Computer Science

Per-Anders Johansson

Magnus Nilsson

Evaluation of Web Application Servers

Bachelor's Project

2000-10

Evaluation of Web Application Servers

Per-Anders Johansson

Magnus Nilsson

This report is submitted in partial fulfillment of the requirements for the Bachelor’s degree in Computer Science. All material in this report which is not my own work has been identified and no material is included for which a degree has previously been conferred.

Per-Anders Johansson

Magnus Nilsson

Approved, 2000-05-31

Advisor: Niklas Nikitin

Examiner: Stefan Lindskog

Abstract

A web/application server is used to deliver statically or dynamically built web content to clients on the World Wide Web. This document describes testing of a number of web/application servers and the criteria with which you can evaluate them. The purpose is to create guidelines that can be used by Ericsson Infotech to evaluate web/application servers.

Credits

We would like to recognize our appreciation to the following people:
Anders Berg, our instructor at Ericsson Infotech: For everything.
Niklas Nikitin, our supervisor at Karlstad University: For guiding us through this project.
Helena Lindskog: For reading our work and for contributing with good ideas on how to evolve it.
Mikael Johansson: For reading our work and for contributing with good ideas on how to evolve it.
Richard Hellberg: Our coach at Ericsson Infotech.

Contents

List of Figures

List of tables

1 Introduction

This document is the result of a bachelor’s project at Ericsson Infotech in Karlstad. Our work began at the end of January 2000 and went on until May 2000. The purpose of our study is to clarify the strengths and weaknesses of different web/application servers, and thus find out if one is better than the others. In this work we have used a number of evaluation criterions and in order to understand the purpose of each evaluation criteria we have chosen to describe them one by one.

The reason why we chose this bachelor’s project is that we wanted to know more about how web/application servers work and why they are used on the Internet. We also felt that Ericsson Infotech provides a good environment to work in and that our project would benefit from the knowledge available there.

2 Terminology

CGI	Common Gateway Interface
CORBA	Common Object Request Broker Architecture
EIN	Ericsson Infotech
EJB	Enterprise Java Beans
FAQ	Frequently Asked Questions
GUI	Graphical User Interface
HTML	Hyper Text Markup Language
IIS	Internet Information Server
JDK	Java Development Kit
JSDK	Java Servlet Development Kit
JSP	Java Server Pages
JVM	Java Virtual Machine
OMG	Object Management Group
SSL	Secure Sockets Layer
XML	Extensible Markup Language

3 Background

In this chapter, we will describe the background to our Bachelor’s project. It also includes some history about web/application servers, such as how they got an important place on the Internet and why they are needed. We will also describe the main problem and the purpose of our Bachelor’s project.

3.1 History

Our task in this Bachelor's project was to evaluate different web/application servers. To give a short description on the area of web/application servers we have tried to put together a short resume on the subject.

Ever since the day that Tim Berners-Lee built the first web server at CERN[1] [1], descendants to this early version have been delivering static web-content to whom ever connected to the web server. The desire to make web sites more application-like demanded that web servers could display web pages with content that was dynamically built based on output from server-side applications. An example of server-side applications is a calendar application, allowing the site user to schedule his or her week.

The entity that provides the application behavior is often viewed as the middle-tier in a three-tier application [2] as can be viewed in Figure 3.1. The first tier, the front-end, is often a web based user interface provided by a web server. Usually some kind of terminal[2] will receive and show the interface to a user but for simplicity we will leave this part out and just view the first three tiers. The second tier, the middle-tier, is the business logic that provides the ability to get things done in the application. This part of the application often executes on an application server.

In many enterprise-based applications a storage facility is required. This is often a database of some sort and it belongs to the third-tier, the back-end of the application.

[1] European Centre For Nuclear Research
[2] The terminal could be a standard PC with a web browser or a cellular phone supporting the WAP-technology.

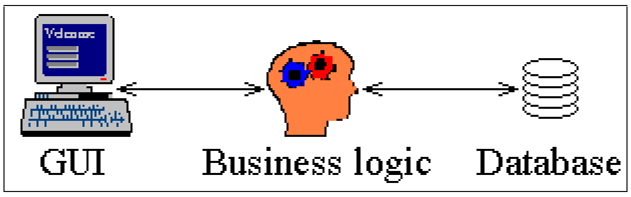

Figure 3.1: Three-tier application overview.

In some cases the application server is integrated into the web server but it might as well be a standalone server. If the application server is integrated into a web server we will see both servers as one entity and call it a web/application server. A web/application server has one major advantage over a standalone application server; namely it can provide both a web based user interface and the business logic in one entity.

In order to deploy applications that can be executed on a web/application server, developers needs a platform. Several platforms or technologies are available and we will give a short description on how some of the technologies work in chapter 5. Some of the technologies described are based on the Java technology e.g. Servlets and JSP, and some are based on separate technologies e.g. CGI.

3.2 Task

In this bachelors project we will try to answer the following question: Is there a web/application server that is better than other web/application servers?

3.3 Purpose

When EIN needs a web/application server they can't say if one server is better than the other. There are also a lot of web/application servers on the market, which makes it even harder to choose one in specific. Each web/application server has it's own special trademark and characteristics.

Our task is to find out if there exists a web/application server that can meet all the requirements that EIN want from such a server.

4 Method

In the beginning of this bachelor's project, our knowledge in this field was inadequate and some studies were needed to understand the given problem. Our studies contained both literature studies and information seeking on the Internet. More information about our studies is available in chapter 5.1. The first month of this project was used to raise the level of our knowledge. As our understanding of web/application servers grew we began to look at web/application servers available on the market. A discussion on how we narrowed the number of web/application servers down to the five we have decided to use can be viewed in chapter 4.1.

Once the contenders were picked we concentrated our work on the evaluation criterions, trying to find what was interesting to compare web/application servers with. Our intention was to produce a list of evaluation criterions that could be used to simplify the evaluation procedure. For further details on how our web/application server evaluation criterion checklist emerged, view chapter 4.2.

4.1 Choosing the web/application servers

There are two things you have to consider when you should compare a group of web/application servers.

- The scope, how many web/application servers should be included?
- Which web/application servers are of interest?

When we decided how many web/application servers to evaluate we tried to balance two opposite factors. First the number of web/application servers that EIN were interested in and then the amount of time available for this project. Considering the given situation we decided that five contending web/application servers would be appropriate. In the remaining part of chapter 4.1 we will discuss how we selected the five contenders.

