Secured Pace Web Server with Collaboration and Error Logging Capabilities

By

Rochelle Ribeiro

Submitted in partial fulfillment
of the requirements for the degree of
M.S. in Computer Science

at

School of Computer Science and Information Systems

Pace University

December 2003
We hereby certify that this dissertation, submitted by Rochelle Ribeiro, satisfies the dissertation requirements for the degree of *M.S. in Computer Science* and has been approved.

Dr. Lixin Tao, Supervisor
Chairperson of Dissertation Committee

Dr. Narayan Murthy, Chair, CS Dept.
Dissertation Committee Member

Dr. Mehdi Badii
Dissertation Committee Member

Dr. Susan M. Merritt
Dean, CSIS

School of Computer Science and Information Systems
Pace University 2003
Abstract

Secured Pace Web Server with Collaboration and Error Logging Capabilities

by
Rochelle Ribeiro

Submitted in partial fulfillment of the requirements for the degree of M.S. in Computer Science

December 2003

Web servers today must be scalable, reliable and secure while providing tools that can benefit its users.

This thesis builds upon an existing Pace Web Server that was developed by graduate students. To further enhance its versatility, the following capabilities have been added to it: Secure Sockets Layer (SSL) using the Java Secure Socket Extension (JSSE) API, error logging, and a collaboration chat tool that implements the Java Shared Data Toolkit (JSDT) API. An analysis is also given on the necessity of security for web servers, the importance of logging errors, and the need for a collaboration server that would contain collaboration tools.
Acknowledgements

I would like to gratefully acknowledge my thesis advisor Dr. Tao for his continued guidance and support in the preparation of this thesis. He has provided much assistance to me and was available to help me anytime. His motivation and enthusiasm have inspired me tremendously throughout this project.

I also thank my friends for their patience and encouragement.

Words cannot express my deepest gratitude and appreciation to my family for their love and moral support. I thank you for always being there for me.
# Table of Contents

Abstract ........................................................................................................................................................................... vii

List of Tables ................................................................................................................................................................. vi

List of Figures ............................................................................................................................................................... vii

Chapter 1  Introduction .................................................................................................................................................... 1

Chapter 2  Background ..................................................................................................................................................... 3

  2.1 Web Server Concepts .................................................................................................................................................. 3

  2.2 Commercial Servers .................................................................................................................................................... 6

Chapter 3  Secure Pace Web Server with SSL .................................................................................................................. 24

  3.1 Introduction to SSL ................................................................................................................................................... 24

  3.2 Overview of SSL ....................................................................................................................................................... 24

  3.3 Java and SSL ............................................................................................................................................................ 29

  3.4 SSL in Pace Web Server .............................................................................................................................................. 35

Chapter 4  Online Collaboration ....................................................................................................................................... 41

  4.1 Overview of Collaborative Computing ....................................................................................................................... 41

  4.2 Java and Collaboration ................................................................................................................................................. 42

  4.3 JSDT in Pace Web Server ............................................................................................................................................. 46

Chapter 5  Logging ............................................................................................................................................................ 56

  5.1 Overview of Logging ................................................................................................................................................... 56

  5.2 Logging in Pace Web Server ....................................................................................................................................... 57

Chapter 6  Running the New Pace Web Server Features .................................................................................................. 59

  6.1 Running SSL .............................................................................................................................................................. 59

  6.2 Running the Collaboration tools ................................................................................................................................ 59

  6.3 Using the Logging Features ........................................................................................................................................ 60

Chapter 7  Conclusion ......................................................................................................................................................... 61

Appendix  A Troubleshooting SSL Errors ..................................................................................................................... 65
List of Tables

Table 1 – Pricing for today’s commercial web/collaboration servers. ......................... 7
Table 2 - Apache Server Features .................................................................................. 8
Table 3 - Microsoft IIS features ................................................................................... 9
Table 4 - Oracle HTTP server features ........................................................................ 10
Table 5 - Sun ONE features. ......................................................................................... 13
Table 6 – AOLserver features ...................................................................................... 14
Table 7 – Xitami features ............................................................................................. 16
Table 8 – Zeus Web Server features ............................................................................. 18
Table 9 – SSL Cipher Suites Supported by Sun’s JSSE Implementation.[13] ................. 35
List of Figures

Figure 1 – Physical structure of the Web and DNS translation process. [10] ................. 5
Figure 2 – Pace Web Server Java classes. ................................................................. 22
Figure 3 – Screenshot of Pace Web Server console with enhancements. ................. 22
Figure 4. Some people who cause security problems and why. [22] ......................... 25
Figure 5 – SSL in the TCP/IP model. [11] ................................................................. 26
Figure 6. SSL Handshake [11] ............................................................................. 28
Figure 7 – Certificate warning. .............................................................................. 38
Figure 8 – Certificate information. ....................................................................... 39
Figure 9 – Success after using the HTTPS protocol in Pace Web Server. ............... 40
Figure 10 - Main Pace Collaboration Server page .................................................. 48
Figure 11 – Initial screen for Collaboration Chat applet. ....................................... 49
Figure 12 – Pop up box to enter the user’s handle for the Collaboration Chat tool. .... 50
Figure 13 – After user logs in s/he can see other users logged into the Collaboration Chat tool. ........................................................................................................... 51
Figure 14 – Users chatting with the collaboration chat tool. .................................. 52
Figure 15 – Whiteboard utility .............................................................................. 54
Figure 16 – User using whiteboard utility. ............................................................... 55
Figure 17 – Apache’s Tomcat Web Server Administration tool login screen .......... 62
Figure 18 – Configurable items in the Tomcat Web Server Administration Tool. .... 63
Chapter 1

Introduction

Web servers today are becoming increasingly more powerful, robust, and highly scalable. Corporations take into consideration the ease of use of administering these web servers, as well as tracking problems and also providing a secure environment for their users when determining what type of web server to obtain for their business needs. The pervasiveness of the Internet makes it vulnerable for web site users to have their personal information stolen from them. Providing a secure channel of communication, such as Secure Sockets Layer (or SSL), between the client and the server is an important feature for any web server to have. As a result, most web servers in existence now must have SSL since there are risks involved with having insecure web servers. This thesis will discuss the significance of SSL and the need for users to be content in knowing that their data is safe from any possible intruders.

Error logging is another vital feature that a web server should have. It can show any errors that the web server produced. It is beneficial in diagnosing the problem and ascertaining a possible solution for correcting the problem. An important aspect is that the error log file should be easily accessible to the server administrators so that they may be able to diagnose the problem quickly.

Another feature that is useful is a collaboration server that allows users to work together on something without having to physically meet at a certain place. Collaboration tools can help users to solve problems remotely. For example, a group of three students are assigned to work on a Java project together. Most of the time, the group has to meet at a certain place at a certain time to look at their Java files and modify them when they are together. However, with an online collaboration chat tool, they can connect to a server and discuss their project in real-time. This saves them time and money (if they have to
travel far to the place where they intended to meet). This tool would be very helpful for these students, as well as companies who need to meet with people on a regular basis.

The Pace Web Server has been enhanced to include SSL, a collaboration tool and error logging. These features will be discussed throughout the thesis.
Chapter 2
Background

2.1 Web Server Concepts

2.1.1 Http Servers

Web servers, also known as HTTP servers, store and fetch contents of a web page based on requests from various clients. The user would open a web browser (the client) and enter a Uniform Resource Locator (URL). The web browser would try to locate the server based on the URL that was entered. Once the web server is found, a connection is made between the client and server. The server would then receive the request, process it and send a response back to the client and the connection is closed. The web server will have no recollection of the connection that was made after it is closed. It is considered “stateless” because the server does not store information about the connection. This is the best approach because there is no overhead involved in keeping the connection alive for long periods of time and maintaining information about the connection would not be required. The server would also be able to handle multiple requests and not bog down the system.

A server can run various types of services, like FTP, Telnet, etc. When a client is making a request to the server, the server needs to know what type of service the client is requesting. The server reserves well-known ports for these services. It listens to these ports for any incoming requests. If a user types in a URL like http://www.pace.edu, by default it goes to port 80 since this port number is reserved for HTTP requests. Port 21 is reserved for FTP requests and Telnet is on port 23. The default port for SSL is 443. The Pace Web Server ports are configurable, so almost any port can be used except the privileged ports which range from 0 to 1024.
A web page can also be accessed through a TELNET session to the web server daemon via a well-known port. The following is an example of how to access a Web page using a TELNET session:

```
Prompt > telnet www.pace.edu http
```

The GET method would be used to retrieve a page and the HTML text would be shown and the connection would be closed. Most web servers have blocked this capability because of security reasons so it may not be possible to do this nowadays but when the Internet was made to the public it was possible. Also, it is not necessary to do all this work just to get an HTML file since web browsers are readily available to handle this.

A web server connects to a network that consists of other machines, which include other web servers. Apache’s Tomcat and Microsoft Internet Information Server are a couple of the more prevalent web servers available today. The Internet is, fundamentally, an interconnected network of other networks and other machines. The servers need to contact each other and communicate messages to one another. TCP/IP helps web servers find each other and send data. TCP/IP uses static addresses to locate machines. Each machine that connects to the Internet will have a static, unique address known as an IP address. Since people can not remember an IP address which consists of numbers, the IP address can be assigned names using a distributed hierarchical lookup system called a Domain Name System.

When a user types in a URL in the web browser there are many things that take place and have to be in place for the web server to process the request. The web server should have a web application that people need to get access to. A web server that is connected to the Internet has to be assigned a static IP address, for example, 123.456.789.123 and a domain name of samplesite.com. On the other end, there are several clients connected to the Internet with web browsers running on each machine. The users at the client machines can access this web server by entering its IP address of 123.456.789.123 or it can enter the domain name samplesite.com. If the IP address is entered then a direct connection is made. If the domain name is entered then the Domain Name Server translates the domain name into an IP address and then connects to the web server.
Web servers should minimally support web applications that are written in the following languages:

- Server-based CGI forms,
- Active Server Pages (ASP),
- Client-based Java applets,
- Java Server Pages (JSP),
- Servlets

CGI has been around for a very long time. It was one of the first server side processing scripts/languages available when the Internet became popular. However, there was much overhead involved because CGI scripts required a separate process. Other technologies were created so that the web server would not have to be bogged with several processes.

The Active Server Pages technology was developed by Microsoft to do server side processing. It contains HTML tags with VBScript embedded within the HTML. The
VBScript is processed on the server side, which results in the contents of the web pages being dynamic.

Applets are mini Java applications that can run within a web browser. There are many security restrictions involved with applets because they run on the clients’ machines.

Java Server Pages is similar to Active Server Pages but the major difference is that the language embedded within the HTML is JSP, which is Java for the web, while in ASP it is VBScript.

Servlets are Java classes that run on the server side. They can access values from an HTML page and redirect to other HTML pages. It can also interact with the database, providing data to be displayed on an HTML page or insert data from an HTML page.

2.2 Commercial Servers

There are several servers available in the market today ranging from free full-fledged servers like Apache Tomcat to more expensive servers like Oracle’s web servers. They all offer a range of features (described below) designed to meet the needs of any type of web application. There are also collaboration servers that offer collaborative services like chatting, collaborative editing, etc. The collaboration servers are fairly new and provide an important service for corporations that require meeting people daily to discuss papers, delivering presentations, or providing customer service.

<table>
<thead>
<tr>
<th>Server</th>
<th>Server Type</th>
<th>OS</th>
<th>Minimum Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache</td>
<td>Web</td>
<td>Windows/Unix/Linux</td>
<td>Free</td>
</tr>
<tr>
<td>Microsoft Internet Information Server</td>
<td>Web</td>
<td>Windows</td>
<td>Incl. with Windows 2003 Server</td>
</tr>
<tr>
<td>Sun ONE Web Server</td>
<td>Web</td>
<td>Windows/Unix/Linux</td>
<td>$1495</td>
</tr>
</tbody>
</table>
Zeus          | Web       | Windows/Unix/Linux | $1700 \\
AOLserver  | Web       | Windows/Unix       | Free \\
Xitami      | Web       | Windows/Unix/Linux | Free \\
Oracle HTTP Server | Web       | Windows/Unix/Linux | Incl. with Oracle Database Server and the Oracle 9ias \\
Stellent Collaboration Server | Collaboration | Windows | $50,000+ \\
Cisco Collaboration Server | Collaboration | Windows, Solaris | $15,000 \\

<table>
<thead>
<tr>
<th>Documentation</th>
<th>Power</th>
<th>Ease</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installation/configuration</td>
<td>Good</td>
<td>Poor</td>
</tr>
<tr>
<td>Management/administration</td>
<td>Good</td>
<td>Poor</td>
</tr>
<tr>
<td>Logging Features</td>
<td>Good</td>
<td>Fair</td>
</tr>
</tbody>
</table>

Table 1 – Pricing for today’s commercial web/collaboration servers.

