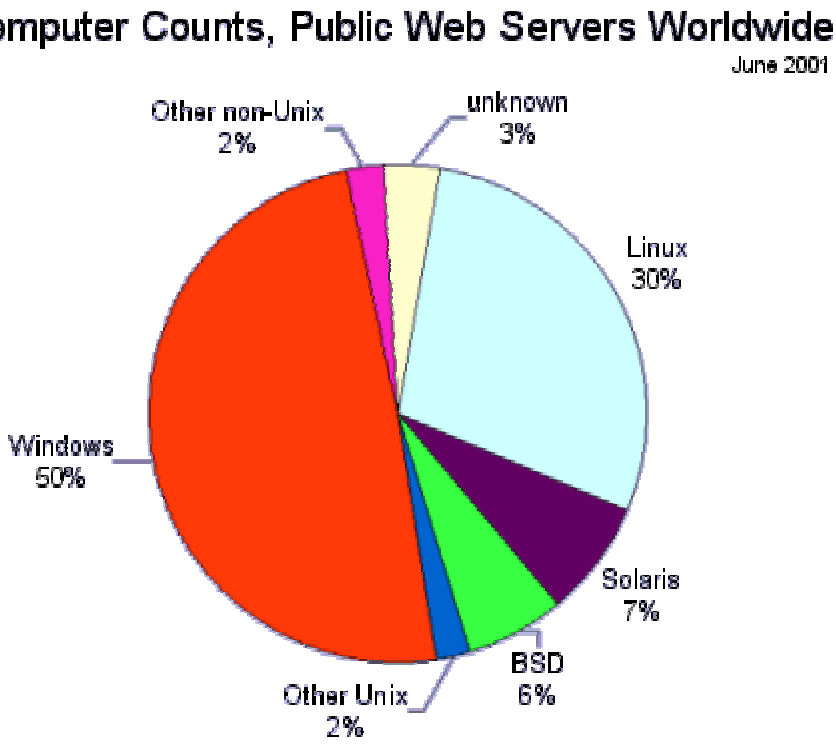

Figure 1. Operating Systems used by Computers Running Public Internet Web Sites, June 2001 (From: Ref 4)

OS group	Percentage	Composition
Windows	49.6%	Windows 2000, NT4, NT3, Windows 9x/Me
Linux	29.6%	Linux
Solaris	7.1%	Solaris 2, Solaris 7, Solaris 8
BSD	6.1%	BSDI BSD/OS, FreeBSD, NetBSD, OpenBSD
Other Unix	2.2%	AIX, Compaq Tru64, HP-UX, IRIX, SCO Unix, SunOS 4 and others
Other non-Unix	2.4%	MacOS, NetWare, proprietary IBM OSs
Unknown	3.0%	not identified by the Netcraft operating system detector

Table 2. Operating systems, June 2001 (From: Ref.4)

b. Reliability

Quantitative data confirm that mature open-source software is more reliable than some proprietary software. For example, GNU/Linux is more reliable than Windows NT, according to a one-year Bloor Research experiment [Ref. 5] and a server uptime study by Netcraft in 2001, which indicated that of the 50 sites with the highest uptime, 92% used Apache and 50% run on open-source operating systems [Ref.3].

c. Performance

Low-level benchmarks by IBM found that GNU/Linux had better performance than Windows for pipes (an input/output mechanism) [Ref.6], and also process and thread creation. Servlet/JSP technology processes each request by creating thread, which will give better performance than Windows Active Server Pages.

d. Scalability

GNU/Linux can use the same software for both small and large projects. From the hardware perspective, GNU/Linux works on PDAs, obsolete hardware, common modern PCs (not just Intel x86), mainframes, massive clusters, and a number of supercomputers. This allows the developers to replace small hardware with massive parallel or extremely high-speed processors or very different CPU architectures without changing the operating system [Ref.7].

e. Security

Some records indicate that the Microsoft Internet Information Services (IIS) is attacked more frequently than Apache [Ref.8]. The number of attacks on IIS was bigger than Apache even though it was used less than a half of Apache. Security is an important issue for Web-based applications. Using Apache integrated with GNU/Linux enhances security to those applications.

f. Cost

Open-source software is generally free unlike proprietary software. This also is true for upgrade and maintenance. For this thesis, the Web-based application prototype cost nothing due to downloading the source code through the Internet.

B. AN OVERVIEW OF OPEN SOURCE SOFTWARE

This part gives an overview of open-source software used in Web applications development for this thesis. It includes an operating system (Linux), Web technologies, and a database system.

1. Linux Overview

a. Introduction

Linux, a Unix-based operating system, is popular today. The Linux kernel, first announced on the Internet in 1991 was initially created by Linus Torvalds at the University of Helsinki in Finland. The kernel has been improved and included in other software, especially software written by the GNU organization that became the operating system named Linux or GNU/Linux. Some Linux features are multitasking, virtual memory, fast TCP/IP driver, shared-libraries, multi-user capability, multiprocessor supporting, and protected mode [Ref. 9]. Initially, Linux was designed to run on an i386 processor but has been ported to run on various architectures including Compaq's Alpha, Sun's SPARC, and Motorola's PowerPC chips. Linux is developed under a General Public License. Differentiated by performance and availability of source code, Linux has been released by commercial or non-commercial organizations in many distributions.

b. Linux Distributions

In technical term, Linux refers specifically to the kernel of the operating system. In order to make the completed operating system, one who releases Linux for distribution always adds to or enhances the basic functions. Improving look and feel and adding a utilities program with an installer to the kernel are some enhanced function alternatives. Popular Linux distributions [Ref. 10] include Linux-Mandrake, Red Hat Linux, Gentoo Linux, Debian GNU/Linux, SuSe Linux, Knoppix, Slackware, Lycrois, and Lindows/OS.

Each Linux distribution's flavor differs from one to the other. One Linux distribution may satisfy programmers but not beginners. However, all Linux distributions are built based on the same kernel with the same additional software packages providing the same functions. One feature making Linux more popular is the configurable nature of this operating system. Users can tune their systems to meet their specific requirements. The number of Linux users has steadily increased during the past decade because Linux is not only becoming easier to use, but it is also providing some specific performance metrics that cannot be met in other operating systems.

For the prototype in this thesis, Gentoo Linux was chosen as the operating system. In addition to the basic features provided by Linux kernel, Gentoo provides the ability to automatically optimize and customize applications. Open-source software integration can be done easily when using it. This was helpful when developing the project.

2. Web Technologies Overview

Performance of Web servers is often critical to Internet applications since handling information through the Web must be done efficiently. Web server performance can be ranked on criteria such as robustness, processor speed, security and others. Among the Web servers that exist on the market, the Apache server has been the most popular Web server on the Internet since April of 1996. The September 2003 Netcraft Web Server Survey found that 63% of the Web sites on the Internet are using it [Ref.3], and usage is growing.

a. The Apache HTTP Server

The Apache HTTP server is the creation of a group of volunteers named the Apache Group with goal to create a robust, commercial-grade, full-featured server whose source code is freely available. Initially, using NCSA http 1.3 as the base, a group of developers who contacted each other via private e-mail, gathered together for the purpose of coordinating their changes (in the form of "patches"). The first official release (0.6.2) was made in April 1995. [Ref. 11] Since then, Apache has been continuously developed to enhance its performance. It took less than a year after the group was formed for the Apache Web server to surpass NCSA httpd as the most popular server on the

Internet. The current version of Apache Web server is 2.0.47. Some of features provided by Apache are:

- Serves static and dynamic CGI Web pages, which can interface with many dynamic content generation technologies such as Java Servlets, JSP, PHP, or Perl.
- Acts as a caching-proxy server.
- Provides language-specific document variants in response to a request.
- Allows many modules to be added or removed, as well as configuring for specific requirements.
- Extends the security feature with several forms of authentication, including SSL encryption.
- Provides ports to many platforms, including Windows, UNIX, Linux, and OS/2.

b. A Tomcat Servlet/JSP Container

Java Web technologies were used in the prototype of this thesis. Servlets and Java Server Pages are Java technologies used for creating such applications. A servlet is a Java Web program that runs on the server, as opposed to applets and other client-side technology. JSP is an alternative interface to servlet technology.

Jakarta Tomcat is the reference implementation for Java Servlet and Java Server Pages (JSP) technologies. It is a servlet container with a JSP environment. A servlet container is a runtime shell that manages and invokes servlets on behalf of users. Tomcat was developed under the Apache Software License but it is based on Java Servlet and JSP specifications developed by Sun [Ref.12]. In addition to being a Java Servlet/JSP container, Tomcat can also be used as a stand-alone Web server. Although this is very useful during development and testing, it is recommended to use Tomcat with a Web Server such as Apache, IIS, and others.

c. Jetty Web Server and Servlet Container

Jetty is a HTTP Server and Servlet Container written from Java that can be used in stand-alone mode to deploy static content, servlets, JSPs and Web applications. It is developed under open source license, derived from the Artistic License. Full source code is included in all releases. Features in Jetty are [Ref. 13]:

- Includes a HTTP/1.1 server configured as a jar file under 300 KB
- Provide one of the fastest servlet servers.
- Scales well to thousands of simultaneous connections.
- Degrades gracefully under stress.
- Integrates into application servers, such as JBoss and Jonas.
- Bundles with many open source product such as JXTA, Tasperty, and Cocoon.
- Runs on embedded systems and handle devices.
- Runs on all Java supported platforms.

The prototype integrates the Apache HTTP Server with Tomcat Servlet/JSP container to enable further development, security, and cooperation. Although Tomcat can be a stand-alone Web server, Web-based applications are more efficient and extendable when integrated with Apache.

3. Database Administration System Overview

The two most popular open-source database systems are MySQL and PostgreSQL.

a. MySQL

MySQL is an implementation of the SQL language. It is provided by a commercial company named MySQL AB, which generates its business by providing services around the MySQL Database. The MySQL software has dual licenses, which are the GNU General Public License (GPL) and the commercial non-GPL MySQL license.

Additionally, MySQL is built on top of a fast indexed sequential engine that is faster than MySQL but takes more resources to gain this speed. It is fully multi-threaded using kernel threads. This software enjoys excellent support in Unix operating systems with an ODBC driver available. Features of MySQL include [Ref. 14]:

- ANSI SQL syntax support.
- Cross-platform support.
- Independent storage engines.
- Transaction support.
- Flexible security system, including SSL support.
- Query caching.
- Full-text indexing and searching.

b. PostgreSQL

PostgreSQL is a free relational database server from PostgreSQL Inc. released under the BSD license. This server uses the SQL language to run queries on data organized as a series of tables with ″foriegn keys" linking related data together. The primary advantage of PostgreSQL is programmability: It allows the user to define new types from the normal SQL types to represent complex data. Its features are [Ref.15]:

- Defining new basic types.
- Defining new operators.
- Defining new functions, using C, SQL, Perl, Python, Tcl, Ruby, sh, or PL/PgSQL .
- Concurrency is managed via a Multi-Version Concurrency Control (MVCC) design, which ensures excellent performance even under heavy concurrent access.
- Inheritance
- Functional indexes, partial indexes.

Between MySQL and PostgreSQL, the most two widely used open-source databases, MySQL was chosen to manage the databases in the prototype of this thesis. According to the comparison by eWeek/PC Labs [Ref. 16], the overall performance of MySQL is comparable to proprietary Oracle database systems. Overall, Oracle9i and MySQL had the best performance and scalability; Oracle9i was slightly ahead of MySQL in most cases, but Oracle costs far more. However, MySQL has a unique "query cache" capability that allows MySQL to use stored results to save time.

In summary, most open-source software related to the Web-based application architecture, including GNU/Linux, Apache, Tomcat, and MySQL meet or even beat most of their proprietary competitors in performance and reliability (in terms of being able to run a correctly-written application). Additionally, the open-source software beat all proprietary software in term of cost.

THIS PAGE INTENTIONALLY LEFT BLANK

III. INTRODUCTION TO SERVER-SIDE TECHNOLOGIES

Web development using server-side technologies provides advantages over old-fashioned CGI/Perl and client-side technologies. Server-side products include ASP, PHP, JSP, and Cold Fusion. Each has its advantages and disadvantages.

A. SEVER-SIDE SCRIPTING

1. ASP

Microsoft Active Server Pages (ASP) allow developers to combine scripting languages (VBScript and JavaScript being the most popular) with an expandable set of software components. These components are treated as objects by the scripting language. Most of the components are Windows-specific and require the Microsoft Internet Information Service Web server software [Ref.17].

The advantages of ASP are:

- It is user-friendly.
- It is a Windows 2000 server component.
- Professional support is available.

The disadvantages are:

- Cost.
- Specialized functionalities may require purchasing commercial components.

2. CGI/Perl

Perl is a multipurpose scripting language designed for text manipulation. Perl has been expanded with modules that adapt it to various specialized purposes. One supports the creation of dynamic Web content via a Common Gateway Interface (CGI) supported by most Web servers. This technology is a standard mechanism by which a Web server can hand a browser's request for a Web page off to any program [Ref 17].

The advantages of CGI/Perl are:

- As a mature language, Perl is less likely to run into bugs than in any other server-side language.
- It is open-source.
- Perl is supported by most Web hosting.

The disadvantages are:

- It is not optimized either for speed, scalability, or ease of use in a Web server setting.
- It is optimized for the Unix platform.
- There is no formal support for Perl.

3. Cold Fusion

Cold Fusion is different from other server-side technologies. Instead of a scripting language, Cold Fusion gives a developer a set of tags to learn. It contains a built-in library of over 300 tags in the latest version and the ability to add custom tags with more traditional programming languages like C/C++ and Java. Cold Fusion is a commercial server platform but is not tied to a Windows based server. It can integrate as easily into Apache running under Linux as it can with IIS on Windows 2000 [Ref.18].

The advantages of Cold Fusion are:

- User-friendly (no programming required).
- Powerful and very scalable technology.
- Professionally supported by Macromedia.
- Cross-platform.

The disadvantages are:

- Expensive to set up.
- Not attractive to those who do not like tag-based development methods.

4. JSP

Java Server Pages (JSP) technology allows Web developers and designers to rapidly and easily maintain dynamic Web pages. This technology uses XML-like tags that encapsulate code-generating content for the page.

Java Server Pages are an extension of the Java Servlet technology. Servlets are platform-independent pure Java server-side modules that can extend a Web server with minimal overhead, maintenance, and support. Unlike other scripting languages, servlets involve no platform-specific considerations. They provide platform independence, enhanced performance, separation of logic from display, ease of administration, and simpler extensibility into the enterprise domain [Ref. 19].

The advantages of JSP are:

- Extremely powerful and scalable.
- Cross-platform.
- Free for personal and development purpose for most Java server plug-ins.

The disadvantages are:

- More time spent in processing compared to other technologies.
- Required payment for most JSPs used to host a commercial Website.

5. PHP

PHP is a widely used open-source general-purpose scripting language specially suited for Web development and that can be embedded into HTML. Its syntax draws upon C, Java, and Perl. This language is mainly focused on server-side scripting, so developers can do anything a CGI program can do, such as collect from data, generate dynamic page content, and send and receive cookies. It can be used on all major operating systems, including Linux, many Unix variants (including HP-UX, Solaris and Open-BSD), Microsoft Windows, Mac OS X, RISC OS, and others. Additionally, PHP supports most Web servers, including Apache, IIS, Netscape, and iPlanet servers [Ref. 20].

The advantages of PHP are:

- Many built-in functions.
- Free, cross-platform open source software.
- A big and active user community.
- Easy to learn, especially for developers with programming experience.

The disadvantages of PHP are:

- Restricted to programmers.
- Not very popular.

B. COMPARISON

For the prototype built in this thesis, I have chosen Servlets/JSP technology for the reasons listed below.

1. Efficient

Servlets/JSP provides superior performance compared to traditional CGI which starts a new process for each HTTP request. With servlets, the Java Virtual Machine runs consistently and handles each request using a lightweight Java thread, not a heavyweight operating system process. This means when a servlet is loaded into the server's memory, it generally remains there, along with Java objects. When a server using servlets receives a request, there are no interpreters to spawn or variables to instantiate (beyond the first time). This as a result makes servlets/JSP serving more efficient than CGI.

2. Convenient

Servlets have an extensive infrastructure for automatically parsing and decoding HTML-form data, reading and setting HTTP headers, handling cookies, tracking sessions, and many other such high-level utilities. Third-party servlet containers are available for Apache Web Server, Microsoft IIS, and other. Servlet containers are also a component of Web and application servers, such as BEA WebLogic Application Server, IBM Websphere, Sun One Web Server, Sun One Application Server, and others.

3. Powerful

Servlets/JSP provides all of the power of the Java language for the client/server interaction. Portability, multithreading, extensive class libraries, object-oriented code, strong safety features, robust security measures, elegance and extensibility are advantages. Several capabilities supported by Servlets/JSP are difficult or impossible to accomplish with CGI. Servlets can talk directly to the Web server, whereas regular CGI programs cannot, at least not without using a server-specific API. For instance, communicating with the Web server makes it easy to translate relative URLs into concrete path names. Multiple servlets can share data, making it easy to implement database connection pooling and similar resource-sharing optimizations.

4. Inexpensive

Numerous free or inexpensive Servlet containers and Web servers are also available. Most of these are for personal use, but Apache also provides commercial-quality support. Setting up the system to support Servlet/JSP technology is an inexpensive endeavor. This is in contrast to many of the CGI alternatives, which require a significant initial investment to purchase proprietary packages.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. INTRODUCTION TO WEB APPLICATION DEVELOPMENT

A. WEB APPLICATION ARCHITECTURE

The architecture of a World Wide Web application includes a browser (the "client"), a network, and a server (Figure 2). Browsers request “Web pages” from the server. Each page is a mix of content and formatting instructions expressed in HTML. Some pages include scripts that define dynamic behavior for the display page through applets, ActiveX controls, and plug-ins contained in the page. For some pages the user enters information in field elements on the page and submits them to the server for processing. The user can also navigate to different pages in the system via hyperlinks.

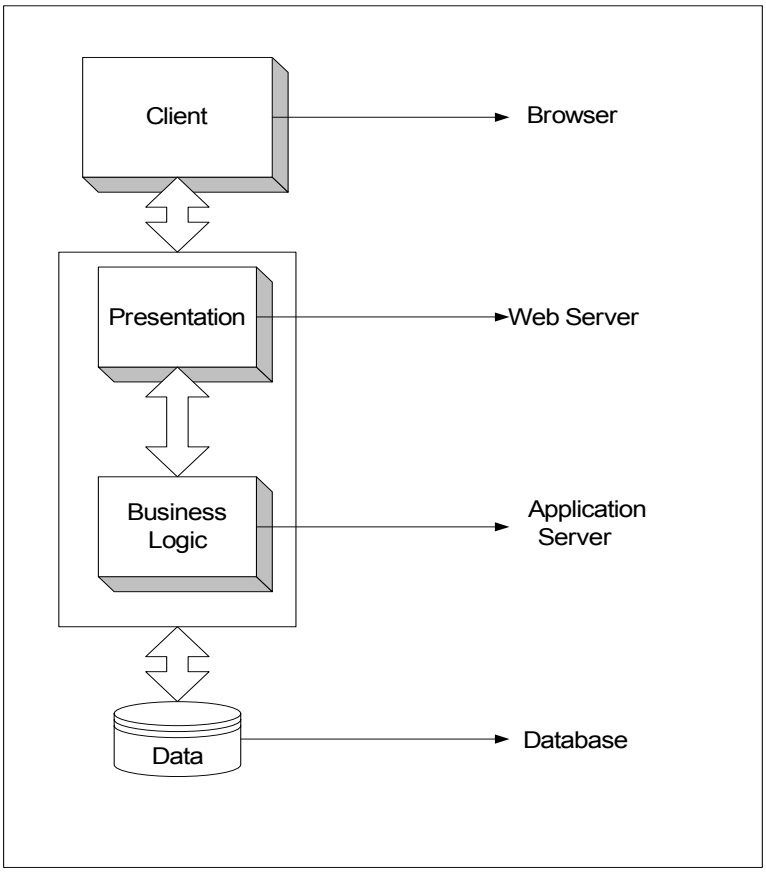

Figure 2. Web Application Architecture

From the client’s perspective, the Web page is always an HTML document. On the server, however, a Web page may manifest itself several ways. In the older Web

applications, dynamic Web pages were built with the Common Gateway Interface (CGI). In a CGI-based system, a special directory is configured on the Web server for executing scripts in response to client page requests, using an appropriate interpreter (usually a Perl shell) and streaming the output back to the requesting client. "Business logic" is executed while processing the file, which can interact with server side resources such as databases and middle-tier components.

Today's Web servers are much more security-conscious, and include features like management of client state on the server, transaction processing, remote administration, and resource pooling. Current Web servers process dynamic pages in three categories: scripted pages, compiled pages, and hybrids [Ref. 21]. In the first category, the page is a mix of HTML and some other scripting language. When the page is requested, the Web server delegates the processing of the scripted portion of the page to an engine that recognizes it. Examples are Microsoft's Active Server Pages, Java Server Pages, and Cold Fusion.

In the second category, the Web server loads and executes binary code that has access to information associated with the page's request (form fields and parameters) and produces the HTML stream returned to the client. Although not a rule, compiled pages tend to do more than scripted pages. Examples of this type of architecture are Microsoft's ISAPI and NSAPI.

The third category mostly includes scripted pages that were once requested and are cached for use by subsequent requests. Only when the original page's content changes, will the page undergo another compile. This is a compromise between the flexibility of scripted pages and the efficiency of compiled pages.

B. THE DEVELOPMENT OF WEB APPLICATIONS WITH SERVLETS/JSP

The term "Java Web Applications" covers all Java-compatible dynamic extensions of a Web server. It includes servlets, JSP, HTML pages, classes, and other resources that can implement an information system on the web. "Web components" are either Java Servlets or JSP pages. This section gives an overview of Servlets/JSP technologies.

1. Servlets

A servlet is a Java class used to extend the capabilities of servers that host applications access via a request-response programming model. Typically servlets extend Web servers even though they can response to any type of request. Java provides HTTP-specific servlet classes. All servlets must implement the Servlet interface, which defines life-cycle methods. Servlet developers can use or extend the generic Servlet class, and the HttpServlet class provides methods such as doGet and doPost for handling HTTP-specific services [Ref.22]. Figure 3 shows HTTP Servlet model.

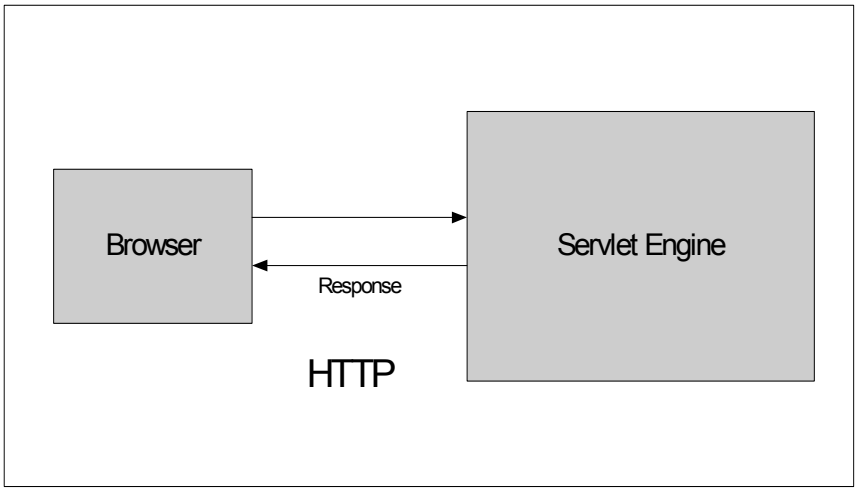

Figure 3. HTTP Servlets

The life cycle of servlet is show in Figure 4. When a request is mapped to a servlet, its "container" performs the following:

- If an instance of the servlet does not exist, the Web container loads the servlet class, creates an instance of the servlet class and initializes the servlet instance by calling the init method.
- On each request, the container invokes the service method, passing a request and response object.
- If the container needs to remove the servlet, it finalizes the servlet by calling the servlet's destroy method.

