

# Win2K Server Installation for a National Political Machine on a Laptop

By R.W. Clark  
copyright 2005

## Table of Contents

Introduction.....	4
Installing Win2K.....	4
Configuring Win2K for Connectivity.....	4
Testing Win2K for Security.....	4
Installing Apache Server.....	5
Configuring Apache Server.....	5
Testing Apache Server.....	5
Installing OpenSSH Server.....	5
Configuring OpenSSH Server.....	5
Testing OpenSSH Server.....	5
Installing FileZilla FTP Server.....	6
Configuring FileZilla FTP Server.....	6
Testing FileZilla FTP Server.....	7
Installing FileZilla FTP Client (abbreviated).....	7
Configuring FileZilla FTP Client (abbreviated).....	7
Testing FileZilla FTP Client (abbreviated).....	8
Introducing XAMPP, an Expanded Apache Server.....	8
Installing XAMPP.....	8
Configuring XAMPP.....	9
Testing XAMPP.....	10
Setting XAMPP Bootstrap.....	11
PHP version switching.....	11
Mercury Mail Server Configuration.....	11
Installing Java.....	11
Configuring Java.....	12
Testing Java.....	12
Installing Java for a JSP Server.....	12
Configuring Java for a JSP Server.....	13
Testing Java for a JSP Server.....	13
Installing Tomcat, the JSP Server.....	13
Testing Tomcat.....	13
Configuring Tomcat.....	14
Testing Tomcat.....	14
Testing Tomcat Servlets.....	14
Testing Tomcat JSPs.....	16
Setting Tomcat Bootstrap.....	17
Setting Tomcat Permissions.....	18
How do I override the default home page loaded by Tomcat?.....	18
MySQL as it is delivered with XAMPP.....	19
MySQL User Privileges.....	19
MySQL 4.1.10.....	20
phpMyAdmin 2.6.1.....	20

MySQL connector installation.....	20
MySQL Servlet Connection Testing.....	20
Linux Server Environment.....	21
NetBeans Integrated Development Environment.....	22
Telnet FTP SSH feature Appendix.....	23
Apache Appendix.....	25
After I rebooted my Linux box XAMPP stopped running! How can I fix this?.....	25
How can I make my XAMPP installation more secure?.....	26
Server Parameter Appendix.....	28
Uninstall.....	29
Java Appendix.....	29
Tomcat Appendix.....	31
Sample Servlet Application Construction.....	33
Sample Servlet Application Deployment.....	36
Sample Servlet Application Testing.....	37
Sample JSP Application Testing.....	38
Top Ten Tomcat Configuration Tips.....	39
1. Configuring the Admin Web Application.....	39
2. Configuring the Manager Web Application.....	40
3. Deploying a Web Application.....	41
4. Configuring Virtual Hosts.....	42
5. Configuring Basic Authentication.....	43
6. Configuring Single Sign-On.....	44
7. Configuring Customized User Directories.....	45
8. Using CGI Scripts with Tomcat.....	46
9. Changing Tomcat's JSP Compiler.....	47
10. Restricting Access to Specific Hosts.....	47
MySQL Connector Appendix.....	55
MySQL Servlet Appendix.....	57

## **Introduction**

This document is a clone of the Linux Server Installation document published in this series. As such, much of that content persists within this one, and creates a rather much larger document than necessary until editing is complete. This editing is proceeding in a rather linear fashion, following the actual installation of the operating system and server components onto a lap top computer. Further, there is another document in this series that relates to applications installations.

What follows is written in as a linear, technical journal of the steps taken from the initial unformatted laptop to the finished operating system holding a set of core servers. As a laptop is a workstation, distinct from the Linux headless system, the companion document for applications should be consulted.

## **Installing Win2K**

1. Obtained PentiumIII based PC w/512MBytes of RAM and 30GBytes of disc storage. This box is a laptop system that arrived without any operating system.
2. Using another Win2K system, I assembled 4 boot floppies for installing Win2K on the system. This was required because using the CDROM as a boot source did not work.
3. The floppies attended to partitioning the hard drive and establishing a boot installation that could read the CDROM.
4. Remaining activities related to obtaining and installing drivers unique to the platform.
5. Of particular note, hackers immediately penetrated the system when I connected to the router and through to the Web.

## **Configuring Win2K for Connectivity**

1. After withdrawing from the Web, I instead used the other Win2K system to surf for the 4 available service packs to upgrade Win2K; these were installed (the first was rejected by the existing system).
2. The Agnitum Firewall was installed.
3. The firewall need to allow services.exe and svchost.exe to communicate with the web.

## **Testing Win2K for Security**

Reconnect to the Web and visited <http://grc.com/default.htm> to test ports and general issues of security to tailor the firewall rules.

## Installing Apache Server

Version 2 of Apache is currently available, however, for a more complete package with the usual suite of support servers like PHP and MySQL, it may be more convenient to skip this install and instead install XAMPP in the next section. **Notice, that doing this install (v.2 of Apache) will mask XAMPP Apache from running without other tailoring. Right now, such tailoring is not described in this paper.**

Apache Web Server available from:

<http://httpd.apache.org/download.cgi>

The download and installation of Apache was enough to get it up and running. It is available for viewing by pointing a browser at:

<http://127.0.0.1/>

## Configuring Apache Server

No actions performed pending XAMPP discussion.

Consult the discussion about PHP version switching in this document. It relates to setting the correct version for applications that demand specific version.

## Testing Apache Server

No actions performed pending XAMPP discussion.

## Installing OpenSSH Server

The only working solution to allow an OpenSSH Server is to run it in Win2K under Cygwin. At this time there is little need for command line access to this project aside from moving files which can be done simpler.

For secure ftp services, open a secure ftp session with server type explicitly set to SFTP using SSH2. The FileZilla FTP Server supports this connection.

## Configuring OpenSSH Server

N/A

## Testing OpenSSH Server

Standard calling nomenclature is:

ssh [root@67.171.24.169](mailto:root@67.171.24.169)

respond to the prompt for password in the conventional way.

## Installing FileZilla FTP Server

FileZilla is a powerful FTP-client for Windows 9x, ME, NT4, 2000 and XP. It has been designed for ease of use and with support for as many features as possible, while still being fast and reliable. It supports SFTP.

FileZilla FTP Client and Server available from:  
<http://sourceforge.net/projects/filezilla/>

## Configuring FileZilla FTP Server

Of particular note, FileZilla needs to have permissions with the firewall. This, as a stumbling point, will be obvious in the screen log by the inability to connect to the server at boot.

We will take only those FileZilla Server Options that are to be changed, as they are offered from the menu bar selection “Edit – Settings”

### General settings

Max. number of users: 2

this is to limit access to the situation that would allow admin to sign in both locally and remotely – and no one else.

Number of threads: 2

this is not a high performance server, simply an admin function.

IP Filter:

\*.\*.\*.\*

this is to remove EVERYONE from access;

n.n.\*.\*

this is to add a range of IPs who CAN connect (which means you need to know their IP, or at least the first two levels denoted by n's here). This setting over-rides the first exclusionary filter. Thus, you reject everyone, except someone from FTP access.

### Security Settings

select all four check boxes.

### SSL/TLS Settings

select the check box “Enable SSL/TLS support”

this will ungrey all of the entry areas in this pane, the next thing to do is:

Press the “Generate new certificate...” button at the bottom right of the pane;

This opens another pane where you now enter a country code (US);

assign a filename (certificate.crt as it suggests is good enough);

and a path to it (where ever you keep important data that is backed up);

select the “Generate Certificate” button at the bottom;

Select the OK button to save and exit this portion of configuration.

Next we move on to Groups and their permissions. First we will add a group called “elite.” To do this, select the menu item “Edit – Groups,” and we will proceed on the same process of describing only those changes to the standard configuration.

General

Go to the far right of the pane that appears and in the sub-pane also called groups, press the “add” button.

Observing that a new window opens, fill in the name “elite” and press OK.

Within the “Connection settings” pane, select the check box “Force SSL for user login.”

Shared Folders

In the middle pane, select the “Add” button and point the file browser at the directory, or path to a folder that is to be accessible to FTP.

To the right of this pane, you should notice a vertical line of check boxes that allow your user to perform all read/write/erase functions for files and directories. This is where the rubber hits the road as far as permissions go.

Select the OK button to save and exit this portion of configuration.

Next we move last to Users and their permissions by association to a group membership. First we will add a User called “admin.” To do this, select the menu item “Edit – Users,” and we will proceed on the same process of describing only those changes to the standard configuration.

General

Go to the far right of the pane that appears and in the sub-pane also called Users, press the “add” button and assign a group once a new member name has been entered.

Select the OK button to save and exit this portion of configuration.

## **Testing FileZilla FTP Server**

The best method to test the FTP server is using a FileZilla FTP client on another machine, through the Web. This, of course, means that both ends need to have their IP addresses known, and for the server to allow the client's IP as described above.

## **Installing FileZilla FTP Client (abbreviated)**

FileZilla Client arrives by virtue of the same downloads of the server. Launch the window installer for FileZilla Client.

## **Configuring FileZilla FTP Client (abbreviated)**

Select the menu item “File – Site Manager,”

At the bottom of the page, press the button marked “New Site;”  
When this ungrays the entry fields on the right;  
Enter the IP of the FileZilla FTP server (on another system, connected to the Web);  
Enter Servertype: FTP over SSL (explicit encryption);  
Select the Logontype radio button marked “Normal;”  
Enter the User name (established above) and  
Enter the user's Password;  
For this new connection, give it a name (by changing from the default “New FTP site”)  
Select the button “Save and Exit.”

### **Testing FileZilla FTP Client (abbreviated)**

This is a concurrent test of the FileZilla FTP server on the remote machine (which in the sense of this document is actually the local machine).

Select the menu item “File – Site Manager;”  
Choose the previously established account;  
Press the “Connect” button.

This will no doubt cause the Firewall to spring into action demanding rules for the connection. Provide such rules as are necessary to complete the connection and perform file transfers and normal FTP operations.

## **Introducing XAMPP, an Expanded Apache Server**

XAMPP is a complete server package. Complete means that it contains not only the standard Web Server, Apache, but it also contains many of the extensions to Apache that allow it to serve CGI and Java pages supported by a database. In this case, the PHP interpreter is the language of choice for many Web applications with MySQL being the database of choice to support them.

### **Installing XAMPP**

Following the Apache server installed and running, I decided to obtain a more complete server package called XAMPP (distribution file [xampp-win32-1.4.15-installer.exe](#) ) that contains an easily installed server complete with MySQL, PHP, and other features that are already integrated – and presumably running when Apache fires up. **Entering into this section of the Server Installation Journal will encounter Apache calling problems if Apache v.2 (described above) was already installed.**

The distribution for Windows 98, NT, 2000 and XP. This version contains: Apache,

MySQL, PHP + PEAR, Perl, mod\_php, mod\_perl, mod\_ssl, OpenSSL, phpMyAdmin, Webalizer, Mercury Mail Transport System for Win32 and NetWare Systems v3.32, JpGraph, FileZilla FTP Server, mcrypt, eAccelerator, SQLite, and WEB-DAV + mod\_auth\_mysql.