2.2.1  *Apache Tomcat*

Apache Tomcat (http://jakarta.apache.org/) is one of the most prevalent freeware, open-source servers available today. Developers from all around the world have worked together to assist in the development of Tomcat. It was a project started up as a small scale server but developers realized that there was much potential in building a scalable server and added more robust features to rival the server giants like Oracle and Microsoft.
Table 2 - Apache Server Features

Apache has a separate SSL module which allows administrators to configure the Apache server to use SSL. It is in a separate package called the mod_ssl package which was first released in August 1998. When starting the SSL Apache version, the server administrator has to enter the “password” for the SSL private key file but there are ways to circumvent this since servers may be rebooted automatically on a scheduled basis. Apache offers a step by step help guide to create an SSL server certificate. It is basically a series of commands using the openssl tool that need to be executed. It also provides some more helpful instructions to create and use your own Certificate Authority. Apache provides a decent amount of documentation for getting SSL up and running on its server, as well as a mailing list specifically for their mod_ssl package. This information tends to be hidden so a search would have to be done to find it. Other than the Apache documentation there is not much information available about using SSL in Apache.

The Apache HTTP web server provides several types of logging, from logging the hostnames that contact the server to conditional logging, which is logging based on predefined conditions that the administrator can set. Error logging is also available. It is redirected into a separate error log file.

2.2.2 Microsoft Internet Information Server (IIS)

The Microsoft Internet Information Server (http://www.microsoft.com) has been available for several years now. It integrates only with the Microsoft Windows Operating System and provides the capability of hosting applications using the .NET framework, COM+, Microsoft Message Queuing (MSMQ) without any additional configurations. IIS does not provide much support for the use of Java applications. The current version of IIS, IIS 6.0, offers several improvements from the previous versions. Approximately 50% of the SSL web sites have been using IIS 5.0 because of its performance. Microsoft has provided better performance and scalability in IIS 6.0.[12] Microsoft has developed
a Remote Certification Object which allows administrators to remotely access and manage their certificates.

IIS does not seem to offer a collaboration server. However, they do provide collaboration capabilities within their applications. For example, Microsoft Word offers an “Online Collaboration” menu item under “Tools”. It allows users to connect to other users over the same network. Any document written through Microsoft Office application that can be viewed in a web browser can be edited through online collaboration.

IIS also provides error logging capabilities. A new feature of IIS 6.0 is the logging of HTTP substatus codes. The HTTP substatus codes works as follows: “For example, if a request cannot be served because the application needed has not been unlocked (like ASP by default on clean installations), the client will get a generic 404. IIS 6.0 actually generates a 404.2, which will now be logged to W3C and Binary log files.”[12]

<table>
<thead>
<tr>
<th></th>
<th>Power</th>
<th>Ease</th>
</tr>
</thead>
<tbody>
<tr>
<td>Documentation</td>
<td>N/A</td>
<td>Good</td>
</tr>
<tr>
<td>Installation/configuration</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>Management/administration</td>
<td>Excellent</td>
<td>Poor</td>
</tr>
<tr>
<td>Logging Features</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>Security</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>Web development</td>
<td>Good</td>
<td>Good</td>
</tr>
</tbody>
</table>

Table 3 - Microsoft IIS features

2.2.3 Oracle HTTP Server

The Oracle HTTP Server (http://www.oracle.com) is prevalent among corporations that use Java as their main application development language. Oracle’s web server is
powered by Apache to make it reliable and secure. It has added several technologies, including some using the Apache modules, for developers to develop applications in like Perl, C, C++, and PLSQL (Oracle’s language). It is included with the Oracle Database Server and the Oracle 9ias Application Server. Some features of the Oracle HTTP Server are:

- **SSL support** – Oracle HTTP Server supports SSL and it is also preconfigured with a demo certificate, to ease development and testing in https mode.[14]

- **Single Sign-on Security** – “…, a new module – mod_osso – has been included in Release 2 to support single sign on across sites and across applications. This provides for a much better end user experience (they have to login only once), and a much easier development cycle.”[14]

- **PLSQL Server Pages** – a module is included to support PLSQL Server Pages which are HTML pages with PLSQL used as the scripting language embedded within the HTML.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Power</th>
<th>Ease</th>
</tr>
</thead>
<tbody>
<tr>
<td>Documentation</td>
<td>N/A</td>
<td>Good</td>
</tr>
<tr>
<td>Installation/configuration</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>Management/administration</td>
<td>Excellent</td>
<td>Excellent</td>
</tr>
<tr>
<td>Logging Features</td>
<td>Fair</td>
<td>Fair</td>
</tr>
<tr>
<td>Security</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>Web development</td>
<td>Fair</td>
<td>Poor</td>
</tr>
</tbody>
</table>

*Table 4 - Oracle HTTP server features*

The Oracle HTTP Server provides full support for SSL. Its server installation guide walks administrators through the process of generating certificate requests using
openssl and modifying files to enable SSL. If a certificate is needed for testing then the demo certificate that comes with the server can be used. If a certificate is required for a production site then openssl would need to be used to generate certificate requests which are sent to the Certificate Authority who will return a valid certificate.

The Oracle web server keeps track of any errors “that the server encounters by placing the error in a separate error log file. It makes it easier for the server administrators to diagnose the problem and to find a way to solve the problem. There is a mod_log_config module that can be configured to send all logging into a single file or multiple files.[15] Browser and referrer information can be added to log files by using a CustomLog directive as follows:

CustomLog logs/access_log "%h %l %u %t "%r"\n %s %b \"%{Referer}i\" \"%{User-Agent}i\""

“This will add the values of the User-agent: and Referer: headers, which indicate the client and the referring page, respectively, to the end of each line in the access log.”[15]

2.2.4 Sun ONE Server

Sun has its own server, Sun ONE Web Server (http://www.sun.com) that provides many features and fully supports the use of, of course, Java applications. Some of its features include:

- Conformity to US government requirements for protecting data that passes through the web server[21]
- Support for SSL, TLS and X.509 digital certificates that will provide “encrypted and authenticated transactions between the client and server”[21]
- SSL to LDAP communications which secures the communication of directory servers.
- Secure web server administration which allows the administration server to be SSL enabled.
- Dynamic log rotation that “allows access and error log files to be automatically rotate and archived”[21]
- Being fully compliant with J2EE standards, as well as supporting applications
written in Active Server Pages, ColdFusion, CGI, and PHP which makes Sun
One web server ideal for administrators wanting to support all types of web
applications.

The Sun ONE web server provides complete support for SSL and TLS which allows it to
offer a full range of security services and capabilities that can be customizable. For each
port that a Sun ONE web server instance listens to, the port can be enabled for SSL.
These SSL-enabled ports can be configured as follows:

- “Which server certificate will be used to identify the server on a given port.
- Which crypto module is used to process the requests on a given port.
- Which digest/encryption/signature algorithms are supported on a given port.
- Whether client authentication (two-way SSL authentication) is required on a
given port…
- File paths may be specified that require a certain level of protection on the
connection, based on active encryption key size (this enables the user to prevent
certain documents from being transmitted over a channel deemed insufficiently
secure for the type of document.”[22]

The Sun ONE web server has a user interface consisting of wizards to guide the
administrators through the management of certificates. This simplifies the process of
creating or modifying certificates.

<table>
<thead>
<tr>
<th></th>
<th>Power</th>
<th>Ease</th>
</tr>
</thead>
<tbody>
<tr>
<td>Documentation</td>
<td>N/A</td>
<td>Fair</td>
</tr>
<tr>
<td>Installation/configuration</td>
<td>Good</td>
<td>Poor</td>
</tr>
<tr>
<td>Management/administration</td>
<td>Good</td>
<td>Poor</td>
</tr>
<tr>
<td>Logging Features</td>
<td>Fair</td>
<td>Fair</td>
</tr>
<tr>
<td>Security</td>
<td>Good</td>
<td>Fair</td>
</tr>
<tr>
<td>----------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>Web development</td>
<td>Fair</td>
<td>Fair</td>
</tr>
</tbody>
</table>

Table 5 - Sun ONE features.

2.2.5 **AOLserver**

America Online bought a company called NaviSoft and transformed their NaviServer into what is now known as AOLserver (http://www.aolserver.com). AOLserver was released as an open-source web server in 1999. Its key features are its C and Tcl APIs that can be used to add more modules to the web server and to provide powerful, robust ways to access databases. It can also connect directly to SQL databases like SOLID and Postgres, and it provides an external database connection to Sybase. There are also third party databases drivers provided for Oracle, Informix, and Interbase.[1] Some of its other features include:

- **Access Control System** – permits administrators to define groups and users, assign users to groups, and assign permissions to the individual users and groups.
- **Multi-homing** – administrators can “run multiple servers on one host by attaching them to different port and/or IP address combinations.”[1]
- **Full CGI Support** – AOLserver can recognize .cgi files as CHI scripts and execute them in a separate process.
- **Tcl Interface** – The Tcl language can be used to crate additional modules. “Tcl functions are available to interact with the connection, implement scheduled procedures, maintain state, access the sockets and threads interfaces, roll access logs, maintain permission, access the database, …”[1]
- **C Interface** – “The C API lets you write your own logging routines, access control system, communications drivers, or database drivers. It provides all the necessary functions and structures you need to access AOLserver’s core functionality and the database.”[1]
- **Multithreading** – AOLserver can handle multiple connections, dynamically create
more threads as needed, and handle each request in its own thread and all under one process.

- Database Connection Pool Management

The AOLserver also supports AOLserver Dynamic Pages, or ADPs, which produce dynamic html pages by using Tcl scripts embedded within the html.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Power</th>
<th>Ease</th>
</tr>
</thead>
<tbody>
<tr>
<td>Documentation</td>
<td>N/A</td>
<td>Fair</td>
</tr>
<tr>
<td>Installation/configuration</td>
<td>Good</td>
<td>Fair</td>
</tr>
<tr>
<td>Management/administration</td>
<td>Good</td>
<td>Fair</td>
</tr>
<tr>
<td>Logging Features</td>
<td>Good</td>
<td>Fair</td>
</tr>
<tr>
<td>Security</td>
<td>Good</td>
<td>Fair</td>
</tr>
<tr>
<td>Web development</td>
<td>Good</td>
<td>Good</td>
</tr>
</tbody>
</table>

Table 6 – AOLserver features.

The AOLserver has an nsssl module that is a communication driver, used for implementing SSL in its server. The nsssl module supports SSLv2 but it does not include client authentication because it was deemed futile since most clients do not have client certificates. Tcl scripts handle requests for keys and certificates. Commands are entered to generate the necessary files: keyfile.pem is stored in the nsssl directory of AOLserver and certreq.pem is sent to the Certificate Authority. The Certificate Authority will return the necessary information is needed for the certificate. Only certain algorithms are permitted for use within the nsssl module of AOLserver. “…RSA, RC4, RC2, and some other algorithms, the 40-bit/512-bit export version is the only binary of nsssl that America Online is permitted to distribute from http://aolserver.com”[2] AOLserver suggests that if a stronger encryption algorithm
needs to be used then the administrative users of AOLserver should write their own nsssl module to encompass a stronger encryption algorithm. When the nsssl module has been properly loaded into the AOLserver, the SSL module can be configured using the AOLserver administration web site which will have a link to the SSL control. From there, the keys and certificates can be installed using a web-based interface.

The server administrators for AOLserver can configure the logging facility by using an ns_menu command in the control port interface of the server. The control port interface is part of a module that allows you to telnet to a specific host and port where the server can be administered and database commands can be executed while the server is running.[3] The logging menu provides two functions: Tail and Adjust. The Tail option allows administrators to see the log in the background while using the control port. The Adjust option adjusts the amount of logging information that is outputted to the log.

2.2.6 Xitami

The Xitami Web Server (http://www.xitami.com) was developed by iMatrix Corporation. It is available for free but any additional plug-ins or modules will cost more. There is also a commercial version that provides SSL support. Web server administration and maintenance is done through a graphical user interface. The Xitami web server consists of a WSX protocol (a plug-in extension) that allows for programming under WSX. “…WSX is a unique program written as a multithreaded SMT agent and is linked into the server executable.”[8] Xitami states that WSX can be used in place of CGI and will not crash like CGI. Xitami does support CGI scripts written in Perl, Awk, Rexx, Python, C, and Java. It also allows administrators to configure the server with changes taking effective immediately and without stopping and starting the server.
<table>
<thead>
<tr>
<th>Documentation</th>
<th>N/A</th>
<th>Fair</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installation/configuration</td>
<td>Good</td>
<td>Poor</td>
</tr>
<tr>
<td>Management/administration</td>
<td>Good</td>
<td>Fair</td>
</tr>
<tr>
<td>Logging Features</td>
<td>Good</td>
<td>Fair</td>
</tr>
<tr>
<td>Security</td>
<td>Good</td>
<td>Fair</td>
</tr>
<tr>
<td>Web development</td>
<td>Good</td>
<td>Good</td>
</tr>
</tbody>
</table>

**Table 7 – Xitami features.**

Xitami also provides full SSL and TLS support using the OpenSSL library (http://openssl.org), which Apache also uses for its SSL module. The SSL module is part of the Xitami/Pro package, so it would cost money to obtain that version of the Xitami server. The Xitami/Pro server would appropriately select the type of ciphers (methods for encrypting text) that should be used for a certain country since the use of the cipher depends on the country that it is being used for. Certificates can be created with the Xitami/Pro server. They have instructions that deal with how to create and use certificates (ssldoc.txt) and they have an sslhttp.cfs file that administrators can edit to specify the certificates to be used.