EIN had special interest in four web/application servers and that left one place open in our survey. Since the thoughts behind EIN´s interest in some of the contending servers[3] are unknown to us we will focus this discussion on how we made our choice. At first we knew little about different servers on the market so we began our search on a web site [4] that has spe-

[3] Apache + JServ, Apache + Tomcat, Sybase Enterprise Application Server and IBM WebSphere.

cialized on evaluating different server software e.g. web/application servers. In their opinion BEA WebLogic Server is a five star[4] web/application server and one of the best web/application servers they have tested. With this in mind we thought that BEA WebLogic Server would be a worthy opponent to the web/application servers chosen by EIN.

4.2 Choosing evaluation criterions

This part of chapter 4 will discuss the thoughts behind our evaluation criterions. When we first received the specifications for this project made by EIN, some evaluation criterions were already mentioned. Table 4.1 describes the criterions used in this survey, the units in which they are measured and their origin.

Evaluation criterion	Unit of measurement	Origin
Performance	Milliseconds (ms)	EIN
Technologies	Qualitative	EIN
Price	USD / Entity	EIN
Support	Qualitative	EIN
Tools	Qualitative	The authors

Table 4.1: Evaluation criterions

As can be seen in Table 4.1 several criterions are measured in qualitative form. This means that it is hard to measure the criterion on any given scale. To be able to give some kind of answer to the qualitative criterions we will try to conduct a discussion on the outcome of the evaluation.

4.2.1 The performance criterion

This criterion is an outcome of EIN's desire to obtain information on how fast (in milliseconds) a web/application server can respond to a certain request. We will decide this by letting a client invoke three different tasks on a web/application server. We have decided to use a test suit containing three different applications and we will describe them in Table 4.2 and a complete view of the source code can be found in appendix B. When building our Servlet applications we have used Sun JDK-1.2.2-001[5].

[4] The scale reaches from one to five, one being the lowest and five the highest grade.
[5] This information can be obtained by executing "java -fullversion" in the command prompt in Windows NT 4.0.

Name	Purpose
HelloWorld.html	Test the response time for a simple html request.
HelloWorld.class	Test the response time for a simple Java Servlet displaying "Hel-loWorld" on the calling client.
StringManip.class	Test the response time for a Java Servlet that concatenates string A on to string B until string B is 20000 characters long. From the beginning B is empty and A con-tains 29 characters.

Table 4.2: Evaluation test suit

When testing the performance of our web/application servers we used the freeware program Apache JMeter[6]. Apache JMeter can be used to test overall performance of a web/application server. For example: Apache JMeter can simulate heavy load on a server or network in order to evaluate the performance. The output from the program can be presented in a various number of graphical diagrams or you can simply write the output-data to a file.

The performance test will be done in the following way. Apache JMeter will simulate one thousand (1000) requests to each server and we will then calculate the highest response time, the average response time, the median response time and the lowest response time for each web/application server. The thousand (1000) requests are done by ten (10) different threads[7] and each tread will wait three hundred (300) milliseconds between each call. As Table 4.1 implies we will measure the response time in milliseconds (ms).

4.2.2 The technologies criterion

This criterion will be used to evaluate which technologies the web/application server's support. EIN needs this information to decide if a web/application server could be used in their projects. We have decided to limit the number of technologies in this survey and we will only

[6] Apache JMeter can be downloaded at http://java.apache.org/jmeter/dist/
[7] Approximately 100 requests by each thread

cover the ones EIN showed interest in. For further details about the technologies view chapter 5.

4.2.3 The price criterion

In order for EIN to be able to include a web/application server in a project the price of the web/application server is essential since it will affect the price of the complete product.

When looking at the price criterion we will simply check how much the web/application server will cost. We will only consider the price for the version used in this survey. As a unit of measurement we will use the currency USD (US Dollar). If we have failed to obtain the price in USD and instead found a price in SEK (Swedish krona) we will transform SEK to USD using an exchange rate of 9:1. Usually we will obtain the price information from the manufacturer homepage or equal.

4.2.4 The support criterion

When evaluating the support criterion we will look at what kind of support the manufacturer will supply with their product. We will check whether upgrades are included in the package and if so, for how long time. Maybe you will have to pay extra for any upgrade. We will also investigate if there are any support number you can call or a mail address where you can get help. To proceed with this work we have chosen to use five questions:

- How long after purchase can the customer get free support?
- In what ways can the customer receive support? (Fax, phone, e-mail)
- At what hours is the support available?
- Are upgrades free?
- For how long are upgrades free?

Our methods in this work are fairly arbitrary. We will try to illustrate the support available for each web/application server by conducting a discussion based on the questions mentioned above. We have obtained the support information through direct contact with representatives from the manufacturers.

4.2.5 The tools criterion

The list of criterions produced by EIN seems to cover all of the interesting parts of the evaluation work. We felt however that something was missing. As with all new areas of computer science web/application servers tend to require a certain amount of knowledge in order to profit from all built-in features. To make it easier to understand and administrate, some of the web/application servers on the market are equipped with easy to use administration tools.

We will use this criterion to evaluate if this is the case with any of the web/application servers in our survey.

5 Background Information

In this chapter we will try to give some background information concerning web/application servers. We will discuss the source of our information and a short description of our lab environment.

5.1 Literature

When we first began this work we didn't know much about the different technologies available to deploy web applications on the Internet. Even though our main concern in this bachelor's project was to evaluate which technologies the different web/application servers support, we thought that a small introduction to the different technologies would come in handy. To make this introduction we have used several books available at EIN [3][5] and some resources on the Internet [2].

5.2 Lab environment

As you can see in Figure 5.1, our lab environment consists of one server and one client. The client (3) is a standard PC running Microsoft Windows NT 4.0, with 64 MB RAM and a Pentium processor working at 133 MHz. The server (1) is a PC running Microsoft Windows NT 4.0, with 384 MB RAM and a Pentium II processor working at 266 MHz. The computers are connected via a switched (2) network, running at 10 Mbit with half-duplex transfer rate.

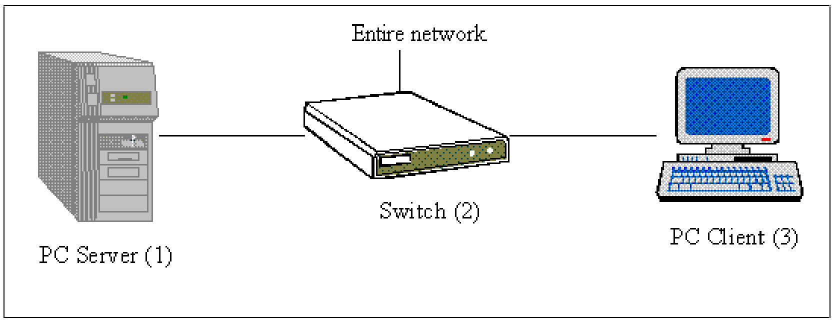

Figure 5.1: Overview of our lab environment

If another platform is desired the server (1) could easily be replaced with another system e.g. Solaris. As mentioned in chapter 4.2.1 we will use the Java based Apache JMeter as our test tool. This makes it possible to run the performance test from any Java supporting platform and we can easily replace our workstation (3) to fit our platform needs.