This is available from:

<http://sourceforge.net/projects/xampp>

During installation and configuration, the machine should be taken offline for security issues.

During the installation from the executable, there is a prompting

Install XAMPP servers (Apache, MySQL or FileZilla FTP) as service?  
respond Yes.

This results in the prompt:

Apache 2 Service seems to be installed! Abort.

Followed by the prompt:

install MySQL as service?  
respond Yes.

This results in the prompt:

FileZilla Service seems to be installed! Abort.

In the end of all these aborts, we find the prompt:

Congratulations! The installation was successful!....

The XAMPP Control Panel Application opens to reveal the three significant servers running.

Consult Apache Appendix below.

## **Configuring XAMPP**

Taken from:

<http://www.apachefriends.org/en/xampp-windows.html#1031>

As mentioned before, XAMPP is not meant for production use but only for developers in a development environment. The way XAMPP is configured is to be open as possible and allowing the developer anything he/she wants. For development environments this is great but in a production environment it could be fatal.

Here a list of missing security in XAMPP:

The MySQL administrator (root) has no password.

The MySQL daemon is accessible via network.

PhpMyAdmin is accessible via network.

Examples are accessible via network.

To fix most of the security weaknesses simply call the following URL:

<http://127.0.0.1/xampp/xamppsecurity.php>

The root password for MySQL+PhpMyAdmin and also a XAMPP directory protection can be established here.

Observing this last point, visit the page described above for security settings:

<http://127.0.0.1/xampp/xamppsecurity.php>

If, on calling this page it does not reveal Password editing fields, then check the firewall for blocking Apache.

Enter the MySQL SuperUser UID root and PW <yourpasswordhere>;

Press the Password changing button;

Stop and Start the MySQL server.

Enter the XAMPP User and P/W;

Press the Make safe the XAMPP directory button.

When you refresh the page (which should still show):

<http://127.0.0.1/security/index.php>

the XAMPP pages should be secure;

the MySQL admin user root should be secure;

PhpMyAdmin password should be secure;

the FileZilla FTP password should be secure;

PHP should NOT be running in “safe mode” (as recommended)

## **Testing XAMPP**

Point a browser at

<http://127.0.0.1>

and observe that instead of serving that page (which would indicate Apache v.2 only is running) that it instead is redirected to, and serves

<http://127.0.0.1/xampp/>

(it also contains v.2 of Apache, but configured under XAMPP). You should clear the browser's cache before this call to eliminate a cache hit and confusing the situation.

Reconnect to the Web and visit

<http://grc.com/default.htm>

to test ports and general issues of security to tailor the firewall rules. This should result in at least three ports open to the outside world. Those ports are for FTP, HTTP, and HTTPS. Other ports may report closed instead of stealth. This is alright because this machine is meant to be visible to the world as a server, we rely on the servers themselves to maintain security as established above – please insure to perform those steps for this very reason.

## Setting XAMPP Bootstrap

This is quite simple through selecting the XAMPP Control Panel. When it is open, observe that the **Svc** check box is selected for each of the servers you want to start at bootstrap. You also use this utility to selectively start/stop the servers for testing and configuration.

Now XAMPP should start and stop automatically if you boot or shutdown your machine.

## PHP version switching

Within XAMPP, at the page

<http://127.0.0.1/xampp/>

and from the menu at the left, there is a link called

PHP Switch

which details the procedures necessary to select the correct PHP version necessary for any particular application. This is a selection that is enforced from the time of selection ever afterward until a new selection is made. As such, the application that requires the lowest numbered version dumbs down all other applications' usage of PHP. Those applications must be downward compatible, or you will encounter problems.

## Mercury Mail Server Configuration

Go to

C:\Program Files\xampp\htdocs\xampp\Mercury Mail

[select mercury.exe](#)

From the menu bar,

[select Configuration-MercuryS SMTP Server](#)

## Installing Java

Consult Java Appendix below.

Visit:

<http://java.sun.com/j2ee/1.4/download.html#sdk>

and download J2EE 1.4 SDK AS:

[j2eesdk-1\\_4\\_02...](#)

Enter admin name and admin password as prompted. Pass over all default settings.

## Configuring Java

The following may conflict with the needs of the JSP Server installation of Java (which follows this discussion) – if so, consider using full pathname compiler, and classpath namings.

In Control panel, System, System Properties, Advanced, Environmental Variables, set the environmental variable:

```
JAVA_HOME=C:\Sun\AppServer\jdk\bin\
```

## Testing Java

Note that all calls should be made with a very verbose command line to eliminate any confusion with other installed packages that come with other server packages. When that confusion comes, it will announce itself as:

```
error: Invalid class file format: ... , wrong version: 48, expected 45
```

or any other version numbers found in this style of statement.

Use the following test program: C:\Test.java

```
public class Test
{
    public static void main(String[] args)
    {
        System.out.println("Hello world");
    }
}
```

Open a DOS session in the directory that holds this test program and enter:

```
C:\Sim\AppServer\jdk\bin\javac C:\Test.java
```

The result of the compile is the file: C:\Test.class

Then in this DOS session, run:

```
C:\> C:\Sun\AppServer\jdk\bin\java -classpath . Test
```

```
Hello world
```

## Installing Java for a JSP Server

This requires jre1.5.04 at

[http://www.java.com/en/download/windows\\_xpi.jsp](http://www.java.com/en/download/windows_xpi.jsp)

## Configuring Java for a JSP Server

In Control panel, System, System Properties, Advanced, Environmental Variables, set the environmental variable:

JAVA\_HOME=C:\Program Files\Java\jre1.5.0\_04\

## Testing Java for a JSP Server

To be done in co-ordination with Tomcat testing.

## Installing Tomcat, the JSP Server

Tomcat installation is for the benefit of adding JSP service to Apache; if this is not required, then such installation is unnecessary.

Consult the Tomcat Appendix below.

Download jakarta-tomcat-5.5.9.exe from:

<http://jakarta.apache.org/tomcat/index.html>

Install Tomcat by launching the download, during the progression of installation respond to the prompt for Administration Login by entering User Name and Password, leave the Connector Port at the traditional 8080.

There are admin packages for Tomcat at

<http://www.signal42.com/mirrors/apache/jakarta/tomcat-5/v5.5.9/bin>

right next to the main downloads, select:

jakarta-tomcat-5.5.9-admin.zip

This is wholly undocumented, but on review of the contents of the zip file, it appears that the way to go is to extract all files to a temporary directory. Open that directory, and the directory:

C:\Program Files\Apache Software Foundation\Tomcat 5.5\

and compare the contents of both to combine the temporary directory into the existing directory. This should result in no overwrites – only new files in

...\conf\Catalina\localhost\

and new directory

...\server\webapps\admin

## Testing Tomcat

In Control Panel, Administrative Tools, Services, start Apache Tomcat.

Enter the following URL into a browser:

<http://127.0.0.1:8080/>

It should open a page saying “**If you're seeing this page via a web browser, it means you've setup Tomcat successfully. Congratulations!**”

## Configuring Tomcat

Permissions are stored in

C:\Program Files\Apache Software Foundation\Tomcat 5.5\conf\tomcat-users.xml

The server may need to be re-started to engage the admin page if it is unavailable at:

<http://127.0.0.1:8080/admin/index.jsp>

When that page's prompt for username and password succeeds

<http://127.0.0.1:8080/admin/frameset.jsp>

will appear.

Set environmental variable:

CATALINA\_HOME=/opt/jakarta-tomcat-5.5.7/

jmx.jar may not be in CLASSPATH (or anywhere for that matter, not part of the distribution of Tomcat nor Apache).

obtain jmx-1\_2\_1-ri.zip from:

<http://java.sun.com/products/JavaManagement/download.html>

remove jmxri.jar from the zip file and rename it to jmx.jar

ftp this file to /opt/jakarta-tomcat-5.5.7/common/endorsed

## Testing Tomcat

```
cd $CATALINA_HOME
```

```
./bin/jsvc -home /opt/j2sdk1.4.2_07 -verbose -Djava
```

```
.endorsed.dirs=./common/endorsed -cp ./bin/bootstrap.jar -outfile ./logs/catalina.out
```

```
-errfile ./logs/catalina.err org.apache.catalina.startup.Bootstrap
```

Enter the following URL into a browser:

<http://67.171.24.169:8080/>

It should open a page saying **“If you're seeing this page via a web browser, it means you've setup Tomcat successfully. Congratulations!”**

## Testing Tomcat Servlets

Servlets, JSPs, are java files compiled to a secure directory under CATALINA\_HOME/webapps. The class file from the source's compilation requires that it be pointed to by the server's xml file for that directory under which it is installed. In the following case that will be in the servlets-examples directory that already exists in the distribution. Notable by the ALL CAPS format of directory naming, within WEB-INF web.xml will be found that describes all the servlets that may be accessed by a browser

through the Tomcat server.

Place the following file, HelloBraveNewWorldServlet.java, in

/opt/jakarta-tomcat-5.5.7/webapps/servlets-examples/WEB-INF/classes/

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

public class HelloBraveNewWorldServlet extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse
response)
throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<body>");
        out.println("<head>");
        out.println("<title>Hello Brave New World!</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Hello Brave New World!</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

Compile it in the usual manner:

```
/opt/j2sdk1.4.2_07/bin/javac -classpath
/opt/j2sdk1.4.2_07/lib/tools.jar:/opt/j2sdk1.4.2_07/jre/lib/rt.jar:/opt/jakarta-tomcat-
5.5.7/common/lib/servlet-api.jar HelloBraveNewWorldServlet.java
```

Change

/opt/jakarta-tomcat-5.5.7/webapps/servlets-examples/WEB-INF/web.xml  
to include

```
<servlet>
  <servlet-name>HelloBraveNewWorldServlet</servlet-name>
  <servlet-class>HelloBraveNewWorldServlet </servlet-class>
</servlet>
```

and

```
<servlet-mapping>
  <servlet-name>HelloBraveNewWorldServlet </servlet-name>
```

```
        <url-pattern>/servlet/HelloBraveNewWorldServlet </url-pattern>
    </servlet-mapping>
```

All that remains is to visit the servlet by going to:

<http://24.19.52.157:8080/servlets-examples/servlet/HelloBraveNewWorldServlet>

## Testing Tomcat JSPs

Create the following file, `hellobravenewworld.jsp`:

```
<html>
<head>
<title>My first JSP page
</title>
</head>
<body>
<%@ page language="java" %>
<% System.out.println("Hello Brave New World"); %>
</body>
</html>
```

And place it in

`/opt/jakarta-tomcat-5.5.7/webapps/jsp-examples/`

Change

`/opt/jakarta-tomcat-5.5.7/webapps/jsp-examples/WEB-INF/web.xml`

to include:

```
<servlet>
    <servlet-name>org.apache.jsp.hellobravenewworld_jsp</servlet-name>
    <servlet-class>org.apache.jsp.hellobravenewworld_jsp</servlet-class>
</servlet>
```

and

```
<servlet-mapping>
    <servlet-name>org.apache.jsp.hellobravenewworld_jsp</servlet-name>
    <url-pattern>/hellobravenewworld.jsp</url-pattern>
</servlet-mapping>
```

Change the file `ShowOrganization.jsp` (found in Tomcat Appendix) to match the test

database built in the next section.