The Xitami server can be administered via a control panel or commands can be executed on the command line. Server administrators can execute commands to not log any messages or to trace any socket input/output operations to a log file. The name of the log file can also be customized to whatever the administrator wants. The default name of the log file is xitami.log. All HTTP requests are logged to this file with all the essential information like IP address of the requesting host, the HTTP request sent by the client, status code returned by the server, etc. Xitami saves all log files. When it is restarted it renames the old log file with a different name and starts a new xitami.log file.
2.2.7 Zeus Web Server

The Zeus Web Server (http://www.zeus.com) is a newcomer to the web server industry. It can deploy applications written in several languages, including J2EE, PHP, ASP, and Perl. It offers a web-based user interface for managing their servers. They provide optional modules so that the web administrator can customize the server to meet the company’s needs.

The Zeus Web Server also offers the latest implementations of SSL. SSL transactions can weaken performance on a server, causing slow client-server communications but Zeus has optimized its hardware so that performance does not have to be compromised when the server is fetching/receiving data during SSL transactions. “Zeus Web Server will alleviate these difficulties through its efficient process architecture and through its ability to perform more native SSL transactions per second than any other web server on comparable hardware.”[24] Zeus also consists of a web-based interface to manage certificates. It “is able to store securely ‘certificate sets’ and to ensure that these are made available to the web server or servers that it manages.”[24] This can help with deploying certificates to many servers at a time. Through the user interface, web administrators can obtain certificates directly from Verisign or they can choose another certificate authority. Self-signed certificates can also be created with the user interface.

<table>
<thead>
<tr>
<th></th>
<th>Power</th>
<th>Ease</th>
</tr>
</thead>
<tbody>
<tr>
<td>Documentation</td>
<td>N/A</td>
<td>Fair</td>
</tr>
<tr>
<td>Installation/configuration</td>
<td>Good</td>
<td>Poor</td>
</tr>
<tr>
<td>Management/administration</td>
<td>Good</td>
<td>Poor</td>
</tr>
<tr>
<td>Logging Features</td>
<td>Fair</td>
<td>Fair</td>
</tr>
<tr>
<td>Security</td>
<td>Good</td>
<td>Fair</td>
</tr>
</tbody>
</table>
2.2.8 **Cisco Collaboration Server**

The Cisco Collaboration Server (http://www.cisco.com) is tailored to businesses that are involved in sales and offer customer services like technical support. Customers can contact customer service representatives via the web while conducting a voice conversation (PSTN or Voice Over IP [VoIP]) or text chat. “Contact center agents and callers can share Web pages – including personalized or dynamically generated pages – complete forms in a collaborative fashion, and share any Windows desktop application using nothing more than a browser.”[7] The Cisco Collaboration Server can be integrated with a company’s telephone infrastructure to offer phone and Web-based inquiries. Presentations, demos and training session can be conducted with the Cisco Collaboration Server. Some web-based collaboration features include:

- Two-way Web page sharing
- Text chat
- Collaborative whiteboard

2.2.9 **Stellent Collaboration Server**

The Stellent Collaboration Server (http://www.stellent.com) allows businesses to work on projects in a collaborative effort. Users are added to projects once they are created. Each user has security levels when accessing data on the collaboration server. Users can use any application like Microsoft Office or Visio when they are “collaborating” and since this is all web-based, applications on the client side does not need to be installed. The collaborative server provides a centralized location to store project files and a secure “digital workspace” to collaborate on project content.[17] The Stellent Collaboration Server has the following features:

- General discussion boards for projects and multi-threaded discussion for specific
content items.[17]

• User-initiated content routing that enables contributors to route content to specific
project members prior to it becoming available to the rest of the project team.[17]

• Subscription services that allow users to monitor project content and receive
notification when new content is added or updated.[17]

2.2.10 Pace Web Server

The Pace Web Server is a lightweight server that was designed to emulate the Apache
Tomcat web server. It provides several features like multi-threading, HTTP 1.1-
compliancy, servlet container integration, support of CGI scripts using a servlet to handle
CGI requests. It was originally developed by Qing Jiang Lee, a graduate student from
Concordia University in Canada, in November 2000. It was written with JDK 1.3 and
supports the Java Servlet Development Kit (JSDK) 2.1 standard.

The Pace Web Server includes several enhancements that can possibly rival simple web
servers that exist in the market today. One feature that has been added is a Secure
Sockets Layer (SSL). The SSL capability will allow users of the server to securely send
private documents through the server. The documents are encrypted with a TLS
algorithm. The Pace Web Server will allow administrators to use their own certificates
for authentication.

Another feature that has been added to the Pace Web server is a collaborative chat utility
and a whiteboard tool. This allows users to chat via a web-based chat utility and to draw
or write text in a collaborative environment. The chat utility and the whiteboard tool run
as applets so there are no additional programs that need to be downloaded by the user.

Another feature is error logging. This is a system administrative tool that allows
administrators to view any errors that occurred from within the server. Any errors that
occur will be logged and administrators will be able to have some understanding as to
why or how an error has occurred. The administrator will have an option to enable or
disable general logging and/or error logging.
Most of the code that I had written to accommodate these changes was drawn from the existing code because I did not want to deviate from the approach that was taken in designing original code. This makes it easier for any future developers of this code to maintain and add more features to it.

The classes for the Pace Web Server are listed in the diagram below. The new classes are represented in **bold/italic**. They were added for the SSL, Collaboration chat and whiteboard tools, and the error logging. The PWS.java file is the starting point for running the Pace Web Server.
<table>
<thead>
<tr>
<th>edu.pace.web.ServletManagement</th>
<th>edu.pace.web.net</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServletManager</td>
<td>CollabServeConnection</td>
</tr>
<tr>
<td>ServletException</td>
<td>CollabServerContext</td>
</tr>
<tr>
<td>SessionManager</td>
<td>PipedSocket</td>
</tr>
<tr>
<td>ExtendServletConfig</td>
<td>ServeConnection</td>
</tr>
<tr>
<td></td>
<td>ServerContext</td>
</tr>
<tr>
<td></td>
<td>ServerInputStream</td>
</tr>
<tr>
<td></td>
<td>ServerOutputStream</td>
</tr>
<tr>
<td></td>
<td>SSLServeConnection</td>
</tr>
<tr>
<td></td>
<td>SSLServerContext</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>edu.pace.web.utils</th>
<th>edu.pace.web.servlets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configurations</td>
<td>CgiServlet</td>
</tr>
<tr>
<td>ConfigurationsRepository</td>
<td>Counter</td>
</tr>
<tr>
<td>ErrorLog</td>
<td>DateServlet</td>
</tr>
<tr>
<td>ExtendedProperties</td>
<td>GBTextServlet</td>
</tr>
<tr>
<td>Fmt</td>
<td>GuestBook</td>
</tr>
<tr>
<td>Utils</td>
<td>HelloWorld</td>
</tr>
<tr>
<td>WildcardDictionary</td>
<td>ParamServlet</td>
</tr>
<tr>
<td></td>
<td>SessExample</td>
</tr>
<tr>
<td></td>
<td>ViewGBText</td>
</tr>
<tr>
<td></td>
<td>ViewGuestBook</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>edu.pace.web.win32</th>
<th>edu.pace.web.service</th>
</tr>
</thead>
<tbody>
<tr>
<td>AboutBox</td>
<td>CollabLocalService</td>
</tr>
<tr>
<td>GeneralOption</td>
<td>ConnectionService</td>
</tr>
<tr>
<td>MyHelpFrame</td>
<td>LocalService</td>
</tr>
<tr>
<td>PWSconfFrame</td>
<td>RemoteService</td>
</tr>
<tr>
<td>TrayIconButton</td>
<td>ServiceManager</td>
</tr>
<tr>
<td>SessionOption</td>
<td>SSLLocalService</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>edu.pace.web.service</th>
<th>edu.pace.web.win32</th>
</tr>
</thead>
<tbody>
<tr>
<td>CollabLocalService</td>
<td>AboutBox</td>
</tr>
<tr>
<td>ConnectionService</td>
<td>GeneralOption</td>
</tr>
<tr>
<td>LocalService</td>
<td>MyHelpFrame</td>
</tr>
<tr>
<td>RemoteService</td>
<td>PWSconfFrame</td>
</tr>
<tr>
<td>ServiceManager</td>
<td>TrayIconButton</td>
</tr>
<tr>
<td>SSLLocalService</td>
<td>SessionOption</td>
</tr>
<tr>
<td></td>
<td>ServletOption</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>edu.pace.web.usage</th>
<th>edu.pace.web.service</th>
</tr>
</thead>
<tbody>
<tr>
<td>CollabClient</td>
<td>Controls</td>
</tr>
<tr>
<td>CollabConsumer</td>
<td>DrawingArea</td>
</tr>
<tr>
<td>CollabObj</td>
<td>WBConsumer</td>
</tr>
<tr>
<td>CollabServer</td>
<td>WhiteBoardClient</td>
</tr>
<tr>
<td>CollabUser</td>
<td>WhiteBoardDebugFlags</td>
</tr>
<tr>
<td></td>
<td>WhiteBoardServer</td>
</tr>
<tr>
<td></td>
<td>WhiteBoardUser</td>
</tr>
</tbody>
</table>
Figure 2 – Pace Web Server Java classes.

The console has also been to reflect the changes that were made.

Figure 3 – Screenshot of Pace Web Server console with enhancements.
The major visible changes in the console from the previous version of the Pace Web Server and the current version are the addition of three entry fields for three different ports. The Pace Web Server administrator can enter a separate port for HTTP, another one for HTTPS (HTTP + SSL), and a collaboration port to listen to requests that are for the collaboration chat utility. The “default” port (this is configurable so the Pace Web Server may not come with these ports as the default) for HTTP is 8989, HTTPS port is 7777 and Collaboration port is set to 8787. If no port is specified for the collaboration and SSL ports then these services will not be available through the Pace Web Server. There are checkboxes to enable general logging and error logging of the Pace Web Server. It will initially be set to enable both types of logging since this is vital for gathering important data when an error occurs. The “SSL Certificate Information” section has also been added to the console. It allows administrators to select the location of their certificate and enter the password for it. These values, along with the other configurable items displayed in the configuration screen, are stored in conf\pws.conf. The Pace Web Server reads the values from this file and uses it within the application code and displays the values on the configuration screen.
Chapter 3

Secure Pace Web Server with SSL

3.1 Introduction to SSL

How to make data secure on the web has been an ongoing question and a major problem with the Internet. It is so effortless for an intruder to intercept emails, so why not try intercepting a client/server communication channel? Information that is valuable to the user is passed between the client and the server so it makes perfect sense for intruders to listen in on the server’s ports and obtain the data. Unauthorized users can and already have attacked servers to get personal information, such as credit card numbers, social security numbers, etc, from web surfers.

All servers are prone to these invasive types of attacks but Secure Socket Layers, or SSL, can prevent this from occurring. SSL encrypts data that is sent between the client and server. Only the server can decrypt the data using a key. This prohibits the intruder from procuring any vital information about a person. This section goes into detail about the security threat of unencrypted data sent over the Internet, as well as the solution to the threat, SSL.

3.2 Overview of SSL

The Internet has been rapidly expanding over the last several years because the price of getting the Internet into residential homes has decreased, along with the cost of computers which have seen a dramatic decrease in price and an increase in processing speed and features. As a result many people from all around the world connect to the Internet for its intended purpose of accessing information. There are also people who would connect to the Internet for malicious reasons like hacking into web sites, spreading
viruses or accessing unauthorized information. It is easy for people to do these kinds of attacks because in the Internet world no one knows who they are or how they look so they are anonymous. Their identity is not known so they can easily pose as someone else and obtain that person’s private information without being caught.