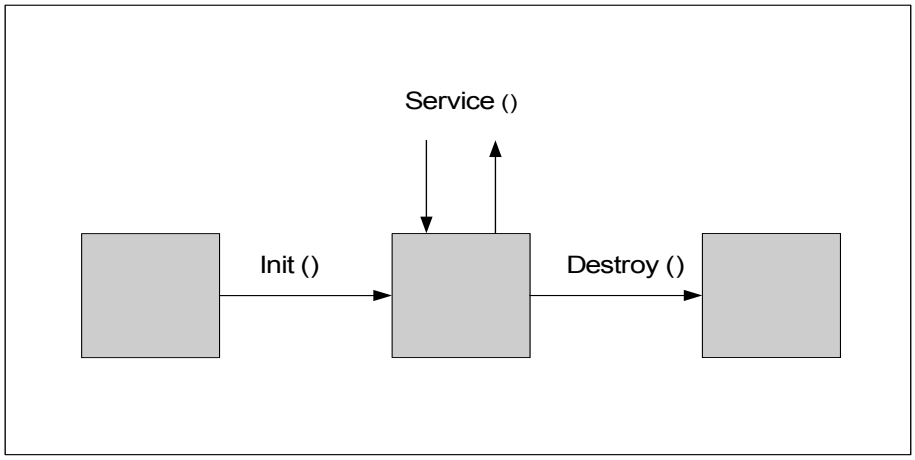

Figure 4. Servlet Lifecycle

The servlets container handles each request using a lightweight Java thread. In most environments, many servlets run in parallel with the same process as the server; therefore it responds quickly to client requests. Servlet multithreading is illustrated in Figure 5.

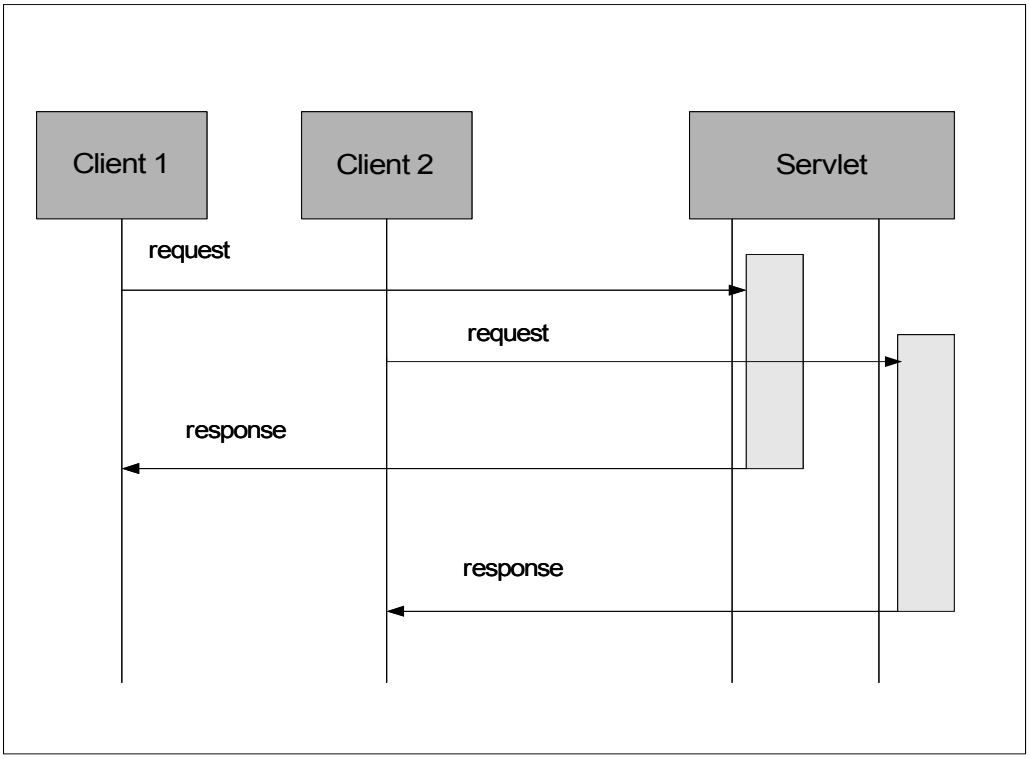

Figure 5. Servlet Multithreading

How servlets are loaded varies with the server. Each server knows how to map a request coming from browser to a servlet. It will be done as either:

- Mapping to a particular servlet.
- Mapping a class of requests to a single servlet.
- Invoking servlets to filter the output of other servlets.

An HTTP servlet overrides the doGet() or doPost() method of the HttpServlet class. These methods take two parameters as input, a request object (instantiated) and a response object (to be instantiated by the servlet). Servlets can accept input parameters from a Java applet, the URI of the request, from some other servlet or network service, or from the parameters passed from a HTML form. Those parameters will be used to generate HTML-formatted responses.

2. JSP

Java Server Pages (JSP) technology is an extension to Java Servlet technology. It separates presentation from application logic. Web development using JSP technology can be accomplished by HTML or XML pages, which contain scripting elements (scriptlets) and tags in addition to the regular page contents. The JSP engine receives requests from a client from JSP Directives including: [Ref. 19]

- JSP Declarations (enclosed with <%! and %>) for variable or method declarations.
- JSP Expressions (enclosed with <%= and %>) which insert values directly into output page.
- JSP Scriptlets (enclosed with <% and %>) which insert arbitrary code.
- JSP actions or tags (enclosed with <jsp: and />) which call methods implemented with Java Beans and similar encapsulations.

The life cycle and many of the capabilities of JSP pages are accomplished by Java Servlet technology. When a request is mapped to a JSP page, the JSP page is translated into input to a servlet automatically [Ref. 19]. For the prototype of this thesis, JSP

technology is used to implement a Web-based application. This approach permits more reusable components that can be shared between applications.

3. Web Application Life Cycle

A Web application consists of Web components, libraries, and static resource files, such as HTML pages and image files. It can also define links to outside resources such as Enterprise Java Beans (EJBs). Java Servlets and JSP pages are supported by the services runtime platform called a Web container which provides services, such as request dispatching, security, concurrency, and life cycle management. It also gives Web components access to APIs, such as naming, transactions, and e-mail. Aspects of the behavior of web applications can be configured in a Web application deployment descriptor in an XML file web.xml located in the /<SERVER_ROOT>/applicationname/ WEB-INF/ directory.

Developing a Web application can be summarized as follows [Ref. 23]:

- Develop the Web-component code.
- Develop the Web application deployment descriptor.
- Build the Web application components along with static resources and helper classes needed.
- Package the application into a deployable unit.
- Install or deploy the application into the Web container.
- Access the URL that references the Web application.

A Web application can be distributed in a Web application archive (WAR), which is similar to JAR, the mechanism used for Java class libraries. A Web application can run from a WAR file or from a corresponding unpacked directory.

V. DATABASE MANAGEMENT

We overview here database management systems, relational databases, and the Structure Query Language (SQL).

A. DATABASE MANAGEMENT SYSTEM

A database management system (DBMS) is system software that manages and controls access to a collection of interrelated data. This collection of data, the database, contains information relevant to an enterprise. A database system is partitioned into modules as illustrated in Figure 6.

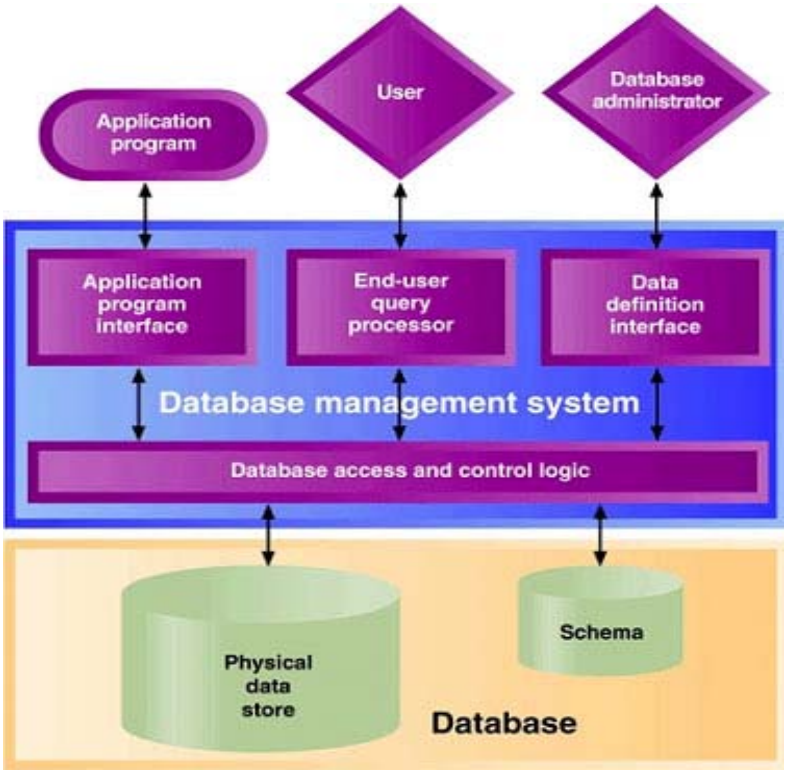

Figure 6. Database Components (From: Ref. 24)

- The Data Definition Interface (DDI) is used to modify the schema. Typical DDI-related tasks include defining a new data stored in the database, defining or modifying data relationships, adding or modifying access controls, and adding or modifying forms or reports.

- The query processor allows a user to interact with a database without writing a program.
- The Application Program Interface (API) allows application programs to interact with the database. Common API functions include the run-time interface and function library.
- The schema defines a database's structure, which can be interpreted in user view, administrator view, or system-software view.

B. RELATIONAL DATABASES

Among the existing database models, the relational model is the most widely used. This model uses a collection of tables to present both data and their relationships among those data. Each table has multiple columns and each column has a unique name [Ref.25]. Figure 7 presents a sample relational schema for columns customer, order, order line, and product.

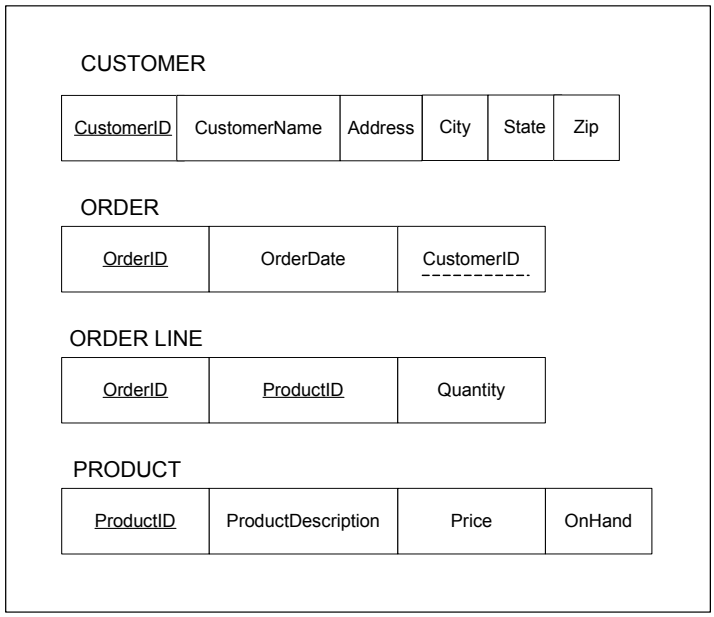

Figure 7. Schema of Four Relations

Keys allow primary data access to the information. For instance, the primary key for ORDER LINE consists of the attributes OrderID and ProductID. Figure 8 shows an example of this database.

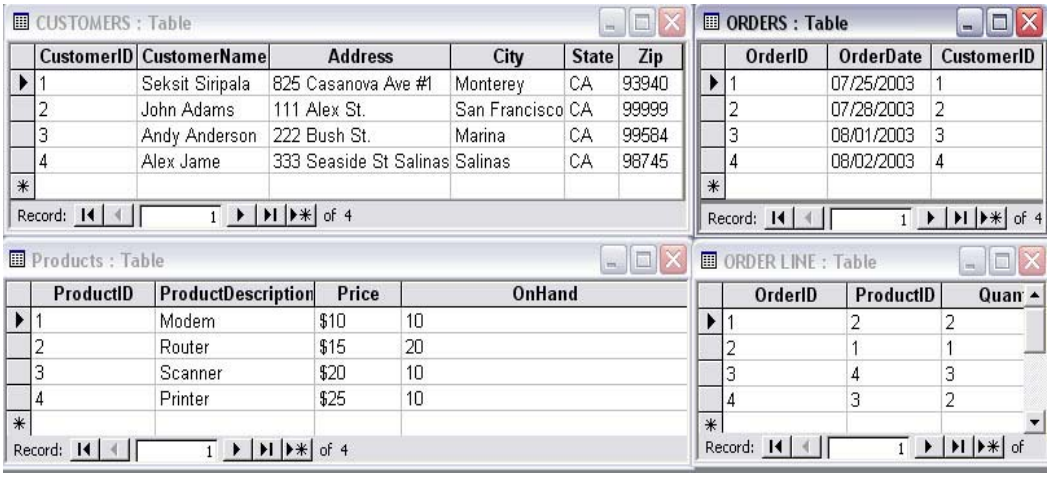

Figure 8. Relational Database Tables

C. THE STRUCTURED QUERY LANGUAGE

The Structured Query Language (SQL) has become the de facto standard language used for creating and querying relational databases. This language has been accepted as a standard by the American National Standard Institute (ANSI) and the International Organization for Standardization (ISO) since 1986 [Ref.25]. SQL versions have been implemented in both mainframe and personal computer systems. Today most products are SQL-92 compliant and moving toward Core SQL-99 compliance.

The SQL environment includes the SQL DBMS along with accessible databases and associated users and programs. The information contained in this database is maintained by the DBMS as a result of the SQL commands, which can be classified into three types including Data Definition Language (DDL), Data Manipulation Language (DML), and Data Control Language (DCL) [Ref.25].

- The Data Definition Language provides commands for defining relation schema, deleting relations, and modifying relation schemas, such as, CREATE SCHEMA, CREATE TABLE, CREATE VIEW and others.

- The Data Manipulation Language includes a query language covering both the relational algebra and the tuple relational calculus. It also includes commands used for updating, inserting, modifying, and querying the data in the database.

- The Data Control Language provides commands for the Database Administrator (DBA) to control the database. They include commands to grant or revoke privileges to access the database or particular objects within the database, and to store or remove transactions that would affect the database.

D. JDBC API

The JDBC (Java Database Connectivity) API (Application Program Interface) is a call-level programming interface allowing external access to SQL database manipulation and update commands. They allow the integration of SQL calls into a general programming environment by providing library routines, which interface with the database. Using standard library routines, developer can write a normal Java program to open a connection to the databases, send SQL commands to them, process the results that are returned and then close the connection.

VI. DESIGN AND IMPLEMENTATION

This chapter reports an experiment using open-source Java Technologies to build Web-based application. The prototype manages simulated personal information in the Thai military organization.

A. DESIGN

1. System Architecture and Technologies

The Personal Information Management System (PIMS) prototype was built based on the Web Application architecture introduced in Chapter IV. Clients use a Web browser to connect to the system. Each client's request is executed at the middle tier. The connection between the middle tier and the database is established via the application program interface (API). The following system technologies were used in the prototype.

- Gentoo Linux runs as the operating system.
- Java 2 Platform, Standard Edition runs as the core Java Technology.
- The Apache HTTP Server runs as the main Web server.
- The Jakarta Tomcat 4 runs as the Servlet/JSP container.
- The Mod_jk module runs as the Apache-Tomcat connector.
- MySQL runs as the database server.
- JDBC runs as the Java Database Connectivity.

Gentoo Linux provides the abilities mentioned in Chapter II, but other Linux distributions and other Unix-based operating systems can be used. The Java server-side technologies used in this prototype are based on Jakarta Tomcat 4.0.6, which is the reference implementation of Servlets 2.3 and Java Server Pages 1.2 technologies. Even though Tomcat can run as a stand-alone Web server, it is recommended to configure the "mod_jk" module for forwarding the client's request from Apache to Tomcat running as Servlet/JSP engine [Ref.25]. One reason is to allow the Web server to host different applications written with a variety of languages. Figure 9 illustrates the framework being used in this prototype as well as alternatives of running Tomcat as a stand-alone or

running it as a Servlet/JSP engine. For the database server, the prototype runs MySQL. The connection between Tomcat and MySQL is established via the standard JDBC.

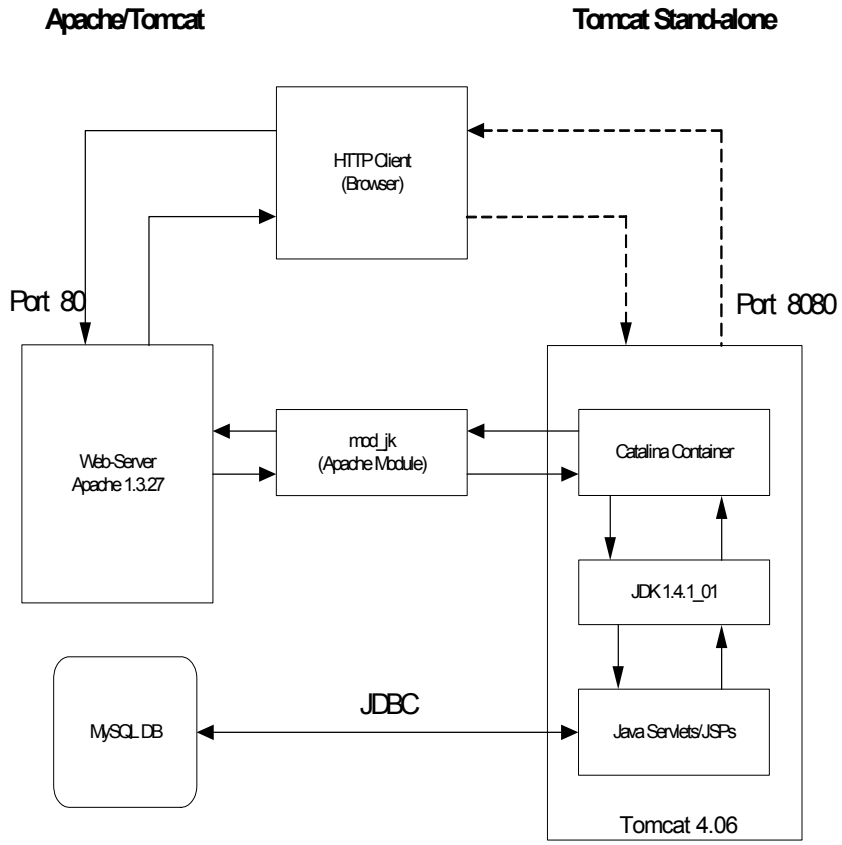

Figure 9. System Architecture and Technologies

2. PIMS Model

The modeling of PIMS is in two parts: use cases and the object model.

a. Use Cases

The application provides access to the following users: (1) administrator, (2) department commander, (3) administrative office personnel, and (4) department officer. The application allows all users to access the system by entering his/her login name and password. They can view a welcome message, announcements, the FAQ (frequently asked questions) page, the monthly schedule page, department information such as the chain of command and mission, the discussion board page where comments on specific topics are posted, the weather page, personal information about

himself/herself (which they can also update), and a menu of available tools. Additional access rights include:

1. Administrator:

- Ability to create a new user profile
- Personal information about all department personnel
- Ability to create, delete and update a message on the announcement page.

2. Department commander:

- Personal information of the person under their command
- The department's projects page

3. Administrative office personnel:

- Ability to create and update user profiles
- Personal information about all department personnel

b. Object Model

Following the use cases in the previous section, the object model of PIMS was designed and developed as illustrated in Figure 10. The session for each user is initialized via the login page, which is authenticated by a servlet. The servlet then forwards the result to display user's options by JSP.

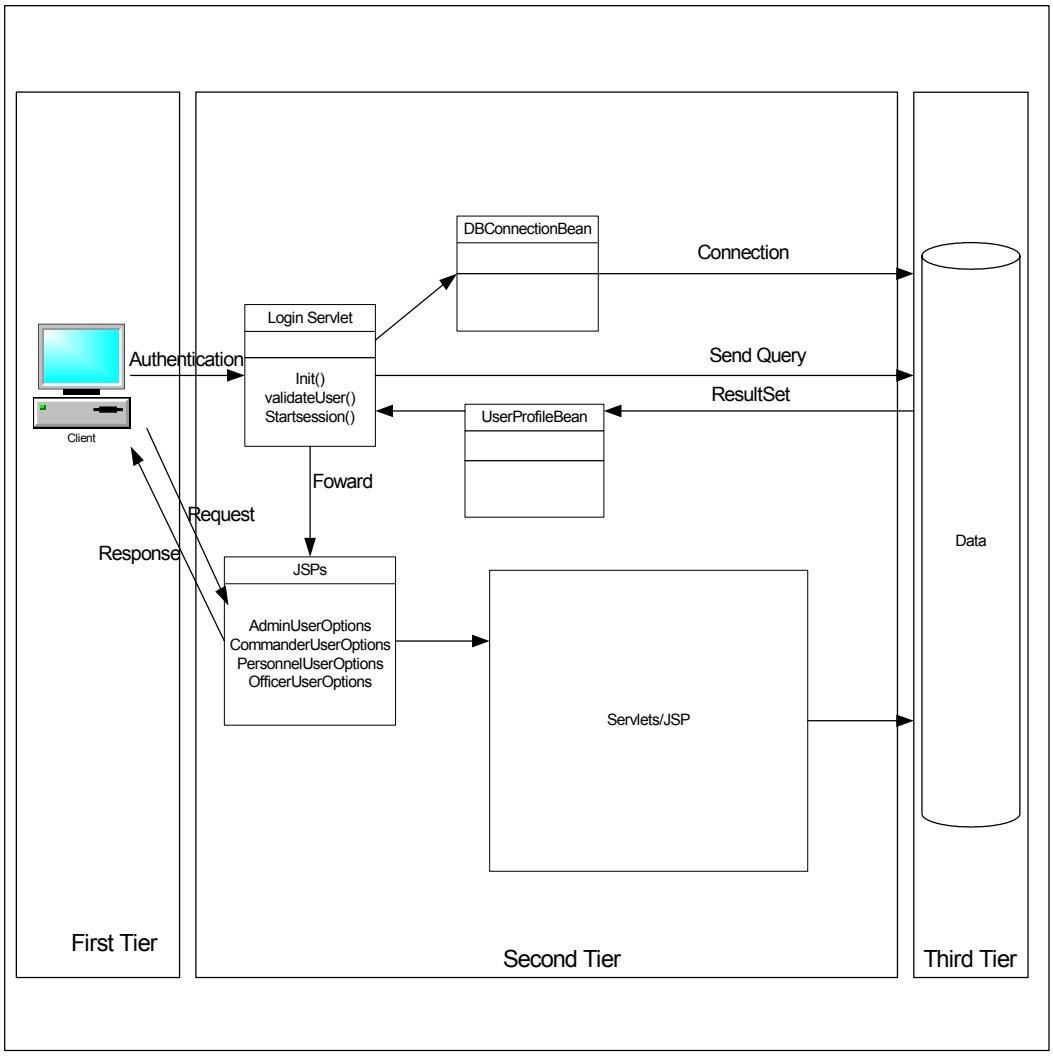

Figure 10. PIMS Object Model

c. PIMS Database Schema

The design of the database for PIMS is illustrated by the entity-relationship diagram in Figure 11.