5.3 Technologies

This part of chapter 5 will discuss the different technologies mentioned in this evaluation. Even though several pages or even books could be written about some of the technologies mentioned below, our intention is to give a brief information about the different topics that follows. Figure 5.2 shows the relationship between the technologies mentioned later in this chapter. CORBA and EJB can to some extent be used in several of these areas, but for simplicity we will leave them out of this figure.

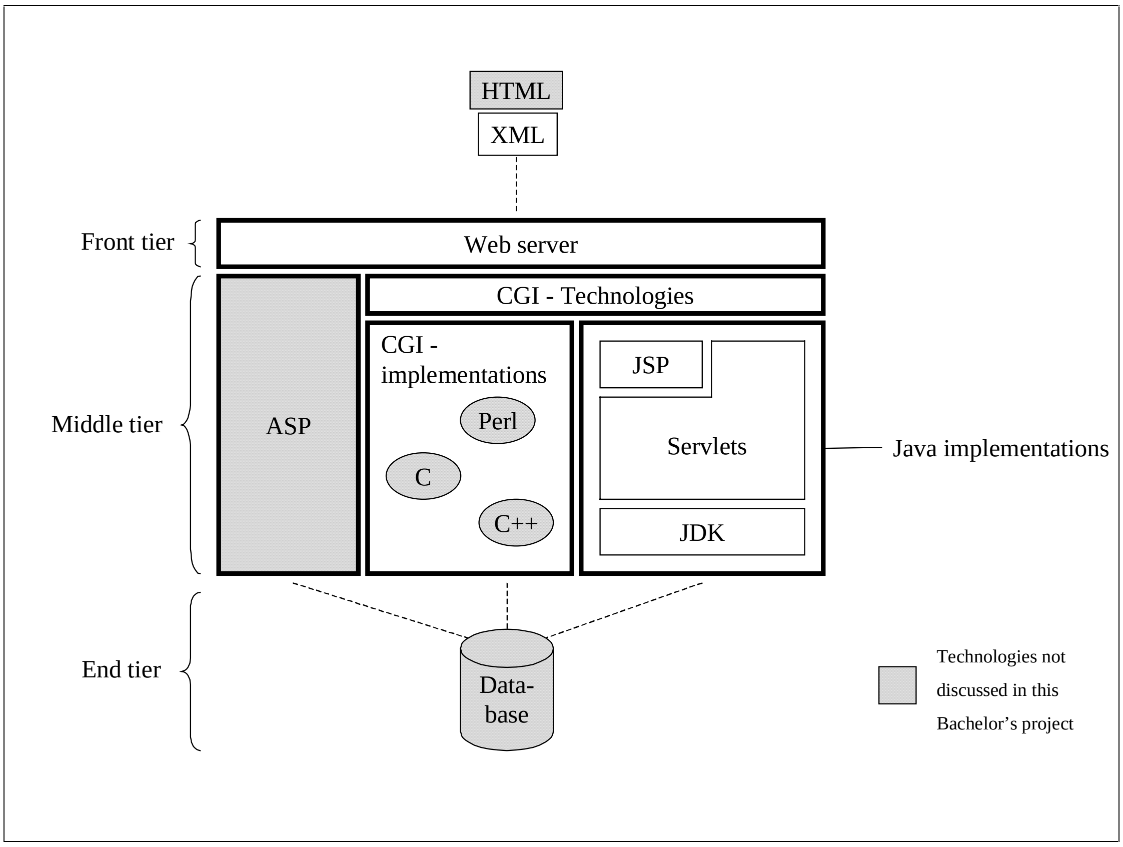

Figure 5.2: Relationship between technologies.

5.3.1 CGI-Technologies

GET and POST are two ways of delivering data from a web server to an application[5]. Several technologies use GET or POST or equivalent methods to obtain the transfer between web server and application. In this document we will use the term CGI-Technologies to describe this relationship.

5.3.2 CGI-implementations

CGI was designed to enable a web server to communicate and execute applications outside of the web server. A side effect of this is that a web server can receive parameters through an HTML form, pass them to a CGI application, which returns the result to the server. The web server then sends the result as an HTML document back to the client who can use a browser to view it. A web server can use two different methods to transfer data to a CGI application, the GET method and the POST method [5].

Because of the way that CGI works, it requires time and a lot of server resources [2], making it less desirable as a development platform for web applications. To overcome the weaknesses of CGI new techniques have been developed, techniques that are less resource consuming. A CGI application can be based on different programming languages such as C, C++ or PERL.

5.3.3 CORBA

CORBA was developed by OMG; it is an architecture for creating, distributing and managing distributed program objects in a network. This means that different programs can communicate between each other over a network via an “interface broker”. CORBA allows applications to communicate with one another no matter where they are located or who has designed them.

The essential component in CORBA is the Object Request Broker (ORB). The (ORB) is the middleware that establishes the client-server relationship between objects. Using an ORB, a client can transparently invoke a method on a server object, which can be on the same machine or across a network. The client does not have to know where the object is located, its programming language, its operating system, or any other system aspects that are not part of an object's interface. Doing so, the ORB provides interoperability between applications on different machines in distributed environments.

5.3.4 Enterprise Java Beans

Enterprise Java Beans are reusable components written in Java, for Java application development. They run in the server-side of a client server network relationship. An example of this could be a button that appears on a web page. Beans are "capsules" of code, designed for a specific purpose. The advantage of Enterprise Java Beans over other standard components is that Beans are independent. They are not specific to operating systems or development environments. If you create a bean in one development environment, you can easily copy and modify it in an another. This allows Java Beans to be more flexible in enterprise computing, as components are easily shared between developers.

So instead of upgrading a lot of clients when a component is changed or added, you only have to update the server instead of all clients. And this way you can distribute program components to clients in a network.

5.3.5 JDK

A JDK contains all you need to develop, compile and run Java applications [9]. As with all software, newer versions are frequently released and the latest version available at the current time is 1.2.2.

5.3.6 JSP

JSP, or Java Server Pages, is a technology invented by Sun Microsystems. With Java Server Pages you can create and maintain server-side HTML pages. These pages can be used both as a kind of dynamic HTML and a replacement for CGI. Java Server Pages mixes static HTML with dynamically generated HTML. The good side to this is that it’s more convenient to write and/or modify regular HTML code than to have a lot of *println* statements that generates HTML, which is the case in Servlets. You can compare Java Server Pages with Microsoft’s Active Server Pages (ASP) or Embedded Perl scripts.