Copy the file ShowOrganization.jsp to:

```
/opt/jakarta-tomcat-5.5.7/webapps/jsp-examples/WEB-INF/classes/
```

## Setting Tomcat Bootstrap

The various files distributed with Tomcat cover a lot of bootstrap shell scripts that introduce a lot of extraneous information that obscure a simple command already exhibited in the test above. As such the complete script Tomcat\_5\_5\_7.sh appears as:

```
cd /opt/jakarta-tomcat-5.5.7
case "$1" in
  start)
    #
    # Start Tomcat
    #
    /opt/jakarta-tomcat-5.5.7/bin/jsvc -home /opt/j2sdk1.4.2_07 -verbose -Djava
    .endorsed.dirs=./common/endorsed -cp ./bin/bootstrap.jar -outfile ./logs/catalina.out
    -errfile ./logs/catalina.err org.apache.catalina.startup.Bootstrap
    ;;
  stop)
    #
    # Stop Tomcat
    #
    PID=`cat /var/run/jsvc.pid`
    kill $PID
    ;;
  *)
    echo "Usage tomcat.sh start/stop"
    exit 1;;
esac
```

Place Tomcat\_5\_5\_7.sh at the CATALINA\_HOME root (/opt/jakarta-tomcat-5.5.7/)

**Make sure that the write permissions are set to 755.**

There is no real standard way to configure the boot process of a Linux system, but most of them should allow you to start Tomcat at boot time using the following steps.

First, find out your default runlevel.

Simply type **egrep :initdefault: /etc/inittab**.

You should no see a line containing a number between two colons.

**id:5:initdefault: shows up for SuSE Linux**

Go into the directory which configures this runlevel.

```
cd /etc/rc.d/rc5.d for SuSE Linux
```

Now carry out the actual configuration by typing:

```
ln -s /opt/jakarta-tomcat-5.5.7/Tomcat_5_5_7.sh S99Tomcat
```

```
ln -s /opt/jakarta-tomcat-5.5.7/Tomcat_5_5_7.sh K01Tomcat
```

```
shutdown -r 1
```

to reboot after configuration changes. Confirm all services configured appear following reboot.

## Setting Tomcat Permissions

NOTE: For security reasons, using the administration webapp is restricted to users with role "admin". The manager webapp is restricted to users with role "manager". Users are defined in /opt/jakarta-tomcat-5.5.7/conf/tomcat-users.xml.

```
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
  <role rolename="tomcat"/>
  <role rolename="role1"/>
  <role rolename="manager"/>
  <user username="tomcat" password="tomcat" roles="tomcat"/>
  <user username="role1" password="tomcat" roles="role1"/>
  <user username="both" password="tomcat" roles="tomcat,role1"/>
  <user username="root" password="secret" roles="manager"/>
</tomcat-users>
```

Unfortunately when the correct settings (**root/secret/manager**) are made to the users file, the system announces:

“Tomcat's administration web application is no longer installed by default. Download and install the "admin" package to use it.”

However, the Status and Tomcat Manager links function (after a successful login).

## How do I override the default home page loaded by Tomcat?

After successfully installing Tomcat, you usually test it by loading <http://localhost:8080>. The contents of that page are compiled into the `index_jsp` servlet. The page even warns against modifying the `index.jsp` files for this reason. Luckily, it is quite easy to override that page. Inside `$TOMCAT_HOME/conf/web.xml` there is a section called `<welcome-file-list>` and it looks like this:

```
<welcome-file-list>
  <welcome-file>index.html</welcome-file>
  <welcome-file>index.htm</welcome-file>
```

```
<welcome-file>index.jsp</welcome-file>
</welcome-file-list>
```

The default servlet attempts to load the `index.*` files in the order listed. You may easily override the `index.jsp` file by creating an `index.html` file at `$TOMCAT_HOME/webapps/ROOT`. It's somewhat common for that file to contain a new static home page or a redirect to a servlet's main page. A redirect would look like:

```
<html>

<head>
<meta http-equiv="refresh"
content="0;http://mydomain.com/some/path/to/servlet/homepage/">
</head>

<body>
</body>

</html>
```

This change takes effect immediately and does not require a restart of Tomcat.

---

## MySQL as it is delivered with XAMPP

Visit page:

[http://24.19.36.144/xampp/installation\\_index.php](http://24.19.36.144/xampp/installation_index.php)

and in the sidebar menu, then select:

phpMyAdmin

This should cause an access pane to open up for user id and password entry. When completed, if you have root privileges, then the rest will follow. Otherwise consult passwords above.

## MySQL User Privileges

To test functionality from the perspective of the typical user, an account for the typical user needs to be set up. Select the Privileges selection at the left of the line of icons.

When the admin page for Privileges opens, add the appropriate login information suitable for the typical user. On this first pass, use the default settings of the pull down menus that introduce each of the listings: User name, Host, and Password. This means you have to provide a name, accept "Any Host," and provide a password. An example:

```
User name: harvey
"Any Host"
Password: therabbit
```

Also, select the "Check All" selection box for simplicity's sake. Push the "Go" button to complete the entry.

The screen will update to show new configuration items like which database that user harvey has privileges to. As we have yet to develop the database, this will be taken care of in the separate document, *Data Base Design*.

## MySQL 4.1.10

The details of originating a new data base are contained in *Data Base Design* documentation.

## phpMyAdmin 2.6.1

### MySQL connector installation

In taking the lead from the appendix material, the `x.jsp` (developed in the document *Data Base Design*) is placed into:

```
/opt/jakarta-tomcat-5.5.7/webapps/ROOT/
```

and the Data Base connector, `mysql-connector-java-3.1.8-bin.jar`, is placed into:

```
/opt/jakarta-tomcat-5.5.7/common/lib/
```

### MySQL Servlet Connection Testing

Copy the file `ShowBedrock.java` (found in the MySQL Servlet Appendix) into

```
/opt/jakarta-tomcat-5.5.7/webapps/servlets-examples/WEB-INF/classes/
```

cd to (the same directory):

```
/opt/jakarta-tomcat-5.5.7/webapps/servlets-examples/WEB-INF/classes/
```

compile this file with:

```
/opt/j2sdk1.4.2_07/bin/javac -classpath /opt/jakarta-tomcat-5.5.7/common/lib/mysql-connector-java-3.1.8-bin.jar:/opt/jakarta-tomcat-5.5.7/common/lib/servlet-api.jar ShowBedrock.java
```

Change

```
/opt/jakarta-tomcat-5.5.7/webapps/servlets-examples/WEB-INF/web.xml
```

to include

```
<servlet>
  <servlet-name>ShowBedrock</servlet-name>
  <servlet-class>ShowBedrock</servlet-class>
</servlet>
```

and

```
<servlet-mapping>
  <servlet-name>ShowBedrock</servlet-name>
  <url-pattern>/servlet/ShowBedrock</url-pattern>
</servlet-mapping>
```

All that remains is to visit the servlet by going to:

<http://24.19.52.157:8080/servlets-examples/servlet/ShowBedrock>

## Linux Server Environment

Post the following file to:

`/etc/profile.local`

```
JAVA_HOME=/opt/j2sdk1.4.2_07
export JAVA_HOME
CATALINA_HOME=/opt/jakarta-tomcat-5.5.7
export CATALINA_HOME
CLASSPATH=/opt/jakarta-tomcat-5.5.7/server/lib:/opt/jakarta-tomcat-5.5.7/common/lib:/opt/jakarta-tomcat-5.5.7/shared/lib
export CLASSPATH
PATH=$PATH:/opt/jakarta-tomcat-5.5.7/bin
export PATH
```

## **NetBeans Integrated Development Environment**

This is not a server, but rather a tool installed on a separate computer for developing Servlets and Java Server Pages (JSPs).

## Telnet FTP SSH feature Appendix

### 6.4. FTP or Telnet Server Won't Allow Logins.

This applies to server daemons that respond to clients, but don't allow logins. On new systems that have Pluggable Authentication Modules installed, look for a file named, "ftp," or "telnet," in the directory /etc/pam/ or /etc/pam.d/. If the corresponding authentication file doesn't exist, the instructions for configuring FTP and Telnet authentication and other PAM configuration, should be in /usr/doc/pam-<version>. Refer also to the answer for "FTP server says: "421 service not available, remote server has closed connection."."

If it's an FTP server on an older system, make sure that the account exists in /etc/passwd, especially "anonymous."

This type of problem may also be caused a failure to resolve the host addresses properly, especially if using Reverse Address Resolution Protocol (RARP). The simple answer to this is to list all relevant host names and IP addresses in the /etc/hosts files on each machine. (Refer to the example /etc/hosts and /etc/resolv.conf files in: "Sendmail Pauses for Up to a Minute at Each Command..") If the network has an internal DNS, make sure that each host can resolve network addresses using it.

If the host machine doesn't respond to FTP or Telnet clients at all, then the server daemon is not installed correctly, or at all. Refer to the manual pages: inetd and inetd.conf on older systems, or xinetd and xinetd.conf, as well as ftpd, and telnetd.

Upon inspection of /etc/host.deny, it appears that no one is allowed to ftp or telnet as a root account (much too common a login and too much power).

In Googlegroups use the query

"connection refused" telnet linux suse

some suggested looking into /etc/hosts.deny hosts.allow ...

from : <http://support.novell.com/>

#### ✦ symptom

When you try to establish a connection e.g. to your own computer via telnet with:

```
telnet localhost
```

the connection interrupts and you obtain the following error message:

```
Trying ::1...
```

```
telnet: connect to address ::1: Connection refused
```

```
Trying 127.0.0.1...
```

```
telnet: connect to address 127.0.0.1: Connection refused
```

#### ✦ cause

As default setting and due to security reasons, most of the network services are disabled. The central network daemon, `inetd`, which activates the telnet service if required, is also deactivated.

#### ✦ solutions

Activate `inetd`. You can do this by introducing as `root`

```
rcinetd start
```

The system must be told to automatically start `inetd` the next time it starts. This can be done for instance in YaST2:

```
YaST2-->System-->RC.Config-Editor-->Start-Variables-->Start-Network-->start_inetd
```

Once there, set the variable `START_INETD` to `yes`.