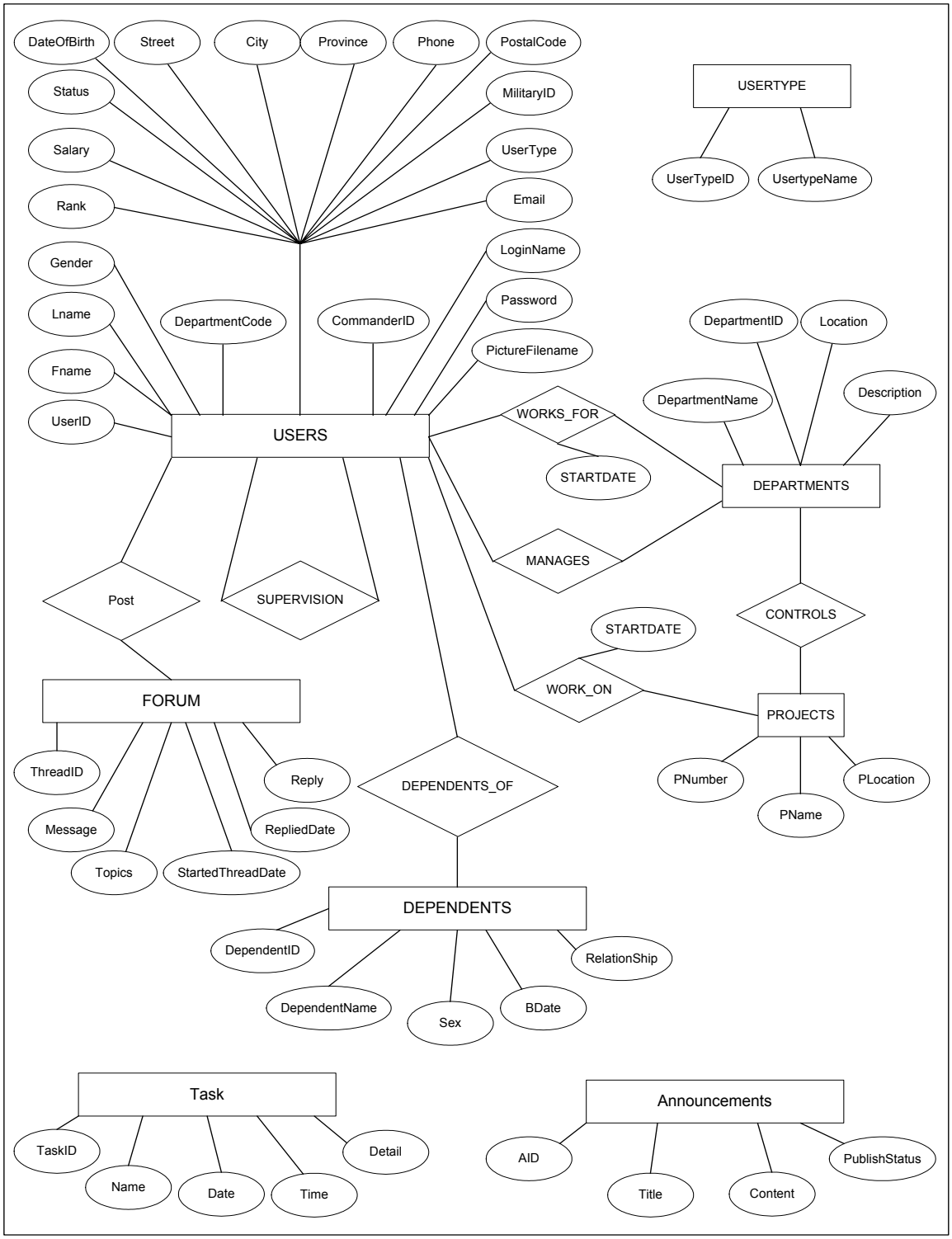

Figure 11. ER Diagram for PIMS System

B. IMPLEMENTATION OF THE PROTOTYPE

1. Implementation of the Database in MySQL

Originally, the database in MySQL was interfaced via a command line; an alternative is a graphical user interface. The database for PIMS is named "schq"; Table 3 briefly describes its tables and their attributes.

Table Name	Primary Key(s)	Description
Announcements	AnnouncementID	Current announcements
DepartmentLocation	DCode, Location	Location information for the department
Departments	DepartmentID	Important information for the department
Dependents	DependentID	Information about dependent's PIMS users
Projects	PNumber	Projects in the organization
SCHQForum	ThreadID	Discussion boards
Task	TaskID	The monthly schedule
UserType	UserCode	User types in PIMS
Users	UserID	Personal information of PIMS users
WorkOn	UID, PNO	People for a specific project

Table 3. Description of "schq" Database Tables

Figure 12 illustrates all tables and their attributes in "schq" database.

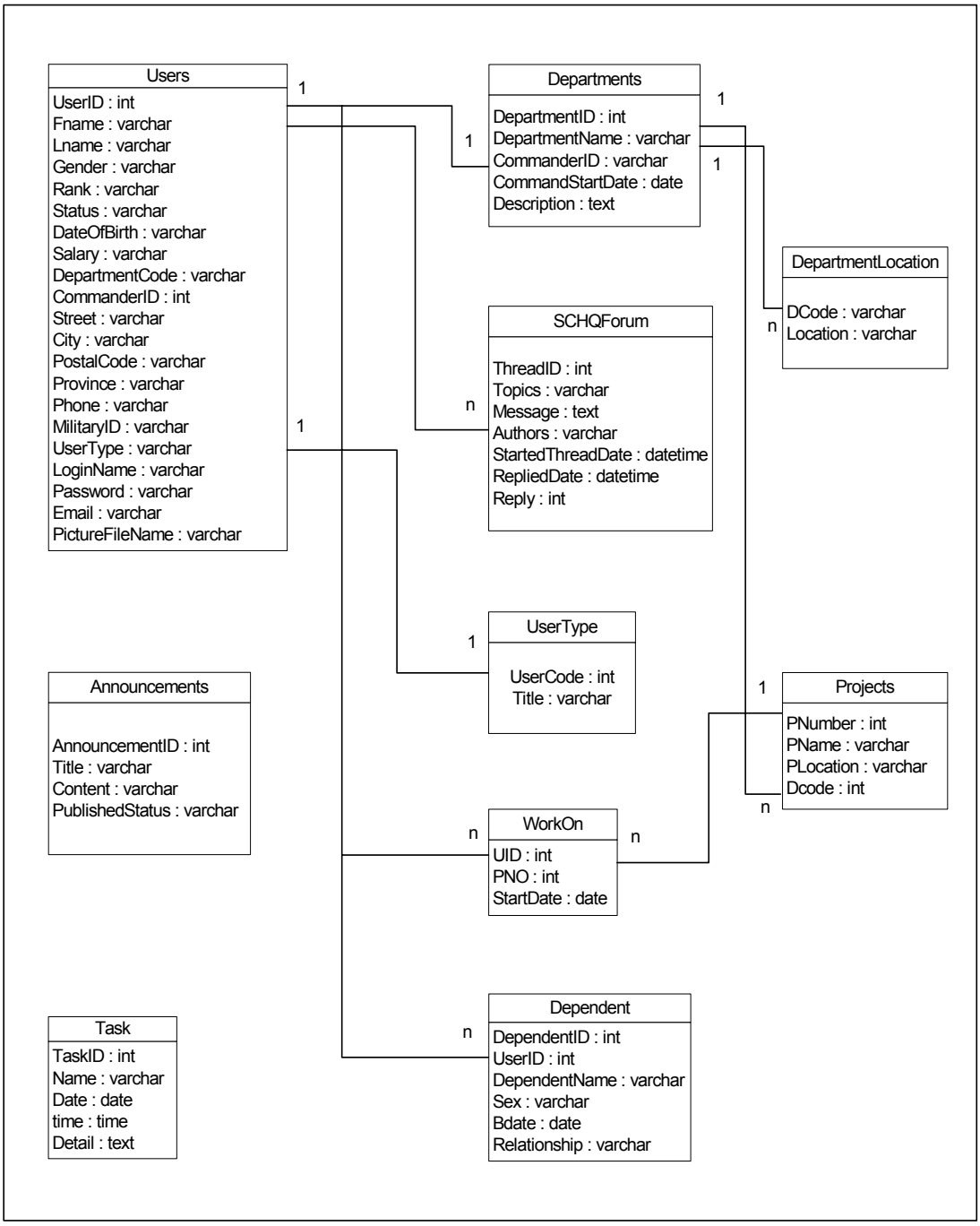

Figure 12. Database Tables and Relationships

PIMS is connected to the database via the MM MySQL JDBC driver. It uses a JavaBean class to connect and to execute SQL statements. The connection is closed after MySQL returns a set of results.

2. Implementation of the Application Logic

In PIMS, JSP pages are the front end, while servlets execute the application logic and present results of that execution. To accommodate the development process and to maintain the system, servlets and JSP pages are divided into packages. The session is initialized in the login servlet after entering the login name and password. The login servlet instantiates the appropriate user profile and forwards a JSP page with options matching user type.

a. SchqAdmin Package

A SCHQAdmin package contains the Java classes that are used in the main part of the application:

- **DBConnectionBean**

PIMS uses Java Beans to connect to MySQL. A Java Bean class uses the database connection properties stored in the "CommonData" class to create a connection to the database server via the JDBC driver. Due to the low expected load of the application, no connection pooling was implemented but could be added.

- **UserProfileBean**

This Java Bean stores the user's personal information. It is instantiated when the login and password is approved and presents each user the appropriate menu options. It also updates user information in the subsequent pages.

- **Login**

The Login servlet authenticates users, creates bean and forwards an appropriate JSP page. After the input information is verified, it instantiates the user-profile bean and starts the session. The session is maintained until the user logs off. One of four JSP pages are displayed depending on user type, each with specific menu options.

- **Servlet classes**

Other servlets are invoked via the hyperlinks in menu options as described in Table 4.

Class Name	Description
Announcements	Class for current announcement from the database
CommonData	Class for information used in the database connection process and generation of the HTML page
CreateGuestUserProfile	Servlet that creates, updates, and deletes personal data
DBConnectionBean	Maintains the connection to the database and executes the SQL command
DeleteAnnouncement	Servlet that deletes an announcement from the database
DeleteDependent	Servlet that deletes a dependent from the database
DepartmentInfo	Servlet that displays department information
DisplayTaskDetail	Servlet that displays a task description
DisplayWorker	Servlet that displays the workers on specific projects
ListMonthlyTask	Servlet that displays the task list in the current month
ListOfficer	Servlet that displays the officer matching the given name
ListOfficerDependent	Servlet that displays the dependents of a user
ListProject	Servlet that displays a project list
Login	Servlet that authenticates the user, creates a user profile, and forwards an appropriate JSP file
Logoff	Servlet that terminates the session for the user
ProcessAnnouncement	Servlet that creates, updates and deletes announcements
ProcessGuestUserProfile	Servlet that updates the personal information
ProcessOfficerDependent	Servlet that creates and updates dependents information
ProcessUserProfile	Creates and updates personal information
SearchEngine	Servlet that submits a query string to the Google search

Class Name	Description
	engine.
ServletUtilities	Servlet that accommodates other servlet classes
UserProfileBean	Class that contains user information during a session.

Table 4. Description of Classes in "schqAdmin" Package

b. Forum, Date, and Weather Packages

The Forum package contains the Java classes for the discussion board. They allow the user to view forums, post comments, and create a new topic. Table 5 describes each class.

Class	Description
ListForum	Servlet that displays the forum topics matching the given query string.
ListLatestTopics	Servlet that displays the forum topics by the latest date.
ListPopularTopics	Servlet that displays popular messages.
PostMessage	Servlet that posts a message.
ShowForum	Servlet that shows all messages on a chosen topic.

Table 5. Description of all classes in "forum" Package

The Date package contains the Java class as utility class for the system. The class in this package is slightly modified from the original date package accompanied with Tomcat. The class arranges the date format to display in the appropriate format. The Weather package contains the aggregator class that collects the weather information from the United States National Service (NWS) web site (http://www.nws.noaa.gov/, September 2003).

c. JSP Files

In PIMS, JSP pages are used to display static pages, create forms to submit to servlets, and to present the results of servlet execution. The first JSP file in the system after initiating the user session is the user-options JSP page. Pages in PIMS are divided into subfolders similar to packages in servlets. The most frequently used package is "User" package containing the core JSPs of the system. Table 6 contains a summary of the JSP files in the system.

Package	JSP	Description
Admin	AdminUserOptions	Illustrates the administrator options.
Admin	EditAnnouncement	Sends an announcement to the database.
Commander	CommanderUserOptions	Illustrates the commander options.
Personnel	GuestUserProfile	Gives the user a form to submit updated personal information.
Personnel	ManageUser	Handles the manage user options.
Personnel	PersonnelUserOptions	Illustrates the administrative-office personnel options.
Tools	Footer, Nav	Gives layout of pages.
User	EditDependentForm	Gives a form to submit updated dependent information.
User	EditOfficerUserProfile	Gives a form to submit updated user information.
User	FAQ	Illustrates the FAQ.
User	NewTopicForm	Gives user a form to post a new topic in the forum.
User	OfficerUserOptions	Illustrates the department officer options.
User	OfficerUserProfile	Illustrates the user's personal information.
User	PostMessageForm	Gives a form to post a message to the specific forum's topic.
User	SearchWWW	Gives the user a form to submit a query string search to the Google search engine.
User	Weather	Gives the current local weather.

Table 6. Description of JSP files

d. Execution of the Request and Session Tracking

In PIMS, processing occurs when users click on a menu option or button. The Web server forwards the command and data to Servlet/JSP container which processes the request and creates the results. PIMS uses a session-tracking technique of the HttpSession API. It uses a session object associated with the current request, creates a new session object when necessary, looks up information associated with a session, stores information in a session, and discards completed sessions.

e. Deployment of PIMS

The technologies using in the system must be installed and configured as shown in Figure 9. The details of this configuration can be found in their Website. The directory "PIMS" is created under "webapps" directory of Tomcat, and Figure 13 shows the structure of PIMS directory under a stand-alone Tomcat server. The "PIMS" directory

having the pages linking to the system must be under the “httpdoc” directory of Apache. Figure 14 shows the structure of “PIMS” directory under the Apache server.

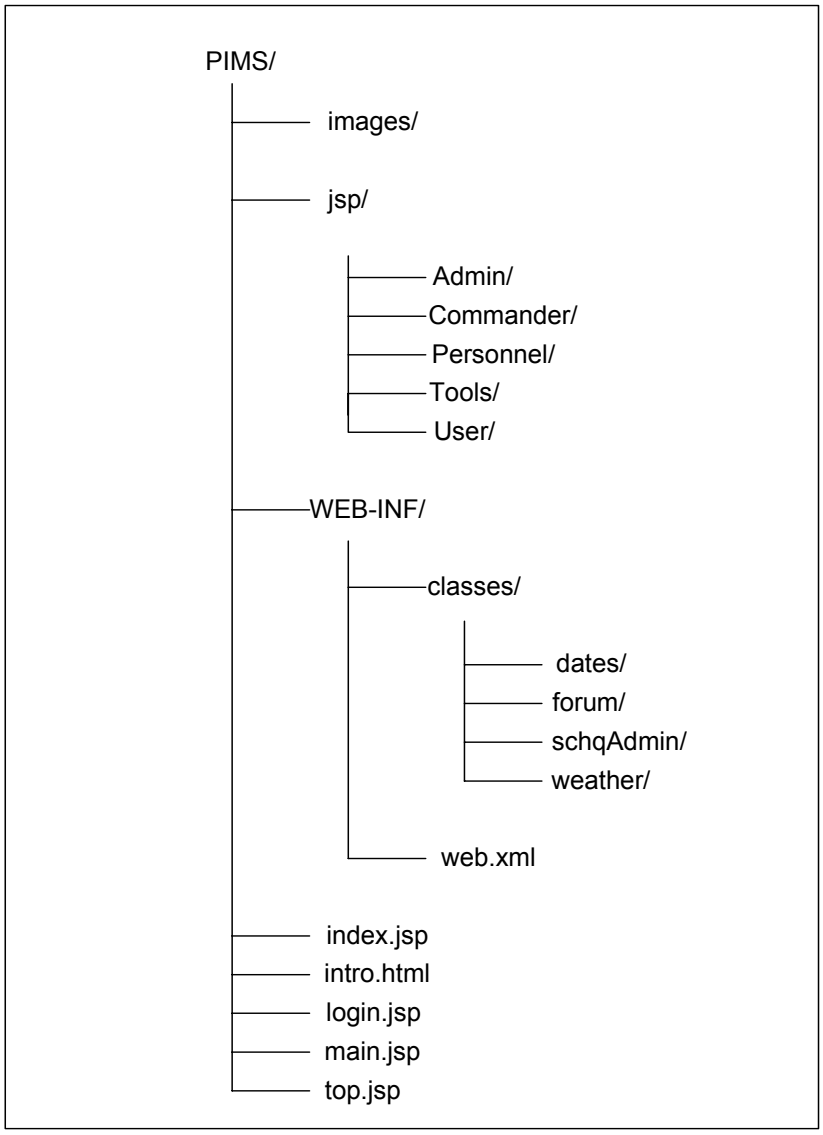

Figure 13. The Structure of PIMS Directory Under Tomcat

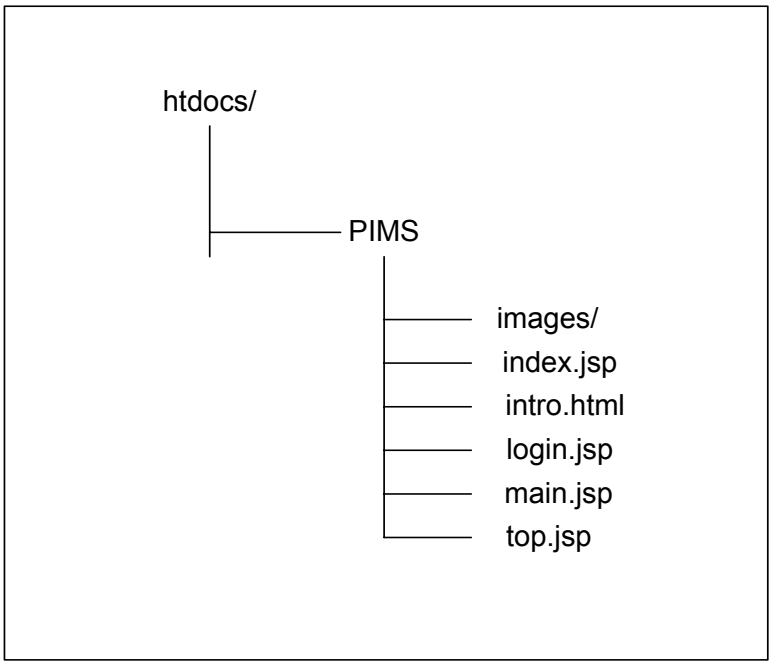

Figure 14. The Structure of PIMS directory under Apache

3. Screenshot of the Prototype

In this section some screen shots of the system are shown.

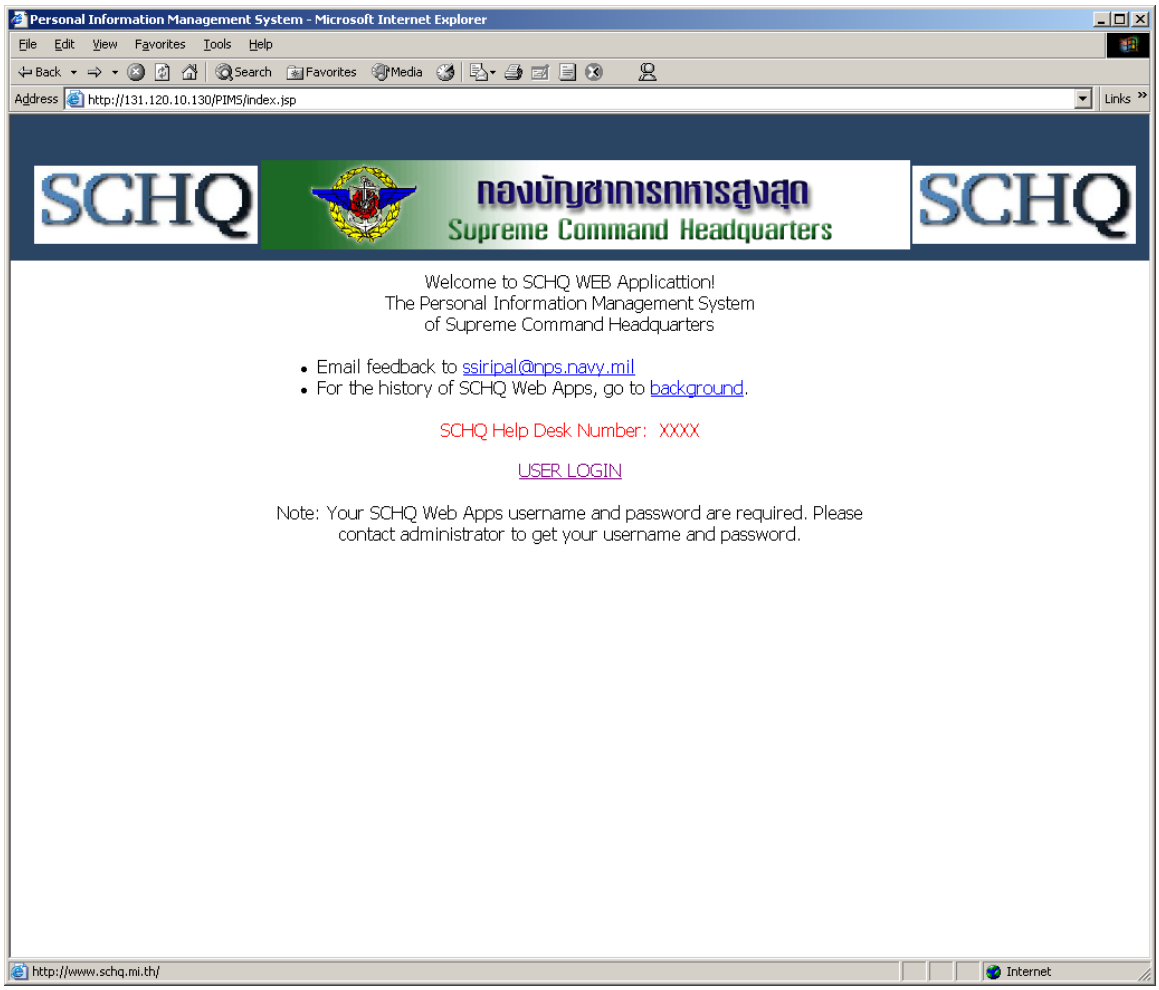

Figure 15. Screen Shot for PIMS Home Page

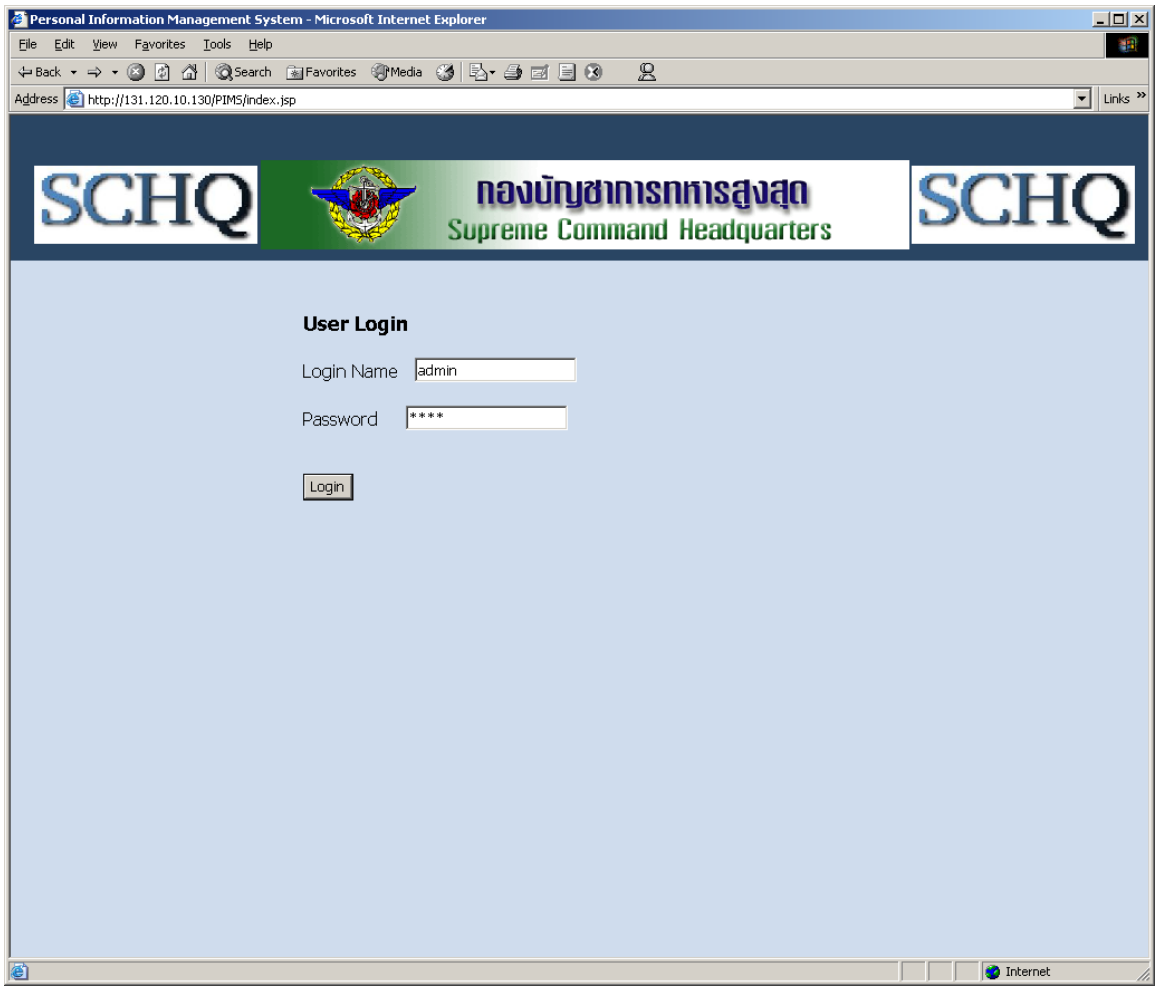

Figure 16. Screen Shot for Login Page

This screen shot is for login page. The user uses login name and password to access the system.