5.3.7 Java Servlets

Servlets are small Java programs (small servers), that run on web/application servers and build web pages dynamically. Therefore you can compare it with CGI. Building web pages dynamically can be very useful. An example is when you use a search engine or put something in a shopping-cart in a Webshop.

A Java thread in a single daemon process handles every request that comes in to a Servlet. Servlets can also share data among other Servlets, which is an advantage when handling with databases. Because Servlets are written in Java they are highly portable. For example they can be run under Apache, Microsoft IIS or almost any major web/application server.

5.3.8 XML

XML is like HTML a “tag” language. But unlike HTML, where you have a fixed set of tags, you define the tags as you go along. For example, if you like a tag to be named “<BACHE-LORSPROJECT>” it’s ok. Also, unlike HTML, XML is a “valid” language, that means for every start-tag you will need a stop-tag or else the parser will complain. When you make your XML page you separate content from appearance, that means that you have the data in one file and the layout/style in another. This way authors and designers can work more independently.

XML was derived from SGML (Standard Generalized Markup Language), and is a meta-language[8].

[8] A meta-language is a language describing another language

6 Evaluation

In this chapter we will evaluate the web/application servers Apache-JServ, Apache-Tomcat, IBM WebSphere, BEA WebLogic Server and Sybase Enterprise application server. Together with EIN we came up with the following questions:

- Performance

  - How will the web/application server handle stressful situations like many users at the same time? Will it deal with it in a satisfying way?
- Technologies

  - Does the web/application server support any special technologies, like Servlets, CORBA or CGI?
- Price

  - What is the cost of buying the web/application server?
- Support

  - Is the web/application server supported? Does it cost extra? Are upgrades included in the price?
- Tools

  - Does the web/application server come with any development environments, or anything else useful, like a GUI configuration tool?

We will try to answer all these questions and summarize it and see if there exists a web/application server that is better than all the others.

6.1 Apache-JServ

Apache is one of the most used web servers on the Internet today. Combined with JServ Servlet engine, Apache works as a web/application server. In this evaluation we have used Apache web server version 1.3.12 and Apache JServ 1.1.

6.1.1 Installing Apache-JServ on Windows NT

Installing Apache-JServ on Windows NT is pretty straightforward. First we installed Apache web server by following the installation wizard included. The next step was to integrate JServ into apache; this to was easily done using the installation wizard for JServ.

6.1.2 Performance

The results from the Apache-JServ test on NT can be viewed in table 6.1. As you can see the result differs a lot between the highest and lowest result. The lowest value could be the result of a denied request and due to weaknesses in Apache JMeter we were unable to measure them. This will affect the calculation of the average and median values and we think that the highest value would be the most accurate value to use in this evaluation.

	HelloWorld.html (ms)	HelloWorld.class (ms)	StringManip.class (ms)
Highest	511	531	9804
Average	96	126	5715
Median	70	110	6469
Lowest	10	10	70

Table 6.1: Result from performance test on Apache-JServ

As mentioned in chapter 7, a higher degree of accuracy could be obtained by using another performance-test application. You should however see this results as guidelines and not as statistically correct. Nevertheless it can be useful data when comparing between different web/application servers.

6.1.3 Technologies

The different technologies supported by Apache-JServ can be viewed in Table 6.2. The lack of support for new "hot" technologies like CORBA and EJB and the fact that JServ never will support any higher version of Java Servlet API than 2.0 makes this combination less attractive to web/application developers. Neither Apache nor JServ supports XML but there are solutions to this weakness and one of them is Cocoon from The Apache XML project [8].

Technology	Version/Supported
CGI	Yes
CORBA	No
EJB	No
JDK	1.2.2
JSP	No
Servlets	2.0
XML	No

Table 6.2: Supported technologies in Apache-JServ

Apache-JServ is not JDK dependent, this means that you can use any JDK in your web/application server runtime environment. The ability to execute CGI-applications is included in the Apache web server and this might be useful if older[9] web/applications are still being used.

6.1.4 Price

Apache web server and JServ Servlet engine are both freeware and new version can easily be downloaded at The Apache Software Foundation homepage [7]. Apart from some legal details concerning the Apache-JServ product name and under which form redistribution is allowed, anyone is free to use and develop Apache-JServ to fit there needs.

6.1.5 Support

The Apache Software Foundation does not have any support agreement included in their products. Instead the user can obtain information about bugs, upgrades and more on their homepage [7]. The online support contains FAQ’s and guides that will help the user solve the most ordinary problems.

6.1.6 Tools

No tools are included in the Apache-JServ configuration. The web/application environment is established through the use of configuration files e.g. *httpd.conf* for the Apache web server. Since the source code for both Apache and Jserv are available for everyone, tools may already be available or perhaps under development. This type of resources usually ends up on the Internet and can be downloaded from there.

6.2 Apache-Tomcat

In this chapter we will try to evaluate Apache combined with Tomcat. Tomcat is a Servlet engine that is supposed to take over after the JServ Servlet engine and even further down the line be integrated into Apache. In this survey we will use Tomcat 3.1 together with Apache 1.3.12. As can be seen in chapter 6.2.3 Tomcat supports newer versions of the Java Servlet API compared to Jserv.

6.2.1 Installing Apache-Tomcat on Windows NT

Installing Apache-Tomcat on Windows NT does not differ too much from the installation of Apache-JServ. First we installed Apache web server by following the installation wizard included. The next step was to integrate Tomcat into apache. This was done with a little help from the user manual included in the Tomcat distribution.

6.2.2 Performance

The result from the performance test on Apache-Tomcat can be viewed in Table 6.3. As mentioned earlier this result is not statistically correct but it could be used as a guideline during evaluations.

	HelloWorld.html (ms)	HelloWorld.class (ms)	StringManip.class (ms)
Highest	481	1712	15032
Average	99	215	8425
Median	70	140	9153
Lowest	10	20	110

Table 6.3: Result from performance test on Apache-Tomcat

6.2.3 Technologies

Apache with Tomcat is a solid base for developers with interest for Java Servlet and JavaServer Pages technology. Through the concern of Apache this web/application server configuration grants access to the CGI technology. Even though CGI is on the verge of extinct several web applications still use this technology and they should be able to use Apache-Tomcat to overcome the transition. Table 6.4 describes in short the technologies supported by Apache-Tomcat web/application server.

[9] By older we mean applications based on C, C++, Perl and other.

Technology	Version/Supported
CGI	Yes
CORBA	No
EJB	1.1
JDK	1.2.2
JSP	1.1
Servlets	2.2
XML	No

Table 6.4: Supported technologies in Apache-Tomcat

As with Apache-JServ XML support can be obtained through the Cocoon [8] add-on from the Apache Software Foundation.