#### ✦ note

##### **Tip:**

Please note that all data transmitted via `telnet` are unencrypted and thus can be easily tapped. Even passwords that are not displayed on the screen can be found out during network transmissions. Therefore, we recommend you to use `ssh` instead of `telnet`, since in the former *all* data are encrypted. As a standard, `ssh` is activated in SuSE Linux.

## Apache Appendix

Ported xampp-linux-1.4.12.tar.gz to the Linux box per simple instructions which follow for completeness' sake:

1. Obtain latest copy of XAMPP from [sourceforge.net](http://sourceforge.net);
2. Extract the downloaded archive file to /opt:  

```
tar xvfz xampp-linux-1.4.12.tar.gz -C /opt
```
3. Start  

```
/opt/lampp/lampp start
```

You should now see something like this on your screen:

```
Starting XAMPP 1.4.12...
LAMPP: Starting Apache...
LAMPP: Starting MySQL...
LAMPP started.
```

4. Test  
Point a browser at the server (<http://67.171.24.169>) and confirm web page is served. Proceed to configure security.

### After I rebooted my Linux box XAMPP stopped running! How can I fix this?

Correct. That's normal Linux behaviour (which applies to any other Unix-like system. It's the admin's job to make sure a particular application is started at bootup.

There is no real standard way to configure the boot process of a Linux system, but most of them should allow you to start XAMPP at boot time using the following steps.

1. First, find out your default runlevel.  
Simply type **egrep :initdefault: /etc/inittab**.  
You should no see a line containing a number between two colons.  
In most cases 3 or 5 (2 if you're using Debian).  
**id:5:initdefault: shows up for SuSE Linux**
2. Go into the directory which configures this runlevel. If for example your runlevel is 3, then you have to change into the /etc/rc.d/rc3.d directory.  
**cd /etc/rc.d/rc5.d for SuSE Linux**
3. Now carry out the actual configuration by typing:  
**ln -s /opt/lampp/lampp S99lampp**  
**ln -s /opt/lampp/lampp K01lampp**

Now XAMPP should start and stop automatically if you boot or shutdown your machine.

In file /opt/lampp/lampp in the following snippet add **killproc httpd** at begin as shown:  
"startapache")

```
killproc httpd
```

```

if testrun /opt/lampp/logs/httpd.pid httpd
then
    $de && echo "XAMPP: XAMPP-Apache laeuft bereits."
    $de || echo "XAMPP: XAMPP-Apache is already running."
else
    if testport 80
    then
        $de && echo "XAMPP: Ein anderer Webserver laeuft bereits."
        $de || echo "XAMPP: Another web server daemon is already running."

```

**Make sure that the write permissions are 755 in the file changed (lampp).**

### **How can I make my XAMPP installation more secure?**

In the default installation, XAMPP has no passwords set and it is not recommended to run XAMPP with this configuration accessible by others (e. g. on the Internet).

Simply type the following command (as root) to start a simple security check:

```
/opt/lampp/lampp security
```

Now you should see the following dialog on your screen (user input is highlighted):

```

LAMPP: Quick security check...
LAMPP: Your LAMPP pages are NOT secured by a password.
LAMPP: Do you want to set a password? [yes] yes (1)
LAMPP: Password: *****
LAMPP: Password (again): *****
LAMPP: Password protection active. Please use 'lampp' as
user name!
LAMPP: MySQL is accessable via network.
LAMPP: Normaly that's not recommended. Do you want me to
turn it off? [yes] yes
LAMPP: Turned off.
LAMPP: Stopping MySQL...
LAMPP: Starting MySQL...
LAMPP: The MySQL/phpMyAdmin user pma has no password set!!!
LAMPP: Do you want to set a password? [yes] yes
LAMPP: Password: *****
LAMPP: Password (again): *****
LAMPP: Setting new MySQL pma password.
LAMPP: Setting phpMyAdmin's pma password to the new one.
LAMPP: MySQL has no root passwort set!!!
LAMPP: Do you want to set a password? [yes] yes
LAMPP: Write the passworde somewhere down to make sure you
won't forget it!!!
LAMPP: Password: *****
LAMPP: Password (again): *****

```

```
LAMPP: Setting new MySQL root password.
LAMPP: Setting phpMyAdmin's root password to the new one.
LAMPP: The FTP password is still set to 'lampp'.
LAMPP: Do you want to change the password? [yes] yes
LAMPP: Password: *****
LAMPP: Password (again): *****
LAMPP: Reload ProFTPD...
LAMPP: Done.
```

(1) Setting a password will protect the XAMPP demo pages (<http://localhost/xampp/>) using this password. The user name is 'lampp'!

After calling this command your XAMPP installation should be "secure". For my part I've no idea what else could be insecure.

eAccelerator is a php cache of compiled calls. To activate eAccelerator simply find the following lines in your **/opt/lampp/etc/php.ini**:

```
;extension="eaccelerator.so"
;eaccelerator.shm_size="16"
;eaccelerator.cache_dir="/opt/lampp/tmp/eaccelerator"
;eaccelerator.enable="1"
;eaccelerator.optimizer="1"
;eaccelerator.check_mtime="1"
;eaccelerator.debug="0"
;eaccelerator.filter=""
;eaccelerator.shm_max="0"
;eaccelerator.shm_ttl="0"
;eaccelerator.shm_prune_period="0"
;eaccelerator.shm_only="0"
;eaccelerator.compress="1"
;eaccelerator.compress_level="9"
```

Remove the semicolon at the beginning of each line and restart XAMPP. eAccelerator is now active. For more information about eAccelerator, check the eAccelerator home page: <http://eaccelerator.net>.

To activate the OCI8/Oracle extension for PHP please execute the following command:

```
/opt/lampp/lampp oci8
```

The following dialog will start:

```
Please enter the path to your Oracle installation:
ORA_HOME [/opt/oracle/OraHome1]
installing symlinks...
patching php.ini...
OCI8 add-on activation likely successful.
LAMPP: Stopping Apache with SSL...
LAMPP: Starting Apache with SSL...
```

Now the extension should be active. I have only had a few chances to test this feature, so

please report if this worked for you or not: [oswald@apachefriends.org](mailto:oswald@apachefriends.org).

## Server Parameter Appendix

IMPORTANT FILES AND DIRECTORIES	
File/Directory	Purpose
/opt/lampp/bin/	The XAMPP commands home. /opt/lampp/bin/mysql for example, calls the MySQL monitor.
/opt/lampp/htdocs/	The Apache DocumentRoot directory.
/opt/lampp/etc/httpd.conf	The Apache configuration file.
/opt/lampp/etc/my.cnf	The MySQL configuration file.
/opt/lampp/etc/php.ini	The PHP configuration file.
/opt/lampp/etc/proftpd.conf	The ProFTPD configuration file.
/opt/lampp/phpmyadmin/config.inc.php	The phpMyAdmin configuration file.

### START AND STOP PARAMETERS

Parameter	Description
start	Starts XAMPP.
stop	Stops XAMPP.
restart	Stops and starts XAMPP.
startapache	Starts only the Apache.
startssl	Starts the Apache SSL support. This command activates the SSL support permanently, e.g. if you restarts XAMPP in the future SSL will stay activated.
startmysql	Starts only the MySQL database.
startftp	Starts the ProFTPD server. Via FTP you can upload files for your web server (user "nobody", password "lampp"). This command activates the ProFTPD permanently, e.g. if you restarts XAMPP in the future FTP will stay activated.
stopapache	Stops the Apache.
stopssl	Stops the Apache SSL support. This command deactivates the SSL support permanently, e.g. if you restarts XAMPP in the future SSL will stay deactivated.

`stopmysql` Stops the MySQL database.  
`stopftp` Stops the ProFTPD server. This command deactivates the ProFTPD permanently, e.g. if you restarts XAMPP in the future FTP will stay deactivated.  
`security` Starts a small security check programm.

For example: To start Apache with SSL support simply type in the following command (as root):

```
/opt/lampp/lampp startssl
```

You can also access your Apache server via SSL under `https://localhost`.

## Uninstall

To uninstall XAMPP just type in this command:

```
rm -rf /opt/lampp
```

## Java Appendix

Consult: <http://java.sun.com/j2se/1.4.2/install-linux.html> which offers much the same discussion that follows.

From: <http://java.sun.com/j2se/1.5.0/jre/install-linux.html>

**1. Download and check the download file size** to ensure that you have downloaded the full, uncorrupted software bundle.

You can download to any directory you choose; it does not have to be the directory where you want to install the J2SE Runtime Environment.

Before you download the file, notice its byte size provided on the download page on the web site. Once the download has completed, compare that file size to the size of the downloaded file to make sure they are equal.

**2. Make sure that execute permissions are set** on the self-extracting binary.

Run this command:

```
chmod +x jre-1_5_0_<version>-linux-i586.bin
```

**3. Change directory** to the location where you would like the files to be installed.

The next step installs the J2SE Runtime Environment into the current directory.

**4. Run the self-extracting binary.**

Execute the downloaded file, prepended by the path to it. For example, if the

file is in the current directory, prepend it with `./` (necessary if `.` is not in the PATH environment variable):

```
./jre-1_5_0_<version>-linux-i586.bin
```

The binary code license is displayed, and you are prompted to agree to its terms.

The J2SE Runtime Environment files are installed in a directory called `jre1.5.0_<version>` in the current directory. Follow this link to see its [directory structure](#).

**Note about Root Access:** Unbundling the software automatically creates a directory called `jre1.5.0_<version>`. Note that if you choose to install the J2SE Runtime Environment into system-wide location such as `/usr/local`, you must first become root to gain the necessary permissions. If you do not have root access, simply install the J2SE Runtime Environment into your home directory, or a subdirectory that you have permission to write to.

**Note about Overwriting Files:** If you unpack the software in a directory that contains a subdirectory named `jre1.5.0_<version>`, the new software overwrites files of the same name in that `jre1.5.0_<version>` directory. Please be careful to rename the old directory if it contains files you would like to keep.

**Note about System Preferences:** By default, the installation script configures the system such that the backing store for system preferences is created inside the J2SE Runtime Environment's installation directory. If the JRE is installed on a network-mounted drive, it and the system preferences can be exported for sharing with Java runtime environments on other machines. As an alternative, root users can use the `-localinstall` option when running the installation script, as in this example:

```
jre-1_5_0_<version>-linux-i586.bin -localinstall
```

This option causes the system preferences to be stored in the `/etc` directory from which they can be shared only by VMs running on the local machine. You must be root user for the `-localinstall` option to work.

See the [Preferences API](#) documentation for more information about preferences in the Java platform.

## Tomcat Appendix

As always, consult readme files such as RELEASE-NOTES.txt that comes with the distribution. This is particularly significant for Java implementation issues.

What follows is Linux discussion that as of yet has not been edited for Win2K application.