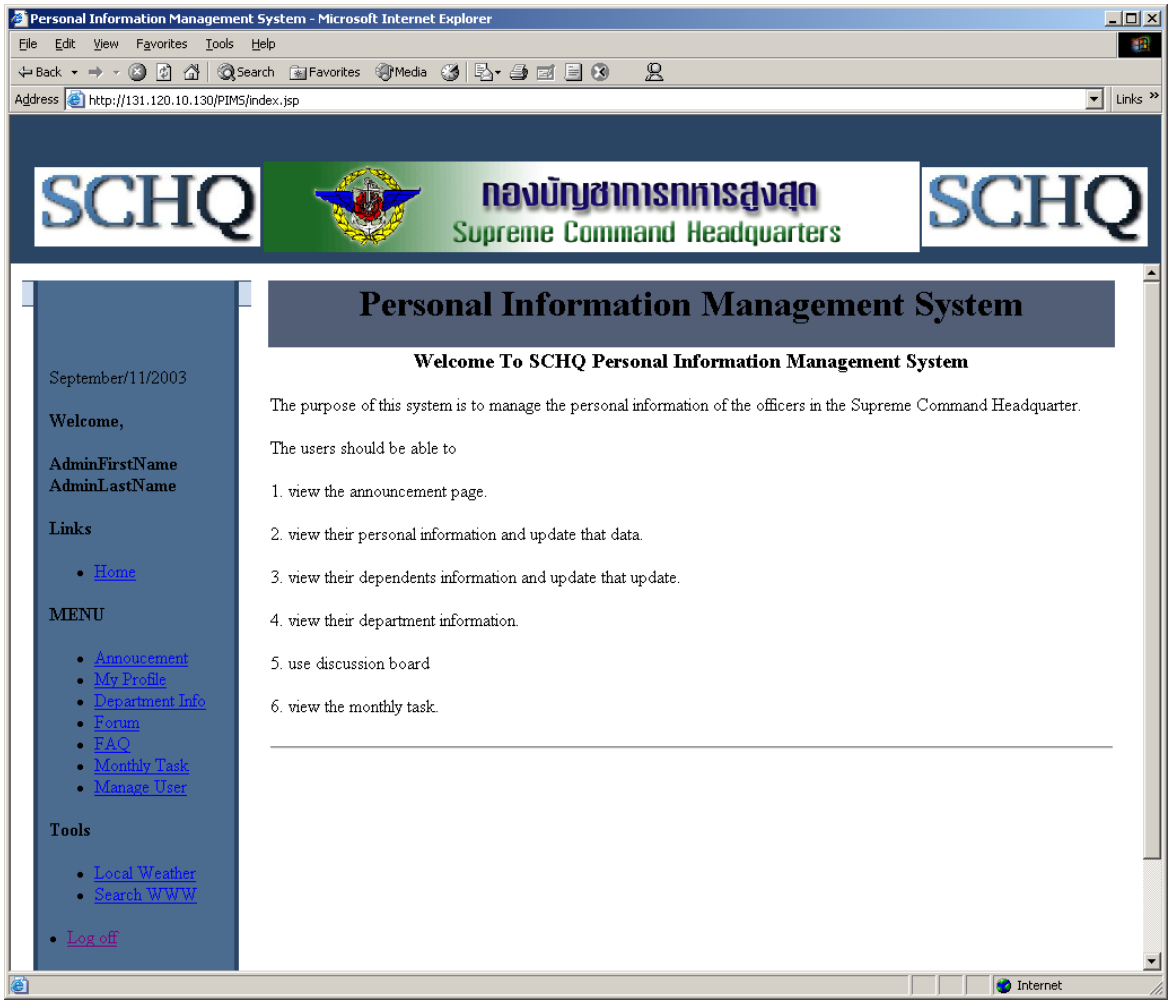

Figure 17. Screen Shot for Administrator Options Page

After the login name and password are verified, the servlet forwards the information to related JSP page depending on the user type.

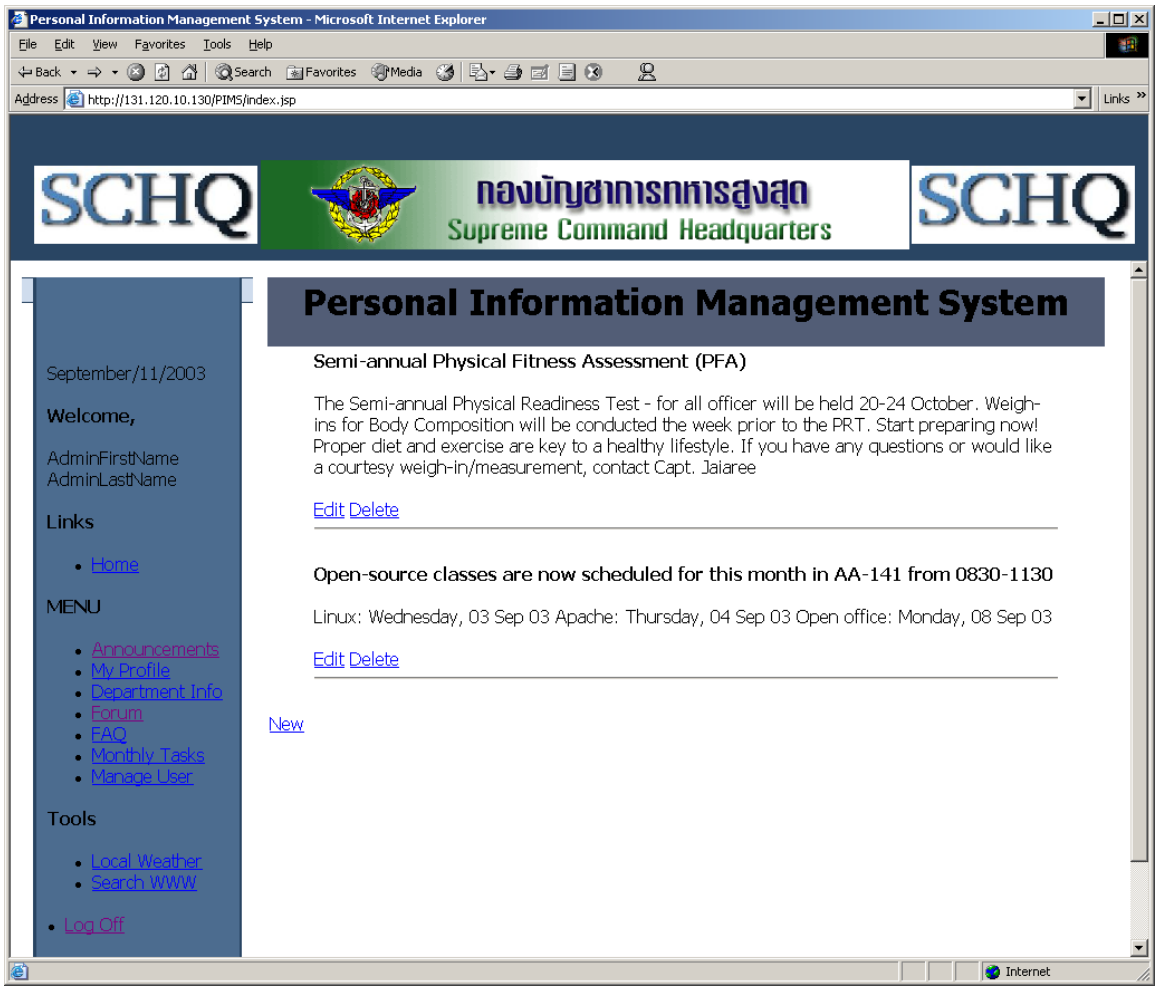

Figure 18. Screen Shot for the Administrator's Announcement Page

The administrator views the announcement page differently from everyone else. He/She can create, edit, and delete the information in the page.

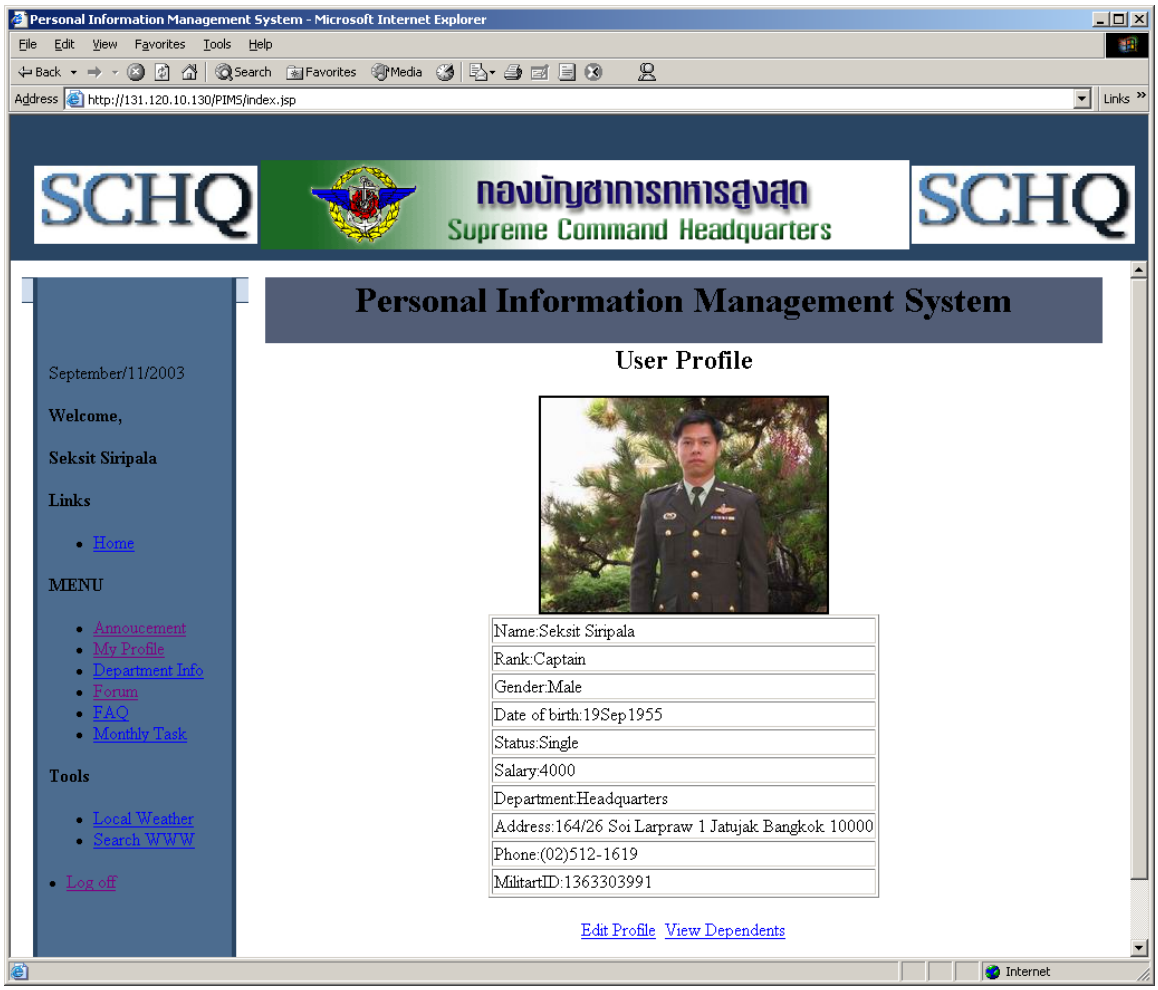

Figure 19. Screen Shot for Personal Information Page

Users can view their personal information using the “My Profile” hyperlink. The page illustrates personal information that can be edited in the subsequent page. The page also has the hyperlink option to display dependents.

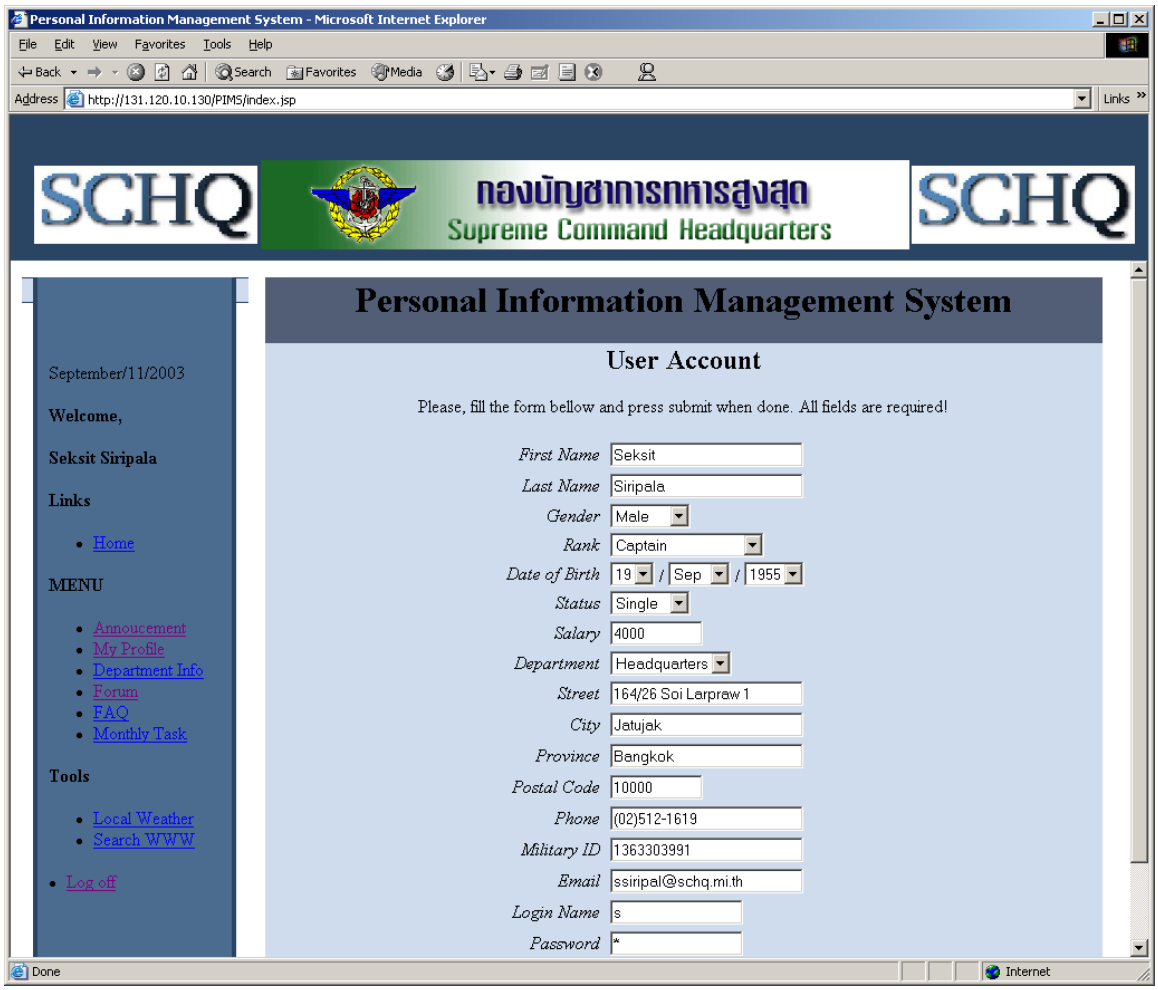

Figure 20. Screen Shot for Page for Updating Personal Information

This page allows users to update their personal information.

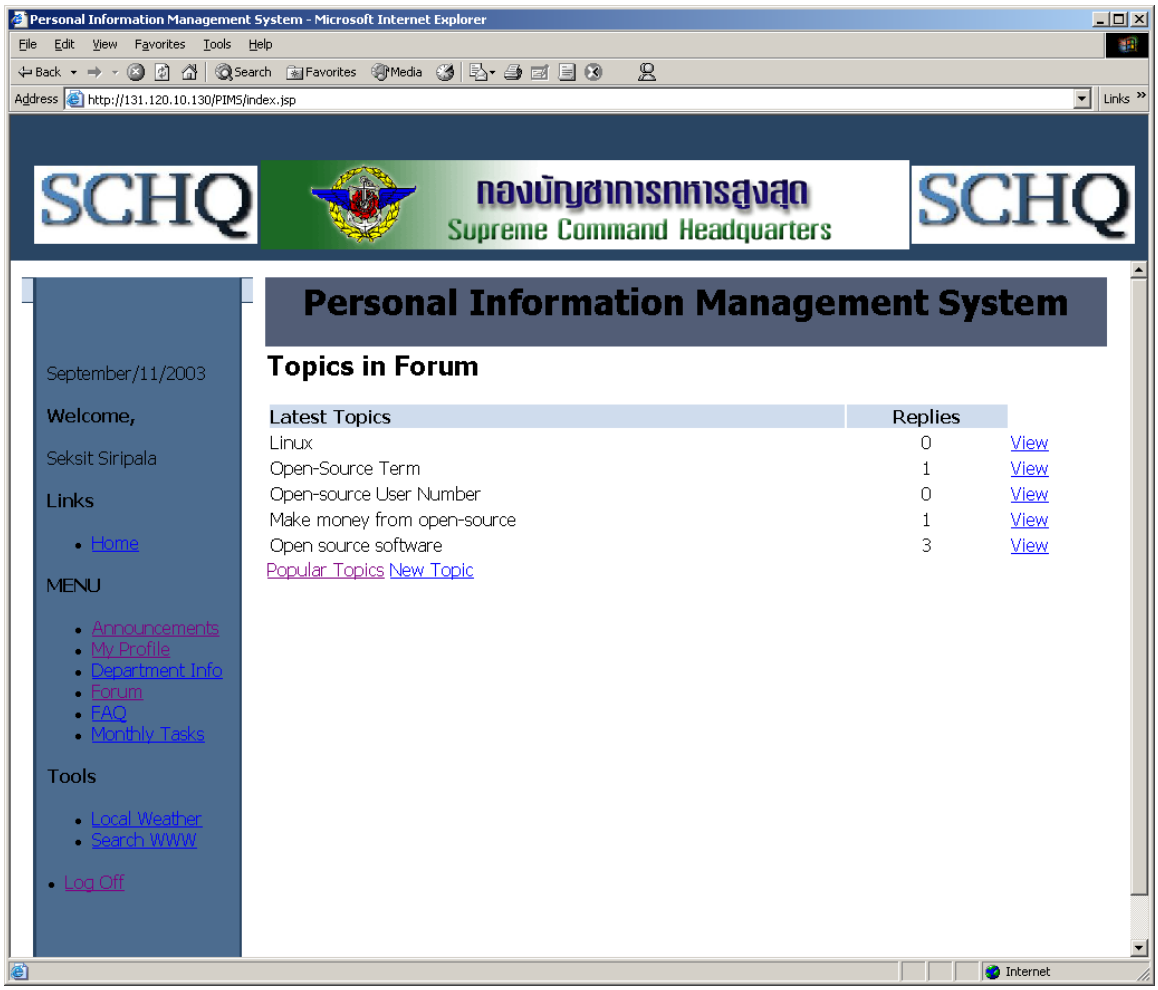

Figure 21. Screen Shot for Latest Topics in Discussion Board Page

This page displays the list of the latest topic on the discussion board. The users can click the hyperlinks to view the topics in popularity order.

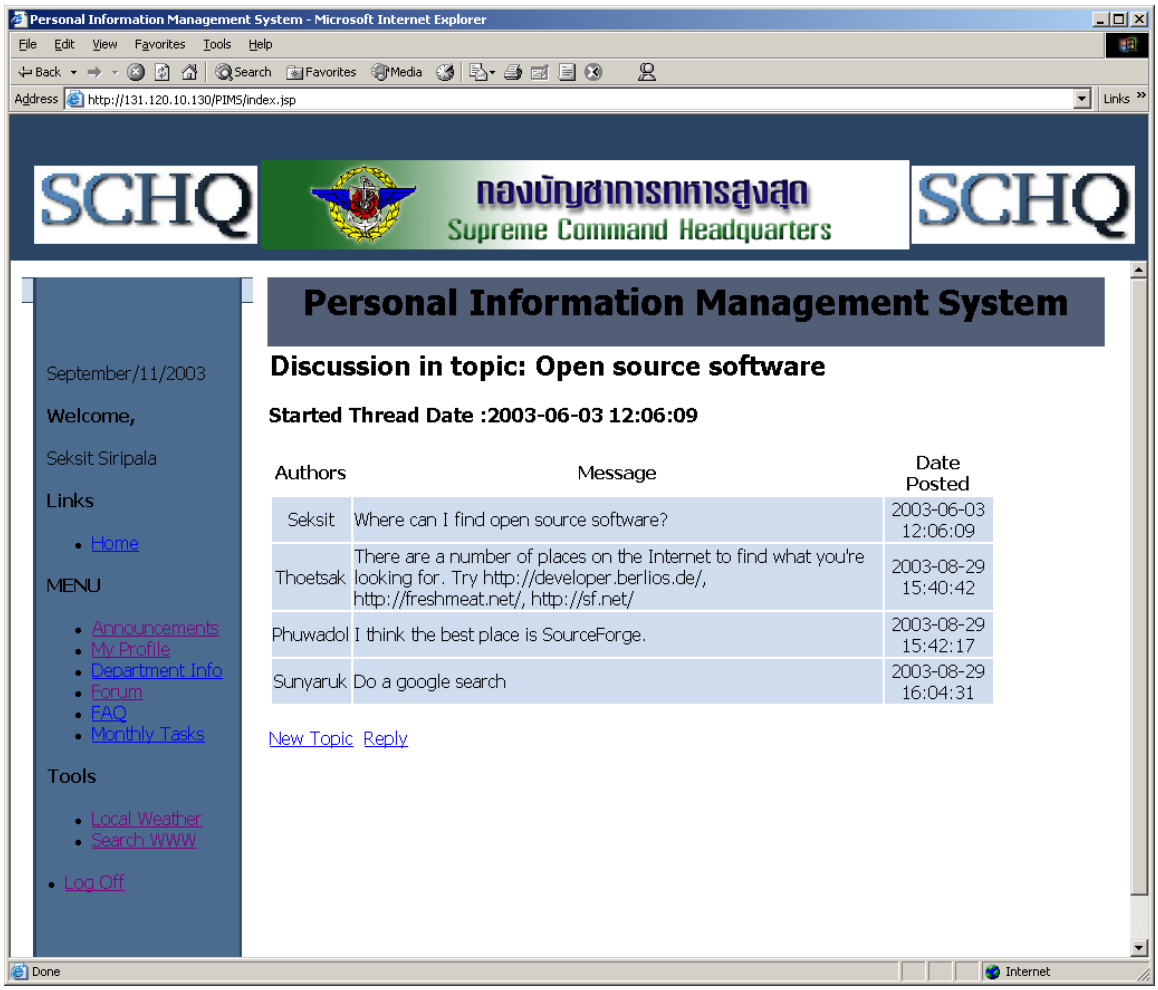

Figure 22. Screen Shot for Specific Topics Page

This page displays all comments posted in the chosen topic. It has hyperlinks to allow user to reply current topic or post a new topic.