<table>
<thead>
<tr>
<th>Adversary</th>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student</td>
<td>To have fun snooping on people’s email</td>
</tr>
<tr>
<td>Hacker</td>
<td>To test out someone’s security system; steal data</td>
</tr>
<tr>
<td>Sales rep</td>
<td>To claim to represent all of Europe, not just Andorra</td>
</tr>
<tr>
<td>Businessman</td>
<td>To discover a competitor’s strategic marketing plan</td>
</tr>
<tr>
<td>Ex-employee</td>
<td>To get revenge for being fired</td>
</tr>
<tr>
<td>Accountant</td>
<td>To embezzle money from a company</td>
</tr>
<tr>
<td>Stockbroker</td>
<td>To deny a promise made to a customer by email</td>
</tr>
<tr>
<td>Con man</td>
<td>To steal credit card numbers for sale</td>
</tr>
<tr>
<td>Spy</td>
<td>To learn an enemy’s military strength</td>
</tr>
<tr>
<td>Terrorist</td>
<td>To steal germ warfare secrets</td>
</tr>
</tbody>
</table>

Figure 4. Some people who cause security problems and why.[23]

Evidently, something had to be done about the open communication of data between the client and server. In 1994 Netscape developed a protocol that can allow clients (mostly, web browsers) and web servers to communicate via a secure channel. This protocol became known as Secure Sockets Layer, or SSL which it is commonly referred by. “It offers encryption, source authentication, and data integrity as means to protect information exchanged over insecure, public networks.”[11] Encryption transforms the data by using some type of algorithm to change the data, then the data is sent to its intended receiver and it is decrypted by the receiver and the data is processed. Intruders would not be able to view the actual data, thereby protecting the data from them. Source authentication verifies “the data sender’s identity. The first time a browser or other client attempts to communicate with a web server over a secure connection, the server presents the client with a set of credentials in the form of a certificate.”[11] A certificate is a
document that provides authentication of its owner and it is verified by a Certificate Authority. This process will be discussed later in detail. The third feature that SSL offers is data integrity which “refers to means of ensuring that data has not been modified in transit.”[11]

In the TCP/IP stack order, SSL runs above the TCP/IP and below the application layer, which includes HTTP, LDAP and IMAP protocols, to name a few.

SSL “uses TCP/IP on behalf of the higher-level protocols, and in the process allows an SSL-enabled server to authenticate itself to an SSL-enabled client, allows the client to authenticate itself to the server, and allows both machines to establish an encrypted connection.”[18] SSL was designed to work within an environment with few servers and many clients. SSL-enabled servers need to provide a certificate to authenticate themselves to their clients but clients do not need to provide a certificate to the server unless it is requested by the server.

Servers that are SSL-enabled need to have a certificate to provide authentication to the clients. Certificates are “electronic documents used to uniquely identify principals and entities over networks such as the Internet.”[4] Certificate authorities issue the certificates. When they are created, the certificates are signed with the certificate
authority’s private key to avoid tampering of it. The certificates contain the following information:

- “Name of the subject (holder, owner) and other identification information required to uniquely identify the subject, such as the URL of the Web server using the digital certificate, or an individual’s email address.
- The subject’s public key.
- The name of the certificate authority that issued the certificate.
- A serial number.
- The validity period (or lifetime) of the certificate.”[4]

3.2.1 SSL Handshake

When a client and server begin communicating, they begin an exchange of information known as a handshake. The following figure shows how the SSL handshake protocol functions.
1. The client connects to the server and sends information to the server like “SSL protocol version, session id, and cipher suites information such as cryptographic algorithms and key sizes supported.”[11]

2. The server responds to the client by sending the cipher suite that is supported by the client and the server.

3. “The server sends the client its certificate which contains the server’s public key. While this message is optional, it is used when server authentication is required.”[11] Basically, the server is sending a certificate to verify its identity to the client.

4. If the server requests a certificate from the client to authenticate it then this message would be sent but this rarely occurs.

5. “The server initiates a key exchange algorithm, based in part on the information
contained in the certificate it has just sent, and sends the necessary key
exchange information to the client.”[13]
6. “This message informs the client that the server finished the initial negotiation
process.”[11]
7. If the server has requested that the client authenticate itself then this message
would be sent.
8. “The client generates a secret key to be shared between the client and server. If
the Rivest-Shamir-Adelman (RSA) encryption algorithm is used, the client
encrypts the key using the server’s public key and sends it to the server. The
server uses its private or secret key to decrypt the message and retrieves the
shared secret key. Now, client and server share a secret key that has been
distributed securely.”[11]
9. This message is sent to complete the authentication process if the server requested
that the client be authenticated.
10. The server is asked by the client to change to encrypted mode.
11. The client lets the server know that it is ready for secure communications.
12. The client is asked by the server to change to encrypted mode.
13. The server notifies the client that it is ready for secure communications. At this
point, the SSL handshake ends.
14. The client and server begin exchanging their messages that are encrypted over a
secure channel.

3.3 Java and SSL

Sun has developed an API that allows developers to easily incorporate SSL features in
their Java applications. Java Secure Socket Extension (JSSE) “provides a means to create
and use SSL sockets as well as protocol handlers to support the HTTPS protocol.”[13] It
gives the option of implementing either SSL 3.0 or TLS (Transport Layer Security) 1.0
protocols. TLS is actually SSL but it is the next version of SSL. The JSSE API is
included in J2SE 1.4 but users of previous versions (1.2.1 or higher) of the Java
Development kit can download the API separately. After downloading JSSE, there are
three JAR files (jcert.jar, jnet.jar, jsse.jar) that need to be placed on the classpath before
using it. Also, a line has to be added to the java.security file to register the
cryptography provider:

    Security.provider.[#]=com.sun.net.ssl.internal.ssl.Provider

In this case, the Sun internal SSL provider is used, which is shipped with the JRE. The #
sign should be replaced with the next available number from the list of providers in the
java.security file. Another way to add the provider is by programmatically adding it as
follows:

    Security.addProvider(new com.sun.net.ssl.internal.ssl.Provider());

There are four packages defined in the JSSE API:

**javax.net.ssl**

“The abstract classes that define Java’s API for secure network
communication.”[9]

**javax.net**

“The abstract socket factory classes used instead of constructors to create secure
sockets.”[9]

**javax.security.cert**

“A minimal set of classes for handling public key certificates that’s needed for
SSL in Java 1.1. (In Java 1.2 and later, the java.security.cert package should be
used instead).”[9]

**com.sun.net.ssl**

“The concrete classes that implement the encryption algorithms and protocols in
Sun’s reference implementation of the JSSE. Technically, these are not part of the JSSE
standard. Other implementers may replace this package with one of their own; for
instance, one that uses native code to speed up the CUP-intensive key generation and
encryption process.”[9]
There are several useful tools provided by Sun for key management. Java’s `keytool` utility which is included in the JRE is a command-line tool that allows administrators to create keys, import digital certificates, and export existing keys. It basically provides interaction with the key management system. Keys and certificates are stored in a file with a `.keystore` extension. To create a key entry that holds a private key and certificate, the `–genkey` option should be used. The following is an example of creating a key using `keytool`:

```
keytool -genkey -alias newkey -keystore mykeys -keyalg rsa
```

An entry named “newkey” has been generated and placed in the keystore named “mykeys”. The steps that follow show the process of creating the key:

Enter keystore password: newkey
What is your first and last name?
    [Unknown]: Pace University
What is the name of your organizational unit?
    [Unknown]: Pace University
What is the name of your organization?
    [Unknown]: Pace University
What is the name of your City or Locality?
    [Unknown]: White Plains
What is the name of your State or Province?
    [Unknown]: NY
What is the two-letter country code for this unit?
    [Unknown]: US
Is CN=Pace University, OU=Pace University, O=Pace University,
L=White Plains, ST=NY, C=US correct?
    [no]: y

Enter key password for <newkey>
    (RETURN if same as keystore password): [Enter]
The SSL-enabled server must have a private key and a certificate to confirm its identity. “The private key is used by the server as part of the key exchange algorithm, and the certificate is sent to the client to tell the client who the server is.”[13] To generate a certificate request from a certificate authority, the –certreq option must be used. The following is an example of using this option:

```
keytool -certreq -alias newkey -file newkey.cer
```

The newkey.cer file is a Certificate Signing Request, or CSR, that has to be sent to the CA so that a certificate can be obtained. When the file is sent to the CA, a certificate will be returned, verifying the public key that was sent with the CSR.

Once the certificate is generated, it can be used through the major web browsers like Internet Explorer and Netscape Navigator.

A client and server need to communicate using a secure channel of communication. In the JSSE API the `javax.net.SSLServerSocket` abstract class would need to be used to create this secure channel. An instance of the `SSLServerSocket` class is created by the abstract factory class `javax.net.SSLServerSocketFactory`. To get an instance of the `SSLServerSocketFactory`, the method `getDefault()` from the `SSLServerSocketFactory` class should be used. When the instance of `SSLServerSocketFactory` is created then the `createServerSocket()` method from the `SSLServerSocketFactory` class can be called. This creates a secure socket on the server side. This provides only server authentication, and not encryption. More has to be done in the initialization process to allow encryption of data but the procedures for doing this varies.

To create a secure server socket, the following steps must be completed:

- “Generate public keys and certificates using `keytool`.
- Pay money to have your certificates authenticated by a trusted third party such as Verisign.
- Create an `SSLContext` for the algorithm you’ll use.
• Create a TrustManagerFactory for the source of certificate material you’ll be using.

• Create a KeyManagerFactory for the type of key material you’ll be using.

• Create a KeyStore object for your key and certificate database. (Sun’s default is JKS).

• Fill the KeyStore object with keys and certificates; for instance, by loading them from the filesystem using the pass phrase they’re encrypted with.

• Initialize the KeyManagerFactory with the KeyStore and its pass phrase.

• Initialize the context with the necessary key managers from the KeyManagerFactory, trust managers from the TrustManagerFactory, and a source of randomness…”[9]

These instructions are the programmatic steps that need to be taken to define an SSL-enabled server socket. A server can request that a client get authenticated by setting the setNeedClientAuth() method of the SSLSocket class to true. If the client cannot be authenticated then the server will refuse any communication from that client.

Creating a secure client socket is not a difficult task since the encryption schemes and exchanging of keys are transparent to the developer. The createSocket() method from the javax.net.ssl.SSLSocketFactory abstract class will return a secure socket which is a subclass of the java.net.Socket class. An instance of the SSLSocketFactory will need to be obtained since it is an abstract class. This instance can be obtained by calling the SSLSocketFactory.getDefault() method, which will return an instance of the SSLSocketFactory. Once this is done, the createSocket() method would create the secure client socket which can then be used like a regular socket, using all the methods that a java.net.Socket object has.
There is much overhead involved when using SSL. The client and server need to exchange keys and encryption methods. They also have to know which ones are supported. This can be done by calling the `getSupportedCipherSuites()` method of the `SSLSocket` class. A cipher suite is an “SSL encryption method that includes the key exchange algorithm, the symmetric encryption algorithm, and the secure hash algorithm used to protect the integrity of the communication.”[4] All the cipher suites that are returned may not be suitable to use because they may be weak so the `getEnabledCipherSuites()` method should be used to determine which cipher suites are suitable for the socket to use. The suite used would ultimately be chosen by the client and server. The following table shows the cipher suites that JSSE supports in its implementation. The strings are concatenated together to come up with an SSL specification: `SSL_<key exchange algorithm>_WITH_<encryption algorithm>_WITH_<hash algorithm>`. The number in the encryption algorithm is significant. “When a number appears in the encryption algorithm, it refers to the key strength of the encryption: higher numbers are more secure…”[13]

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Key Exchange</th>
<th>Encryption</th>
<th>Hash</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSL</td>
<td>DH_anon_EXPORT</td>
<td>WITH_DES40_CBC</td>
<td>SHA</td>
</tr>
<tr>
<td>SSL</td>
<td>DH_anon_EXPORT</td>
<td>WITH_RC4_40</td>
<td>MD5</td>
</tr>
<tr>
<td>SSL</td>
<td>DH_anon</td>
<td>WITH_3DES_EDE_CBC</td>
<td>SHA</td>
</tr>
<tr>
<td>SSL</td>
<td>DH_anon</td>
<td>WITH_DES_CBC</td>
<td>SHA</td>
</tr>
<tr>
<td>SSL</td>
<td>DH_anon</td>
<td>WITH_RC4_128</td>
<td>MD5</td>
</tr>
<tr>
<td>SSL</td>
<td>DHE_DSS_EXPORT</td>
<td>WITH_DES40_CBC</td>
<td>SHA</td>
</tr>
<tr>
<td>SSL</td>
<td>DHE_DSS</td>
<td>WITH_3DES_EDE_CBC</td>
<td>SHA</td>
</tr>
<tr>
<td>SSL</td>
<td>DHE_DSS</td>
<td>WITH_DES_CBC</td>
<td>SHA</td>
</tr>
<tr>
<td>SSL</td>
<td>RSA_EXPORT</td>
<td>WITH_RC4_40</td>
<td>MD5</td>
</tr>
<tr>
<td>SSL</td>
<td>RSA</td>
<td>WITH_3DES_EDE_CBC</td>
<td>SHA</td>
</tr>
<tr>
<td>SSL</td>
<td>RSA</td>
<td>WITH_DES_CBC</td>
<td>SHA</td>
</tr>
<tr>
<td>SSL</td>
<td>RSA</td>
<td>WITH_NULL</td>
<td>MD5</td>
</tr>
</tbody>
</table>
Table 9 – SSL Cipher Suites Supported by Sun’s JSSE Implementation.[13]

<table>
<thead>
<tr>
<th>SSL_</th>
<th>RSA_</th>
<th>WITH_NULL_</th>
<th>SHA</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSL_</td>
<td>RSA_</td>
<td>WITH_RC4_128_</td>
<td>MD5</td>
</tr>
<tr>
<td>SSL_</td>
<td>RSA_</td>
<td>WITH_RC4_128_</td>
<td>SHA</td>
</tr>
</tbody>
</table>

All the cipher suites listed in the table are supported but the cipher suites in bold are the ones that are enabled in the JSSE implementation. The other cipher suites have to be enabled explicitly. This is done by calling the `setEnabledCipherSuites()` method and pass the array that is returned from the `getSupportedCipherSuites()` method.