From: <http://www.yolinux.com/TUTORIALS/LinuxTutorialTomcat.html>

Config files:

- `/opt/jakarta-tomcat-5.5.7/conf/server.xml` - Servlet container configuration: Defines services offered, TCP port numbers for services, SSL, JDBC configuration for MySQL (user/passwd), PostgreSQL, Oracle, ODBC, etc. Later we will configure Apache to use one of these services.
- `/opt/jakarta-tomcat-5.5.7/conf/web.xml` - Tomcat's built-in http server **global** config file: Defines servlets to be processed by Tomcat. Some are predefined to perform pre-configured tasks like SSI, JSP, etc.
- `/opt/jakarta-tomcat-5.5.7/conf/catalina.policy` - Security policy permissions for Tomcat
- `/var/tomcat4/webapps/examples/WEB-INF/web.xml` - **Application specific** config file.

From: <http://jakarta.apache.org/tomcat/tomcat-5.5-doc/setup.html>

Tomcat can be run as a daemon using the `jsvc` tool from the commons-daemon project. Source tarballs for `jsvc` are included with the Tomcat binaries, and need to be compiled. Building `jsvc` requires a C ANSI compiler (such as GCC), GNU Autoconf, and a JDK.

Before running the script, the `JAVA_HOME` environment variable should be set to the base path of the JDK. Alternately, when calling the `./configure` script, the path of the JDK may be specified using the `--with-java` parameter, such as `./configure --with-java=/usr/java`.

Using the following commands should result in a compiled `jsvc` binary, located in the `$CATALINA_HOME/bin` folder. This assumes that GNU TAR is used, and that `CATALINA_HOME` is an environment variable pointing to the base path of the Tomcat installation.

Please note that you should use the GNU make (`gmake`) instead of the native BSD make on FreeBSD systems.

```
cd $CATALINA_HOME/bin
```

```
tar xvfz jsvc.tar.gz
cd jsvc-src
autoconf
./configure
make
cp jsvc ..
cd ..
```

Tomcat can then be run as a daemon using the following commands.

```
cd $CATALINA_HOME
./bin/jsvc -Djava.endorsed.dirs=../common/endorsed -cp ../bin/bootstrap.jar \
-outfile ../logs/catalina.out -errfile ../logs/catalina.err \
org.apache.catalina.startup.Bootstrap
```

jsvc has other useful parameters, such as `-user` which causes it to switch to another user after the daemon initialization is complete. This allows, for example, running Tomcat as a non privileged user while still being able to use privileged ports. `jsvc --help` will return the full jsvc usage information. In particular, the `-debug` option is useful to debug issues running jsvc.

The file `$CATALINA_HOME/bin/jsvc/native/tomcat.sh` can be used as a template for starting Tomcat automatically at boot time from `/etc/init.d`. The file is currently setup for running Tomcat 4.1.x, so it is necessary to edit it and change the classname from `BootstrapService` to `Bootstrap`.

Note that the Commons-Daemon JAR file must be on your runtime classpath to run Tomcat in this manner. The Commons-Daemon JAR file is in the `Class-Path` entry of the `bootstrap.jar` manifest, but if you get a `ClassNotFoundException` or a `NoClassDefFoundError` for a Commons-Daemon class, add the Commons-Daemon JAR to the `-cp` argument when launching jsvc.

Where options include:

`-jvm <JVM name>`

use a specific Java Virtual Machine. Available JVMs:

`-cp / -classpath <directories and zip/jar files>`

set search path for service classes and resources

`-home <directory>`

set the path of your JDK or JRE installation (or set the `JAVA_HOME` environment variable)

-version  
show the current Java environment version (to check correctness of -home and -jvm. Implies -nodetach)

-help / -?  
show this help page (implies -nodetach)

-nodetach  
don't detach from parent process and become a daemon

-debug  
verbosely print debugging information

-check  
only check service (implies -nodetach)

-user  
user used to run the daemon (defaults to current user)

-verbose[:class|gc|jni]  
enable verbose output

-outfile </full/path/to/file>  
Location for output from stdout (defaults to /dev/null)  
Use the value '&2' to simulate '1>&2'

-errfile </full/path/to/file>  
Location for output from stderr (defaults to /dev/null)  
Use the value '&1' to simulate '2>&1'

-pidfile </full/path/to/file>  
Location for output from the file containing the pid of jsvc  
(defaults to /var/run/jsvc.pid)

-D<name>=<value>  
set a Java system property

-X<option>  
set Virtual Machine specific option

## Sample Servlet Application Construction

From: <http://www.javaworld.com/javaworld/jw-02-2001/jw-0223-servletweblogic.html>

## Web application directory structure

The Servlet 2.2 specifications define the directory structure of the files in a Web application. The top directory -- or root directory -- should be given the name of your Web application and will define that *document root* for your Web application. All files beneath this root can be served to the client except for files under the special directories META-INF and WEB-INF in the root directory. All private files -- such as servlet class files -- should be stored under the WEB-INF directory.

The directory structure of your Web application should look something like that shown in Figure 1.

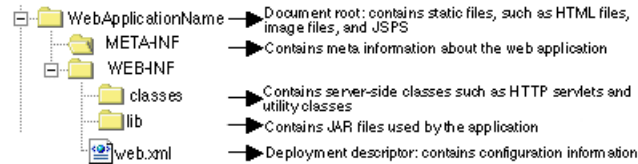


Figure 1. Web application directory structure

To create a Web application, begin by creating this directory structure. Take your compiled servlet classes and place them in the WEB-INF/classes directory. If you have defined your servlet to belong in a package, you must follow the standard Java rules and create the appropriate subdirectories so the JVM will be able to find your classes. For example, if your servlet is defined in a package `com.mycompany.myproject`, you should create the following directory structure:

```
.../WEB-INF
|-- classes
    |-- com
        |-- mycompany
            |-- myproject
```

Place your Java classes in the myproject subdirectory.

## Modify the deployment descriptor

You should now have all of your files in place to create your first Web application. At that point, one other task needs to be done: update the deployment descriptor to register your servlets with the servlet container. To easily create a deployment descriptor, simply edit an existing one. A skeletal `web.xml` file is given below:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<!DOCTYPE web-app
```

```
PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
"http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
```

```
<web-app>
```

```
<!-- Your servlet definitions go here -->
```

```
</web-app>
```

You insert your servlet deployment descriptors between the `<web-app>` and `</web-app>` tags in this file. Servlet deployment descriptors must include the following tags (in this order):

```
<servlet>
  <servlet-name>name</servlet-name>
  <servlet-class>package.name.MyClass</servlet-class>
</servlet>
```

Various optional tags, which define the servlet's other runtime properties, are allowed before the closing `</servlet>`. Those tags define properties such as initialization parameters, whether the servlet should be loaded at startup, security roles, and display properties (including small and large icons, a display name, and a description). Refer to the specifications for more details on those.

So far, the deployment descriptors describe the servlet to the servlet container. Next, we must describe when the servlet container should invoke the servlet -- referred to as *mapping*. In other words, we must describe how to map a URL to a servlet. In the web.xml file URL, mapping follows this form:

```
<servlet-mapping>
  <servlet-name>name</servlet-name>
  <url-pattern>pattern</url-pattern>
</servlet-mapping>
```

OK, enough theory. Let's look at an example of a real Web application deployment descriptor. Below you'll find a minimal web.xml file describing our sample RequestDetails servlet:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<!DOCTYPE web-app
PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
"http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
```

```
<web-app>
```

```
<servlet>
  <servlet-name>RequestDetails</servlet-name>
  <servlet-class>org.stevengould.javaworld.RequestDetails</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>RequestDetails</servlet-name>
  <url-pattern>SampleServlet</url-pattern>
</servlet-mapping>

</web-app>
```

As you can see from the servlet mapping tags, we have chosen to map our RequestDetails servlet to the URL /SampleServlet.

That's it. We have created our first Web application containing a single servlet. We should now be able to deploy that Web application to any Servlet 2.2-compliant servlet container.

More than likely, that will be how you work with and deploy your Web applications in a development mode. In a production environment, however, keeping related files bundled together is more convenient. In the next section, we look at creating Web application archive files (WARs) that do just that.

### **Create WARs**

As mentioned earlier, a WAR file is simply a Java archive with the extension changed to reflect its different purpose. We have already seen the directory structure required by a Web application. To create a WAR file, we use that same directory structure.

To create a WAR for your Web application, go to the root directory containing your Web application and type the following command:

```
jar cvOf myWebApp.war .
```

Note the required period at the end of the above line; it tells the jar program to archive the current directory.

The aforementioned jar command will create a WAR file called myWebApp.war. Next, we shall look at how to deploy that WAR file in both Tomcat 3.2 and WebLogic Server 6.0.

## **Sample Servlet Application Deployment**

To deploy your Web application in Tomcat, copy your root Web application directory -- the one containing web.xml and its subdirectories -- into the webapps/ROOT/ subdirectory of your Tomcat installation. You may want to save a copy of the default Web application before overwriting it.

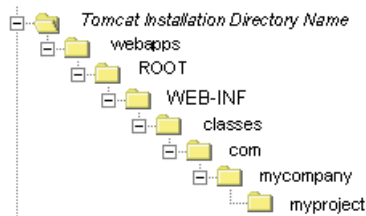
Under Unix, for example, if you installed Tomcat into the directory `/opt/jakarta-tomcat-3.2.1`, you would copy your servlet classes into the directory beneath:

`/opt/jakarta-tomcat-3.2.1/webapps/ROOT/`

If you run Tomcat under Windows and installed Tomcat into the `C:\Program Files\Jakarta-Tomcat-3.2.1` directory, you would copy your servlet classes into the directory beneath:

`C:\Program Files\Jakarta-Tomcat-3.2.1\webapps\ROOT\`

The `webapps/ROOT/WEB-INF/classes` subdirectory is the default directory in which Tomcat will look for your Java classes. If you have defined your servlet to belong in a package, you must follow the standard Java rules and create the appropriate subdirectories so that the JVM will be able to find your classes, as we did before. For example, if you defined your servlet in a package `com.mycompany.myproject`, then you should have the directory structure shown in Figure 2.



**Figure 2. Package layout in Tomcat**

Your Java servlet classes will then be in the `myproject` subdirectory.

That is all that is involved. There is no further configuration. Keeping with the `RequestDetails` example, try copying the Web application files into Tomcat's default Web application.

## Sample Servlet Application Testing

### Test the servlet

To test your servlet, start the Tomcat server, open a Web browser, and open a URL of the following form:

`http://{address}:{port}/{servletName}`

where:

- `address` is the name or IP address of the machine running Tomcat. You can use `localhost` if the browser is running on the same machine as Tomcat.
- `port` is the port on which Tomcat is listening. By default, that is port 8080.
- `servletName` is the name of the servlet you want to invoke. That should match the value contained in the `<url-pattern></url-pattern>` tags in the `web.xml` deployment

descriptor file.