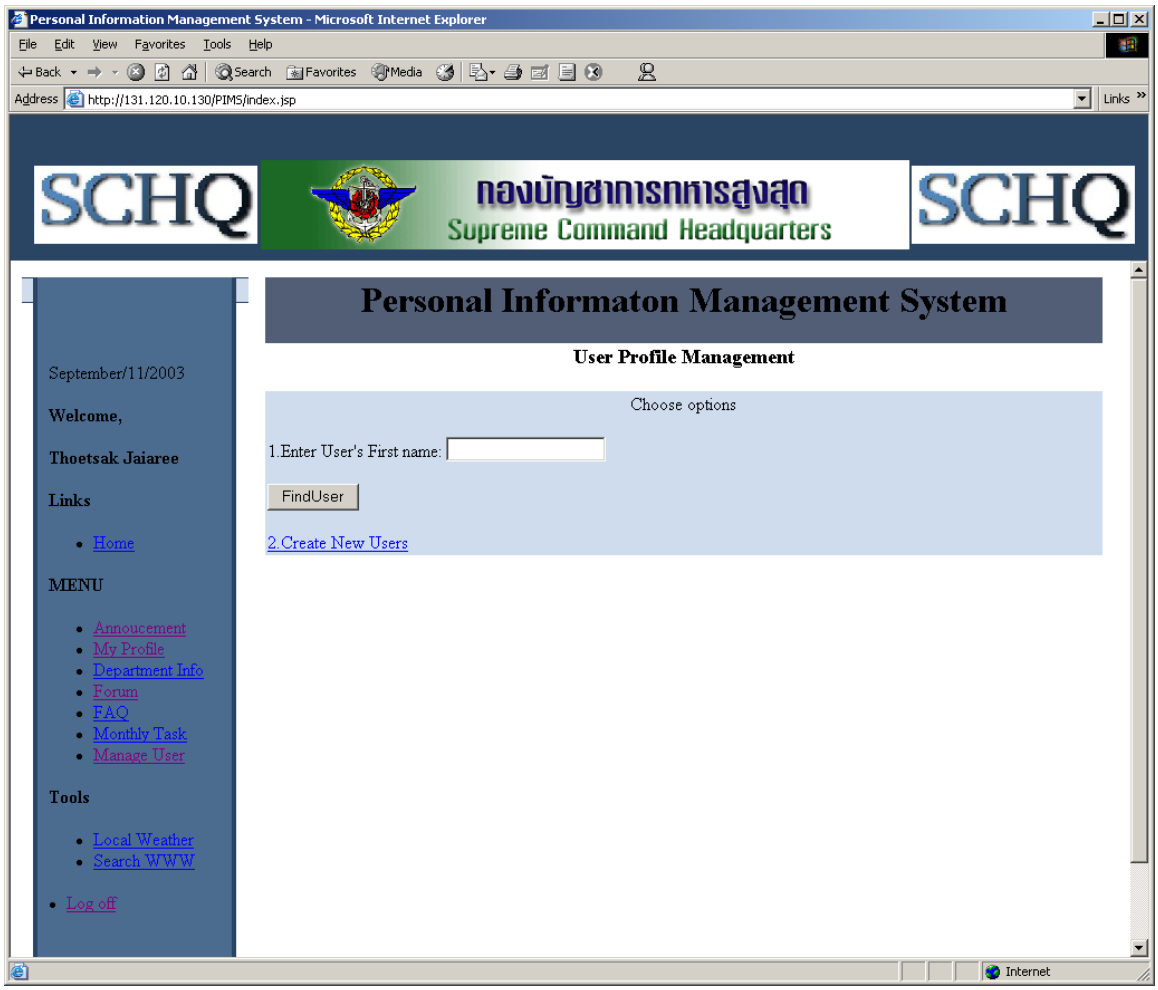

Figure 23. Screen Shot for Manage User Page

This page allows users (Administrator and Administrative office personnel) to manage other user profiles. The functions include updating the existing profiles and creating new ones.

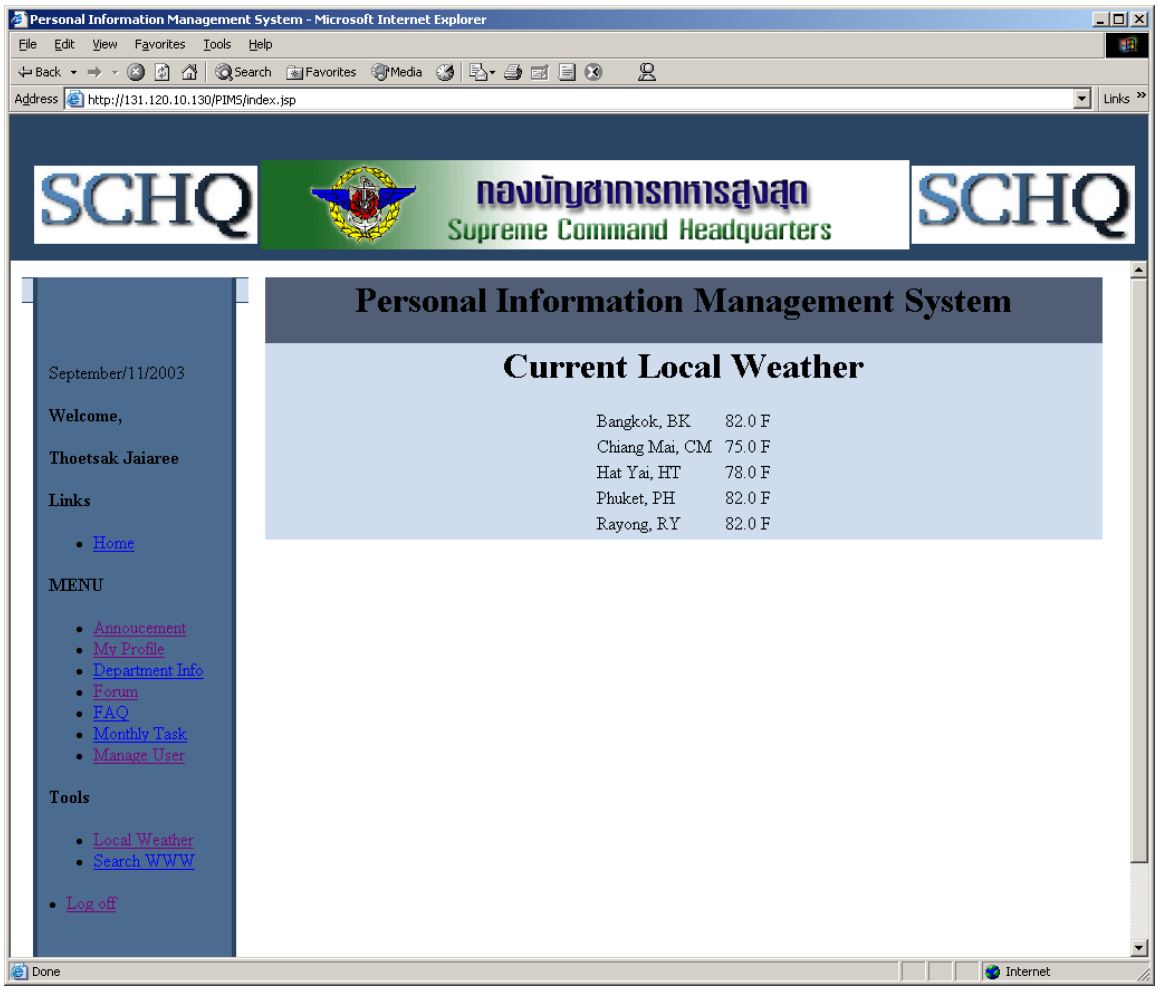

Figure 24. Screen Shot for the Local Weather Page

This page displays local weather information obtained by a "bot" we wrote.

THIS PAGE INTENTIONALLY LEFT BLANK

VII. CONCLUSIONS

A. SUMMARY OF THE PROTOTYPE

With the open-source and Java-Technologies approach to Web applications in this thesis, a prototype using the framework proposed in Chapter VI was implemented as an experiment. The design of the prototype use case was based on the functions of a personal-information management system for military organization. However, the system easily allows adding other functions or modules.

The prototype proved successful in using open-source technologies. Although the Java server-side technologies (Servlet/JSP) are time-consuming to develop and require some knowledge of the Java programming language, the performances of the Web application using those technologies are more scalable, powerful and secure than any other server-side technologies.

Similar ideas to the framework proposed in this thesis for building Web applications are commonly used today. Linux, Apache, Tomcat and MySQL are designed well for building a Web-based application system. Even though support from an open-source software vendor may not be good as from a proprietary software vendor, the overall expense of an open-source approach beats that of proprietary software.

Government organizations can benefit from the proposed approach because it reduces unnecessary expense in Web technologies. Migrating from the proprietary software to open-source software requires much work initially, but it benefits several areas of performance in the long term, including reliability, portability, maintainability, and economy.

B. FUTURE WORK

This thesis introduced an approach using open-source and Java Technologies for Web applications. To improve or enhance the performance of future Web applications, the following approaches are recommended.

The Java 2 Platform Enterprise Edition defines a standard for developing a multi-tier enterprise application. It simplifies enterprise applications by basing them on standardized, modular components, by providing the complete set of services of those components, and by handling many details of application behavior automatically without complex programming. Future work can take advantage of J2EE by integrating the existing technologies in this thesis (Servlet/JSP and JDBC API) with Enterprise JavaBeans (EJB) and XML to enhance the performance of Web application.

Currently, the prototype does not address Java's Model-View-Controller (M-V-C) architecture. This architecture becomes important when designing and developing a large-scale application. Completely separating the business logic and presentation output will make the application easy to develop and maintain. An open-source technology suitable for M-V-C architecture is that of "struts", an M-V-C framework released under the Apache Software license. Future work could take advantage of the M-V-C architecture by separating each part completely. This will be useful when implementing an enterprise application.

APPENDIX A. SOURCE CODE OF THE IMPLEMENTATION

A. SAMPLE SOURCE CODE OF SCHQADMIN PACKAGE

1. UserProfileBean

```
package schqAdmin;


/**
 * This class defines a bean used to maintain user data during the duration
 * of a session.
 * @author Seksit Siripala
 */
public class UserProfileBean {
  // Data members
  private int userID;
  private String firstName;
  private String lastName;
  private String gender;
  private String rank;
  private String dateOfBirth;
  private String status;
  private String salary;
  private String departmentCode;
  private String commanderID;
  private String street;
  private String city;
  private String postalCode;
  private String province;
  private String phone;
  private String militaryID;
  private String pictureFileName;
  private String userType;
  private String loginName;
  private String email;
  private String password;
  /**
   * Default constructor.
   */
  public UserProfileBean()
  {
    // Initialize properties to a valid state
```

```
  getReset();
}
/**
 * Resets all data
 */
public String getReset()
{
  userID = -1;
  firstName = new String("");
  lastName = new String("");
  gender = new String("");
  rank = new String("");
  dateOfBirth = new String("01Jan1983");
  status = new String("");
  salary = new String("");
  departmentCode = new String("");
  commanderID = new String("");
  street = new String("");
  city = new String("");
  postalCode = new String("");
  province = new String("");
  phone = new String("");
  militaryID = new String("");
  pictureFileName = new String("/schq/images/defaultPix.jpg");
  userType = new String("3");
  loginName = new String("");
  password = new String("");
  email = new String("");
  return "";
}
public int getUserID()
{
  return userID;
}
public void setUserID(int newUserID)
{
  userID = newUserID;
}
public String getFirstName()
{
  return firstName;
}
public void setFirstName(String newFirstName)
```

```
{
 firstName = newFirstName;
}
public String getLastName()
{
 return lastName;
}
 public void setLastName(String newLastName)
{
 lastName = newLastName;
}
public String getFullName()
{
 return (getFirstName() + " "  + getLastName());
}
 public String getGender()
{
 return gender;
}
 public void setGender(String newGender)
{
 gender = newGender;
}
 public String getRank()
{
 return rank;
}
public void setRank(String newRank)
{
 rank = newRank;
}
 public String getDateOfBirth()
{
 return dateOfBirth;
}
public void setDateOfBirth(String newDateOfBirth)
{
 dateOfBirth = newDateOfBirth;
}
 public String getBDate()
{
 return dateOfBirth.substring(0,2);
}
```

```
 public String getBMonth()
{
 return dateOfBirth.substring(2,5);
}
 public String getBYear()
{
 return dateOfBirth.substring(5);
}
public String getStatus()
{
 return status;
}
public void setStatus(String newStatus)
{
 status = newStatus;
}
public String getSalary()
{
 return salary;
}
public void setSalary(String newSalary)
{
 salary = newSalary;
}
public String getDepartmentCode()
{
 return departmentCode;
}
public void setDepartmentCode(String newDepartmentCode)
{
 departmentCode = newDepartmentCode;
}
public String getCommanderID()
{
 return commanderID;
}
public void setCommanderID(String newCommanderID)
{
 commanderID = newCommanderID;
}
public String getStreet()
{
 return street;
```

```
}
public void setStreet(String newStreet)
{
 street = newStreet;
}
public String getCity()
{
 return city;
}
public void setCity(String newCity)
{
 city = newCity;
}
public String getPostalCode()
{
 return postalCode;
}
public void setPostalCode(String newPostalCode)
{
 postalCode = newPostalCode;
}
public String getProvince()
{
 return province;
}
public void setProvince(String newProvince)
{
 province = newProvince;
}
public String getPhone()
{
 return phone;
}
public void setPhone(String newPhone)
{
 phone = newPhone;
}
public String getMilitaryID()
{
 return militaryID;
}
public void setMilitaryID(String newMilitaryID)
{
```

```
  militaryID = newMilitaryID;
}
public String getPictureFileName()
{
  return pictureFileName;
}
public void setPictureFileName(String newPictureFileName)
{
  pictureFileName = newPictureFileName;
}
public String getUserType()
{
  return userType;
}
public void setUserType(String newUserType)
{
  userType = newUserType;
}
public String getLoginName()
{
  return loginName;
}
public void setLoginName(String newLoginName)
{
  loginName = newLoginName;
}
public String getPassword()
{
  return password;
}
public void setPassword(String newPassword)
{
  password = newPassword;
}
public String getEmail()
{
  return email;
}
public void setEmail(String newEmail)
{
  email = newEmail;
}
public String toString()
```

```
  {
    String str = new String("User is:\n");
    str += "UserID = \t" + userID + "\n";
    str += "First Name = \t" + firstName + "\n";
    str += "Last name  = \t" + lastName + "\n";
    str += "Street     = \t" + street + "\n";
    str += "City       = \t" + city + "\n";
    str += "Postal Code= \t" + postalCode + "\n";
    str += "Province      = \t" + province + "\n";
    str += "Phone      = \t" + phone + "\n";
    str += "Military card No.  = \t" + militaryID + "\n";
    str += "Login name = \t" + loginName + "\n";
    str += "Password   = \t" + password + "\n";
    str += "Email      = \t" + email + "\n";
    return str;
  }
} // End of UserProfileBean class
```

2. DBConnectionBean

```
package schqAdmin;
import java.io.*;
import java.util.*;
import java.sql.*;

/**
 * This class implements a bean that is shared among all modules to process
 * all transactions with the backend database. It is a unique point of
 * connection with the database, using the JDBC API.
 * @author Serksit Siripala
 */
public class DBConnectionBean
{
  private boolean loaded;
  private boolean connected;
  private String url;
  private String driver;
  private String user;
  private String password;
  private Connection con;
  private Statement statement;
  private String query;
  private String update;
```

```
private String primaryKeyQuery;
/**
 * The default constructor.
 * @return void.
 */
public void DBConnectionBean()
{
  loaded = false;
  connected = false;
  url =CommonData.URL;
  driver =CommonData.DRIVER;
  user = CommonData.USER;
  password =CommonData.PASSWORD;
  con = null;
  query = new String("");
  primaryKeyQuery = new String();
}
/**
 * Set the DB url.
 * @param newUrl the new url.
 * @return void.
 */
public void setUrl(String newUrl)
{
  url = newUrl;
}
/**
 * Sets the driver.
 * @param newDriver the new driver.
 * @return void.
 */
public void setDriver(String newDriver)
{
  driver = newDriver;
}
/**
 * Sets the DB user name.
 * @param newUser the new user name.
 * @return void.
 */
public void setUser(String newUser)
{
  user = newUser;
```

```
}
/**
 * Sets the DB password.
 * @param newPassword the new password.
 * @return void.
 */
public void setPassword(String newPassword)
{
  password = newPassword;
}
/**
 * Loads the driver
 * @return true if the driver is loaded.
 */
public boolean isLoaded()
{
  try
  {
    Class.forName(driver);
    loaded = true;
  }
  catch(ClassNotFoundException cnfe)
  {
    loaded = false;
  }
  return loaded;
} // end of isLoaded()
/**
 * Establish the connection with the DB.
 * @return true if the connection is established.
 */
private boolean isConnected()
{
  try
  {
    con = DriverManager.getConnection(url, user, password);
    statement = con.createStatement();
    connected = true;
  }
  catch(SQLException sqle) {
    connected = false;
  }
  return connected;
```

```
}
/**
 * Sets the query string.
 * @param query the new query string.
 * @return void.
 */
public ResultSet query(String query)
{
  this.query = query;
  return getQuery();
}
/**
 * Queries the DB and return the result resultset.
 * @return the result set.
 */
public ResultSet getQuery()
{
  ResultSet rs = null;
  if(isConnected())
  {
    try
    {
      rs = statement.executeQuery(query);
    }
    catch(Exception e) {
    }
  }
  return rs;
}
/**
 * Sets the query.
 * @param query the new query.
 * @return void.
 */
public void setQuery(String query)
{
  this.query = query;
}
/**
 * Sets the update query.
 * @param update the update query.
 * @return void.
 */
```

```
public void setUpdate(String update)
{
 this.update = update;
}
/**
 * Executes an update query.
 * @param updateQuery the update query
 * @return void.
 */
public int update(String updateQuery)
{
 update = updateQuery;
 return getUpdate();
}
public int getUpdate()
{
 int result = -1;
 if(isConnected())
 {
  try
  {
   result = statement.executeUpdate(update);
   isClose();
  }
  catch(Exception e) {
   e.printStackTrace();
  }
 }
 return result;
}
public boolean isClose()
{
 try
 {
  con.close();
  statement.close();
 }
 catch(Exception e) {}
 connected = false;
 return true;
}
public void setPrimaryKeyQuery(String newPrimaryKeyQuery)
{
```

```
    primaryKeyQuery = newPrimaryKeyQuery;
  }
  /**
   * Obtains a new primary key in the table specified by the current primary.
   * The result set of the ordered set of existing primary keys is examined
   * sequentially, until the smalest non used positive integer is found.
   * This method may only be used for tables using integers as a primary key.
   * @return the new primary key or -1 if an error happens.
   */
  public int getPrimaryKey()
  {
    int primaryKey = -1;
    try
    {
      ResultSet rs = query(primaryKeyQuery);
      if(rs.next())
      {
        primaryKey = rs.getInt(1);
        do
        {
          ++primaryKey;
        }while(rs.next() && (primaryKey == rs.getInt(1)));
      }
    }
    catch(SQLException e)
    {
      e.printStackTrace();
      primaryKey = -1;
    }
    return primaryKey;
  }
} // End of DBConnectionBean class
```

3. Login Servlet

```
package schqAdmin;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;
import java.net.*;
import java.text.*;
```

```
/**
 * The class Login defines a servlet that is responsible to authenticate
 * users trying to login as user. Upon positive authentication,
 * a new user session is created with two attached beans: the user profile
 * bean and the database connection module.
 * @author Seksit Siripala
 */
public class Login extends HttpServlet
{
 // Data members
 private final int DRIVER_ERROR = 1;
 private final int CONNECTION_ERROR = 2;
 private final int QUERY_ERROR = 3;
 private final int INVALID_USER = 4;
 private final int ADMIN_USER = 1;
 private final int COMMANDER_USER = 2;
 private final int PERSONEL_USER = 3;
 private final int OFFICER_USER = 4;
 private HttpServletRequest request;
 private HttpServletResponse response;
 private Connection connection;
 private PrintWriter out;
 private ResultSet rs;
 private DBConnectionBean dbBean;
 /**
  * One-time initilization of the servlet. If the Connection Module has not
  * yet initiated, it ensures its creation.
  * @param config local server configuration parameters.
  * @return void.
  * @exception ServletException
  */
 public void init(ServletConfig config) throws ServletException
 {
  super.init(config);
  rs = null;                          // for the result set
  DBConnectionBean dbBean =
   (DBConnectionBean)getServletContext().getAttribute("connectionBean");
  if(dbBean == null)
  {
   dbBean = new DBConnectionBean();       // for the connection module
   // Initialize the DB connection module
   dbBean.setDriver(CommonData.DRIVER);
   dbBean.setUrl(CommonData.URL);
```

```
    dbBean.setUser(CommonData.USER);
    dbBean.setPassword(CommonData.PASSWORD);
    // Load the driver
    if(!dbBean.isLoaded())
    {
      exitPoint(DRIVER_ERROR);
    }
  }
  this.dbBean = dbBean;
 getServletContext().setAttribute("connectionBean", dbBean);
}  // end of init()
/**
 * doPost processing. The same as doGet.
 * @param request the client request.
 * @param response the response to client.
 * @return void.
 * @exception ServletException.
 * @exception IOException.
 */
public void doPost(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException
{
  // Simply pass pass control to doGet()
  this.doGet(request, response);
  return;
} // end of doPost
/**
 * doGet processing. The same as doPost.
 * @param request the client request.
 * @param response the response to client.
 * @return void.
 * @exception ServletException.
 * @exception IOException.
 */
public void doGet(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException
{
  getServletContext().setAttribute("connectionBean", dbBean);
  // Initialization
  this.request = request;
  this.response = response;
  out = response.getWriter();
  // Get the login parameters from the request
```

```
  String name = request.getParameter("loginName");
  String password = request.getParameter("password");
 //displayOutput(name+"  " +password);
  validateUser(name, password);
} // end of doGet()
/**
 * Authenticates a user with the given name and password.
 */
private void validateUser(String name, String password)
{
  rs = dbBean.query(
    "SELECT * " +
    "FROM Users " +
    "WHERE LoginName like '"+ name +"' " +
    "AND Password  like '"+ password +"'");
  try
  {
    if(rs.next())
    {
      // Successful user login
         startSession();
    }
    else
    {
      // Invalid login data
        exitPoint(QUERY_ERROR);
    }
  }
  catch(Exception e)
  {
     exitPoint(QUERY_ERROR);
  }
} // end of validateUser()
/**
 * Starts a new session after a user is correctly authenticated.
 * @return void.
 * @exception Exception.
 */
private void startSession() throws Exception
{
    RequestDispatcher dispatcher = null;
    switch(rs.getInt("UserType"))
  {
```

```
      case(ADMIN_USER):
        // Get the dispatcher
         dispatcher =
              getServletContext().getRequestDispatcher("/jsp/Admin/AdminUserOptions.jsp");
       break;
       case(COMMANDER_USER):
        // Get the dispatcher
         dispatcher =
              getServletContext().getRequestDispatcher("/jsp/Commander/CommanderUserOptions.jsp");
       break;
       case(PERSONEL_USER):
      // Get the dispatcher
        dispatcher =
              getServletContext().getRequestDispatcher("/jsp/Personnel/PersonelUserOptions.jsp");
       break;
      case(OFFICER_USER):
       // Get the dispatcher
        dispatcher =
              getServletContext().getRequestDispatcher("/jsp/User/OfficerUserOptions.jsp");
       break;
      default:
      exitPoint(999);
    }//end switch
  if (dispatcher == null)
    {
     // No dispatcher means the given file could not be found
     response.sendError(response.SC_NO_CONTENT);
    }
    else
    {
     // Get or start a new session for this user
     HttpSession session = request.getSession(true);
     if(!session.isNew())
     {
      // Ensure the session is newly created
      session.invalidate();
      session = request.getSession(true);
     }
     // Create user object as a bean. Set it with the user information.
     UserProfileBean user = new UserProfileBean();
     user.setUserID(rs.getInt("UserID"));
     user.setFirstName(rs.getString("Fname"));
     user.setLastName(rs.getString("Lname"));
```

```
    user.setGender(rs.getString("Gender"));
    user.setRank(rs.getString("Rank"));
    user.setDateOfBirth(rs.getString("DateOfBirth"));
    user.setStatus(rs.getString("Status"));
    user.setSalary(rs.getString("Salary"));
    user.setDepartmentCode(rs.getString("DepartmentCode"));
    user.setCommanderID(rs.getString("CommanderID"));
    user.setStreet(rs.getString("Street"));
    user.setCity(rs.getString("City"));
    user.setPostalCode(rs.getString("PostalCode"));
    user.setProvince(rs.getString("Province"));
    user.setPhone(rs.getString("Phone"));
    user.setMilitaryID(rs.getString("MilitaryID"));
    user.setPictureFileName(rs.getString("PictureFileName"));
    user.setUserType(rs.getString("UserType"));
    user.setLoginName(rs.getString("LoginName"));
    user.setPassword(rs.getString("Password"));
    user.setEmail(rs.getString("Email"));
    // Close connection
    dbBean.isClose();
    // Attach user bean to this session object
    session.setAttribute("user", user);
    // Pass control to a different page
    dispatcher.forward(request, response);
  }
} // end of startSession()
/**
 * Defines several types of exit conditions depending on the specified
 * exit time.
 * @param the time of exit.
 * @return void.
 */
private void exitPoint(int exitCondition)
{
  String output = new String();
  switch(exitCondition)
  {
    case DRIVER_ERROR:
      output += "Application error: Driver eror ";
       break;
    case CONNECTION_ERROR:
      output += "Application error: unable to establish connection to database";
      break;
```

```
            case QUERY_ERROR:
              output += "Application error: invalid database query";
              break;
            case INVALID_USER:
              output += "User not found";
              break;
             default:
              output += "Application error";
           } // end switch
          displayOutput(output);
         }//end method
          private void displayOutput(String text){
          out.println(ServletUtilities.headWithTitle("SCHQ Login"));
          out.println("<body bgcolor=\"#cfdced\">");
          out.println("<H3>SCHQ login process</H3>");
          out.println("<P>" + text + "</P>\n");
          out.println("<P>");
          out.println(ServletUtilities.hyperLink(response,
            "/PIMS/login.jsp", "Try to login again") + "<br>\n");
          out.println(ServletUtilities.hyperLink(response,
            "http://www.schq.mi.th", "Home"));
          out.println("<P>\n");
          out.println("</BODY>\n");
          out.println("</HTML>");
         } // end displayOutput
} // end of Login class
```