6.2.4 Price

Apache web server and Tomcat Servlet engine are both freeware and new version can easily be downloaded at The Apache Software Foundation homepage [7]. Apart from some legal details concerning the Apache-Tomcat product name and some limits concerning redistribution, anyone is free to use and develop Apache Tomcat to fit there needs.

6.2.5 Support

The Apache Software Foundation does not have any support agreement included in their products. Instead the user can obtain information about bugs, upgrades and more on their homepage [7]. The online support contains FAQ's and guides that will help the user solve the most ordinary problems.

6.2.6 Tools

Apache-Tomcat web/application server does not come with any fancy development tools but it does include the Ant build tool [7]. Ant can be viewed as a Java based equivalent to "make" in the Unix world.

6.3 BEA WebLogic Server

BEA WebLogic Server has won a lot of awards and they claim to be the market leader with their web/application server. The BEA WebLogic Server also supports Java, and is thought by many to be the best web/application server for e-commerce. In this test we will use version 5.1.

6.3.1 Installing BEA WebLogic Server on Windows NT

Like most major Windows applications BEA WebLogic Server came with an installation wizard. By using the default settings, it was very easy to install the web/application server on our machine.

6.3.2 Performance

The result from the performance test on BEA WebLogic Server can be viewed in Table 6.5. As mentioned earlier this result is not statistically correct but it could be used as a guideline during evaluations.

	HelloWorld.html (ms)	HelloWorld.class (ms)	StringManip.class (ms)
Highest	921	461	6780
Average	102	74	4184
Median	60	50	4757
Lowest	10	10	100

Table 6.5: Result from performance test on BEA WebLogic Server

6.3.3 Technologies

As you can see in Table 6.6, BEA WebLogic Server provides a number of features that today are industry standard components. Since BEA WebLogic Server is strongly connected to Java, it supports the latest versions of EJB, JSP and Servlets.

Technology	Version/Supported
CGI	Yes
CORBA	2.2 (2.3)
EJB	1.1
JDK	1.2.2
JSP	1.1
Servlets	2.2
XML	Yes

Table 6.6: Supported technologies in BEA WebLogic Server

BEA WebLogic Server also provides XML integration examples that will work with any XML-compliant parser, and everything is provided for developers to build XML-enabled applications. If CORBA support is wanted you can easily integrate that with BEA WebLogic Enterprise. CGI is also supported so those with many CGI applications will not be left out in the cold. More information is available at the BEA homepage [11].

6.3.4 Price

As with many of the web/application server developers on the market, BEA delivers several different editions of their server. We will list some of them below.

- WebLogic Server Base Edition/CPU USD 13 000
- WebLogic Server Cluster Edition/CPU USD 20 000
- WebLogic Commerce Server/CPU USD 52 000
- WebLogic Personalization Server/CPU USD 33 000
- WebLogic Server Developer license/license USD 3 300

As you can se the price varies depending on the version you are interested in.

6.3.5 Support

Once you purchase the BEA WebLogic Server you will have the option to choose what type of support you want. Depending on the amount of availability you desire you’ll have various packages to choose from. The lightest package stretches over office-hours (8 hours, 5 days/week), and the heaviest is “around the clock” support 365 days a year. Support is included in the price the first year. After that the support fees are paid annually and entitle you to new BEA product upgrades as long as the maintenance and support remains current. Customers can receive support via phone, fax or e-mail.

6.3.6 Tools

The tools included in the test package wasn’t that rich, but BEA has a package, WebGain Studio, which can be purchased separately. WebGain Studio is an integrated suite of tools to support development of server applications. WebGain Studio provides an all-round suite that contains all of the products required designing, developing, debugging, and deploying web applications. Unfortunately no price information were available for WebGain Studio when we wrote this evaluation.

6.4 IBM WebSphere

IBM WebSphere can be purchased in three different editions, Standard edition, Advanced edition and Enterprise edition. In this evaluation we will use IBM WebSphere Standard edition version 3.02. Some of the differences between the editions will be discussed in chapter 6.4.3.

6.4.1 Installing IBM WebSphere on Windows NT

The installation was performed in three steps. First we installed IBM's own JDK 1.1.7p for Windows NT. Then we installed the IBM HTTP Server version 1.3.6. And then finally IBM WebSphere Application Server 3.02 Standard Edition. The four other web/application servers we tested came with an installation wizard, and IBM WebSphere was no exception. By following the wizard it was quite simple to perform the installation.

6.4.2 Performance

Because IBM WebSphere requires a special JDK version[10], it’s not completely fair to compare the IBM WebSphere performance test with the other web/application performance tests. Nevertheless the results between the two JDK versions shouldn’t differ too much.

A complete overview of the test result can be viewed in Table 6.7. As with all the other performance tests done during this bachelor's project the statistical value can be discussed. We will only consider these values as guidelines.

[10] JDK 1.1.7 IBM

	HelloWorld.html (ms)	HelloWorld.class (ms)	StringManip.class (ms)
Highest	671	480	10014
Average	85	123	2717
Median	50	100	2073
Lowest	10	10	60

Table 6.7: Result from performance test on IBM WebSphere

6.4.3 Technologies

The Standard edition of IBM WebSphere is pretty modest and some of the techniques supported are of older versions. IBM WebSphere lacks the ability to run on any JDK and as you can se in our extra task in appendix C, the JDK version can have a crucial effect on the performance. In Table 6.8 a complete listing of the supported technologies can be viewed.

Technology	Version/Supported
CGI	Yes
CORBA	No
EJB	No
JDK	1.1.7p (IBM JDK)
JSP	1.0
Servlets	2.1
XML	1.0

Table 6.8: Supported technologies in IBM WebSphere

IBM WebSphere Standard edition comes with support for XML technology, this makes it complete as a lightweight application server but with no support for EJB and CORBA demanding web/application developers will be better off using the Advanced or Enterprise edition [10].

6.4.4 Price

As with most web/application servers IBM provides several distributions of WebSphere application server. Potential customers can choose between Standard, Advanced or Enterprise edition, all with different configuration and function. In this evaluation the price criterion will be represented by WebSphere Standard Edition version 3.0.2 distributed on CD-ROM. As mentioned above several other distributions are available and the price for these can easily be obtained at ShopIBM [6]. The price for the distribution we have used is USD 755 and it to was found at ShopIBM [6].

6.4.5 Support

IBM's support service is quite extensive. Several support configurations are available and the price depends on the content of each configuration. One support configuration contains an upgrade protection, which can be signed for one or two years. If the customer choose a support configuration with upgrade protection IBM will provide newer versions as they appear.

Support is available during office hours but for customers with other needs, special agreements can be arranged. Support from IBM can be obtained through phone, fax or e-mail.

6.4.6 Tools

IBM delivers WebSphere with two built-in tools, the HTTP server administration tool based on a GUI and the site analysis tool [10]. The HTTP server administration tool will help the server administrator configure and manage the HTTP server[11]. The site analysis tool can be used to help the server administrator increase the performance of the HTTP server.