For example, if Tomcat is running on the same machine as the browser and listening on the default port (8080), you can test your RequestDetails sample servlet (which is mapped to the URL SampleServlet) by opening the following URL:

`http://localhost:8080/SampleServlet`

Notice how little work was involved deploying the Web application. Copy some files and test. The ease of use is made possible by the Java Servlet 2.2 specifications and the use of the deployment descriptors.

## Sample JSP Application Testing

Basic JSP elements:

Define/execute Java statements

```
<jsp: ... %>
```

Example:

```
Year: is <jsp:getProperty name="clock" property="year"/>
```

```
<BR>
```

```
Month: is <jsp:getProperty name="clock" property="month"/>
```

Execute Java statements. Nothing displayed.

```
<% ... %> :
```

Example

```
<% numguess.reset(); %>
```

Execute Java expression and place results here

```
<%= ... %>
```

Example

```
Calendar:<%= table.getDate() %>
```

Declare Java variable or method

```
<%@ ... %>
```

Example:

```
<%@ page language="java" import="cal.*" %>
<jsp:useBean id="table" scope="session" class="cal.TableBean"
/>
```

JSP's are rarely self contained. JSP's most often require use of classes and methods defined in Java programs.

The samples delivered with Tomcat show numerous JSP examples and the source code

Test the JSP

from: <http://www.planet-source-code.com/vb/scripts/ShowCode.asp?txtCodeId=2162&lngWId=2>

## File ShowOrganization.jsp

```
<%@ page import="java.sql.*" %>
<%
String connectionURL =
"jdbc:mysql://localhost:3306/mydatabase?user=myname;password=mypassword";
Connection connection = null;
Statement statement = null;
ResultSet rs = null;
%>

<html><body>

<%
Class.forName("com.mysql.jdbc.Driver").newInstance();
connection = DriverManager.getConnection(connectionURL, "", "");
statement = connection.createStatement();
rs = statement.executeQuery("SELECT * FROM mytable");

while (rs.next()) {
out.println(rs.getString("myfield")+"<br>");
}

rs.close();
%>

</body></html>
```

---

from: [http://www.onjava.com/pub/a/onjava/2003/06/25/tomcat\\_tips.html](http://www.onjava.com/pub/a/onjava/2003/06/25/tomcat_tips.html)

## Top Ten Tomcat Configuration Tips

### 1. Configuring the Admin Web Application

Most commercial J2EE servers provide a fully functional administrative interface, and many of these are accessible as web applications. The Tomcat Admin application is on its way to becoming a full-blown Tomcat administration tool rivaling these commercial offerings. First included in Tomcat 4.1, Admin already provides control over contexts, data sources, and users and groups. You can also control resources such as initialization parameters, as well as users, groups, and roles in a variety of user databases. The list of capabilities will be expanded upon in future releases, but the present implementation has proven itself to be quite useful.

The Admin web application is defined in the auto-deployment file *CATALINA\_BASE/webapps/admin.xml*.

You must edit this file to ensure that the path specified in the docBase attribute of the Context element is absolute; that is, the absolute path of *CATALINA\_HOME/server/webapps/admin*. Alternatively, you could just remove the auto-deployment file and specify the Admin context manually in your *server.xml* file. On machines that will not be managed by this application, you should probably disable it altogether by simply removing *CATALINA\_BASE/webapps/admin.xml*.

If you're using a `UserDatabaseRealm` (the default), you'll need to add a user and a role to the *CATALINA\_BASE/conf/tomcat-users.xml* file. For now, just edit this file, and add a role named "admin" to your users database:

```
<role name="admin"/>
```

You must also have a user who is assigned to the "admin" role. Add a user line like this after the existing user entries (changing the password to something a bit more secure):

```
<user name="admin" password="deep_dark_secret"
roles="admin"/>
```

Once you've performed these steps and restarted Tomcat, visit the URL *http://localhost:8080/admin*, and you should see a login screen. The Admin application is built using container-managed security and the Jakarta Struts framework. Once you have logged in as a user assigned to the admin role, you will be able to use the Admin application to configure Tomcat.

## 2. Configuring the Manager Web Application

The Manager web application lets you perform simple management tasks on your web applications through a more simplified web user interface than that of the Admin web app.

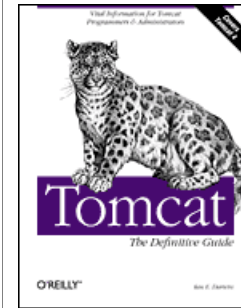
The Manager web application is defined in the auto-deployment file *CATALINA\_BASE/webapps/manager.xml*.

You must edit this file to ensure that the path specified in the docBase attribute of the Context element is absolute; that is, the absolute path of *CATALINA\_HOME/server/webapps/manager*.

If you're using the default `UserDatabaseRealm`, you'll need to add a user and role to the *CATALINA\_BASE/conf/tomcat-users.xml* file. For now, just edit this file, and add a role named "manager" to your users database:

```
<role name="manager"/>
```

### Related Reading



[Tomcat: The Definitive Guide](#)  
By [Jason Brittain](#),  
[Ian F. Darwin](#)

[Table of Contents](#)  
[Index](#)  
[Sample Chapter](#)

[Read Online--](#)  
[Safari](#) Search this  
book on Safari:

  
  
  
 Code Fragments  
only

You must also have a user who is assigned the "manager" role. Add a user line like this after the existing user entries (changing the password to something a bit more secure):

```
<user name="manager" password="deep_dark_secret" roles="manager"/>
```

Then restart Tomcat and visit the URL `http://localhost/manager/list` to see the plain-text manager interface, or `http://localhost/manager/html/list` for the simple HTML manager interface. Either way, your Manager application should now be working.

The Manager application lets you install new web applications on a non-persistent basis, for testing. If we have a web application in `/home/user/hello` and want to test it by installing it under the URI `/hello`, we put `/hello` in the first text input field (for Path) and `file:/home/user/hello` in the second text input field (for Config URL).

The Manager also allows you to stop, reload, remove, or undeploy a web application. Stopping an application makes it unavailable until further notice, but of course it can then be restarted. Users attempting to access a stopped application will receive an error message, such as *503 - This application is not currently available*.

Removing a web application removes it only from the running copy of Tomcat -- if it was started from the configuration files, it will reappear the next time you restart Tomcat (i.e., removal does not remove the web application's content from disk).

### 3. Deploying a Web Application

There are two ways of deploying a web application on the filesystem:

1. Copy your WAR file or your web application's directory (including all of its content) to the `$CATALINA_BASE/webapps` directory.
2. Create an XML fragment file with just the `Context` element for your web application, and place this XML file in `$CATALINA_BASE/webapps`. The web application itself can then be stored anywhere on your filesystem.

If you have a WAR file, you can deploy it by simply copying the WAR file into the directory `CATALINA_BASE/webapps`. The filename must end with an extension of `.war`. Once Tomcat notices the file, it will (by default) unpack it into a subdirectory with the base name of the WAR file. It will then create a context in memory, just as though you had created one by editing Tomcat's `server.xml` file. However, any necessary defaults will be obtained from the `DefaultContext` element in Tomcat's `server.xml` file.

Another way to deploy a web app is by writing a Context XML fragment file and deploying it into the `CATALINA_BASE/webapps` directory. A context fragment is not a complete XML document, but just one `Context` element and any subelements that are appropriate for your web application. These files are like `Context` elements cut out of the `server.xml` file, hence the name "context fragment."

For example, if we wanted to deploy the WAR file `MyWebApp.war` along with a realm for accessing parts of that web application, we could use this fragment:

```

<!--
  Context fragment for deploying MyWebApp.war
-->
<Context path="/demo" docBase="webapps/MyWebApp.war"
  debug="0" privileged="true">
  <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
    resourceName="UserDatabase"/>
</Context>

```

Put that in a file called "MyWebApp.xml," and copy it into your *CATALINA\_BASE/webapps* directory.

These context fragments provide a convenient method of deploying web applications; you do not need to edit the *server.xml* file and, unless you have turned off the default *liveDeploy* feature, you don't have to restart Tomcat to install a new web application.

## 4. Configuring Virtual Hosts

The *Host* element normally needs modification only when you are setting up virtual hosts. Virtual hosting is a mechanism whereby one web server process can serve multiple domain names, giving each domain the appearance of having its own server. In fact, the majority of small business web sites are implemented as virtual hosts, due to the expense of connecting a computer directly to the Internet with sufficient bandwidth to provide reasonable response times and the stability of a permanent IP address.

Name-based virtual hosting is created on any web server by establishing an aliased IP address in the Domain Name Service (DNS) data and telling the web server to map all requests destined for the aliased address to a particular directory of web pages. Since this article is about Tomcat, we don't try to show all of the ways to set up DNS data on various operating systems. If you need help with this, please refer to [DNS and Bind](#), by Paul Albitz and Cricket Liu (O'Reilly). For demonstration purposes, I'll use a static hosts file, since that's the easiest way to set up aliases for testing purposes.

To use virtual hosts in Tomcat, you just need to set up the DNS or hosts data for the host. For testing, making an IP alias for localhost is sufficient. You then need to add a few lines to the *server.xml* configuration file:

```

<Server port="8005" shutdown="SHUTDOWN" debug="0">
  <Service name="Tomcat-Standalone">
    <Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
      port="8080" minProcessors="5" maxProcessors="75"
      enableLookups="true" redirectPort="8443"/>
    <Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
      port="8443" minProcessors="5" maxProcessors="75"
      acceptCount="10" debug="0" scheme="https"
      secure="true"/>
    <Factory
      className="org.apache.coyote.tomcat4.CoyoteServerSocketFactory"
      clientAuth="false" protocol="TLS" />
  </Connector>

```

```

<Engine name="Standalone" defaultHost="localhost" debug="0">
  <!-- This Host is the default Host -->
  <Host name="localhost" debug="0" appBase="webapps"
    unpackWARs="true" autoDeploy="true">
    <Context path="" docBase="ROOT" debug="0"/>
    <Context path="/orders" docBase="/home/ian/orders" debug="0"
      reloadable="true" crossContext="true">
    </Context>
  </Host>

  <!-- This Host is the first "Virtual Host": www.example.com -->
  <Host name="www.example.com" appBase="/home/example/webapp">
    <Context path="" docBase="."/>
  </Host>

</Engine>
</Service>
</Server>

```

Tomcat's *server.xml* file, as distributed, contains only one virtual host, but it is easy to add support for additional virtual hosts. The simplified version of the *server.xml* file in the previous example shows in bold the overall additional structure needed to add one virtual host. Each `Host` element must have one or more `Context` elements within it; one of these must be the default `Context` for this host, which is specified by having its relative path set to the empty string (for example, `path=""`).