4. ProcessUserProfile Servlet

```
package schqAdmin;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;

/**
 * The class defines a servlet that is responsible to process user's information.
 * This class retrive data from frontend user and update that data into database.
 * @author Seksit Siripala
 */
public class ProcessUserProfile extends HttpServlet
{
  // Data members
```

```
private final int ERROR_LOADING_DRIVER = 1;
private final int ERROR_SESSION = 2;
private final int ERROR_RETRIVE_DATA = 3;
private final int ERROR_QUERY = 4;
private final int ERROR_DUPLICATE_LOGINNAME = 5;
private final int ERROR_ACC_UPDATING = 6;
private final int SUCCESS_ACC_UPDATED = 7;
private HttpServletRequest request;
private HttpServletResponse response;
private PrintWriter out;
private ResultSet rs;
private DBConnectionBean dbBean;
private UserProfileBean user;
private String errorInfo;
/**
 * One-time initilization of the servlet. If the Connection Module is not
 * yet initiated, it ensures its creation.
 * @param config local server configuration parameters.
 * @return void.
 * @exception ServletException
 */
public void init(ServletConfig config) throws ServletException
{
 super.init(config);
 // Check if already exists the connection bean
 DBConnectionBean dbBean =
  (DBConnectionBean)getServletContext().getAttribute("connectionBean");
 if(dbBean == null)
 {
  // Did not exist. Lets create it.
  dbBean = new DBConnectionBean();       // for the connection module
  // Initialize the DB connection module
  dbBean.setDriver(CommonData.DRIVER);
  dbBean.setUrl(CommonData.URL);
  dbBean.setUser(CommonData.USER);
  dbBean.setPassword(CommonData.PASSWORD);
  // Load the driver
  if(!dbBean.isLoaded())
  {
   exitPoint(ERROR_LOADING_DRIVER);
  }
 }
 this.dbBean = dbBean;
```

```
  // Place the connection bean in the servlet context (application scope)
  getServletContext().setAttribute("connectionBean", dbBean);
}  // end of init()
/**
 * doPost processing. The same as doGet.
 * @param request the client request.
 * @param response the response to client.
 * @return void.
 * @exception ServletException.
 * @exception IOException.
 */
public void doPost(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException
{
  // Simply pass pass control to doGet()
  this.doGet(request, response);
  return;
} // end of doPost
/**
 * doGet processing. The same as doPost.
 * @param request the client request.
 * @param response the response to client.
 * @return void.
 * @exception ServletException.
 * @exception IOException.
 */
public void doGet(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException
{
  // Startup settings
  this.request = request;
  this.response = response;
  response.setContentType("text/html");
  out = response.getWriter();
  // Check is session exists and is in a valid state
  HttpSession session = request.getSession(true);
  if(session.isNew()) {
    // Not a valid session. Should abort
    exitPoint(ERROR_SESSION);
  }
  else
  {
    // This is a valid session - retrieve information
```

```
    user = (UserProfileBean)session.getAttribute("user");
    dbBean = (DBConnectionBean)getServletContext().getAttribute("connectionBean");
          // user modifies his account
             if(isValidateData() == true){
               if(!isDuplicateLoginName()){
                     modifyAccount();
               }
               else{
                    exitPoint(ERROR_DUPLICATE_LOGINNAME);
               }
             }
             else{
               exitPoint(ERROR_RETRIVE_DATA);
          }
    }
} // end of doGet()
  /**
 * Modifies an existing user account by updating data with the information
 * submited in the request.
 * @return void.
 */
private void modifyAccount()
{
  retrieveData();
  int result = dbBean.update("UPDATE Users SET "
    + "Fname = " + "'" +user.getFirstName()  + "', "
    + "Lname = " + "'" + user.getLastName()  + "', "
    + "Gender = " + "'" + user.getGender() + "', "
    + "Rank = " + "'" + user.getRank() + "', "
    + "Status = " + "'" + user.getStatus() + "', "
    + "DateOfBirth = "  + "'" + user.getDateOfBirth() + "', "
    + "Salary = "  + "'" + user.getSalary() + "', "
    + "DepartmentCode = " + "'" + user.getDepartmentCode()  + "', "
    + "CommanderID = " + "'" + user.getCommanderID() + "', "
    + "Street = " + "'" + user.getStreet() + "', "
    + "City = "+ "'" + user.getCity()  + "', "
    + "PostalCode = " + "'" + user.getPostalCode() + "', "
    + "Province = "+ "'" + user.getProvince()  + "', "
    + "Phone = " + "'" + user.getPhone()  + "', "
    + "MilitaryID = " + "'" + user.getMilitaryID() + "', "
    + "UserType = " + "'" + user.getUserType() + "', "
    + "LoginName = "  + "'" + user.getLoginName()     + "', "
    + "Password = "  + "'" + user.getPassword() + "', "
```

```
    + "Email = " + "'" + user.getEmail()  + "', "
    + "PictureFileName = " + "'" + user.getPictureFileName() + "' "
    + "WHERE UserID = " + user.getUserID()
   );
   if(result == 1)
   {
    exitPoint(SUCCESS_ACC_UPDATED);
   }
   else
   {
    exitPoint(ERROR_ACC_UPDATING);
   }
  } // end of modifyAccount()
  private boolean isValidateData(){
     boolean status = true;
     String fName = request.getParameter("firstName");
     String lName = request.getParameter("lastName");
     String salary = request.getParameter("salary");
     String street = request.getParameter("street");
     String city = request.getParameter("city");
     String province = request.getParameter("province");
     String postalCode = request.getParameter("postalCode");
     String phone = request.getParameter("phone");
     String militaryID = request.getParameter("militaryID");
     String loginName = request.getParameter("loginName");
     String password = request.getParameter("password");
     String email = request.getParameter("email");
     errorInfo = new String();
     if(fName.length() == 0){
            errorInfo += ("missing firstname !<br>");
               status = false;
     }
     if(lName.length() == 0){
           errorInfo += ("missing lastname !<br>");
               status = false;
     }
     if(salary.length() == 0){
           errorInfo += ("missing salary !<br>");
               status = false;
     }
     if(street.length() == 0){
           errorInfo += ("missing street !<br>");
               status = false;
```

```
    }
    if(city.length() == 0){
            errorInfo += ("missing city !<br>");
              status = false;
    }
    if(province.length() == 0){
            errorInfo += ("missing province !<br>");
              status = false;
    }
    if(postalCode.length() == 0){
            errorInfo += ("missing postalCode !<br>");
              status = false;
    }
    if(militaryID.length() == 0){
           errorInfo += ("missing militaryID !<br>");
              status = false;
    }
    if(loginName.length() == 0){
            errorInfo += ("missing loginName !<br>");
              status = false;
    }
    if(password.length() == 0){
           errorInfo += ("missing password !<br>");
              status = false;
    }
    if(email.length() == 0){
           errorInfo += ("missing email !<br>");
              status = false;
    }
    return status;
     }//end isvalidateData
  private boolean isDuplicateLoginName(){
    boolean temp = false;
    String loginName = request.getParameter("loginName");
    String firstName = user.getFirstName();
    rs = dbBean.query(
    "SELECT * " +
    "FROM Users " +
    "WHERE LoginName like '"+loginName +"' " +
    "AND FName  not like '"+ firstName +"'");
    try
      {
         if(rs.next()){
```

```
            temp = true;
        }
     }
    catch(Exception e)
    {
          displayOutput(e.toString());
          exitPoint(ERROR_QUERY);
    }
      return temp;
    }//end isDuplicateName
  /**
   * Retrieves the data from the request and assignes them to the user
   * object.
   * @return void.
   */
  private void retrieveData()
  {
    user.setFirstName(ServletUtilities.filter(request.getParameter("firstName")));
    user.setLastName(request.getParameter("lastName"));
    user.setGender(request.getParameter("gender"));
    user.setRank(request.getParameter("rank"));
    user.setStatus(request.getParameter("status"));
    user.setDateOfBirth(retriveDateOfBirth());
    user.setSalary(request.getParameter("salary"));
    user.setDepartmentCode(request.getParameter("departmentCode"));
    user.setCommanderID(request.getParameter("commanderID"));
    user.setStreet(request.getParameter("street"));
    user.setCity(request.getParameter("city"));
    user.setPostalCode(request.getParameter("postalCode"));
    user.setProvince(request.getParameter("province"));
    user.setPhone(request.getParameter("phone"));
    user.setMilitaryID(request.getParameter("militaryID"));
    user.setLoginName(request.getParameter("loginName"));
    user.setPassword(request.getParameter("password"));
    user.setEmail(request.getParameter("email"));
    System.err.println(user);
  } // end of retrieveData()e
 public String retriveDateOfBirth()
 {
    StringBuffer dateOfBirth = new StringBuffer().append(request.getParameter("bDate"));
    dateOfBirth = dateOfBirth.append(request.getParameter("bMonth"));
    dateOfBirth = dateOfBirth.append(request.getParameter("bYear"));
    String dateOfBirthInfo = dateOfBirth.toString();
```

```
  return dateOfBirthInfo;
}//end retriveDateOfBirth
 /**
  * Defines several types of exit conditions depending on the specified
  * exit time.
  * @param the time of exit.
  * @return void.
  */
 private void exitPoint(int exitCondition)
 {
  String output = new String();
  output += ServletUtilities.headWithTitle("Process user Profile");
  output += "<body bgcolor=\"#cfdced\">";
  // Conditional message
  switch(exitCondition)
  {
   case ERROR_LOADING_DRIVER:
    output +=
     "<h3>Application error</h3>\n" +
     "<p>Error generated when trying to load DB connection driver</p>\n";
    break;
   case ERROR_SESSION:
    output +=
     "<h3>Session error</h3>\n" +
     "<p>The system is not able to process your request.<br>\n" +
     "Access denied!</p>\n";
    break;
   case ERROR_RETRIVE_DATA:
    output +=
     "<h3>Application error</h3>\n" +
     "<p>There was an error when trying to update your account!, cannot retrive data</p>\n";
    break;
   case ERROR_QUERY:
    output +=
     "<h3>QUERY ERROR</h3>\n";
     break;
   case ERROR_DUPLICATE_LOGINNAME:
    output +=
     "<h3>Application error</h3>\n" +
     "<p>There was an error when trying to update your account! Duplicate Login Name</p>\n";
    break;
   case ERROR_ACC_UPDATING:
    output +=
```

```
      "<h3>Application error</h3>\n" +
      "<p>There was an error when trying to update your account! </p>\n";
    break;
   case SUCCESS_ACC_UPDATED:
    output +=
      "<h3>Account updated</h3>\n" +
      "<p>Your account was sucessfully updated.</p>\n";
    break;
  } // end of switch
  // Conditional exit buttons
  switch(exitCondition)
  {
   case ERROR_LOADING_DRIVER:
   case ERROR_SESSION:
   case ERROR_RETRIVE_DATA:
    output += errorInfo;
    output += "<p>" + ServletUtilities.hyperLink(response,
     "/PIMS/jsp/User/EditOfficerUserProfile.jsp", "Try again") + "</p>\n";
    break;
   case ERROR_QUERY:
   case ERROR_DUPLICATE_LOGINNAME:
     output += "<p>" + ServletUtilities.hyperLink( response,
      "/PIMS/jsp/User/EditOfficerUserProfile.jsp", "Try again") + "</p>\n";
     break;
   case ERROR_ACC_UPDATING:
     output += "<p>" + ServletUtilities.hyperLink(response,
      "/PIMS/jsp/User/EditOfficerUserProfile.jsp", "Try again") + "</p>\n";
     break;
   case SUCCESS_ACC_UPDATED:
    output += "<p>" + ServletUtilities.hyperLink(response,
     "/PIMS/jsp/User/OfficerUserOptions.jsp", "Continue") + "</p>\n";
    break;
  } // end of switch
  out.println(output);
  out.println("</BODY>\n");
  out.println("</HTML>");
 } // end of exitPoint()
 private void displayOutput(String text){
  out.println(ServletUtilities.headWithTitle("Process User Profile"));
  out.println("<body bgcolor=\"#cfdced\">");
  out.println("<H3>SCHQ process</H3>");
  out.println("<P>" + text + "</P>\n");
  out.println("<P>");
```

```
    out.println(ServletUtilities.hyperLink(response,
      "/PIMS/login.jsp", "Try to login again") + "<br>\n");
    out.println("<P>\n");
    out.println("</BODY>\n");
    out.println("</HTML>");
     } // end displayOutput
} // end of ProcessUserProfile class
```

5. Announcements Servlet

```
package schqAdmin;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;
import dates.*;
/**
 * This class provides the processing module to display the available announcements
 *
 */
public class Announcements extends HttpServlet
{
  // Data members
  private final int ERROR_LOADING_DRIVER = 1;
  private final int ERROR_SESSION = 2;
  private final int ERROR_QUERY = 3;
  private HttpServletRequest request;
  private HttpServletResponse response;
  private PrintWriter out;
  private ResultSet rs;
  private DBConnectionBean dbBean;
  private UserProfileBean user;
  private JspCalendar d;
  private String output;
  /**
   * doPost processing. The same as doGet.
   * @param request the client request.
   * @param response the response to client.
   * @return void.
   * @exception ServletException.
   * @exception IOException.
   */
```

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException
{
  // Simply pass pass control to doGet()
  this.doGet(request, response);
  return;
} // end of doPost
/**
 * doGet processing. The same as doPost.
 * @param request the client request.
 * @param response the response to client.
 * @return void.
 * @exception ServletException.
 * @exception IOException.
 */
public void doGet(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException
{
  // Startup settings
  this.request = request;
  this.response = response;
  response.setContentType("text/html");
  out = response.getWriter();
  d = new JspCalendar();
  output = new String();
  // Check is session exists and is in a valid state
  HttpSession session = request.getSession(true);
  if(session.isNew()) {
    // Not a valid session. Should abort
    exitPoint(ERROR_SESSION);
  }
  else
  {
    dbBean =(DBConnectionBean)getServletContext().getAttribute("connectionBean");
    user = (UserProfileBean)session.getAttribute("user");
    queryAnnouncements();
    displayAnnouncements();
    dbBean.isClose();
  }
} // end of doGet()
/**
 * Set the query string for available announcements
 * @return void
```

```
 */
private void queryAnnouncements()
{
        String qry =
                "SELECT AnnouncementID, Title, Content "+
                "FROM Announcements " +
                "WHERE PublishedStatus like 'on' ";
        rs = dbBean.query(qry);
} // end of queryAnnouncements()
private void displayAnnouncements(){
    output += ServletUtilities.headWithTitle("Announcements");
    output += "<BODY text=\"#000000\" bgColor=\"#ffffff\">";
    output += CommonData.navLayout;
    output += "<p>"+d.getDate()+"</p>"+
    "<p><b>Welcome,</b></p>"+user.getFullName()+
    "<p><strong>Links</strong></p>"+
    "<ul>";
    output += "<li><a>" + ServletUtilities.hyperLink(response,
             "http:/www/schq.mi.th","Home") + "</a></li></ul>";
    output += "<p><strong>MENU</strong></p>";
    output += "<ul><li><a>" + ServletUtilities.hyperLink(response,
            "/PIMS/servlet/schqAdmin.Announcements","Announcements") + "</a></li>";
    output += "<li><a>" + ServletUtilities.hyperLink(response,
            "/PIMS/jsp/User/OfficerUserProfile.jsp","My Profile") + "</a></li>";
    output += "<li><a>" + ServletUtilities.hyperLink(response,
            "/PIMS/servlet/schqAdmin.DepartmentInfo","Department Info") + "</a></li>";
    output += "<li><a>" + ServletUtilities.hyperLink(response,
            "/PIMS/servlet/forum.ListPopularTopics","Forum") + "</a></li>";
    output += "<li><a>" + ServletUtilities.hyperLink(response,
            "/PIMS/jsp/User/FAQ.jsp","FAQ") + "</a></li>";
    output += "<li><a>" + ServletUtilities.hyperLink(response,
            "/PIMS/servlet/schqAdmin.ListMonthlyTask","Monthly Tasks") + "</a></li>";
    if(user.getUserType().equals("2")){
             output += "<li><a>" + ServletUtilities.hyperLink(response,
            "/PIMS/servlet/schqAdmin.ListProject","Projects") + "</a></li>";
    }//end if
    if(user.getUserType().equals("1")|| user.getUserType().equals("3")){
              output += "<li><a>" + ServletUtilities.hyperLink(response,
      "/PIMS/jsp/Personnel/ManageUser.jsp","Manage User") + "</a></li>";
    }//end if
    output +="</ul>";
    output +="<p><strong>Tools</strong></p><ul>";
    output += "<li><a>" + ServletUtilities.hyperLink(response,
```

```
        "/PIMS/jsp/User/Weather.jsp","Local Weather") + "</a></li>";
 output += "<li><a href=\""+response.encodeURL("/PIMS/jsp/User/SearchWWW.jsp")+
            "\" target=\"_blank\" >Search WWW </a></li></ul>";
 output += "<li><a>" + ServletUtilities.hyperLink(response,
        "/PIMS/servlet/schqAdmin.Logoff","Log Off") + "</a></li>";
 output += CommonData.bottomNavLayout;
 output += CommonData.mainLayout;
try{
          while(rs.next()){
                    int aID = rs.getInt("AnnouncementID");
                    String title = rs.getString("Title");
                    String content = rs.getString("Content");
                    output + = "<tr>" +
                              "<td><blockquote>"+
                              "<p>" +
                              "<b>" +  title + "</b>\n" +
                              "</p>\n"+
                              "<p>" + content+"</p>\n";
                    if(user.getUserType().equals("1")){
                     output += "<a>" + ServletUtilities.hyperLink(response,
                              "/PIMS/jsp/Admin/EditAnnouncement.jsp?AID="+aID+
                              "&Title="+title+"&Announcement=old"+
                              "&Content="+content,"Edit") + "</a>";
                     output += " <a>" + ServletUtilities.hyperLink(response,
                              "/PIMS/servlet/schqAdmin.DeleteAnnouncement?AID="+aID,"Delete") + "</a>";
                    }//end if
                    output +="<HR>" +
                    "</blockquote></TD>" +
                    "</TR>";
          }// end while
          if(user.getUserType().equals("1")){
            output += "<td><a>" + ServletUtilities.hyperLink(response,
            "/PIMS/jsp/Admin/EditAnnouncement.jsp?AID=0&Title=UNDEFINED"+
            "&Content=UNDEFINED&Announcement=new","New") + "</a></td>";
          }//end if
          output += CommonData.bottomMainLayout;
          output += CommonData.footerLayout;
          output += "</body></html>";
          out.println(output);
        }
        catch(Exception e){
                    System.out.println(e.toString());
                    exitPoint(ERROR_QUERY);
```

```
            }//end try-catch
}
/**
 * Defines several types of exit conditions depending on the specified
 * exit time.
 * @param the time of exit.
 * @return void.
 */
private void exitPoint(int exitCondition)
{
 out.println(ServletUtilities.headWithTitle("Announcements"));
 out.println("<body bgcolor=\"#cfdced\">");
 switch(exitCondition)
 {
  case ERROR_LOADING_DRIVER:
    output +=
      "<h3>Application error</h3>\n" +
      "<p>Error generated when trying to load DB connection driver</p>\n";
    break;
  case ERROR_SESSION:
    output +=
      "<h3>Session error</h3>\n" +
      "<p>The system is not able to process your request.<br>\n" +
      "Access denied!</p>\n";
    break;
  case ERROR_QUERY:
    output +=
      "<h3>Application error</h3>\n" +
      "<p>There was an error when trying to query the database</p>\n";
    break;
            } // end of switch
 // Conditional exit buttons
 switch(exitCondition)
 {
  case ERROR_LOADING_DRIVER:
  case ERROR_SESSION:
  case ERROR_QUERY:
    output += "<p>" + ServletUtilities.hyperLink(response,
      "/PIMS/login.jsp", "Back to login page") + "</p>\n";
    break;
 } // end of switch
 out.println(output);
 out.println("</BODY>\n");
```

```
    out.println("</HTML>");
  } // end of exitPoint()
} // end of Announcements class
```

6. ProcessOfficerDependent Servlet

```
package schqAdmin;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;
import schqAdmin.*;
import dates.*;

/**
 * The class defines a servlet that is responsible to process user's dependent.
 * This class retrive data from frontend user and update that data into database.
 * @author Seksit Siripala
 */
public class ProcessOfficerDependent extends HttpServlet
{
  // Data members
  private final int ERROR_LOADING_DRIVER = 1;
  private final int ERROR_SESSION = 2;
  private final int ERROR_RETRIEVE_MESSAGE =3;
  private final int SUCCESS_CREATE_DEPENDENT = 4;
  private final int ERROR_CREATE_DEPENDENT = 5;
  private final int SUCCESS_UPDATE_DEPENDENT = 6;
  private final int ERROR_UPDATE_DEPENDENT = 7;
  private final int ERROR_DEPENDENTID_CREATION = 8;
  private HttpServletRequest request;
  private HttpServletResponse response;
  private PrintWriter out;
  private ResultSet rs;
  private DBConnectionBean dbBean;
  private UserProfileBean user;
  private JspCalendar d;
  private String name;
  private String bDate;
  private String sex;
  private String relationship;
  private int dependentID;
  /**
```

```
 * One-time initilization of the servlet. If the Connection Module is not
 * yet initiated, it ensures its creation.
 * @param config local server configuration parameters.
 * @return void.
 * @exception ServletException
 */
public void init(ServletConfig config) throws ServletException
{
 super.init(config);
 // Check if already exists the connection bean
 DBConnectionBean dbBean =
  (DBConnectionBean)getServletContext().getAttribute("connectionBean");
 if(dbBean == null)
 {
  // Did not exist. Lets create it.
  dbBean = new DBConnectionBean();       // for the connection module
  // Initialize the DB connection module
  dbBean.setDriver(CommonData.DRIVER);
  dbBean.setUrl(CommonData.URL);
  dbBean.setUser(CommonData.USER);
  dbBean.setPassword(CommonData.PASSWORD);
  // Load the driver
  if(!dbBean.isLoaded())
  {
   exitPoint(ERROR_LOADING_DRIVER);
  }
 }
 this.dbBean = dbBean;
 // Place the connection bean in the servlet context (application scope)
 getServletContext().setAttribute("connectionBean", dbBean);
}  // end of init()
/**
 * doPost processing. The same as doGet.
 * @param request the client request.
 * @param response the response to client.
 * @return void.
 * @exception ServletException.
 * @exception IOException.
 */
public void doPost(HttpServletRequest request, HttpServletResponse response)
 throws ServletException, IOException
{
 // Simply pass pass control to doGet()
```