6.5 Sybase Enterprise Server

In this evaluation we will use Sybase Enterprise Server 3.5 Enterprise edition. This is a package containing several components including Sybase PowerJ 3.5, Sybase PowerBuilder 7, Jaguar CTS and PowerDynamo.

6.5.1 Installing Sybase Enterprise Server on Windows NT

Installing Sybase Enterprise Server on Windows NT or any other Windows[12] system for that matter is often all about clicking the *Next* button, and so we did. During the installation we choosed to include the Jaguar CTS component, which is the servlet-engine, and continued through the installation wizard.

[11] An HTTP server is equal to a web server, both provides the calling client with web pages.
[12] Microsoft Windows 95 or Microsoft Windows 98

6.5.2 Performance

Table 6.9 displays the result obtained during our performance test of Sybase Enterprise Server. Although this test is done with some accuracy the result is not statistically correct, we have only done the performance test once and without statistic methods in mind.

	HelloWorld.html (ms)	HelloWorld.class (ms)	StringManip.class (ms)
Highest	1241	501	17055
Average	131	99	7877
Median	90	80	10135
Lowest	10	10	50

Table 6.9: Result from performance test on Sybase Enterprise Server

6.5.3 Technologies

Sybase Enterprise Application Server seams to cover it all. As can be viewed in Table 4.1 all of the technologies included in this survey are supported. Even if we didn't find information on which version that is supported in all of the technologies, a deeper investigation should give an answer in this matter. In Table 6.10 all the supported technologies are listed.

Technology	Version/Supported
CGI	Yes
CORBA	Yes
EJB	1.1
JDK	1.2.2
JSP	1.1
Servlets	2.2
XML	Yes

Table 6.10: Supported technologies in Sybase Enterprise Server

6.5.4 Price

Sybase Enterprise Application Server is split up in two main packages. The package we tested, the Development Edition, contains among several things a development suite were you can develop your own Java applications. The other package, Small Business Edition, is a little more expensive, about USD 3 200, but on the other hand it supports for example database publishing. The price for the first package is USD 645.

6.5.5 Support

Sybase provides support on different levels or plans as they call it. The support is divided into 5 plans:

- Incident Plan
- Basic Plan
- Extended Plan
- Enterprise Plan
- Enterprise Plan Support Option

The differences between the different plans are the level of support included and the availability of the support. Some of the plans[13] include new version release, which means that newer versions can be obtained through the supplier. Users of the Incident Plan will have to purchase an update subscription plan annually in order to receive new releases. For customers with really high demands, Enterprise Plan Support Option is available. This plan will allow the user to personalize the support to fit the need of their projects. All of the support plans mentioned above expire after some time and renewal is vital in order to avoid difficulties obtaining support.

The support is usually available through several forms e.g. email lists, telephone and through the Technical Library CD, containing technical documents. For multi-country installations global support agreements can be obtained, this will save the customer from having to purchase a separate support agreement in each of the country they are active in.

6.5.6 Tools

Sybase Enterprise Application Server is a complete suit with development tools and servers. Among many other components Sybase Enterprise Application Server contains a tool called PowerJ [12]. PowerJ is supposed to help developers create Java programs without problems and in a less time consuming fashion.

[13] Basic plan, Extended plan and Enterprise plan.

7 Summary

It seems that different web/application servers have different advantages. In this evaluation we have used five different web/application servers and we will now try to summarize the outcome of our evaluation.

First we have the heavy group containing BEA WebLogic Server and Sybase Enterprise Application Server. These servers are capable of handling new technologies and they often come equipped with complete development tools and a large support organization to back them up. Our opinion is that these web/application servers would suit developers of large web based applications like e-commerce web-sites.

In this group we would choose Sybase Enterprise Server as the winner. The main reason for this is that Sybase Enterprise Server comes with an easy-to-understand user interface and good support. Also, you don’t have to edit any configuration files before you can get your Servlets to work. On the other hand, Sybase Enterprise Server didn’t perform as well as BEA WebLogic Server did in the performance test. But the overall impression leads Sybase Enterprise Server to victory.

The other group contains Apache-JServ, Apache-Tomcat and IBM WebSphere. Our opinion, based on the impressions gained during this work, is that these web/application servers though widely spread on the Internet lack the ability to compete with the other two regarding supported technologies and tools available. The support for the Java Servlet technology does however give these web/application servers a place on the Internet. Apache-Tomcat and it's predecessor Apache-JServ has the advantage of being free, you can download them and include them in your projects without having to pay for them.

Even though IBM WebSphere have many similarities with Apache-JServ and Apache-Tomcat one difference is the configuration and management tool included. This makes IBM WebSphere more professional in its appearance but bare in mind that IBM WebSphere is a commercial product.

Trying to summarize this group was not easy but we have decided that IBM WebSphere is the winner. Even though some weak spots can be found (mostly regarding the old version of the JDK), the overall impression is that this web/application server is a commercial product with advantages like full-scale support, fully developed user interface and a fast Servlet engine.

We feel that some future extensions could be made to make this work more complete. From the beginning of this bachelor’s project our intentions was to test the performance in both Windows NT (PC) and Sun Solaris (UNIX) environment. As we experienced a shortage of time we decided to only run the performance test on Windows NT. The method we have used should work fine on any Solaris systems, so an extension of our performance test could easily be done. Some information on how to install Apache-JServ on Solaris can be found in appendix A.

We also feel that more precise performance tests could be achieved if the following components where made more accurate.

- A standalone lab environment with no connection to the other network.
- A more precise test application which reveals more information about each request e.g. if the web/application server is too busy the test application will log this information and not just regard it as a very fast response.
- Our methods regarding the calculation of test results could be made statistically correct.
- More work could be done to optimize the web/application server’s performance. Both the configurations of the JVM and the web/application server could affect the performance.
- On some systems the amount of logging could be of importance for the performance.

We have tried to introduce facts about the different contenders and our intention was to give enough information to be able to compare them on by one. We feel however that more could be done to make this evaluation more fair, other technologies like SSL and database support could be measured and deeper investigations could lead to more information regarding tools available for each web/application server.

References

[1] Ben Laurie and Peter Laurie. *Apache The Definitive Guide.* O'REILLY, 1997.

[2] Whatis.com. *http://www.whatis.com.* Whatis.com Inc, 2000.

[3] Jason Hunter and William Crawford. *JAVA Servlet Programming.* O'REILLY, 1998.

[4] ServerWatch.com. *http://serverwatch.internet.com.* Internet.com Corp, 2000-04-13.

[5] Mohammed J. Kabir. *Apache Server Bible.* IDG Books Worldwide Inc, 1998.