## 5. Configuring Basic Authentication

Container-managed authentication methods control how a user's credentials are verified when a web app's protected resource is accessed. When a web application uses basic authentication (BASIC in the *web.xml* file's `auth-method` element), Tomcat uses HTTP basic authentication to ask the web browser for a username and password whenever the browser requests a resource of that protected web application. With this authentication method, all passwords are sent across the network in base64-encoded text.

Note: using basic authentication is generally considered insecure because it does not strongly encrypt passwords, unless the site also uses HTTPS or some other form of encryption between the client and the server (for instance, a virtual private network). Without this extra encryption, network monitors can intercept (and misuse) users' passwords. But, if you're just starting to use Tomcat, or if you just want to test container-managed security with your web app, basic authentication is easy to set up and test. Just add `<security-constraint>` and `<login-config>` elements to your web app's *web.xml* file, and add the appropriate `<role>` and `<user>` elements to your *CATALINA\_BASE/conf/tomcat-users.xml* file, restart Tomcat, and Tomcat takes care of the rest.

The example below shows a *web.xml* excerpt from a club membership web site with a members-only subdirectory that is protected using basic authentication. Note that this effectively takes the place of the Apache web server's *.htaccess* files.

```

<!--
  Define the Members-only area, by defining
  a "Security Constraint" on this Application, and
  mapping it to the subdirectory (URL) that we want
  to restrict.
-->
<security-constraint>
  <web-resource-collection>
    <web-resource-name>
      Entire Application
    </web-resource-name>
    <url-pattern>/members/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>member</role-name>
  </auth-constraint>
</security-constraint>
<!-- Define the Login Configuration for this Application -->
<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>My Club Members-only Area</realm-name>
</login-config>

```

## 6. Configuring Single Sign-On

Once you've set up your realm and method of authentication, you'll need to deal with the actual process of logging the user in. More often than not, logging into an application is a nuisance to an end user, and you will need to minimize the number of times they must authenticate. By default, each web application will ask the user to log in the first time the user requests a protected resource. This can seem like a hassle to your users if you run multiple web applications and each application asks the user to authenticate. Users cannot tell how many separate applications make up any single web site, so they won't know when they're making a request that crosses a context boundary, and will wonder why they're being repeatedly asked to log in.



The "single sign-on" feature of Tomcat 4 allows a user to authenticate only once to access all of the web applications loaded under a virtual host. To use this feature, you need only add a `SingleSignOn Valve` element at the host level. This looks like the following:

```

<Valve className="org.apache.catalina.authenticator.SingleSignOn"
  debug="0"/>

```

The Tomcat distribution's default `server.xml` contains a commented-out single sign-on Valve configuration example that you can uncomment and use. Then, any user who is considered valid in a context within the configured virtual host will be considered valid in all other contexts for that same host.

There are several important restrictions for using the single sign-on valve:

- The valve must be configured and nested within the same `Host` element that the web applications (represented by `Context` elements) are nested within.
- The `Realm` that contains the shared user information must be configured either at the level of the same `Host` or in an outer nesting.
- The `Realm` cannot be overridden at the `Context` level.
- The web applications that use single sign-on must use one of Tomcat's built-in authenticators (in the `<auth-method>` element of *web.xml*), rather than a custom authenticator. The built-in methods are `basic`, `digest`, `form`, and `client-cert` authentication.
- If you're using single sign-on and wish to integrate another third-party web application into your web site, and the new web application uses only its own authentication code that doesn't use container-managed security, you're basically stuck. Your users will have to log in once for all of the web applications that use single sign-on, and then once again if they make a request to the new third-party web application. Of course, if you get the source and you're a developer, you could fix it, but that's probably not so easy to do.
- The single sign-on valve requires the use of HTTP cookies.

## 7. Configuring Customized User Directories

Some sites like to allow individual users to publish a directory of web pages on the server. For example, a university department might want to give each student a public area, or an ISP might make some web space available on one of its servers to customers that don't have a virtually hosted web server. In such cases, it is typical to use the tilde character (~) plus the user's name as the virtual path of that user's web site:

```
http://www.cs.myuniversity.edu/~username  
http://members.mybigisp.com/~username
```

Tomcat gives you two ways to map this on a per-host basis, using a couple of special `Listener` elements. The `Listener`'s `className` attribute should be `org.apache.catalina.startup.UserConfig`, with the `userClass` attribute specifying one of several mapping classes. If your system runs Unix, has a standard `/etc/passwd` file that is readable by the account running Tomcat, and that file specifies users' home directories, use the `PasswdUserDatabase` mapping class:

```
<Listener className="org.apache.catalina.startup.UserConfig"  
directoryName="public_html"  
userClass="org.apache.catalina.startup.PasswdUserDatabase"/>
```

Web files would need to be in directories such as `/home/users/ian/public_html` or `/users/jbrittain/public_html`. Of course, you can change `public_html` to be whatever subdirectory into which your users put their personal web pages.

In fact, the directories don't have to be inside of a user's home directory at all. If you don't have a password file but want to map from a user name to a subdirectory of a common parent directory such as */home*, use the `HomesUserDatabase` class:

```
<Listener className="org.apache.catalina.startup.UserConfig"
directoryName="public_html" homeBase="/home"
userClass="org.apache.catalina.startup.HomesUserDatabase"/>
```

In this case, web files would be in directories such as */home/ian/public\_html* or */home/jasonb/public\_html*. This format is more useful on Windows, where you'd likely use a directory such as *C:\home*.

These `Listener` elements, if present, must be inside of a `Host` element, but not inside of a `Context` element, as they apply to the `Host` itself.

## 8. Using CGI Scripts with Tomcat

Tomcat is primarily meant to be a servlet/JSP container, but it has many capabilities rivalling a traditional web server. One of these is support for the Common Gateway Interface (CGI), which provides a means for running an external program in response to a browser request, typically to process a web-based form. CGI is called "common" because it can invoke programs in almost any programming or scripting language: Perl, Python, `awk`, Unix shell scripting, and even Java are all supported options. However, you probably wouldn't run a Java application as a CGI due to the start-up overhead; elimination of this overhead was what led to the original design of the servlet specification. Servlets are almost always more efficient than CGIs because you're not starting up a new operating-system-level process every time somebody clicks on a link or button.

Tomcat includes an optional CGI servlet that allows you to run legacy CGI scripts; the assumption is that most new back-end processing will be done by user-defined servlets and JSPs.

To enable Tomcat's CGI servlet, you must do the following:

1. Rename the file *servlets-cgi.renametojar* (found in *CATALINA\_HOME/server/lib/*) to *servlets-cgi.jar*, so that the servlet that processes CGI scripts will be on Tomcat's `CLASSPATH`.
2. In Tomcat's *CATALINA\_BASE/conf/web.xml* file, uncomment the definition of the servlet named `cgi` (this is around line 241 in the distribution).
3. Also in Tomcat's *web.xml*, uncomment the servlet mapping for the `cgi` servlet (around line 299 in the distributed file). Remember, this specifies the HTML links to the CGI script.
4. Either place the CGI scripts under the *WEB-INF/cgi* directory (remember that *WEB-INF* is a safe place to hide things that you don't want the user to be able to view, for security reasons), or place them in some other directory within your context and adjust the `cgiPathPrefix` initialization parameter of the

CGIServlet to identify the directory containing the files. This specifies the actual location of the CGI scripts, which typically will not be the same as the URL in the previous step.

5. Restart Tomcat, and your CGI processing should now be operational.

The default directory for the servlet to locate the actual scripts is *WEB-INF/cgi*. As has been noted, the *WEB-INF* directory is protected against casual snooping from browsers, so this is a good place to put CGI scripts, which may contain passwords or other sensitive information. For compatibility with other servers, though, you may prefer to keep the scripts in the traditional directory, */cgi-bin*, but be aware that files in this directory may be viewable by the curious web surfer. Also, on Unix, be sure that the CGI script files are executable by the user under which you are running Tomcat.

## 9. Changing Tomcat's JSP Compiler

In Tomcat 4.1 (and above, presumably), compilation of JSPs is performed by using the Ant program controller directly from within Tomcat. This sounds a bit strange, but it's part of what Ant was intended for; there is a documented API that lets developers use Ant without starting up a new JVM. This is one advantage of having Ant written in Java. Plus, it means you can now use any compiler supported by the `javac` task within Ant; these are listed in the [javac page](#) of the Apache Ant manual. It is easy to use because you need only an `<init-param>` with a name of "compiler" and a value of one of the supported compiler names:

```
<servlet>
  <servlet-name>jsp</servlet-name>
  <servlet-class>
    org.apache.jasper.servlet.JspServlet
  </servlet-class>
  <init-param>
    <param-name>logVerbosityLevel</param-name>
    <param-value>WARNING</param-value>
  </init-param>
  <init-param>
    <param-name>compiler</param-name>
    <param-value>jikes</param-value>
  </init-param>
  <load-on-startup>3</load-on-startup>
</servlet>
```

Of course, the given compiler must be installed on your system, and the `CLASSPATH` may need to be set, depending on which compiler you choose.

## 10. Restricting Access to Specific Hosts

Sometimes you'll only want to restrict access to Tomcat's web app to only specified host names or IP addresses. This way, only clients at those specified sites will be served content. Tomcat comes with two `Valves` that you can configure and use for this purpose:

RemoteHostValve and RemoteAddrValve.

These Valves allow you to filter requests by host name or by IP address, and to allow or deny hosts that match, similar to the per-directory Allow/Deny directives in Apache httpd. If you run the Admin application, you might want to only allow access to it from localhost, as follows:

```
<Context path="/path/to/secret_files" ...>
  <Valve className="org.apache.catalina.valves.RemoteAddrValve"
    allow="127.0.0.1" deny="" />
</Context>
```

If no allow pattern is given, then patterns that match the deny attribute patterns will be rejected, and all others will be allowed. Similarly, if no deny pattern is given, patterns that match the allow attribute will be allowed, and all others will be denied.

*[Jason Brittain](#) is a Senior Software Engineer at [Symantec Corporation](#)'s Network and Gateway Security Solutions Team, working on the AntiSpam product. He has contributed to many Apache Jakarta projects, and has been an active open source software developer for several years.*

*[Ian F. Darwin](#) has worked in the computer industry for three decades: with Unix since 1980, Java since 1995, and OpenBSD since 1998. He is the author of two O'Reilly books, [Checking C Programs with lint](#) and [Java Cookbook](#), and co-author of [Tomcat: The Definitive Guide](#) with Jason Brittain.*

## Creating your first JSP page

Type the following code into a text file. Name the file helloworld.jsp.

Place this in the correct directory on your JSP web server and call it via your browser.

```
<html>
<head>
<title>My first JSP page
</title>
</head>
<body>
<%@ page language="java" %>
<% System.out.println("Hello World"); %>
</body>
```

</html>

## Using JSP tags

There are four main tags:

1. Declaration tag
2. Expression tag
3. Directive Tag
4. Scriptlet tag
5. Action tag

### Declaration tag ( <%! %> )

This tag allows the developer to declare variables or methods.