```
 this.doGet(request, response);
 return;
} // end of doPost
/**
 * doGet processing. The same as doPost.
 * @param request the client request.
 * @param response the response to client.
 * @return void.
 * @exception ServletException.
 * @exception IOException.
 */
public void doGet(HttpServletRequest request, HttpServletResponse response)
 throws ServletException, IOException
{
 // Startup settings
 this.request = request;
 this.response = response;
 response.setContentType("text/html");
 out = response.getWriter();
 d = new JspCalendar();
 // Check is session exists and is in a valid state
 HttpSession session = request.getSession(true);
 if(session.isNew()) {
  // Not a valid session. Should abort
  exitPoint(ERROR_SESSION);
 }
 else
 {
  // This is a valid session - retrieve information
  user = (UserProfileBean)session.getAttribute("user");
  dbBean = (DBConnectionBean)getServletContext().getAttribute("connectionBean");
  name =(request.getParameter("Name"));
  bDate =(request.getParameter("BDate"));
  sex =(request.getParameter("Sex"));
  relationship =(request.getParameter("Relationship"));
  dependentID = Integer.parseInt(request.getParameter("DependentID"));
  String dependent = request.getParameter("Dependent");
   if(dependent.equals("new")){
    newDependent();
   }
   else{
    modifyDependent();
   }
```

```
  }
 } // end of doGet()
 /**
  * Creates a new user's dependent information with the information submited in the request.
  * @return void
  */
 private void newDependent()
 {
    // Get user id (obtain a new primary key)
  dbBean.setPrimaryKeyQuery(
    "SELECT DependentID FROM Dependent ORDER BY DependentID");
  int pk = dbBean.getPrimaryKey();
 //displayOutput("primary key "+pk);
  if(pk == -1)
  {
    // Error getting the primary key
    exitPoint(ERROR_DEPENDENTID_CREATION);
  }
  else
  {
    // Get the data from the request
  int uID = user.getUserID();
  int result = dbBean.update("INSERT INTO Dependent (UserID, DependentName, sex, BDate, "
                              +"Relationship, DependentID) VALUES("
                              + """+uID + "', "
                              + """ + ServletUtilities.filter(name) + "', "
                              + """ + sex + "', "
                              + """ + bDate + "', "
                              + """ + relationship + "', "
                              + """ + pk + "') " );
               if(result == 1){
              exitPoint(SUCCESS_CREATE_DEPENDENT);
               }
               else{
                      displayOutput("result"+ result);
                      exitPoint(ERROR_CREATE_DEPENDENT);
              }
    }//end else
  } // end of newDependent()
 /**
   * Modify dependent with the information submited in the request.
   * @return void
   */
```

```
private void modifyDependent()
{
   // Get the data from the request
   int uID = user.getUserID();
   int result = dbBean.update("UPDATE Dependent SET "
                + "UserID = " +"'"+uID + "', "
                + "DependentName = " +"'"+name+ "', "
                + "sex = " +"'"+sex+ "', "
                + "BDate = " +"'"+bDate+ "', "
                + "Relationship = " +"'"+relationship+"', "
                + "DependentID = " +"'"+dependentID+"' "
                + "Where DependentID = " +"'"+dependentID+"' );
   if(result == 1){
            exitPoint(SUCCESS_UPDATE_DEPENDENT);
  }
 else{
            displayOutput("result"+ result);
            exitPoint(ERROR_UPDATE_DEPENDENT);
 }
} // end of modifyDependent()
/**
 * Defines several types of exit conditions depending on the specified
 * exit time.
 * @param the time of exit.
 * @return void.
 */
private void exitPoint(int exitCondition)
{
 String output = new String();
 // Conditional message
 switch(exitCondition)
 {
  case ERROR_LOADING_DRIVER:
    output +=
      "<h3>Application error</h3>\n" +
      "<p>Error generated when trying to load DB connection driver</p>\n";
    break;
  case ERROR_SESSION:
    output +=
      "<h3>Session error</h3>\n" +
      "<p>The system is not able to process your request.<br>\n" +
      "Access denied!</p>\n";
    break;
```

```
 case ERROR_RETRIEVE_MESSAGE:
  output +=
   "<h3>Application error</h3>\n" +
   "<p>There was an error when trying to retrieve your message, please try again</p>\n";
  break;
 case SUCCESS_CREATE_DEPENDENT:
   output +=
   "<h3>Success Create</h3>\n" +
   "<p>Your request has been processed</p>\n";
  break;
 case ERROR_CREATE_DEPENDENT:
  output +=
   "<h3>Application error</h3>\n" +
   "<p>There was an error when trying to create dependent information</p>\n";
  break;
 case SUCCESS_UPDATE_DEPENDENT:
   output +=
   "<h3>Success Update</h3>\n" +
   "<p>Your request has been processed</p>\n";
  break;
 case ERROR_UPDATE_DEPENDENT:
  output +=
   "<h3>Application error</h3>\n" +
   "<p>There was an error when trying to update dependent information</p>\n";
  break;
 case ERROR_DEPENDENTID_CREATION:
  output +=
   "<h3>Application error</h3>\n" +
   "<p>There was an error when trying to create dependent ID</p>\n";
  break;
} // end of switch
// Conditional exit buttons
switch(exitCondition)
{
 case ERROR_LOADING_DRIVER:
 case ERROR_SESSION:
 case ERROR_RETRIEVE_MESSAGE:
     output += "<p>" + ServletUtilities.hyperLink(response,
     "/PIMS/servlet/schqadmin.ListOfficerDependent", "Try again") + "<br>\n";
  break;
  case SUCCESS_CREATE_DEPENDENT:
   output += "<p>" + ServletUtilities.hyperLink(response,
   "/PIMS/jsp/User/OfficerUserOptions.jsp", "Continue") + "</p>\n";
```

```
        break;
      case ERROR_CREATE_DEPENDENT:
          output += "<p>" + ServletUtilities.hyperLink(response,
          "/PIMS/jsp/User/EditDependentForm.jsp", "try again") + "</p>\n";
        break;
      case SUCCESS_UPDATE_DEPENDENT:
          output += "<p>" + ServletUtilities.hyperLink(response,
          "/PIMS/jsp/User/OfficerUserOptions.jsp", "Continue") + "</p>\n";
        break;
      case ERROR_UPDATE_DEPENDENT:
          output += "<p>" + ServletUtilities.hyperLink(response,
          "/PIMS/jsp/User/EditDependentForm.jsp", "try again") + "</p>\n";
        break;
    } // end of switch
    displayOutput(output);
  } // end of exitPoint()
   private void displayOutput(String text){
    out.println(ServletUtilities.headWithTitle("Process Dependent"));
    out.println("<body bgcolor=\"#cfdced\">");
    out.println("<P>" + text + "</P>\n");
    out.println("<P>\n");
    out.println("</BODY>\n");
    out.println("</HTML>");
  } // end displayOutput
} // end of ProcessOfficerDependent class
```

7. Logoff Servlet

```
package schqAdmin;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;

/**
 * This class defines a servlet that is logout a user and end user's session.
 * @author Seksit Siripala
 */
public class Logoff extends HttpServlet
{
  // Data members
  private final int ERROR_SESSION = 1;
  private HttpServletRequest request;
```

```
private HttpServletResponse response;
private PrintWriter out;
private ResultSet rs;
private DBConnectionBean dbBean;
private UserProfileBean user;
private String output = new String();
/**
 * doPost processing. The same as doGet.
 * @param request the client request.
 * @param response the response to client.
 * @return void.
 * @exception ServletException.
 * @exception IOException.
 */
public void doPost(HttpServletRequest request, HttpServletResponse response)
 throws ServletException, IOException
{
 // Simply pass pass control to doGet()
 this.doGet(request, response);
 return;
} // end of doPost
/**
 * doGet processing. The same as doPost.
 * @param request the client request.
 * @param response the response to client.
 * @return void.
 * @exception ServletException.
 * @exception IOException.
 */
public void doGet(HttpServletRequest request, HttpServletResponse response)
 throws ServletException, IOException
{
 // Startup settings
 this.request = request;
 this.response = response;
 response.setContentType("text/html");
 out = response.getWriter();
 output = "";
 processRequest();
} // end of doGet()
/**
 * Logs user out, terminating current user's session.
 * @return void
```

```
  */
 private void processRequest()
 {
  // Terminate the session for this user
  HttpSession session = request.getSession(false);
  if(session != null)
  {
   session.invalidate();
  }
  out.println(
   ServletUtilities.headWithTitle("Log Off")
   + "<body bgcolor=\"#cfdced\">"
   + "<p>Thank you for using On-line Personal Information Management System."
   + "<br><br>\n"
   + "<a href=\""+response.encodeURL("/PIMS/index.jsp")+
                "\" target=\"_top\" >Home </a>"
   + "</body>"
   + "</html>");
 } // end of processRequest()
} // end of Logoff class
```

B. SAMPLE SOURCE CODE OF FORUM PACKAGE

1. ListLatestTopics Servlet

```
package forum;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;
import dates.*;
import schqAdmin.*;
/**
 * This class defines a servlet that is responsible to list all task
 * in latest topics
 * @author Seksit Siripala
 */
public class ListLatestTopics extends HttpServlet
{
 // Data members
 private final int ERROR_SESSION = 1;
 private final int NO_TOPIC = 2;
 private final int TOPICS_LIST = 3;
```

```
private HttpServletRequest request;
private HttpServletResponse response;
private PrintWriter out;
private ResultSet rs;
private UserProfileBean user;
private DBConnectionBean dbBean;
private JspCalendar d;
private String output = new String();

/**
 * doPost processing. The same as doGet.
 * @param request the client request.
 * @param response the response to client.
 * @return void.
 * @exception ServletException.
 * @exception IOException.
 */
public void doPost(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException
{
  // Simply pass control to doGet()
  this.doGet(request, response);
  return;
} // end of doPost
/**
 * doGet processing. The same as doPost.
 * @param request the client request.
 * @param response the response to client.
 * @return void.
 * @exception ServletException.
 * @exception IOException.
 */
public void doGet(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException
{
  // Startup settings
  this.request = request;
  this.response = response;
  d = new JspCalendar();
  response.setContentType("text/html");
  out = response.getWriter();
  output = "";
  // Check if this is a valid session
```

```
 HttpSession session = request.getSession(false);
 if(session == null) {
  // Not a valid session. Should abort
  exitPoint(ERROR_SESSION);
 }
 else
 {
  // This is a valid session - retrieve information
  user = (UserProfileBean)session.getAttribute("user");
  dbBean = (DBConnectionBean)getServletContext().getAttribute("connectionBean");
  if((user != null)&&(user.getUserID()!=-1)&&(dbBean != null))
  {
   // Session is in a valid state
   processRequest();
  }
  else
  {
   // Session is not in a valid state
   exitPoint(ERROR_SESSION);
  }
 }
} // end of doGet()
/**
 * Displays latest discussion topics.
 * @return void
 */
private void processRequest()
{
 output += ServletUtilities.headWithTitle("Latest Topics");
 output += "<BODY text=\"#000000\" bgColor=\"#ffffff\">";
 output += CommonData.navLayout;
 //<!--LEFT SIDE NAVIGATION-->
 output += "<p>"+d.getDate()+"</p>"+
 "<p><b>Welcome,</b></p>"+user.getFullName()+
 "<p><strong>Links</strong></p>"+
 "<ul>";
  output += "<li><a>" + ServletUtilities.hyperLink(response,
            "http:/www/schq.mi.th","Home") + "</a></li></ul>";
  output += "<p><strong>MENU</strong></p>";
  output += "<ul><li><a>" + ServletUtilities.hyperLink(response,
           "/PIMS/servlet/schqAdmin.Announcements","Announcements") + "</a></li>";
  output += "<li><a>" + ServletUtilities.hyperLink(response,
           "/PIMS/jsp/User/OfficerUserProfile.jsp","My Profile") + "</a></li>";
```

```
output += "<li><a>" + ServletUtilities.hyperLink(response,
          "/PIMS/servlet/schqAdmin.DepartmentInfo","Department Info") + "</a></li>";
output += "<li><a>" + ServletUtilities.hyperLink(response,
          "/PIMS/servlet/forum.ListPopularTopics","Forum") + "</a></li>";
output += "<li><a>" + ServletUtilities.hyperLink(response,
          "/PIMS/jsp/User/FAQ.jsp","FAQ") + "</a></li>";
output += "<li><a>" + ServletUtilities.hyperLink(response,
          "/PIMS/servlet/schqAdmin.ListMonthlyTask","Monthly Tasks") + "</a></li>";
if(user.getUserType().equals("1")|| user.getUserType().equals("3")){
           output += "<li><a>" + ServletUtilities.hyperLink(response,
                    "/PIMS/jsp/Personnel/ManageUser.jsp","Manage User") + "</a></li>";
}//end if
output +="</ul>";
output +="<p><strong>Tools</strong></p><ul>";
output += "<li><a>" + ServletUtilities.hyperLink(response,
          "/PIMS/jsp/User/Weather.jsp","Local Weather") + "</a></li>";
output += "<li><a href=\""+response.encodeURL("/PIMS/jsp/User/SearchWWW.jsp")+
            "\" target=\"_blank\" >Search WWW </a></li></ul>";
output += "<li><a>" + ServletUtilities.hyperLink(response,
          "/PIMS/servlet/schqAdmin.Logoff","Log Off") + "</a></li>";
output += CommonData.bottomNavLayout;
output += CommonData.mainLayout;
try
{
 ResultSet rs = dbBean.query(
   "SELECT distinct Topics, StartedThreadDate, Reply " +
   "FROM SCHQForumn " +
   "ORDER BY StartedThreadDate desc");
 if(rs.next())
 {
    String popularTopicsHyperLink = ServletUtilities.hyperLink(response,
      "/PIMS/servlet/forum.ListPopularTopics","Popular Topics");
      output += "<h2>Topics in Forum </h2>";
      output +=
      "<TABLE width=750>"
    +"<tr bgcolor=\"cfdced\">"
    +"<th align=left><b>Latest Topics</b></th>"
    +"<th aligh=left><b>Replies</b></th>"
    +"</tr>";
    int index = 0;
   do
   {
           String topics = rs.getString("Topics");
```

```
                String startedThreadDate = rs.getString("StartedThreadDate");
                String replies = rs.getString("Reply");
       String forumHyperLink = ServletUtilities.hyperLink(response,
              "/PIMS/servlet/forum.ShowForum"+"?topics=" + topics+
              "&startedThreadDate=" + startedThreadDate +
              "&replies="+replies,"View");
      output +=
                "<tr>"
              +"<td>" + topics + "</td>"
              +"<td align=\"center\">" + replies + "</td>"
              +"<td>" + forumHyperLink + "</td>"
              +"</tr>";
              //rs.next();
              index +=1;
    } while(rs.next()&& index <10);
    output += "</TABLE>";
    output += popularTopicsHyperLink;
    output += " <a>" + ServletUtilities.hyperLink(response,
           "/PIMS/jsp/User/NewTopicForm.jsp","New Topic") + "</a>";
    dbBean.isClose();
    output += CommonData.bottomMainLayout;
    output += CommonData.footerLayout;
    output += "</body></html>";
    exitPoint(TOPICS_LIST);
   }
   else
   {
    exitPoint(NO_TOPIC);
   }
  }
  catch(SQLException e)
  {
   e.printStackTrace();
  }
  //out.println(output);
} // end of processRequest()
/**
 * Defines several types of exit conditions depending on the specified
 * exit time.
 * @param the time of exit.
 * @return void.
 */
private void exitPoint(int exitCondition)
```

```
  {
    out.println(ServletUtilities.headWithTitle("List Latestest Topics"));
    out.println("<body bgcolor=\"#cfdced\">");
    // Conditional message
    switch(exitCondition)
    {
     case ERROR_SESSION:
       output +=
         "<h3>Session error</h3>\n" +
         "<p>The system is not able to process your request.<br></p>\n";
       break;
     case TOPICS_LIST:
       break;
     case NO_TOPIC:
       output += "<h3>List Topics</h3>";
       output += "<p>You have no Topics in forumn.</p>\n";
       break;
    } // end of switch
    out.println(output);
    out.println("</BODY>\n");
    out.println("</HTML>");
  } // end of exitPoint()
} // end of ListLatestTopics class
```

2. ShowForun Servlet

```
package forum;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;
import dates.*;
import schqAdmin.*;

/**
 * This class defines a servlet that is responsible to show specific forum.
 * @author Seksit Siripala
 */
public class ShowForum extends HttpServlet
{
  // Data members
  private final int ERROR_SESSION = 1;
  private final int SHOW_FORUM = 2;
```

```
private final int NO_FORUM = 3;
private HttpServletRequest request;
private HttpServletResponse response;
private PrintWriter out;
private ResultSet rs;
private UserProfileBean user;
private DBConnectionBean dbBean;
private JspCalendar d;
private String output = new String();
/**
 * doPost processing. The same as doGet.
 * @param request the client request.
 * @param response the response to client.
 * @return void.
 * @exception ServletException.
 * @exception IOException.
 */
public void doPost(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException
{
  // Simply pass control to doGet()
  this.doGet(request, response);
  return;
} // end of doPost
/**
 * doGet processing. The same as doPost.
 * @param request the client request.
 * @param response the response to client.
 * @return void.
 * @exception ServletException.
 * @exception IOException.
 */
public void doGet(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException
{
  // Startup settings
  this.request = request;
  this.response = response;
  d = new JspCalendar();
  response.setContentType("text/html");
  out = response.getWriter();
  output = "";
  // Check if this is a valid session
```

```
 HttpSession session = request.getSession(false);
 if(session == null) {
  // Not a valid session. Should abort
  exitPoint(ERROR_SESSION);
 }
 else
 {
  // This is a valid session - retrieve information
  user = (UserProfileBean)session.getAttribute("user");
  dbBean = (DBConnectionBean)getServletContext().getAttribute("connectionBean");
  String topics =(request.getParameter("topics"));
  String startedThreaddate = (request.getParameter("startedThreadDate"));
  String replies =(request.getParameter("replies"));
  if((user != null)&&(user.getUserID()!=-1)&&(dbBean != null)
    &&(topics != null) && (replies != null))
  {
   // Session is in a valid state
   showForum(topics,startedThreaddate,replies);
  }
  else
  {
   // Session is not in a valid state
   exitPoint(ERROR_SESSION);
  }
 }
} // end of doGet()
/**
 * Displays forum.
 * @return void
 */
private void showForum(String topics,String startedThreadDate, String replies)
{
 output += ServletUtilities.headWithTitle("Show Forum");
 output += "<BODY text=\"#000000\" bgColor=\"#ffffff\">";
output += CommonData.navLayout;
    //<!--LEFT SIDE NAVIGATION-->
    output += "<p>"+d.getDate()+"</p>"+
    "<p><b>Welcome,</b></p>"+user.getFullName()+
    "<p><strong>Links</strong></p>"+
    "<ul>";
    output += "<li><a>" + ServletUtilities.hyperLink(response,
            "http:/www/schq.mi.th","Home") + "</a></li></ul>";
   output += "<p><strong>MENU</strong></p>";
```

```
 output += "<ul><li><a>" + ServletUtilities.hyperLink(response,
        "/PIMS/servlet/schqAdmin.Announcements","Announcements") + "</a></li>";
 output += "<li><a>" + ServletUtilities.hyperLink(response,
        "/PIMS/jsp/User/OfficerUserProfile.jsp","My Profile") + "</a></li>";
 output += "<li><a>" + ServletUtilities.hyperLink(response,
        "/PIMS/servlet/schqAdmin.DepartmentInfo","Department Info") + "</a></li>";
 output += "<li><a>" + ServletUtilities.hyperLink(response,
        "/PIMS/servlet/forum.ListPopularTopics","Forum") + "</a></li>";
 output += "<li><a>" + ServletUtilities.hyperLink(response,
        "/PIMS/jsp/User/FAQ.jsp","FAQ") + "</a></li>";
 output += "<li><a>" + ServletUtilities.hyperLink(response,
        "/PIMS/servlet/schqAdmin.ListMonthlyTask","Monthly Tasks") + "</a></li>";
 if(user.getUserType().equals("1")|| user.getUserType().equals("3")){
          output += "<li><a>" + ServletUtilities.hyperLink(response,
   "/PIMS/jsp/Personnel/ManageUser.jsp","Manage User") + "</a></li>";
 }//end if
 output +="</ul>";
 output +="<p><strong>Tools</strong></p><ul>";
 output += "<li><a>" + ServletUtilities.hyperLink(response,
        "/PIMS/jsp/User/Weather.jsp","Local Weather") + "</a></li>";
 output += "<li><a href=\""+response.encodeURL("/PIMS/jsp/User/SearchWWW.jsp")+
              "\" target=\"_blank\" >Search WWW </a></li></ul>";
 output += "<li><a>" + ServletUtilities.hyperLink(response,
        "/PIMS/servlet/schqAdmin.Logoff","Log Off") + "</a></li>";
         output += CommonData.bottomNavLayout;
         output += CommonData.mainLayout;
try
{
 ResultSet rs = dbBean.query(
  "SELECT Topics, Message, Authors, StartedThreadDate, RepliedDate, Reply " +
  "FROM SCHQForumn " +
        "Where Topics = '"+topics+"' and Reply = '"+replies+"' "+
  "ORDER BY RepliedDate ");
 if(rs.next())
 {
   output += "<h2>Discussion in topic: "+topics+"</h2>";
         output += "<h3>Started Thread Date :"+startedThreadDate+ "</h3>";
        output +=
        "<TABLE width=650>"
        +  "<tr>"
        +    "<th align=center><b>Authors</b></th>"
        +    "<th align=center><b>Message</b></th>"
        +    "<th align=center><b>Date Posted</b></th>"
```

```
            +   "</tr>";
    do
    {
            String topic = rs.getString("Topics");
            String authors = rs.getString("Authors");
            String message = rs.getString("Message");
            String startedDate = rs.getString("StartedThreadDate");
            String repliedDate = rs.getString("RepliedDate");
            String reply = rs.getString("Reply");
      output +=
        "<tr>"
      +   "<td align=\"center\" bgcolor=\"#cfdced\">" + authors + "</td>"
      +   "<td align=\"left\" bgcolor=\"#cfdced\">" + message + "</td>"
            +   "<td align=\"center\" bgcolor=\"#cfdced\">" + repliedDate + "</td>"
      + "</tr>";
    } while(rs.next());
    output += "</TABLE>";
    output += "<br><a>" + ServletUtilities.hyperLink(response,
          "/PIMS/jsp/User/NewTopicForm.jsp","New Topic") + "</a>  ";
     output += "<a>" + ServletUtilities.hyperLink(response,
      "/PIMS/jsp/User/PostMessageForm.jsp"+"?topics="+topics+
            "&startedThreadDate="+startedThreadDate+
            "&replies="+replies,"Reply")+"</a>";
    dbBean.isClose();
    output += CommonData.bottomMainLayout;
    output += CommonData.footerLayout;
    output += "</body></html>";
    exitPoint(SHOW_FORUM);
   }
   else
   {
    exitPoint(NO_FORUM);
   }
  }
  catch(SQLException e)
  {
   e.printStackTrace();
  }
  //out.println(output);
} // end of processRequest()
/**
 * Defines several types of exit conditions depending on the specified
 * exit time.
```

```
   * @param the time of exit.
   * @return void.
   */
  private void exitPoint(int exitCondition)
  {
    out.println(ServletUtilities.headWithTitle("Show Forumn"));
    out.println("<body bgcolor=\"#cfdced\">");
    // Conditional message
    switch(exitCondition)
    {
      case ERROR_SESSION:
        output +=
          "<h3>Session error</h3>\n" +
          "<p>The system is not able to process your request.<br></p>\n";
        break;
      case SHOW_FORUM:
        break;
      case NO_FORUM:
        output += "<h3>List Topics</h3>";
        output += "<p>You have no Topics in forum.</p>\n";
        break;
     } // end of switch
    out.println(output);
    out.println("</BODY>\n");
    out.println("</HTML>");
  } // end of exitPoint()
} // end of ShowForum class
```

3. PostMessage Servlet

```
package forum;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.sql.*;
import schqAdmin.*;
import dates.*;

/**
 * The class defines a servlet that is responsible to post user's message.
 * This class retrive data from frontend user and update that data into database.
 * @author Seksit Siripala
```