[6] ShopIBM. *http://commerce.www.ibm.com.* IBM Corp, 2000-04-28.

[7] The Apache Software Foundation. *http://www.apache.org.* The Apache Software Foundation Org, 2000-04-28.

[8] The Apache XML Project. *http://xml.apache.org/cocoon.* The Apache Software Foundation Org, 2000-04-28.

[9] The Source For Java Technology. *http://java.sun.com*. Sun Microsystems Inc, 2000-05-03.

[10] IBM Corporation. *http://ibm.com.* IBM Corp, 2000-05-04.

[11] BEA Systems. *http://www.bea.com.* BEA Systems Inc, 2000-05-04.

[12] Sybase. *http://www.sybase.com.* Sybase Inc, 2000-05-11.

A Installation of Apache and Apache JServ on Solaris

While standing in the Apache source directory we first executed the following line:

```
./configure --prefix=/home/qinxpaj/APP_SERVERS/apache;
make;
make install;
```

Then we edited the *httpd.conf* –file located in the *conf* directory in the Apache directory.
There we set the variable *ServerName*.
To include *so -(shared object)* -support following line was executed:

```
./configure \
--prefix=/home/qinxpaj/APP_SERVERS/apache \
--enable-rule=SHARED_CORE \
--enable-module=so;
make install;
```

Now we were ready to install JServ. While standing in the JServ source directory we executed this line:

```
./configure \
--prefix=/home/qinxpaj/APP_SERVERS/jserv \
--with-apache-src=/home/qinxpaj/shared/solaris/Uncompressed/apache_1.3.12 \
--with-jdk-home=/opt/JDK/jdk1.2_01 \
--with-JSDK=/home/qinxpaj/shared/solaris/JSDK/JSDK2.0/lib/jsdk.jar \
--disable-debugging;
make;
make install
```

After changing back to the Apache source directory we executed:

```
make;
make install
```

Finally we edited the *httpd.conf* –file located in the *conf* directory in the Apache directory.
There we added this line: *Include /home/qinxpaj/APP_SERVERS/jserv/etc/jserv.conf*

Finally finished.

B Performance test

B.1 HelloWorld HTML test page

```
<!------------------------------------------------------------------------------>
<!--Hello World - Test HTML Page                                               -->
<!--Per-Anders Johansson & Magnus Nilsson                                      -->
<!------------------------------------------------------------------------------>
<HTML>
		<TITLE>
				Hello World - Test HTML Page
		</TITLE>

		<BODY TEXT="#000000" BGCOLOR="#FFFFFF">
				<H1>Hello World</H1>
		</BODY>
</HTML>
```

B.2 HelloWorld Java Servlet test application

```
/*****************************************************************/
/*Hello World - Test Servlet                                     */
/*Per-Anders Johansson & Magnus Nilsson                          */
/*****************************************************************/
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet
{
	public void doGet(HttpServletRequest request, HttpServletResponse respons)throws ServletException, IOException
	{
			respons.setContentType("text/html");
			PrintWriter out = respons.getWriter();

			out.println("<HTML>");
			out.println("<HEAD>");
			out.println("<TITLE>HelloWorld-Test Servlet</TITLE>");
			out.println("</HEAD>");
			out.println("<BODY>");
			out.println("<H1>Hello World</H1>");
			out.println("</BODY>");
			out.println("</HTML>");
	}
}
```

B.3 StringManip Java Servlet test application

```
/**********************************************************************/
/* StringManip - Test Servlet                                         */
/*Magnus Nilsson & Per-Anders Johansson                               */
/*********************************************************************/

    import java.io.*;
    import javax.servlet.*;
    import javax.servlet.http.*;
    import java.util.*;

    public class StringManip extends HttpServlet
    {
            String smallStr = new String("abcdefghijklmnopqrstuvwxyzåäö");
            String bigStr = new String();
            long counter = 0;
            long totaltime;  //holds the execution time

            public void manipulate()
            {
                    Date start = new Date();
                    long total_s_time = start.getTime();  //get start time

                    while (bigStr.length() < 100000)        //manip
                    {
                                    counter++;
                                    bigStr = bigStr + smallStr + " " + counter;
                    }

                    Date stop = new Date();
                    long total_e_time = stop.getTime();   //get stop time
                    totaltime = total_e_time - total_s_time;                //calculate the difference
            }

            public void doGet(HttpServletRequest request, HttpServletResponse respons)throws ServletException, IOException
            {
                    respons.setContentType("text/html");
                    PrintWriter out = respons.getWriter();

                    out.println("<HTML>");
                    out.println("<HEAD><TITLE>String Manipulator</TITLE></HEAD>");
                    out.println("<BODY>");
                    out.println("<BIG>Performing String Manipulation...</BIG>");
                    manipulate();
                    out.println("<BR>" + totaltime + " ms");                //print total time
                    out.println("<BR><BR><BIG>DONE!!!</BIG>");
                    out.println("</BODY>");
                    out.println("</HTML>");
                    }
    }
```

C Extra Task

As a little side task we were asked to develop a program that tested how Java’s virtual machine handled multiple threads in multiple CPU environment.

The idea was to start one, two or four threads, and let them each go through an iterative loop. By timing each thread and comparing it to the total time taken, we could see if the threads were split up on the CPU’s and working parallel.
The conclusion was that the JDK version played a big part in the result.

C.1 Results

The result from running our test-program can be viewed in Table C.1.

Multiple threads in multiple CPU environment					
1 Processor UNIX (300MHz) einks648					
Solaris_JDK_1.1.7_08a	Thread 1 (ms)	Thread 2 (ms)	Thread 3 (ms)	Thread 4 (ms)	Total (ms)
1 Thread / 50000000 Iterations	2857				2870
2 Threads 50000000 Iterations	5723	5257			5731
4 Threads 50000000 Iterations	5637	5702	5712	5667	11433
One could think that the total time should be twice as big here because it's a one CPU machine.					
2 Processor UNIX (400MHz) kswas01					
Solaris_JDK_1.1.7_08a	Thread 1 (ms)	Thread 2 (ms)	Thread 3 (ms)	Thread 4 (ms)	Total (ms)
1 Thread / 50000000 Iterations	2151				2159
2 Threads 50000000 Iterations	2192	2172			2199
4 Threads 50000000 Iterations	2185	2189	2140	2172	4369
The result here is quite good. Two threads runs parallel the whole time.					
4 Processor UNIX (400MHz) solstal					
Solaris_JDK_1.1.7_08a	Thread 1 (ms)	Thread 2 (ms)	Thread 3 (ms)	Thread 4 (ms)	Total (ms)
1 Thread / 50000000 Iterations	2130				2137
2 Threads 50000000 Iterations	2132	2130			2137
4 Threads 50000000 Iterations	2130	2127	2126	2128	4263
Here should the threads be split up on all 4 CPU's, but the result is the same as on a 2 CPU machine.					
Solaris_JDK_1.2_01	Thread 1 (ms)	Thread 2 (ms)	Thread 3 (ms)	Thread 4 (ms)	Total (ms)
1 Thread / 50000000 Iterations	1764				1777
2 Threads 50000000 Iterations	1987	1879			1998
4 Threads 50000000 Iterations	2426	2200	2440	2333	2539
This is the same test as the latest above but with a newer JDK-version. As you can see on the result the threads are split up on all 4 CPU's. Interesting.					