### 3.4 SSL in Pace Web Server

The SSL feature requires a separate port the server should listen to for incoming requests. When a user tries to obtain a page using the SSL protocol (HTTPS) there will be a slight delay because there is a great amount of processing involved since it requires getting the necessary encryption/authentication information to proceed with a secure service request.

In the starting point class, `PWS.java`, an instance of the `SSLServerContext` class, which uses the `Runnable` interface, is created. It is passed as an argument to a `Thread` object and the thread is started, as shown below:

```java
if (confs.getInteger("SSL_PORT") != 0) {
    sslserver = new SSLServerContext(confs);
    Thread sslmain = new Thread(sslserver);
    sslmain.start();
}
```

If no port was entered then the SSL will not be enabled.

Within the `SSLServerContext` class, the SSL server socket is created and waits for any incoming request which is how it works for the HTTP and Collaboration ports. The
key difference is that there is a method call to a
getSSLServerSocketFactory() that returns an SSL-enabled server socket.

The getSSLServerSocketFactory() method is responsible for obtaining the
keystore information and getting any other necessary information to allow authentication
on the server side. Servers do not normally request client authentication and as a result of
this, client authentication is not required. The type of protocol used for securing the site
is TLS. TLS is the latest version of SSL to be used for SSL encryption and
authentication.

The keystore implementation that I used is the default provided by SUN Microsystems,
called “JKS”, which is provided based on the provider that is used, namely SUN’s
internal SSL provider, com.sun.net.ssl.internal.ssl.Provider. It is specified in
the java.security file as:

```
keystore.type=jks
```

“A keystore type defines the storage and data format of the keystore information, and the
algorithms used to protect private keys in the keystore and the integrity of the keystore
itself.”[19] If a keystore was created using a different implementation then an exception
like “Invalid keystore format” will be thrown. To prevent this from occurring I called the
method KeyStore.getDefaultType() which will get the default keystore type
specified in the java.security file. Administrators will be able to create their own
keystores using the keystore implementation they specify in that file.

Administrators can specify their own keystores through the Pace Web Server console by
selecting the location of the keystore file and the password of the keystore. A keystore is
“a database of keys. Private keys in a keystore have a certificate chain associated with
them, which authenticates the corresponding public key.”[19] The variables in the
SSLServerContext, namely sslCertPath and sslCertPassword, will hold
the absolute path of the keystore containing the certificate and the password for the
keystore, respectively. The password for the keystore cannot be changed within the
console. Instructions for creating a keystore can be found in the appendix.
The main part of the code to acquire the SSL server socket factory is as follows, with the server socket factory being returned:

```java
SSLContext sslContext;
KeyManagerFactory keyManFactory;
KeyStore keyStore;
char[] keyPassPhrase = sslCertPassword.toCharArray();

// add provider
System.setProperty("java.protocol.handler.pkgs", "com.sun.net.ssl.internal.www.protocol");
java.security.Security.addProvider(new com.sun.net.ssl.internal.ssl.Provider());

// set algorithm
sslContext = SSLContext.getInstance("TLS");
keyManFactory = KeyManagerFactory.getInstance("SunX509");
keyStore = KeyStore.getInstance(KeyStore.getDefaultType());

String keyStoreFile = sslCertPath;
keyStore.load(new FileInputStream(keyStoreFile), keyPassPhrase);
keyManFactory.init(keyStore, keyPassPhrase);
sslContext.init(keyManFactory.getKeyManagers(), null, null);

ssf = sslContext.getServerSocketFactory();
return ssf;
```

After the SSL server socket factory is returned, a server socket is created by calling the `createServerSocket(int port)` method from the SSL server socket factory instance with the value of the port that will use SSL passed to it. Once the SSL server socket is created it can be used like a regular server socket that accepts HTTP requests. Another thing to note is that not all cipher suites are supported by the SSL implementation that is used. To work around this, the following line was used:

```java
serverSocket.setEnabledCipherSuites(serverSocket.getSupportedCipherSuites());
```
This method call to `setEnabledCipherSuites` enables all the cipher suites supported by the SSL provider specified in the code above. After the previous lines are executed, the server socket calls its `accept()` method and waits to service any client requests. All the processing done for the clients’ requests are serviced the same way as the clients’ requests coming in from an HTTP request. This makes it easy to switch the SSL server socket to a regular server socket, if it is necessary.

After a user types the URL to access an HTML page using the HTTPS protocol on the Pace Web Server, the following will appear:

![Security Alert](image)

Figure 7 – Certificate warning.

This lets the user know that the certificate may not be valid because the certificate does not come from a trusted Certificate Authority. If the “View Certificate” button is clicked then the following will appear:
Figure 8 – Certificate information.

The message states that the certificate can be installed so that “trust” may be enabled. Even though this occurs the user can still click on the “Yes” button to proceed and go through a secure connection to the server as the following figure shows, after the user had made a request for https://localhost:7777/index.html.
Figure 9 – Success after using the HTTPS protocol in Pace Web Server.
Chapter 4

Online Collaboration

4.1 Overview of Collaborative Computing

Collaborative computing can be defined as a means in which users can work together in real-time, full duplex mode, which means they can send and receive data simultaneously. It can mean various other things like:

- “Concurrent application usage
- Multi-user application development toolkits
- Interactive desktop conferencing
- Distributed presentations over the network
- Interacting in ‘virtual’ environments
- Workflow; i.e., coordination, transfer and development of information”[20]

There are many groups that are striving to get their collaboration tools developed and made readily available to all users. Sun has a small research group, Collaborative Computing Group or COCO (http://research.sun.com/research/coco/) that has made a group effort to analyze the uses of collaborative computing. Microsoft has an online collaborative feature built into some of its Microsoft Office products. They have developed a desktop application for making video conference connections as their initial project. IEEE also has a collaborative computing research group as well (http://dsonline.computer.org/collaborative/). This group has developed a collaborative editing tool utilizing MS Word. There is a market out there for collaborative tools because they make things easier to deal with. People can work together without having to physically meet somewhere. It omits the need for taking business trips, or occupying office space since people can telecommute if they have such tools. Sun has taken an
effort to join in on the collaborative computing research by building a collaboration
API called Java Shared Data Toolkit.

4.2 Java and Collaboration

4.2.1 Overview of JSDT

Java had developed an API that allows clients to connect to and share the same work
space. It is called the Java Shared Data Toolkit (JSDT) API. It was developed by Rich
Burridge and it was originally released as a technology release on the Java Developer
Connection on java.sun.com and then update were made and finally the first supported
version, JSDT 1.4, was released.[20] Unfortunately, it is no longer being supported by
Sun and there is not much information available about JSDT and there is not a wide use
of it either since many developers gave up on it after it lost support from Sun. Since it is
not being supported, it has been difficult to implement the JSDT application within the
Pace Web Server. The last supported version that was released was JSDT version 2.0
which I have used in developing the chat applet. However, the toolkit can still be
downloaded at http://www.sun.com/software/jsdt/ and there are developers that still use
the JSDT API. They can be contacted through the JSDT mailing list or a question can be
posted in the Sun Java Forums.

4.2.2 How JSDT Works

JSDT “implements a multipoint data delivery service for use in support of highly
interactive, collaborative, multimedia applications.”[20] Chat forums, whiteboards, and
various other types of collaborative applications can be created with ease using the JSDT.
It stores client objects in a single database, or registry and creates a session as a channel
of communication between the clients. Clients can then “talk” to each other
simultaneously in real-time. “It provides the basic abstraction of a session (i.e., a group
of objects associated with some common communications pattern), and supports full-
duplex multipoint communication among an arbitrary number of connected application
entities -- all over a variety of different types of networks.”[20]
Before a JSDT application can be executed, a registry needs to be started. This registry is a transitory database that stores JSDT objects, either a Session or a Client. The registry can be started at the command line or within the JSDT application’s server. To start it at the command line, the following has to be entered at the prompt:

```
Prompt>java com.sun.media.jsdt.<type>.Registry
```

where `<type>` is socket, http, lrimp, or rmi. The `RegistryFactory` class can be used to start the registry as well, as follows:

```
RegistryFactory.startRegistry(String type);
```

where, `type` can be one of the implementation types. When the registry has started it will be empty. When clients or sessions are started then they will be added in the registry.

A JSDT session consists of “a collection of related Clients which can exchange data via defined communication paths.”[5] All the clients that are spawned from one JSDT application session, like a whiteboard session, can be stored in a single session. The session will maintain the communication lines among the whiteboard clients, using channels, tokens or byteArrays to share data. A Session object is created by using the `createSession()` method of the SessionFactory class. An example is shown below:

```
SessionFactory.createSession(client, urlstring, true);
```

The client is a JSDT `Client` object that is specific to the JSDT application, the `urlstring` parameter represents a “URL” to connect to the session. This will be discussed later. The `true` parameter is an autojoin parameter that lets the client automatically join in the session when it is created. Once the session is created clients are able to join the session and share data.

A JSDT `Client` object “is part of a JSDT application or applet and is a participant in an instance of multiparty communications. Once properly associated with one another..., related Clients can transfer data in a point-to-point or multipoint fashion.”[5] Once clients are created, they can be the sender or receiver of data that is exchanged through a
channel of communication amongst each other. The Client is actually an interface so the Client class in a JSDT application needs to implement this Client interface which consists of two methods:

```java
public Object authenticate(AuthenticationInfo info);

public String getName();
```

The authenticate() method is used to authenticate a Client object. A Manager object may request authentication from the Client object. A Manager object “encapsulates some management policy for a Manageable object (i.e. a ByteArray, Channel, Session or Token).”[5] If managed objects are not being used then null can be returned. The getName() method returns the name of the Client object.

The Client objects connect to a session via a URLString object which takes the form of:

```
jsdt://<host>:<port>/<connection type>/<object type>/<object name>
```

where,

- `<host>` represents the name of the server for the JSDT Session,
- `<port>` is the port number to use for the JSDT connections,
- `<object type>` is either a “Client” or “Session” object type,
- `<connection type>` is the connection/implementation type to be used (can be either http, lrmp, or socket),
- `<object name>` is the name of the JSDT Session.

This URLString object is created by using the createSessionURL() method of the URLString class.

The Channel object is the communication path among the clients in a particular session. If a client registers with a particular channel then that client can send or receive
data through that channel. “Channels are session-wide addresses. Every client of a session can join a Channel to receive data sent to it, and by joining an appropriate combination of Channels, and by consuming them, a Client can choose to receive Data sent to these Channels and ignore Data sent to other Channels.”[5] A Channel object is created by calling the `createChannel(Client client, java.lang.String channelName, boolean reliable, boolean ordered, boolean autoJoin)` method of the Session class. The first call to this method will create the channel within a given session and subsequent calls to createChannel method will return a reference to that channel. Multiple channels can be created by calling the `createChannel()` method and specifying a different name for the channel.

The `ChannelConsumer` interface is “a Client object which has registered its interest in receiving Data sent over a given Channel.”[5] The data is sent asynchronously over the channel. It has one method:

```java
public void dataReceived(Data data);
```

The consumer is added to a channel by calling the channel’s `addConsumer()` method and passing the consumer object as its parameter.

The `Data` object contains the content of the message to be sent to the other clients. It “is a discrete unit of data (array of bytes) that is sent by a Client over a Channel to all of the Clients which have currently registered an interest in receiving data on the given Channel…”[5] This Data object is what is passed around, over the communication path. It contains several methods to obtain certain information about the data that is being sent like, the channel the data was sent over, the name of the sender, or the length of the data sent. The data can be prioritized so if some data had to be sent with high priority then the `setPriority()` method would be called as follows:

```java
setPriority(Channel.TOP_PRIORITY);
```

When data is being sent, it can be sent to all the clients who are using the particular channel, or it can be sent to the other clients who are all the clients except the one that has
sent the data, or it can be sent to one client. Data can be received by calling the `dataReceived()` method in which a `Data` object is passed as its parameter.