Before the declaration you must have <%!

At the end of the declaration, the developer must have %>

Code placed in this tag must end in a semicolon ( ; ).

Declarations do not generate output so are used with JSP expressions or scriptlets.

For Example,

```
<%!  
private int counter = 0 ;  
private String get Account ( int accountNo) ;  
%>
```

### Expression tag ( <%= %> )

This tag allows the developer to embed any Java expression and is short for out.println().

A semicolon ( ; ) does not appear at the end of the code inside the tag.

For example, to show the current date and time.

Date :

```
<%= new java.util.Date() %>
```

Directive tag ( <%@ directive ... %>)

A JSP directive gives special information about the page to the JSP Engine.

There are three main types of directives:

- 1) page – processing information for this page.
- 2) Include – files to be included.
- 3) Tag library – tag library to be used in this page.

Directives do not produce any visible output when the page is requested but change the way the JSP Engine processes the page.

For example, you can make session data unavailable to a page by setting a page directive (session) to false.

### 1. Page directive

This directive has 11 optional attributes that provide the JSP Engine with special processing information. The following table lists the 11 different attributes with a brief description:

language	Which language the file uses.	<%@ page language = “java” %>
extends	Superclass used by the JSP engine for the translated Servlet.	<%@ page extends = “com.taglib...” %>
import	Import all the classes in a java package into the current JSP page. This allows the JSP page to use other java classes.	<%@ page import = “java.util.*” %>
session	Does the page make use of sessions. By default all JSP pages have session data available. There are performance benefits to switching session to false.	Default is set to true.
buffer	Controls the use of buffered output for a JSP page. Default is 8kb	<%@ page buffer = “none” %>
autoFlush	Flush output buffer when full.	<%@ page autoFlush = “true” %>

	isThreadSafe Can the generated Servlet deal with multiple requests? If true a new thread is started so requests are handled simultaneously.	
info	Developer uses info attribute to add information/document for a page. Typically used to add author, version, copyright and date info.	<%@ page info = "visualbuilder.com test page, copyright 2001." %>
errorPage	Different page to deal with errors. Must be URL to error page.	<%@ page errorPage = "/error/error.jsp" %>
IsErrorPage	This flag is set to true to make a JSP page a special Error Page. This page has access to the implicit object exception (see later).	
contentType	Set the mime type and character set of the JSP.	

## 2. Include directive

Allows a JSP developer to include contents of a file inside another. Typically include files are used for navigation, tables, headers and footers that are common to multiple pages.

Two examples of using include files:

This includes the html from privacy.html found in the include directory into the current jsp page.

```
<%@ include file = "include/privacy.html" %>
```

or to include a navigation menu (jsp file) found in the current directory.

```
<%@ include file = "navigation.jsp %>
```

Include files are discussed in more detail in the later sections of this tutorial.

### 3. Tag Lib directive

A tag lib is a collection of custom tags that can be used by the page.

```
<%@ taglib uri = "tag library URI" prefix = "tag Prefix" %>
```

Custom tags were introduced in JSP 1.1 and allow JSP developers to hide complex server side code from web designers.

Scriptlet tag ( <% ... %> )

Between <% and %> tags, any valid Java code is called a Scriptlet. This code can access any variable or bean declared.

For example, to print a variable.

```
<%
```

```
String username = "visualbuilder" ;
```

```
out.println ( username ) ;
```

```
%>
```

Visualbuilder.com

Action tag

There are three main roles of action tags :

- 1) enable the use of server side Javabeans
- 2) transfer control between pages
- 3) browser independent support for applets.

Javabeans

A Javabeans is a special type of class that has a number of methods. The JSP page can call these methods so can leave most of the code in these Javabeans. For example, if you wanted to make a feedback form that automatically sent out an email. By having a JSP page with a form, when the visitor presses the submit button this sends the details to a Javabeans that sends out the email. This way there would be no code in the JSP page dealing with sending emails (JavaMail API) and your Javabeans could be used in another page (promoting reuse).

To use a Javabeans in a JSP page use the following syntax:

```
<jsp : usebean id = " .... " scope = "application" class = "com..." />
```

The following is a list of Javabean scopes:

page – valid until page completes.

request – bean instance lasts for the client request

session – bean lasts for the client session.

application – bean instance created and lasts until application ends.

Visualbuilder.com

### Creating your second JSP page

For the second example, we will make use of the different tags we have learnt. This example will declare two variables; one string used to store the name of a website and an integer called counter that displays the number of times the page has been accessed. There is also a private method declared to increment the counter. The website name and counter value are displayed.

```
<HTML>
```

```
<HEAD>
```

```
<TITLE> JSP Example 2</TITLE>
```

```
</HEAD>
```

```
<BODY> JSP Example 2
```

```
<BR>
```

```
<%!
```

```
String sitename = "visualbuilder.com";
```

```
int counter = 0;
```

```
private void increment Counter()
```

```
{
```

```
counter ++;
```

```
}
```

```
%>
```

```
Website of the day is
```

```
<%= sitename %>
```

```
<BR>
```

```
page accessed
```

```
<%= counter %>
```

```
</BODY>
```

```
</HTML>
```

Visualbuilder.com

### Implicit Objects

So far we know that the developer can create Javabeans and interact with Java objects. There are several objects that are automatically available in JSP called implicit objects.

The implicit objects are

Variable Of type

Request `Javax.servlet.http.HttpServletRequest`

Response `Javax.servlet.http.HttpServletResponse`

Out `Javax.servlet.jsp.JspWriter`

Session `Javax.servlet.http.HttpSession`

PageContent `Javax.servlet.jsp.PageContext`

Application `Javax.servlet.http.HttpServletRequest`

Config `Javax.servlet.http.HttpServletRequest`

Page `Java.lang.Object`

#### Page object

Represents the JSP page and is used to call any methods defined by the servlet class.

#### Config object

Stores the Servlet configuration data.

#### Request object

Access to information associated with a request. This object is normally used in looking up parameter values and cookies.

```
<% String devStr = request.getParameter("dev"); %>
```

```
Development language = <%= devStr %>
```

This code snippet is storing the parameter “dev” in the string devStr. The result is displayed underneath.

## MySQL Connector Appendix

re: ReadMe.TXT located in mysql-connector-java-3.1.8a.zip obtained from:  
<http://dev.mysql.com/downloads/connector/j/3.1.html>

### 1.2.1.3. Installing the Driver and Configuring the CLASSPATH

MySQL Connector/J is distributed as a .zip or .tar.gz archive containing the sources, the class files and a class-file only "binary" .jar archive named "mysql-connector-java-[version]-bin.jar". You will need to use the appropriate gui or command-line utility to un-archive the distribution (for example, WinZip for the .zip archive, and "tar" for the .tar.gz archive).

Once you have un-archived the distribution archive, you can install the driver in one of two ways: Either copy the "com" and "org" subdirectories and all of their contents to anywhere you like, and put the directory holding the "com" and "org" subdirectories in your classpath, or put mysql-connector-java-[version]-bin.jar in your classpath, either by adding the FULL path to it to your CLASSPATH environment variable, or by copying the directly specifying it with the commandline switch -cp when starting your JVM

If you are going to use the driver with the JDBC DriverManager, you would use "com.mysql.jdbc.Driver" as the class that implements java.sql.Driver.

#### Example 1.11. Setting the CLASSPATH Under UNIX

The following command works for 'csh' under UNIX:

```
$ setenv CLASSPATH /path/to/mysql-connector-java-[version]-bin.jar:$CLASSPATH
```

The above command can be added to the appropriate startup file for the login shell to make MySQL Connector/J available to all Java applications.

If you want to use MySQL Connector/J with a servlet engine or application server such as Tomcat or JBoss, you will have to read your vendor's documentation for more information on how to configure third-party class libraries, as most application servers ignore the CLASSPATH environment variable. This document does contain configuration examples for some J2EE application servers in the section named "Using Connector/J with J2EE and Other Java Frameworks".

If you are developing servlets and/or JSPs, and your application server is J2EE-compliant, you can put the driver's .jar file in the WEB-INF/lib subdirectory of your webapp, as this is a standard location for third party class libraries in J2EE web applications.

You can also use the `MysqlDataSource` or `MysqlConnectionPoolDataSource` classes in the `com.mysql.jdbc.jdbc2.optional` package, if your J2EE application server supports or requires them. The various `MysqlDataSource` classes support the following parameters (through standard "set" mutators):

- o user
  
- o password
  
- o serverName (see the previous section about fail-over hosts)
  
- o databaseName
  
- o port

## MySQL Servlet Appendix

```
// File: ShowBedrock.java
/* A servlet to display the contents of the MySQL Bedrock database */
import java.io.*;
import java.net.*;
import java.sql.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class ShowBedrock extends HttpServlet
{
    public String getServletInfo()
    {
        return "Servlet connects to MySQL database and displays result of a SELECT";
    }

    // Use http GET

    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException
    {
        String loginUser = "Dude1";
        String loginPasswd = "SuperSecret";
        String loginUrl = "jdbc:mysql://localhost:3306/bedrock";

        response.setContentType("text/html"); // Response mime type

        // Output stream to STDOUT
        PrintWriter out = response.getWriter();
```

```

out.println("<HTML><HEAD><TITLE>Bedrock</TITLE></HEAD>");
out.println("<BODY><H1>Bedrock</H1>");

// Load the mm.MySQL driver
try
{
    Class.forName("org.gjt.mm.mysql.Driver");
    Connection dbcon = DriverManager.getConnection(loginUrl, loginUser,
loginPasswd);
    // Declare our statement
    Statement statement = dbcon.createStatement();

    String query = "SELECT name, dept, ";
    query += "    jobtitle ";
    query += "FROM employee ";

    // Perform the query
    ResultSet rs = statement.executeQuery(query);

    out.println("<TABLE border>");

    // Iterate through each row of rs
    while (rs.next())
    {
        String m_name = rs.getString("name");
        String m_dept = rs.getString("dept");
        String m_jobtitle = rs.getString("jobtitle");
        out.println("<tr> " +
            "<td>" + m_name + "</td>" +
            "<td>" + m_dept + "</td>" +
            "<td>" + m_jobtitle + "</td>" +

```

```

        "</tr>");
    }

    out.println("</TABLE>");

    rs.close();
    statement.close();
    dbcon.close();
}
catch (SQLException ex) {
    while (ex != null) {
        System.out.println ("SQL Exception: " + ex.getMessage ());
        ex = ex.getNextException ();
    } // end while
} // end catch SQLException

catch(java.lang.Exception ex)
{
    out.println("<HTML>" +
        "<HEAD><TITLE>" +
        "Bedrock: Error" +
        "</TITLE></HEAD>\n<BODY>" +
        "<P>SQL error in doGet: " +
        ex.getMessage() + "</P></BODY></HTML>");
    return;
}
out.close();
}
}

```