Table C.1: Performance results from extra task

As you can see, the JVM version plays a big part in the result. The newer JDK_1.2_01 seems to improve the performance a lot compared to the older one.

C.2 Source code

```
/*
*    Thread tester (UNIX version)
*    Modified: 2000-02-25
*    Author: Magnus Nilsson (QINXNIM)
*    Per-Anders Johansson (QINXPAJ)
*/

import java.util.*;

/*-----------------------------------------------------------------------------------------------
*  class Tt
*-----------------------------------------------------------------------------------------------*/
public class Tt extends Thread
{
    public static long thread1 = 1;
    public static long thread2 = 2;
    public static long thread3 = 3;
    public static long thread4 = 4;

    /*------------------------------------------------------------------------------------------
    *  Method startThreads
    *  Creates instances of inner class Process.
    *  Responsible of starting and interrupting the threads.
    *------------------------------------------------------------------------------------------*/
    public void startThreads(int choice, double iterations)
    {
        switch (choice)                          //check the choice made
        {
            /*One Thread*/
            case 1:
            {
                Process p1 = new Process("1", iterations);
                p1.aktivitet.start();

                try
                {
                    p1.aktivitet.join();           //the main thread (this) waits for p1 to finish.
                }
                catch (Exception e_p1)
                {
                    System.out.println("Exception in join()");
                }
            }break;


            /*Two Threads*/
            case 2:
            {
                Process p1 = new Process("1", iterations);
                Processp 2 = new Process("2", iterations);
                p1.aktivitet.start();
                p2.aktivitet.start();

                try
                {
                    p1.aktivitet.join();
                    p2.aktivitet.join();
                }
                catch (Exception e_p2)
                {
                    System.out.println("Exception in join() (2 threads)");
                }
            }break;


            /*Four Threads*/
            case 4:
            {
                Process p1 = new Process("1", iterations);
                Process p2 = new Process("2", iterations);
                Process p3 = new Process("3", iterations);
                Process p4 = new Process("4", iterations);

                p1.aktivitet.start();
                p2.aktivitet.start();
                p3.aktivitet.start();
                p4.aktivitet.start();

                try
                {
                    p1.aktivitet.join();
                    p2.aktivitet.join();
                    p3.aktivitet.join();
                    p4.aktivitet.join();
                }
                catch (Exception e_p4)
                {
                    System.out.println("Exception in join() (4 threads)");
                }
            }break;

            default:
            {
                System.out.println("\nIllegal argument!");
                System.out.println("Valid argument is 1, 2 or 4.");
            }
        }//end switch
    }//end startThreads
```

```
/*-------------------------------------------------------------------------------
*  class Process (inner class)
*  This is where the threads are created
*-------------------------------------------------------------------------------*/
public class Process implements   Runnable
{
    public Thread aktivitet = new Thread(this);            //new thread
    private String text;
    private double iteration;
    private double i;

    //constructor
    public Process(String txt, double _iteration)
    {
        text = txt;
        iteration = _iteration;
    }

        /*-------------------------------------------------------------------------
        *  Method run
        *  This is what every thread do.
        *-------------------------------------------------------------------------*/
        public void run()
        {
            try
            {
                Date start = new Date();                    //get start time
                long s_time = start.getTime();

                for (int i=0; i<iteration; i++);           //loop baby loop...

                Date finish = new Date();         //get the finish time
                long f_time = finish.getTime();

                long time = f_time - s_time;         //how long time did it take

                if (text == "1")                 //checks which thread it is
                    thread1 = time;
                else if (text == "2")
                    thread2 = time;
                else if (text == "3")
                    thread3 = time;
                else if (text == "4")
                    thread4 = time;
            }
            catch (Exception ee)
            {
                System.out.println("Exception in class Process, method run!");
            }
        }
    }
```

```
/*------------------------------------------------------------------------------------
*  Main
*  Checks the arguments and starts the program if the arguments are ok
*----------------------------------------------------------------------------------*/
public static void main(String args[])
{
   int choice = -1;
   double iterations = -1;
   String choiceString;
   String iterationString;

   //check that number of arguments is correct
   if (args.length != 2)
   {
      System.out.println("\nThread Tester 1.0 (unix) by QINXNIM & QINXPAJ  2000");
      System.out.println("USAGE: java Tt <arg1> <arg2>");
      System.out.println("<arg1> is number of threads and\n<arg2> is number of iterations in every thread.");
   }
   else //number of args is ok
   {
      //try to extract the arguments
      try
      {
         choiceString = args[0];
         iterationString = args[1];

         Double d = new Double(iterationString);
         Integer i = new Integer(choiceString);
         iterations = d.doubleValue();
         choice = i.intValue();
      }
      catch(Exception e)  //catch exception but do nada
      {
      }

      //check if less than zero
      if ((choice < 0) || (iterations < 0))
      {
         System.out.println("\nIllegal argument!");
         System.out.println("USAGE: java Tt <arg1> <arg2>");
         System.out.println("<arg1> is number of threads (1, 2 or 4)\n<arg2> is number of iterations in every thread.");
      }
      else //OK
      {
         Tt tt = new Tt();

         //start clocking total time
         Date totalstart = new Date();
         long total_s_time = totalstart.getTime();
```

```
            //run threads
            tt.startThreads(choice, iterations);

            //clock the endtime
            Date totalend = new Date();
            long total_e_time = totalend.getTime();

            long totaltime = total_e_time - total_s_time;

            if (choice == 1)
               System.out.println("Thread 1 finished in " + thread1 + " ms");
            else if (choice == 2)
            {
               System.out.println("Thread 1 finished in " + thread1 + " ms");
               System.out.println("Thread 2 finished in " + thread2 + " ms");
            }
            else if (choice == 4)
            {
               System.out.println("Thread 1 finished in " + thread1 + " ms");
               System.out.println("Thread 2 finished in " + thread2 + " ms");
               System.out.println("Thread 3 finished in " + thread3 + " ms");
               System.out.println("Thread 4 finished in " + thread4 + " ms");
            }

         System.out.println("Total operation time = " + totaltime + " ms.");
                                                          }
      }//end else
   }// end main
}
```