### 4.3 JSDT in Pace Web Server

A chat applet written with the Java Shared Data Toolkit API from Sun was also integrated into the server. It runs off of its own port that can be specified on the Pace Web Server console. I chose to have it run off of another port because it would probably be easier to integrate other collaboration tools off this port and the Pace Web Server could officially be called a collaboration server as well. I tackled many problems using this JSDT API. One major problem is that it is not being supported by Sun and since JSDT has not been in existence for a long time (approximately from 1999-2001) many people have not used this technology. As a result, it was difficult to obtain any information related to JSDT outside of the Sun web site. Since there was no support there was no place to turn to for assistance in overcoming the problems that have occurred while working with JSDT. JSDT would have been a fine technology to use for collaboration and the basic code was straightforward because most of the underlying connections were hidden from the developer.

The JSDT toolkit came with several examples. One of the examples included a chat application which I used as the underlying base for the Pace Web Server version of the collaboration chat. Another example is a whiteboard which I also incorporated into the Pace Web Server.

The basic structures and objects of JSDT classes are similar. Using the chat utility as an example, these objects consist of a client class which represents the chat user (`CollabClient.java`), a consumer class which is responsible for sending the data to the other clients (`CollabConsumer.java`), a server class that starts the JSDT registry and the session (`CollabServer.java`), and a user class which is responsible for displaying the user interface and listening to the events that take place within the chat applet and calling the appropriate methods to get the data across (`CollabUser.java`). An instance of the `CollabServer` class is created within the `CollabServerContext` class. The `CollabServerContext` class initializes the port, name of host, etc. `CollabServer.java` also
contains this information, like the port number and the host name, but this information is specific to the chat tool, not the collaboration server. The chat tool itself has its own port number because it needs to communicate with the other clients so, using the collaboration port specified for the collaboration server would not suffice. Also, if that were used then no other collaboration tools would be able to run from that port since it will be in use by the chat applet. The collaboration port runs within its own thread and the server listens for requests made through this port. If a port was entered in the console for the collaboration port then a thread for the collaboration server would be initialized and started as follows:

```java
if (confs.getInteger("COLLAB_PORT") != 0) {
    collabserver = new CollabServerContext(confs);
    Thread collabmain = new Thread(collabserver);
    collabmain.start();
}
```

The value for “COLLAB_PORT” that is stored in the `pws.conf` file would be retrieved and checked to make sure it is not equal to zero (zero would be stored in the file if no value was entered for the collaboration port). When the user selects the chat tool then the applet would load.

The chat and whiteboard applets can be accessed from the main collaboration server page:
After clicking on the “Chat” link from the menu on the left, the chat applet is opened. The chat applet consists of basic functions like changing the font, style or color of the text. Users of the chat applet can also view other users who are using the chat applet. There are also login and signoff buttons.
Figure 11 – Initial screen for Collaboration Chat applet.

The login button has a popup dialog box where the user enters a chat handle.
Figure 12 – Pop up box to enter the user’s handle for the Collaboration Chat tool.

The names of the users that are logged in would appear on the right side, as shown in the next figure.
Figure 13 – After user logs in s/he can see other users logged into the Collaboration Chat tool.

The users that are logged into the collaboration chat tool can now talk amongst themselves. After the user is connected s/he can begin typing.
Figure 14 – Users chatting with the collaboration chat tool.

If other users are there then the new user will join in on the chat from the point s/he joined the chat session. The user can also log off by pressing the “Log Off” button. This chat tool emulates a GUI-based Internet Relay Chat (IRC) tool, rather than an instant messaging application, so there are not as many features. The reason for this is because the JSDT toolkit is very limited with what they can provide. If it were still supported today there may have been some class or method that allowed only the JSDT clients to exit without ending the whole session.

Another example of a collaboration utility is a whiteboard. It can be beneficial for users who need to work on group projects or do presentations where they would need something like a whiteboard to explain their ideas clearly. The whiteboard utility consists of the following classes, which are similar to the chat tool’s classes:

- **Controls.java**: (GUI) Creates the controls that are used for the whiteboard.
- **DrawingArea.java**: (GUI) Creates the area where the users can draw on.

- **WBConsumer.java**: Sends the data to the other Whiteboard clients.

- **WhiteBoardClient.java**: Represents the user of the whiteboard.

- **WhiteBoardDebugFlags.java**: Interface that contains boolean variables for logging purposes.

- **WhiteBoardServer.java**:STARTS the JSDT registry, if it is not already started up, and creates a WhiteBoard session.

- **WhiteBoardUser.java**: Creates the WhiteBoard application/applet which creates DrawingArea and Controls objects, also is responsible for creating the WhiteBoard client and connecting it to the WhiteBoard session.

The whiteboard server is instantiated in the CollabServerContext class.

The whiteboard applet consists of controls that allow you to select the brush style and the type of object to draw. The whiteboard can be accessed from the Pace Collaboration Server page.
Figure 15 – Whiteboard utility.

After the user clicks on the whiteboard link s/he can write to the whiteboard immediately and everyone else that is logged in will be able to see whatever the user enters in the drawing area.
Figure 16 – User using whiteboard utility.
Chapter 5

Logging

5.1 Overview of Logging

Logging is an important feature that a web server can have. The logging of server activities is essential in the maintenance and support of the applications residing on the server. “Collecting data generated by the web server provides you with critical information that is essential for analyzing the security of the web server and detecting signs of intrusion.”[6] Logging is especially beneficial when an error occurs. Depending on the detail of the error logs, the web administrator would be able to pinpoint the possible cause and, instantly, find a solution. Logs on the web server can be used to:

- Alert you to suspicious activity that requires further investigation
- Determine the extend of an intruder’s activity
- Help you to recover your systems
- Help you to conduct an investigation
- Provide information required for legal proceedings”[6]

The types of logging available to servers are configurable within the server’s administrative tools. Some servers may provide detail logging of every event that occurs on the server. These details can quickly fill up a file if there are many users that visit the web sites that are deployed by the web server. There are some servers that log only suspicious activities that may occur on the server. The following describes four different types of log files that may be available on the web server:

- “Transfer Log: Each transfer is represented as one entry showing the main information related to the transfer.
- Error Log: Each error is represented as one entry including some explanation of the reason for this error report.
• Agent Log: If this log is available, it contains information about the user client software used in accessing your web content.

• Referrer Log: If this log is available, it collects information relevant to HTTP access. This includes the URL of the page that contained the link that the user client software followed to initiate the access to your web page.”[6]

Server log files can be rotated by renaming the old log file to something else and naming the new log file to whatever the name of the log file was previously. The old log file can be archived somewhere else so that the server does not get inundated with huge log files. Saving old log files is useful since administrators can go back to view any activity that happened on a certain date, if there was some problem. Or, if there is a recurring problem, the problem can be retraced from when it first occurred by looking through the old log files. It is important that these files not be deleted because you never know when you may need to refer to them.

5.2 Logging in Pace Web Server

There are many forms of logging available in the Pace Web Server. All the logging is inserted into the PWS.log file. If an error occurs it may be difficult to find it among all the lines of informational text about what the server is doing and what was occurring, etc. To make it easier for the web administrator to find the error logs, a separate error log file (PWSError.log) was created. Any Java exceptions that were caught would be placed in this file.

The error logging feature consists of a single class, ErrorLog.java, which contains several methods to log the errors, shown below:

```java
public static void logError(String message)

public static void logError(Exception exception,
                              String message)

public static void logError(String message,
                              Throwable throwable)
```
public static void logError(Exception exception,
        String message,
        String javaFile,
        String method)

There are varying levels of logging the errors and I used the method that would provide valuable information to diagnose the error, which are the last two methods. The log information would contain the time the error occurred, method and class the error occurred in, a message string and the stack trace from the exception as follows (from PWS.java):

```
ErrorLog.logError(e,
        "Error while reading Session configuration file",
        "PWS.java",
        "restart()");
```

The error logging method is invoked within the catch statement of the Pace Web Server Java classes and is stored in the PWSError.log file.

The Pace Web Server console provides two checkboxes to enable general logging and error logging. By default, these checkboxes will be checked since logging of a web server is critical.
You need to install the Pace Web Server files from the PaceWeb.zip file. Instructions for installing the Pace Web Server can be found in the appendix.

### 6.1 Running SSL

To view the SSL feature, first make sure you are referencing a certificate in the Pace Web Server console. If you need to create a certificate please refer to the appendix “Creating a Keystore and Certificate.” Also, make sure there is a port entered for the SSL Port in the Pace Web Server console. The necessary jar files for SSL (jnet.jar, jcert.jar, jsse.jar) will be included in the classpath when you run the startserver.bat file. The actual jar files can be found in the lib directory off the Pace Web Server home directory.

After these steps are taken, you can now access web sites using SSL. To access the main Pace Web Server page, simply type the following in the web browser’s address bar:

`https://<servername>:<SSL Port>/`

You will then be told that you are about to view pages over a secure connection. Click the “Ok” button to proceed. The next box that will appear is a Security Alert. It says there is a problem with the certificate and asks if you want to proceed. Click “yes” to proceed and you will then be taken to the page you requested.

### 6.2 Running the Collaboration tools

To run the collaboration tools, you can enter a collaboration port on the Pace Web Server console. You can enter the site for the collaboration tool as follows:

`http://<servername>:<Collab Port>/collab/welcome.html`
You will be taken to the main collaboration server web page. From there you can access the chat or whiteboard utilities. If you are running these utilities under SSL mode then you may receive some dialog boxes which occur because the certificate owner does not appear in the trusted list of certificate authorities within your web browser and the certificate that was issued for the server may not match the server’s hostname. You can click “Yes” for both dialog boxes and you will be able to use the collaboration utilities.

6.3 Using the Logging Features

To enable the logging features, you should put a check next to the “Enable Logging” to log general informational messages that are generated by the Pace Web Server and/or put a check next to “Enable Error Logging” to log any error messages. To view the logs you have to go to the conf directory under the Pace Web Server root directory.
Chapter 7

Conclusion

In this thesis, I have demonstrated how adding a few features, like SSL, error logging and a collaboration chat tool, can make a server more appealing. However, more features can still be added to the server to enhance the server’s capabilities.

The collaboration server piece can be expanded to include other utilities like a collaborative editor or some type of video conferencing tool, possibly in another Java technology. The JSDT API can still be used for developing collaboration tools but it is no longer supported by Sun. The JSDT mailing forum can still be used to post questions related to JSDT or the Sun Java Forums can also provide assistance. Some other Java technology to consider for developing collaborative tools is Remote Method Invocation (RMI) which allows an object to be invoked from another object running on a different machine. The Java Collaborative Computing Group (http://research.sun.com/research/coco/) has developed a video conferencing tool in Java called Montage which could probably be integrated into the Pace Web Server.

A web-based server administration utility could be developed. There are several commercial web servers that offer this feature. For example, Apache has a web-based server administration utility with all the functions required for maintaining the server.
Figure 17 – Apache’s Tomcat Web Server Administration tool login screen.
Figure 18 – Configurable items in the Tomcat Web Server Administration Tool.

It would be easier to maintain the server because the administrator would not have to be situated at the physical location of the server. A web based administration tool will allow him/her to view or modify settings from any computer that has a web browser and is connected to a network or Internet. Also, developing or modifying an HTML page is easier than doing the same for a Java Swing based application. Although having a web based administration would be nice there can be some problems associated with it. The major concern with a web based tool is security. Authorized users should have an id and password to enter into the system and unauthorized users should not be able to access the web site or any of its web pages. The web site should be secure enough so that hackers would not be able to pry into the server and crash it.

Another useful feature to have on the Pace Web Server is a utility that tracks the web statistics. This utility should be customizable, allowing the administrator to view
statistics from different perspectives, i.e., it should be able to show the pages that were viewed and how many times, or the browsers used to view the pages, or where the users are coming from (based on their IP addresses). This would be useful in gathering essential information in the type of content users are more interested in, and developers would be able to cater to people who use the most popular browser that is recorded in the statistics. They would be able to provide more features to the most popular browser, basically. Administrators can also gather marketing information based on where the user goes to on the server, and possibly provide appropriate web sites for specific groups of people. Also, a log analysis tool can be developed to analyze the log files generated from the Pace Web Server. It would make it easier for administrators to look through the log files.