```
 */
public class PostMessage extends HttpServlet
{
 // Data members
 private final int ERROR_LOADING_DRIVER = 1;
 private final int ERROR_SESSION = 2;
 private final int ERROR_THREADID_CREATION =3;
 private final int ERROR_RETRIVE_MESSAGE =4;
 private final int ERROR_POST_MESSAGE = 5;
 private final int SUCCESS_POST_MESSAGE = 6;
 private HttpServletRequest request;
 private HttpServletResponse response;
 private PrintWriter out;
 private ResultSet rs;
 private DBConnectionBean dbBean;
 private UserProfileBean user;
 private JspCalendar d;
 private String topics;
 private String startedThreadDate;
 private String replies;
 /**
  * One-time initilization of the servlet. If the Connection Module is not
  * yet initiated, it ensures its creation.
  * @param config local server configuration parameters.
  * @return void.
  * @exception ServletException
  */
 public void init(ServletConfig config) throws ServletException
 {
  super.init(config);
  // Check if already exists the connection bean
  DBConnectionBean dbBean =
   (DBConnectionBean)getServletContext().getAttribute("connectionBean");
  if(dbBean == null)
  {
   // Did not exist. Lets create it.
   dbBean = new DBConnectionBean();        // for the connection module
   // Initialize the DB connection module
   dbBean.setDriver(CommonData.DRIVER);
   dbBean.setUrl(CommonData.URL);
   dbBean.setUser(CommonData.USER);
   dbBean.setPassword(CommonData.PASSWORD);
   // Load the driver
```

```
    if(!dbBean.isLoaded())
    {
     exitPoint(ERROR_LOADING_DRIVER);
    }
   }
   this.dbBean = dbBean;
   // Place the connection bean in the servlet context (application scope)
   getServletContext().setAttribute("connectionBean", dbBean);
  }  // end of init()
  /**
   * doPost processing. The same as doGet.
   * @param request the client request.
   * @param response the response to client.
   * @return void.
   * @exception ServletException.
   * @exception IOException.
   */
  public void doPost(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException
  {
   // Simply pass control to doGet()
   this.doGet(request, response);
   return;
  } // end of doPost
  /**
   * doGet processing. The same as doPost.
   * @param request the client request.
   * @param response the response to client.
   * @return void.
   * @exception ServletException.
   * @exception IOException.
   */
  public void doGet(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException
  {
   // Startup settings
   this.request = request;
   this.response = response;
   response.setContentType("text/html");
   out = response.getWriter();
   d = new JspCalendar();
   // Check is session exists and is in a valid state
   HttpSession session = request.getSession(true);
```

```
 if(session.isNew()) {
  // Not a valid session. Should abort
  exitPoint(ERROR_SESSION);
 }
 else
 {
  // This is a valid session - retrieve information
  user = (UserProfileBean)session.getAttribute("user");
  dbBean = (DBConnectionBean)getServletContext().getAttribute("connectionBean");
  topics =(request.getParameter("topics"));
  startedThreadDate =(request.getParameter("startedThreadDate"));
  replies =(request.getParameter("replies"));
  newMessage();
 }
} // end of doGet()
/**
 * Creates a new  message with the information submited in the request.
 * @return void
 */
private void newMessage()
{
 // Get user id (obtain a new primary key)
 dbBean.setPrimaryKeyQuery(
  "SELECT ThreadID FROM SCHQForumn ORDER BY ThreadID");
 int pk = dbBean.getPrimaryKey();
//displayOutput("primary key "+pk);
 if(pk == -1){
  // Error getting the primary key
  exitPoint(ERROR_THREADID_CREATION);
 }
 else
 {
  // Get the data from the request
  String author = user.getFirstName();
  String message = request.getParameter("message");
  String newTopic = request.getParameter("newTopic");
  String repliedDate = d.getDateTime();
  int reply = ((Integer.parseInt(replies)));
  int result =
  dbBean.update("INSERT INTO SCHQForumn (ThreadID, Topics, Message, Authors, "
                +" StartedThreadDate, RepliedDate, Reply) VALUES("
                + pk + ", "
                + "'" + ServletUtilities.filter(topics)+ "', "
```

```
                        + "'" + ServletUtilities.filter(message) + "', "
                        + "'" + author + "', "
                        + "'" + startedThreadDate  + "', "
                        + "'" + repliedDate + "', "
                        + "'" + reply + "') " );
    if(result == 1){
   // Account successfuly created
            if(newTopic.equals("no")){
            reply += 1;
            int rs = updateReplies(topics,reply);
            }
            exitPoint(SUCCESS_POST_MESSAGE);
     }
     else{
            displayOutput("result"+ result);
            // Error creating the new account
            exitPoint(ERROR_POST_MESSAGE);
     }
   }//end if
  else{
            exitPoint(ERROR_RETRIVE_MESSAGE);
  }//end if-else
 }//end else
} // end of newMessage()
public int updateReplies(String topics, int reply){
     int result = dbBean.update("UPDATE SCHQForumn set Reply ='"+reply+"' where Topics
              ='"+topics+"' ");
     return result;
}//end updateReplies method
 /**
  * Defines several types of exit conditions depending on the specified
  * exit time.
  * @param the time of exit.
  * @return void.
  */
 private void exitPoint(int exitCondition)
 {
  String output = new String();
  // Conditional message
  switch(exitCondition)
  {
   case ERROR_LOADING_DRIVER:
    output +=
```

```
    "<h3>Application error</h3>\n" +
    "<p>Error generated when trying to load DB connection driver</p>\n";
   break;
  case ERROR_SESSION:
   output +=
    "<h3>Session error</h3>\n" +
    "<p>The system is not able to process your request.<br>\n" +
    "Access denied!</p>\n";
   break;
  case ERROR_THREADID_CREATION:
   output +=
    "<h3>Application error</h3>\n" +
    "<p>There was an error when trying to post your message!can not retrive data</p>\n";
   break;
  case ERROR_RETRIVE_MESSAGE:
    output +=
    "<h3>Application error</h3>\n" +
    "<p>There was an error when trying to retrive your message, please try again</p>\n";
   break;
  case ERROR_POST_MESSAGE:
   output +=
    "<h3>Application error</h3>\n" +
    "<p>There was an error when trying to post your message! cannot query primary
          key</p>\n";
   break;
  case SUCCESS_POST_MESSAGE:
   output +=
    "<h3>Message Posted</h3>\n" +
    "<p>Your message was sucessfully posted.</p>\n";
   break;
} // end of switch
// Conditional exit buttons
switch(exitCondition)
{
  case ERROR_LOADING_DRIVER:
  case ERROR_SESSION:
  case ERROR_THREADID_CREATION:
  case ERROR_RETRIVE_MESSAGE:
      output += "<p>" + ServletUtilities.hyperLink(response,
       "/PIMS/servlet/forum.ListPopularTopics"+"?topics="+topics+
      "&replies="+replies, "Try again") + "<br>\n";
   break;
  case ERROR_POST_MESSAGE:
```

```
        output += "<p>" + ServletUtilities.hyperLink(response,
          "/PIMS/login.jsp", "Back to home") + "</p>\n";
        break;
      case SUCCESS_POST_MESSAGE:
        output += "<p>" + ServletUtilities.hyperLink(response,
          "/PIMS/jsp/User/OfficerUserOptions.jsp", "Continue") + "</p>\n";
        break;
    } // end of switch
    displayOutput(output);
  } // end of exitPoint()
  private void displayOutput(String text){
    out.println(ServletUtilities.headWithTitle("Process Post Message"));
    out.println("<body bgcolor=\"#cfdced\">");
    out.println("<P>" + text + "</P>\n");
    out.println(ServletUtilities.hyperLink(response,
      "http://www/schq.mi.th", "Home"));
    out.println("<P>\n");
    out.println("</BODY>\n");
    out.println("</HTML>");
  } // end displayOutput
} // end of PostMessage class
```

C. SAMPLE SOURCE OF WEATHER PACKAGE

1. Weather Class

```
package weather;
import java.io.*;
import java.net.*;
import java.util.*;
/**
 * Weather
 *
 */
public class Weather {
  /**
   * The city name
   */
  public String city;
  /**
   * The temperature in degrees Fahrenheit.
   */
  public double deg;
  /**
```

```
 * Called to get the current temperature.
 * @param code A weather code for the city.
 * @return The temperature in degrees fahrenheit.
 */
public static double getTemp(String code)
{
  try {
    String urlStr;
    urlStr = "http://weather.noaa.gov/weather/current/";
    urlStr += code.toUpperCase();
    urlStr += ".html";
    //HTTPSocket http = new HTTPSocket();
    //http.send(url,null);
   // int i = http.getBody().indexOf("Temperature")+11;
   StringBuffer body = new StringBuffer();
   URL url = new URL(urlStr);
   URLConnection uc = url.openConnection();
   try{
      InputStream in = url.openStream();
      BufferedReader htmlPage = new BufferedReader(new InputStreamReader(in));
      int data;
      int charCounter =0;
      while((data = in.read()) != -1){
              body.append((char)data);
              charCounter++;
     }//end while
   }//end try
   catch(MalformedURLException e)
{
     System.out.println("Error: invalid URL");
}
catch(IOException e)
{
     System.out.println("Can't connect");
   }//end try-catch
   int pageCharLength = body.length();
   String pageBody = body.substring(0,pageCharLength);
   int index = pageBody.indexOf("Temperature") +11;
   while ( !Character.isDigit(pageBody.charAt(index)) )
          index++;
   String str = pageBody.substring(index,pageBody.indexOf(' ',index) );
   System.out.println(str);
   return Double.parseDouble(str);
```

```
  } catch ( Exception e ) {
  }
  return 0;
}
/**
 * A list of cities to aggregate.
 */
static String _city[] = {
  "Bangkok, BK|VTBD",
  "Chiang Mai, CM|VTCC",
  "Hat Yai, HT|VTSS",
  "Phuket, PH|VTSP",
  "Rayong, RY|VTBU"};
public static Weather[] getList()
{
  Weather array[] = new Weather[_city.length];
  for ( int i=0;i<_city.length;i++ ) {
    array[i] = new Weather();
    array[i].city = _city[i].substring(0,_city[i].indexOf("|") );
    array[i].deg = getTemp(_city[i].substring( _city[i].indexOf("|") + 1));
  }
  return array;
}
/**
 * Aggregate this list of cities to a file.
 *
 * @param path Where to write the HTML file.
 */
public static void fileAggregate(String path)
{
  try {
    FileOutputStream fw = new FileOutputStream(path);
    PrintStream ps = new PrintStream(fw);
    ps.println("<html><head><title>Current Weather</title></head>");
    ps.println("<body>");
    ps.println("<h1>Current Weather</h1><table border=0>");
    Weather wx[] = getList();
    for ( int i=0;i<wx.length;i++ ) {
      ps.println("<tr><td>" + wx[i].city + "</td><td>"+ wx[i].deg + "</td></tr>");
    }
    ps.println("</table></body></html>");
    ps.close();
    fw.close();
```

```
    } catch ( Exception e ) {
    }
}
}//end of Weather classs
```

D. SAMPLE SOURCE CODE OF JSP FILES

1. Tools.nav.jsp

```
<TABLE cellSpacing=0 cellPadding=0 summary=menu border=0 width="203">
<TBODY>
<TR><!--start left top NavBar -->
<TD vAlign=top rowSpan=3 width="10">
<TABLE cellSpacing=0 cellPadding=0 summary="blue line" border=0>
<TBODY>
<TR>
<TD bgColor=#294563><IMG height=1 alt=""
src="/PIMS/images/spacer.gif" width=10></TD></TR>
<TR>
<TD bgColor=#cfdced><FONT face="Arial, Helvetica, Sans-serif"
color=#4c6c8f size=4> </FONT></TD></TR>
<TR>
<TD bgColor=#294563><IMG height=1 alt=""
src="/PIMS/images/spacer.gif"
width=10></TD></TR></TBODY></TABLE></TD>
<!-- end left top NavBar -->
<TD bgColor=#294563 width="4"><IMG height=1 alt=""
src="/PIMS/images/spacer.gif" width=1></TD>
<TD vAlign=bottom bgColor=#4c6c8f width="10"><IMG height=10 alt=""
src="/PIMS/images/spacer.gif" width=10></TD>
<TD vAlign=top noWrap bgColor=#4c6c8f width="154">
<!-- start Menu items -->
<DIV class=menu> <p> <BR></p>
<jsp:useBean id="clock"  class="dates.JspCalendar" scope="page" />
<p>  <jsp:getProperty name="clock" property="date"/> </p>
<jsp:useBean id ="user" class="schqAdmin.UserProfileBean" scope="session" />
<p><b>Welcome,</b></p>
<p><b>
<jsp:getProperty name="user" property="fullName" />
</b></p>
<p><strong>Links</strong></p>
<ul><li><a href="http://www.schq.mi.th">Home</a></li></ul>
<p><strong>MENU</strong></p>
<ul>
```

```
<li><a href="/PIMS/servlet/schqAdmin.Announcements">Annoucement</a></li>
<li><a href="/PIMS/jsp/User/OfficerUserProfile.jsp">My Profile</a></li>
<li><a href="/PIMS/servlet/schqAdmin.DepartmentInfo">Department Info</a></li>
<li><a href="/PIMS/servlet/forum.ListPopularTopics">Forum</a></li>
<li><a href="/PIMS/jsp/User/FAQ.jsp">FAQ</a></li>
<li><a href="/PIMS/servlet/schqAdmin.ListMonthlyTask">Monthly Task</a></li>
<% if( user.getUserType().equals("3") || user.getUserType().equals("1") ){ %>
<li><a href="/PIMS/jsp/Personnel/ManageUser.jsp">Manage User</a></li>
<% } %>
<% if( user.getUserType().equals("2")  ){ %>
<li><a href="/PIMS/servlet/schqAdmin.ListProject">Project</a></li>
<% } %>
</ul>
<p><strong>Tools</strong></p>
<ul>
<li><a href="/PIMS/jsp/User/Weather.jsp">Local Weather</a></li>
<li><a href="/PIMS/jsp/User/SearchWWW.jsp" target="_blank">Search WWW </a></li>
</ul>
<li><a href="/PIMS/servlet/schqAdmin.Logoff">Log off</a></li>
<DIV align=center><BR><BR><BR><A href="http://xml.apache.org/forrest/"><IMG height=1 alt=""
src="/PIMS/images/spacer.gif" width=5 border=0></A>
</DIV></DIV></TD>
<TD vAlign=bottom bgColor=#4c6c8f width="10"><IMG height=10 alt=""
src="/PIMS/images/spacer.gif" width=10></TD>
<TD bgColor=#294563 width="4"><IMG height=1 alt=""
src="/PIMS/images/spacer.gif" width=1></TD>
<!-- start left top NavBar -->
<TD vAlign=top rowSpan=3 width="11">
<TABLE cellSpacing=0 cellPadding=0 summary="blue line" border=0>
<TBODY>
<TR>
<TD width="13" bgColor=#294563><IMG height=1 alt=""
src="/PIMS/images/spacer.gif" width=10></TD></TR>
<TR>
<TD bgColor=#cfdced><FONT face="Arial, Helvetica, Sans-serif"
color=#4c6c8f size=4> </FONT></TD></TR>
<TR>
<TD bgColor=#294563><IMG height=1 alt=""
src="/PIMS/images/spacer.gif"
width=10></TD></TR></TBODY></TABLE></TD>
<!-- end left top NavBar -->
</TR>
<TR><TD vAlign=bottom align=left bgColor=#4c6c8f colSpan=2
```

```
rowSpan=2><IMG height=10 alt=""
src="/PIMS/images/menu_left.gif" width=10 border=0></TD>
<TD bgColor=#4c6c8f width="154"><IMG height=10 alt=""
src="/PIMS/images/spacer.gif" width=10 border=0></TD>
<TD vAlign=bottom align=right bgColor=#4c6c8f colSpan=2
rowSpan=2><IMG height=10 alt=""
src="/PIMS/images/menu_right.gif" width=10
border=0></TD></TR>
<TR>
<TD bgColor=#294563 height=1 width="154"><IMG height=1 alt=""
src="/PIMS/images/spacer.gif"
width=1></TD></TR></TBODY></TABLE>
```

2. User.OfficerUserProfile.jsp

```
<%@ page contentType="text/html; charset=iso-8859-1"%>
<HTML>
<TITLE>Welcome</TITLE>
<BODY ><TABLE cellSpacing=0 cellPadding=0 width="100%" bgColor=#ffffff
summary="page content" border=0>
<TBODY>
<TR>
<TD vAlign=top>
<!-- Navigator bar -->
<%@ include file="/jsp/Tools/nav.jsp" %>
</td>
<TD vAlign=top width="100%" align="center">
<TABLE width="750" cellSpacing=0 cellPadding=0 summary=content border=0>
<!-- Main Content  -->
<TBODY><TR><TD align=left width="100%">
<table width="750" cellpadding="2" cellspacing="0" border="0" align="center" >
<tr ><td width="750" bgcolor="#525D76" align="center">
<h1 align="center">Personal Information Management System</h1></td></tr>
<tr><td >
<div align="center">
<h2>User Profile</h2>
<!--  Use connection Bean. If is not yet created, create and set it with application scope -->
<jsp:useBean id = "connectionBean" class="schqAdmin.DBConnectionBean" scope="application" />
 <!-- Form with deault as current user object values -->
<IMG ALIGN="CENTER" SRC="<jsp:getProperty name="user" property="pictureFileName"/>" BORDER="2">
<table border =1 align=center>
<tr><td> Name:<jsp:getProperty name="user" property="firstName"/>
<jsp:getProperty name="user" property="lastName"/></td></tr>
```

```
<tr><td>Rank:<jsp:getProperty name="user" property="rank"/></td></tr>
<tr><td>Gender:<jsp:getProperty name="user" property="gender"/></td></tr>
<tr><td>Date of birth:<jsp:getProperty name="user" property="dateOfBirth"/></td></tr>
<tr><td>Status:<jsp:getProperty name="user" property="status"/></td></tr>
<tr><td>Salary:<jsp:getProperty name="user" property="salary"/></td></tr>
<tr><td>Department:<jsp:getProperty name="user" property="departmentCode"/></td></tr>
<tr><td>Address:<jsp:getProperty name="user" property="street"/>
<jsp:getProperty name="user" property="city"/>
<jsp:getProperty name="user" property="province"/>
<jsp:getProperty name="user" property="postalCode"/></td></tr>
<tr><td>Phone:<jsp:getProperty name="user" property="phone"/></td></tr>
<tr><td>MilitartID:<jsp:getProperty name="user" property="militaryID"/></td></tr>
</table><br>
<a href="/PIMS/jsp/User/EditOfficerUserProfile.jsp">Edit Profile</a> 
<a href="/PIMS/servlet/schqAdmin.ListOfficerDependent">View Dependents</a>
</td></tr></table></td>
</TR><!-- end Content-->
</TBODY></TABLE></TD></TR></TBODY></TABLE>
<!-- Footer -->
<%@ include file="/jsp/Tools/footer.jsp" %>
</BODY>
</HTML>
```

3. Admin.EditAnnouncement.jsp

```
<%@ page contentType="text/html; charset=iso-8859-1" %>
<HTML>
<TITLE>Welcome</TITLE>
<BODY >
<TABLE cellSpacing=0 cellPadding=0 width="100%" bgColor=#ffffff
summary="page content" border=0>
<TBODY>
<TR>
<TD vAlign=top>
<!-- Navigator bar -->
<%@ include file="/jsp/Tools/nav.jsp" %>
</td>
<TD vAlign=top width="100%" align="center">
<TABLE width="750" cellSpacing=0 cellPadding=0 summary=content border=0>
<!-- Main Content  -->
<TBODY><TR><TD align=left width="100%">
<table width="750" cellpadding="2" cellspacing="0" border="0" align="left" >
<tr><td bgcolor="#525D76" align="center">
<h1>SCHQ Personal Administration System</h1></td></tr>
```

```
<tr><td bgcolor="cfdced">
<div align="center">
<h3>Edit Announcement</h3>
<% String aID =request.getParameter("AID");
  String title = request.getParameter("Title");
  String content = request.getParameter("Content");
  String announcement = request.getParameter("Announcement");
%>
<h3>Anouncement: <%= title %> </h3>
<tr><td align ="center" bgcolor="cfdced">
<form METHOD="POST" ACTION="/PIMS/servlet/schqAdmin.ProcessAnnouncement?
AID=<%=aID%>&Announcement=<%=announcement%>">
<input TYPE="TEXT" NAME="Title" SIZE="25"VALUE="<%= title %>" /><br>
<textarea name="Content" rows="10" cols="50"> <%= content %> </textarea>
<center><H3>Publishing Status</H3><input type="radio" name="PStatus" value="on" checked >ON
<input type="radio" name="PStatus" value="off">OFF </center>
<center> <input type="SUBMIT" name="PostAnnouncement" value="Post Announcement" > </center>
</form>
</td></tr></table></td>
</TR><!-- end Content-->
</TBODY></TABLE></TD></TR></TBODY></TABLE>
<!-- Footer -->
<%@ include file="/jsp/Tools/footer.jsp" %>
</BODY>
</HTML>
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] Free Software Foundation, Inc., “The Free Software Definition”, http://www.gnu.org/philosophy/free-sw.html, September 2003.

[2] Open Source Initiative, “The Open Source Definition”, http://www.opensource.org/docs/definition_plain.html, September 2003.

[3] Netcraft Ltd, “WebServerSurvey”, http://news.netcraft.com/archives/web_server_survey.html, September 2003.

[4] Netcraft Ltd, “Web Server Survey”, http://www.netcraft.com/Survey/index-200109.html#computers, September 2003.

[5] Bloor Research Ltd., “Linux and Window NT”, https://www.bloorresearch.com/research_library.php?pid=245, September 2003.

[6] Edward Bradford, “Runtime: Pipes in Linux, Windows 2000, and Windows XP” http://www-106.ibm.com/developerworks/linux/library/l-rt4/, September 2003.

[7] Linux Online, Inc “Hardware Port Projects”, http://www.linux.org/projects/ports.html, September 2003.

[8] Cable News Network LP, LLLP, “RSA: Security in 2002 worse than 2001, exec says”, http://www.cnn.com/2002/TECH/internet/02/25/2002.security.idg/index.html, September 2003.

[9] Linux Online Inc, “What is Linux”, http://www.linux.org, September 2003.

[10] Ladislav Bodnar, “Major Linux Distributions”, http://www.distrowatch.com/dwres.php?resource=major, September 2003.

[11] The Apache Software Foundation, “Apache HTTP Sever Project”, http://httpd.apache.org/ABOUT_APACHE.html, September 2003.

[12] The Apache Software Foundation, “The Apache Jakarta Project”, http://jakarta.apache.org/tomcat/index.html, September 2003.

[13] Mort Bar Consulting, “Jetty Servlet/JSP Container”, http://jetty.mortbay.com/jetty/index.html, September 2003.

[14] MySQL AV, “MySQL Database Server”, http://www.mysql.com/products/mysql/index.html, September 2003.

[15] Bruce Momjian, “PostgreSQL: Introduction an Concepts”, http://www.postgresql.org/docs/aw_pgsql_book/index.html, September 2003.

[16] Ziff Davis Media Inc., “Server Database Clash”, http://www.eweek.com/article2/0,3959,293,00.asp, September 2003.

[17] Deitel, H.M., Deitel, P.J., Neito, T.R. *Internet & the World Wide We: How To Program*, Prentice Hall, second edition, 2002.

[18] Macromedia, Inc., “Macromedia-Cold Fusion MX”, http://www.macromedia.com, September 2003.

[19] Sun Microsystems, Inc., “Java Server Pages Technology”, http://java.sun.com/products/jsp/whitepaper.html, September 2003.

[20] The PHP Group, “PHP: Hypertext Preprocessor”, http://www.php.net, September 2003.

[21] Jim C, “Modeling Web Application Design With UML”, Rational Software White Paper, http://www.rational.com/media/whitepapers/webapps.pdf, September 2003.

[22] Sun Microsystems, Inc., “Java Servlet Technology”, http://java.sun.com/products/servlet/whitepaper.html, September 2003.

[23] Sun Microsystems, Inc., “The J2EE Tutorial”, http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html, September 2003.

[24] Satzinger, J. W., Jackson, R. B., and Stephen D. B, *Systems Analysis and Design in a Changing World,* p. 357, Course Technology, 2000.

[25] Silberschatz A., Korth H. F., and Sudarsham S., *Database System Concepts*, McGraw-Hill, fourth edition, 2002.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Professor Peter J. Denning
   Naval Postgraduate School
   Monterey, California

4. Professor Neil C. Rowe
   Naval Postgraduate School
   Monterey, California

5. CDR Gary L. Kreeger USN
   Naval Postgraduate School
   Monterey, California

6. Captain Seksit Siripala
   Supreme Command Headquarters
   Bangkok, Thailand