With these possible features along with the new features I contributed, the Pace Web Server will prove to be a valuable learning tool for future students to gain understanding and insight into the internal functionalities of a web server and the web technologies that it provides.
## Appendix A

### Troubleshooting SSL Errors

<table>
<thead>
<tr>
<th>ERROR DESCRIPTION</th>
<th>CAUSE</th>
<th>SOLUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>javax.net.ssl.SSLException: Unrecognized SSL handshake (occurs when trying to run the application)</td>
<td>The problem may lie in the protocol used.</td>
<td>Verify that the instance for the SSLContext is TLS.</td>
</tr>
<tr>
<td>javax.net.ssl.SSLException: Unrecognized SSL handshake (occurs when trying to access a web page)</td>
<td>The url may not have been entered correctly</td>
<td>Verify that the protocol used is https, not http.</td>
</tr>
<tr>
<td>java.io.FileNotFoundException: [keyname] (The system cannot find the file specified.)</td>
<td>The application can not find the key.</td>
<td>Verify that the key is spelled correctly in the software. Also make sure that the key is in the correct location (should be in the current working directory)</td>
</tr>
<tr>
<td>Cannot find server or DNS error</td>
<td>Server may not be running or wrong port was used</td>
<td>Make sure the Pace Web Server is running and verify the correct (SSL) port was used.</td>
</tr>
<tr>
<td>java.security.NoSuchAlgorithmException: Algorithm TLS not available</td>
<td>Provider was not found.</td>
<td>2 options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1. Enter the provider in the java.security policy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Add it in the code.</td>
</tr>
</tbody>
</table>
If problems occur during the execution of the application then try using the following option to debug the application: “-Djavax.net.debug=all”.
Appendix B

Pace Web Server User Guide

This section was taken from [16]. Any new items that were added are in \textit{bold/italic} style.

System Requirements

The web server has been written in Java which makes it platform-independent. The Java Virtual Machine must be installed to run the Pace Web Server. It can run under Java version 1.2 and higher and it has been tested using Windows and Linux. \textbf{There must be a java.security file, which is located in JAVA_HOME\jre\lib\security because the code that enables SSL accesses this file.} The console will not be available in Linux since it uses files specific to the Windows Operating System.

Installation

The entire source and compiled code for the Pace Web Server is bundled into a single zip archive, PaceWeb.zip. It contains the Java sources, configuration files, sample HTML and other CGI files, sample servlets, and necessary resource files like images. The zip file has the following structure (item in \textit{bold/italic} signify a new directory):

```
PaceWeb
  classes
  conf
  htmlRoot
    \textcolor{red}{collab}
    examples
    help
    images
```
The htmlRoot > collab directory contains html files, jar files relevant to the collaboration tools. The security directory would contain the keystore file that holds the certificate for server authentication and the src > edu > pace > web > collab and > whiteboard directories contain the source code for the chat and whiteboard collaboration utilities.

Running the Pace Web Server

The web server can be run from the command line or from within an IDE. In a production environment, it will be run from the command line. Follow the steps below.

• Unzip the archive using WinZip or other program in Windows.

• Edit the PWS.conf file found in conf sub directory and modify the following parameters.
HOST: Edit the host to the appropriate name. The default is localhost.
PORT: Edit the port to whatever port you want the web server to listen on. The default is 8989.

SSL_PORT: Edit the port to whatever port you want the web server to listen to for secure communication.

COLLAB_PORT: Edit the port to the port you want the web server to listen to for the online collaboration tools.

ROOT_PATH: Edit the Root Path to specify where the web server should look for files and directories for any client request. The default is htmlRoot. It is recommended that the default value be used.

SSL_CERT_PATH: Select the keystore file that contains the certificate that you want to use for server authentication.

SSL_CERT_PASSWORD: Enter the password for the keystore file.

- Run the web server either using the supplied script startserver.bat (in Windows). Alternately, it can also be run manually.
- To run using the supplied script, open the script startserver.bat in a text editor like Notepad. Edit the JAVA_HOME parameter to point to the location of JDK 1.3.1 or other appropriate version in your machine. I have mine in C:\jdk1.3.1. The script automatically sets the necessary CLASSPATH and PATH variables. To run the server, just type startserver.
- To manually run the web server
  Include the path to java in the PATH environment variable,
  set PATH= C:\jdk1.3.1\bin;%PATH%
  Include the current directory (C:\PaceWeb) in the CLASSPATH
  set CLASSPATH=.\classes;.\lib\servlet.jar;.\lib\jeans.jar;.\lib\jcert.jar,
  .\lib\jnet.jar;.\lib\jsse.jar;.\lib\jsdt.jar;%CLASSPATH%
  Then run the main class as below:
C:\PaceWeb> **java edu.pace.web.PWS -c .**

The same command should also work in Linux.

/home/PaceWeb$ **java edu.pace.web.PWS -c .**

- Once the web server is running, point a browser window to the following URL
  http://localhost:8989

Note: In order to run the CGI scripts written in Perl you need to download and
install Perl in your machine where the Web Server is running. Perl software for
Windows can be downloaded from
http://www.activestate.com/Products/ActivePerl

The server understands the following command line parameters:
- **-c <basepath to conf files>**

The base path tells the server where to find the various configuration files, where
to find the log file to write to if logging is enabled, the icon image file for the
server.

The configuration files are read as
- `<basepath>\conf\PWS.conf`
- `<basepath>\conf\Servlet.conf` and
- `<basepath>\conf\Session.conf`

The log file where it writes trace, diagnostic, and error messages will be
- `<basepath>\log\PWS.log`

The icon image file for the various windows will be
- `<basepath>\conf\leaf.gif`

For example, if the server is started as below
C:\PaceWeb> **java edu.pace.web.PWS –c C:\my\folder**
then it will look for the PWS.conf configuration file in the directory C:\my\folder\conf. The complete file name will then be C:\my\folder\conf\PWS.conf. Similarly for other files. Full path to Servlet.conf will be C:\my\folder\conf\Servlet.conf and Full path to Session.conf will be C:\my\folder\conf\Session.conf Full path to log file will be C:\my\folder\log\PWS.log Similarly it applies to Linux with the appropriate file separators ("/" is the file separator in Linux as compared to "\" in Windows)

Note: Although I have listed some directory names as I used it, the code can be unzipped and run from anywhere in the file system.

**Command line compiling and invocation**

**Windows**

Windows XP was used for compiling and running the Pace Web Server but the code should work identically in Windows NT or Windows 2000 Professional versions though I have not tested in those versions. For the steps below, I assume that I have uncompressed the zip archive to C:\ drive though the files can be uncompressed to any location. The files will be created in PaceWeb directory and its subdirectories as indicated in the previous section.

The following steps are required to compile and run the server.

1. Unzip the PaceWeb zip archive using WinZip utility. Use C:\ as the destination folder to extract the contents to and check the “Use Folder Names” option and “All Files” option.
2. Open a command window (Start > Programs > Accessories > Command Prompt or Start > Run and then type cmd.exe in the dialog and click ok)
3. Change directory to C:\PaceWeb
4. To run the web server, run the script startserver.bat as
C:\PaceWeb> **startserver**

Note 1: All the scripts assume that the Java JDK is available in C:\jdk1.3.1 directory. If it is installed in a different directory then change the setting for JAVA_HOME inside the batch files.

Note 2: By default the root directory from which the server sends html pages is set to htmlRoot sub-directory from the base directory. If the HTML root directory is moved elsewhere or if it needs to be pointed to some other directory it has to be changed in the PWS.conf file.

In addition, to compile all the sources at once, run the script makeall.bat as below:
C:\PaceWeb> **makeall**

To set the environment only for compiling one or more files individually, run the script envset.bat as below
C:\PaceWeb> **envset**

Once the environment is set, any file can be compiled by specifying the full path to the source java file as below (all parameters must be in the same line):
C:\PaceWeb> **javac** **-verbose** **-d classes** **-sourcepath src**
src/edu/pace/web/servlets/HelloWorld.java

Note: The –verbose option prints detailed information as the compiler builds the file. It is optional and need not be specified.[16]

**Linux [Redhat 7.1]**

The compilation and execution of the Pace Web Server has been tested in Redhat Linux 7.1. However, the same shell scripts should work in any version of Linux/Unix as long as the Java compiler and JVM are available. The scripts in Linux are generic and written using the basic shell commands that must be available in any version of Linux or Unix.
The compilation and execution steps are very similar to Windows except that the path and directory information is different. For the steps below, the home directory, /home/username, to uncompress and run the server though the files can be uncompressed to any location. Obviously, in the steps below substitute username with the your actual login name.

1. Open a terminal window. In KDE it is the Konsole window. In GNOME, it is simply the gnome-terminal.

2. Change directory to the user home directory, if not already there, by simply running the cd command without any parameters.

3. Unzip the PaceWeb zip archive using *unzip* or *gunzip* utilities.

4. Change directory to /home/username/PaceWeb

5. Enable execute permissions on startserver.sh and makeall.sh scripts (type the entire command in the same line) if not already available
   
   `/home/username/PaceWeb $ chmod u+x startserver.sh makeall.sh`

6. To run the web server, run the script startserver.sh as below
   
   `/home/username/PaceWeb $ startserver.sh`

**Note 1:** All the scripts assume that the Java JDK is available in /usr/local/jdk1.3.1 directory. If it is installed in a different directory then change the setting for JAVA_HOME inside the batch files.

**Note 2:** By default the root directory from which the server sends html pages is set to htmlRoot sub-directory. If the HTML root directory is moved elsewhere or if it needs to be pointed to some other directory it has to be changed in the PWS.conf file.

In addition, to compile all the sources at once, run the script makeall.bat as below:

`/home/username/PaceWeb $ makeall`
To set the environment only for compiling one or more files individually, source, do not run, the script envset.bat as below:

```
/home/username/PaceWeb $ . envset
```

Note: “Sourcing” a shell script merely sets the environment variables and other settings in the current shell whereas executing a script will run the script in a child shell forked from the parent or current shell.

Once the environment is set, any file can be compiled by specifying the full path to the source Java file as below (all parameters must be in the same line):

```
/home/username/PaceWeb $ javac -verbose -d classes -sourcepath src src/edu/pace/web/servlets/HelloWorld.java
```

Note: The –verbose option prints detailed information as the compiler builds the file. It is optional and need not be specified.

Refer to the appendix for a listing of the scripts.

**Using an IDE to run Pace Web Server**

*Borland JBuilder 9*

Borland’s JBuilder is a good IDE for Java, web and J2EE development similar to JDeveloper. I tried out JBuilder 7 to make sure my code can be built using any IDE. I used JBuilder 7 Personal Edition for my testing purposes. Refer to Borland JBuilder webpage *Error! Reference source not found.* for more information on IDE and tutorials.

Again with JBuilder, you need to create a Project and add all the source files. JBuilder also provides a lot of wizards to create, run, test, and debug code. It also has an integrated debugger by which you can set breakpoints, view the variable and instance values during execution and so on.
I have included the JBuilder project files as a part of the PaceWeb.zip archive. Within JBuilder the project can be opened simply using the steps below:

1. Click File > Open Project
2. Navigate to C:\PaceWeb directory in the File Open dialog
3. Select PaceWeb.jpx file and click OK.

Now the project is ready for compilation and running.

To compile all the files in the Project

Right Click on PaceWeb.jpx and use the context menu to select “Rebuild”.

It should take a few seconds to a minute to compile all the files depending on the processing power of the computer.

To run the pace web server,

Go to the menu Run > Run Project

The Pace Web Server should be up and running.

Alternately, a new project can be created in JBuilder and the existing source files can be added to create the project. The following figures show the project settings required to build and run the pace web server code. The command line parameters “-c .” are optional.

Note: In order to compile and run properly the following two options must be set correctly.

(a) In the required libraries section, include the jeans178a.zip, servlet.jar, jcert.jar, jnet.jar, jsse.jar, and jsdt.jar files as shown in the project settings.

(b) In the Run options create a new Runtime Configuration (I have named it PaceWeb) as shown in the settings above.
In addition to the PaceWeb.jpx project file, JBuilder creates the following files during compilation and build. These are necessary only for JBuilder and are not required while building and running from the command line. I have indicated JBuilder created directories in bold and JBuilder created files underlined to differentiate from regular files and directories. This was taken from [16]

PaceWeb

**Bak**

classes

**package cache**

PaceWeb.jpx.local

PWSlibraries.library
Appendix C

Creating a Keystore and Certificate

Generate key and keystore

To generate a keystore you can use the –genkey option and specify a new keystore with the –keystore option as follows:

```
C:\PaceWeb>keytool -genkey -alias pws -keystore paceks -keyalg rsa
```

Enter keystore password: pace123

What is your first and last name?
[Unknown]: Pace Web Server

What is the name of your organizational unit?
[Unknown]: IT

What is the name of your organization?
[Unknown]: Pace University

What is the name of your City or Locality?
[Unknown]: White Plains

What is the name of your State or Province?
[Unknown]: NY

What is the two-letter country code for this unit?
[Unknown]: US

Is CN=Pace Web Server, OU=IT, O=Pace University, L=White Plains, ST=NY, C=US correct?
[no]: y

Enter key password for <pws>

(RETURN if same as keystore password): <Enter>

Make sure the –keyalg option is specified with a value of “RSA”. If this is not specified then the certificate will not work with Internet Explorer and Netscape web browsers. The certificate will last for 90 days by default. Use the –validity option to specify the validity period (in days) for the certificate. After this command has been successfully executed, the keystore file will be created in the directory in which the command was executed.

Using the –genkey option will generate a private key and a self-signed certificate with the public key placed within the certificate. After these steps are taken, the keystore file can be
selected on the Pace Web Server console and within the keystore will the certificate will be used for authenticating the server.

**Generate Self-Signed Certificate**

It is not necessary to follow this procedure since the –genkey option already creates a self-signed certificate. If you want to create another key, you will have to delete any existing key(s) to use the new one. If you need to change the DN entry of a key, in case your organization name changes, for example, then you can clone your existing key and create a self-signed certificate as follows:

```
C:\PaceWeb>keytool -keyclone -alias newkey -keystore newstore
Enter destination alias name: newPaceKey
Enter keystore password: newstore
Enter key password for <newPaceKey>
    (RETURN if same as for <newkey>) <ENTER>
```

Executing the previous command will generate a clone ("newPaceKey") of the existing key “newkey”. To generate a self-signed certificate execute the following command:

```
C:\PaceWeb>keytool -selfcert -alias newPaceKey -keystore newstore -dname "CN=Pace Web, OU=IT, O=Pace University, L=White Plains, S=NY, C=US" -keyalg rsa
Enter keystore password: newstore
```

A new self-signed certificate will be created with the new DN information. The self-signed certificate will be stored in the keystore specified. To list what is in a keystore, the following can be done:

```
C:\PaceWeb>keytool -list -keystore newstore
Enter keystore password: newstore
```

Keystore type: jks
Keystore provider: SUN
Your keystore contains 2 entries

newkey, Nov 30, 2003, keyEntry,
Certificate fingerprint (MD5):

newpacekey, Nov 30, 2003, keyEntry,
Certificate fingerprint (MD5):

The original key will be listed along with the cloned key that was just created. To use the cloned key, you have to delete the original key as follows:

C:\PaceWeb>keytool -delete -alias newkey -keystore newstore
Enter keystore password: newstore

The Pace Web Server would have to be restarted for the changes to take affect.
Appendix  D

Server Configuration Files

This section was taken from [16]. Any new items that were added are in bold/italic style.

PWS.conf

Session_Configuration_File = Session.conf
ChildThreadPriority = 4
**COLLAB\_PORT = 9999**
Directory_Indexing = true
**LOG\_ENABLED = true**
**ERROR\_LOG\_ENABLED = true**
serverName = PWS
Servlet_Configuration_File = Servlet.conf
port = 8989
ROOT\_PATH = htmlRoot
Default\_Filename = index.html
Default\_Filename = index.htm
Log = true
**SSL\_PORT = 7777**
MAX\_NUMBER\_OF\_CONNECTIONS = 1000
serverVersion = v1.5
serverUrl = http://localhost
RESERVED\_NUMBER\_OF\_CONNECTIONS = 2
HOST = localhost
MainThreadPriority = 9
**SSL\_CERT\_PATH = C:\ PaceWeb\ security\ paceks**
**SSL\_CERT\_PASSWORD = pace123**
ENABLE\_CGI = true

Servlet.conf

servlet.HelloWorld.code = edu.pace.web.servlets.HelloWorld
servlet.HelloWorld.urlpattern = HelloWorld|servlet/HelloWorld

servlet.Counter.initarg = pone=foo
servlet.Counter.initarg = ptwo=bar
servlet.Counter.initarg = pthree=foobar
servlet.Counter.initarg = pfour=barfoo
servlet.Counter.code = edu.pace.web.servlets.Counter
servlet.Counter.urlpattern = servlet/Counter
servlet.DateServlet.code = edu.pace.web.servlets.DateServlet
servlet.DateServlet.urlpattern = DateServlet
servlet.DateServlet.initarg = name=Priya
servlet.DateServlet.initarg = degree=MS
servlet.ParamServlet.code = edu.pace.web.servlets.ParamServlet
servlet.NewDirServlet.code = edu.pace.web.newdir.NewDirServlet
servlet.SessExample.code = edu.pace.web.servlets.SessExample
servlet.ExtDirServlet.code = ExtDirServlet
servlet = HelloWorld
servlet = Counter
servlet = NewDirServlet
servlet = ParamServlet
servlet = ExtDirServlet
servlet = DateServlet
servlet = SessExample

Session.conf

session.useCookies = true
session.newtimeout = 1800
session.checkFrequency = 5
Appendix E

Compilation and Execution Scripts

This section was taken from [16]. Any new items that were added as a result of this thesis are in *bold/italic*.

makeall.bat – Compilation batch file for Windows

@echo off
REM ============================================================
REM Pace Web Server Build Script
REM Author: Priya Srinivasaraghavan
REM ============================================================

IF "%OS\%"="Windows_NT" SETLOCAL

SET JAVA_HOME=C:\JDK1.3.1

IF NOT EXIST %JAVA_HOME%\bin\javac.exe GOTO :nojava

SET PACEWEB_BASE=.
SET
CLASSPATH=%PACEWEB_BASE%\classes;%PACEWEB_BASE%\lib\servlet.jar;%PACEWEB_BASE%\lib\jeans178a.zip;%PACEWEB_BASE%\lib\jnet.jar;%PACEWEB_BASE%\lib\jsdt.jar;%PACEWEB_BASE%\lib\jsse.jar;%PACEWEB_BASE%\lib\jcert.jar;%CLASSPATH%

SET PATH=%JAVA_HOME%\bin;%PATH%

echo Classpath is %CLASSPATH%
echo.
echo Path is %PATH%
echo.

javac -d classes -sourcepath src src/edu/pace/web/PWS.java
javac -d classes -sourcepath src src/edu/pace/web/servlets/HelloWorld.java
javac -d classes -sourcepath src src/edu/pace/web/servlets/DateServlet.java
javac -d classes -sourcepath src src/edu/pace/web/servlets/Counter.java
javac -d classes -sourcepath src
src/edu/pace/web/servlets/Dummy.java
javac -d classes -sourcepath src
src/edu/pace/web/servlets/GBTextServlet.java
javac -d classes -sourcepath src
src/edu/pace/web/servlets/GuestBook.java
javac -d classes -sourcepath src
src/edu/pace/web/servlets/ParamServlet.java
javac -d classes -sourcepath src
src/edu/pace/web/servlets/SessExample.java
javac -d classes -sourcepath src
src/edu/pace/web/servlets/ViewGBText.java
javac -d classes -sourcepath src
src/edu/pace/web/servlets/ViewGuestBook.java

echo.
echo Compilation Completed
echo.

goto end
:nojava
echo Unable to find javac.exe in the Java Home directory
[%JAVA_HOME%] specified
echo %JAVA_HOME%in\javac.exe : No such file exists
echo exiting...
:end
IF "%OS%"="Windows_NT" ENDLOCAL

makeall.sh – Compilation shell script for Linux
#!/bin/sh
# ==============================================================
# Pace Web Server Build Script
# Author: Priya Srinivasaraghavan
# ==============================================================
JAVA_HOME=/usr/local/jdk1.3.1_04
export JAVA_HOME

if [ ! -f $JAVA_HOME/bin/javac ]; then
  echo "Unable to find javac in Java Home directory ( $JAVA_HOME
) specified."
  echo "$JAVA_HOME/bin/javac : No such file exists"
  echo "Exiting..."
  exit
fi

PACEWEB_BASE=.
export PACEWEB_BASE
export CLASSPATH

echo Classpath is $CLASSPATH
echo Path is $PATH
echo 

echo "java version is"
$JAVA_HOME/bin/java -version
echo

javac -d classes -sourcepath src src/edu/pace/web/PWS.java
javac -d classes -sourcepath src src/edu/pace/web/servlets/HelloWorld.java
javac -d classes -sourcepath src src/edu/pace/web/servlets/DateServlet.java
javac -d classes -sourcepath src src/edu/pace/web/servlets/Counter.java
javac -d classes -sourcepath src src/edu/pace/web/servlets/Dummy.java
javac -d classes -sourcepath src src/edu/pace/web/servlets/GBTextServlet.java
javac -d classes -sourcepath src src/edu/pace/web/servlets/GuestBook.java
javac -d classes -sourcepath src src/edu/pace/web/servlets/ParamServlet.java
javac -d classes -sourcepath src src/edu/pace/web/servlets/SessExample.java
javac -d classes -sourcepath src src/edu/pace/web/servlets/ViewGBText.java
javac -d classes -sourcepath src src/edu/pace/web/servlets/ViewGuestBook.java

echo
echo Compilation completed
echo

startserver.bat – Execution batch file for Windows

@echo off
REM ===================================================================
REM Pace Web Server Execution Script
REM Author: Priya Srinivasaraghavan
REM ===================================================================
IF "%OS%"=="Windows_NT" SETLOCAL

SET JAVA_HOME=C:\JDK1.3.1
rem SET JAVA_HOME=C:\JBuilder9\jdk1.4

IF NOT Exist %JAVA_HOME%\bin\java.exe GOTO :nojava

SET PACEWEB_BASE=.
SET CLASSPATH=%PACEWEB_BASE%\classes;%PACEWEB_BASE%\lib\servlet.jar;%PACEWEB_BASE%\lib\jeans178a.zip;%PACEWEB_BASE%\ExtDir;%PACEWEB_BASE%\lib\jnet.jar;%PACEWEB_BASE%\lib\jsdt.jar;%PACEWEB_BASE%\lib\jss.jar;%PACEWEB_BASE%\lib\jcert.jar;%CLASSPATH%

SET PATH=%JAVA_HOME%\bin;%PATH%

echo Classpath is %CLASSPATH%
echo.
echo Path is %PATH%
echo.

tITLE Pace Web Server
java -hotspot edu.pace.web.PWS -c .
goto end

:nojava
echo Unable to find java.exe in the Java Home directory [%JAVA_HOME%] specified
echo %JAVA_HOME%\bin\java.exe : No such file exists
echo exiting...

:end
IF "%OS%"="Windows_NT" ENDLOCAL

startserver.sh – Execution shell script for Linux

#!/bin/sh
#
# ==============================================================
# Pace Web Server Execution Script
# Author: Priya Srinivasaraghavan
# ==============================================================

JAVA_HOME=/usr/local/jdk1.3.1_04
export JAVA_HOME

if [ ! -f $JAVA_HOME/bin/java ]; then
  echo "Unable to find java in Java Home directory ( $JAVA_HOME ) specified."
  echo "$JAVA_HOME/bin/java : No such file exists"
  echo "Exiting..."
  exit
fi
PACEWEB_BASE=
export PACEWEB_BASE

export CLASSPATH

echo Classpath is $CLASSPATH
echo
echo Path is $PATH
echo
echo "java version is"
$JAVA_HOME/bin/java -version
echo

$JAVA_HOME/bin/java edu.pace.web.PWS -c .

echo

envset.bat – Environment setup batch file for Windows

@echo off
REM ============================================================
REM Pace Web Server Environment Setup Script
REM Author: Priya Srinivasaraghavan
REM ============================================================

SET JAVA_HOME=C:\JDK1.3.1
IF NOT EXIST %JAVA_HOME%\bin\javac.exe GOTO :nojava

SET PACEWEB_BASE=.
SET CLASSPATH=%PACEWEB_BASE%\classes;%PACEWEB_BASE%\lib\servlet.jar;%PACEWEB_BASE%\lib\jeans178a.zip;%PACEWEB_BASE%\lib\jnet.jar;%PACEWEB_BASE%\lib\jsdt.jar;%PACEWEB_BASE%\lib\jsse.jar;%PACEWEB_BASE%\lib\jcert.jar;%CLASSPATH%

SET PATH=%JAVA_HOME%\bin;%PATH%

echo Classpath is %CLASSPATH%
echo.
echo Path is %PATH%
echo.
goto end

:nojava
echo Unable to find java.exe in the Java Home directory [%JAVA_HOME%] specified
echo %JAVA_HOME%\bin\java.exe : No such file exists
echo JAVA_HOME setting may be incorrect
echo exiting...
:end
echo.

envset.sh – Environment setup shell script for Linux

# ==============================================================
# Pace Web Server Environment Setup Script
# Author: Priya Srinivasaraghavan
# ==============================================================

JAVA_HOME=/usr/local/jdk1.3.1_04
export JAVA_HOME

if [ ! -f $JAVA_HOME/bin/java ]; then
    echo "Unable to find java in Java Home directory ( $JAVA_HOME ) specified."
    echo "$JAVA_HOME/bin/java : No such file exists"
    echo "JAVA_HOME setting may be incorrect"
    echo "Exiting..."
    exit
fi

PACEWEB_BASE=.
export PACEWEB_BASE

er.jar:$CLASSPATH
export CLASSPATH

PATH=$JAVA_HOME/bin:$PATH

echo Classpath is $CLASSPATH
echo echo Path is $PATH
echo "java version is"
$JAVA_HOME/bin/java -version

echo
References


