

Fedora Linux Server Installation for a National Political Machine on a Laptop

By R. W. Clark
copyright 2005

Table of Contents

Introduction.....	4
Introducing Fedora Linux.....	4
Installing Fedora Linux.....	4
Testing Fedora Linux for Connectivity.....	6
Introducing XAMPP, an Expanded Apache Server.....	6
Installing XAMPP.....	6
Testing XAMPP.....	7
Configuring XAMPP.....	7
Setting XAMPP Bootstrap.....	8
Introducing Java for a JSP Server	9
Installing Java.....	9
Configuring Java.....	9
Testing Java.....	9
Introducing GNU m4.....	10
Installing GNU m4.....	10
Configuring GNU m4.....	10
Testing GNU m4.....	10
Introducing autoconf.....	10
Installing autoconf.....	10
Configuring autoconf.....	11
Testing autoconf.....	11
Introducing GCC, Gnu C Compiler.....	11
Installing GCC.....	11
Configuring GCC.....	11
Testing GCC.....	11
Introducing Tomcat, the JSP Server.....	11
Installing Tomcat.....	11
Configuring Tomcat.....	12
Testing Tomcat.....	12
Testing Tomcat Servlets.....	13
Testing Tomcat JSPs.....	14
Setting Tomcat Bootstrap.....	15
Setting Tomcat Permissions.....	16
How do I override the default home page loaded by Tomcat?.....	16
MySQL as it is delivered with XAMPP.....	17
MySQL User Privileges.....	17
MySQL 4.1.10.....	18
phpMyAdmin 2.6.1.....	18
MySQL connector installation.....	18
MySQL Servlet Connection Testing.....	18
Linux Server Environment.....	19
NetBeans Integrated Development Environment.....	20

Fedora Appendix.....	21
Linux Kernel using RedHat's Open Source "Fedora Core 1".....	21
Linux Grub Appendix.....	26
What is a boot loader?.....	26
What is GRUB?.....	26
Telnet FTP SSH feature Appendix.....	28
Apache Appendix.....	30
After I rebooted my Linux box XAMPP stopped running! How can I fix this?.....	30
How can I make my XAMPP installation more secure?.....	31
XAMPP Directories and Commands.....	33
What is where?.....	34
Stopping XAMPP.....	35
Uninstall.....	35
Server Parameter Appendix.....	36
Uninstall.....	37
Java Appendix.....	38
Tomcat Appendix.....	40
Sample Servlet Application Construction.....	42
Sample Servlet Application Deployment.....	45
Sample Servlet Application Testing.....	46
Sample JSP Application Testing.....	47
Top Ten Tomcat Configuration Tips.....	48
1. Configuring the Admin Web Application.....	48
2. Configuring the Manager Web Application.....	49
3. Deploying a Web Application.....	50
4. Configuring Virtual Hosts.....	51
5. Configuring Basic Authentication.....	52
6. Configuring Single Sign-On.....	53
7. Configuring Customized User Directories.....	54
8. Using CGI Scripts with Tomcat.....	55
9. Changing Tomcat's JSP Compiler.....	56
10. Restricting Access to Specific Hosts.....	57
JSP Syntax Appendix.....	57
MySQL Connector Appendix.....	62
MySQL Servlet Appendix.....	65

Introduction

Linux Server Installation is a from the ground up description of building a generic Linux server. It is meant to be a headless system which means, no keyboard, no display following the initial boot portion of building the operating system. Certainly, during this interval, keyboard and display are necessary; but installation of the servers can be accomplished remotely and these items disconnected and the box parked in a closet or remote corner. This is its intended usage as it will have no other purpose such as acting as a workstation.

This document is written in a progression of linear development. The basic components are introduced and added to with packages of increasing sophistication and utility. As such, the reader of this document should take this into consideration when contemplating whether or not to skip installations or to take them out of this order. In some circumstances this can be accomplished, and in others there are sure to be orders of precedence.

Introducing Fedora Linux

Fedora is the freely available version of Linux found at redhat.com. This “project” is supported only at the web pages offered by Red Hat, otherwise you need to purchase their package of Linux that is professionally supported.

Obtain Red Hat Fedora binary download FC4-i386-disc1.iso, FC4-i386-disc2.iso, FC4-i386-disc3.iso, and FC4-i386-disc4.iso.

Burned each image to a CDROM as prompted by the naming convention, and labeled the Disks 1 – 4.

Installing Fedora Linux

This is referenced to a PentiumIII based PC w/128MBytes of RAM and 20GBytes of disc storage. This box is a headless system that arrived without any operating system (actually, it had a hard drive partitioned for Win95 to be bootable, but otherwise had no OS).

On reboot of the target system, went into configuration to set CDROM as bootable and first to be searched.

Placed Disk 1 CDROM in drive and rebooted.

There is an Internet based guide for installing Fedora Linux located at:

<http://fedora.redhat.com/docs/fedora-install-guide-en/fc4/>

As for a event linear log of actions, when prompted:

select the text mode rather than the GUI mode of installation;

select appropriate OK to welcome, language, and country screens;

select the server configuration from the list of options

A Fedora Core system has at least three partitions:

A data partition mounted at `/boot`

A data partition mounted at `/`

A swap partition

Make the swap partition 2 – 4 times the size of the physical memory size.

Make the `/boot` partition type `ext3`, 100MB, Primary Partition.

Make the `/` partition type `ext3`, remaining memory.

For the disk partitioning:

select Disk Druid;

remove all partitions;

edit “free space” [will be `/dev/hda/hda2`]

mount at will be <not applicable> with next step:

scroll down (rather than tabbing) to swap;

select 2 – 4 times RAM for fixed sized;

select OK

edit remaining “free space” [will be `/dev/hda/hda1`]

mount at `/boot`

scroll down (rather than tabbing) to `ext3`;

select 100MB for fixed sized;

forcing as primary partition;

select OK

edit remaining “free space” [will be `/dev/hda/hda3`]

mount at `/`

scroll down (rather than tabbing) to `ext3`;

select fill all available space;

select OK

select Grub boot loader and accept all defaults from promptings that follow (no parameters to be passed);

enter Grub boot loader password;

accept default boot loader configuration (there is only one OS);

select Master Boot Record, `mbr`, for installation of boot loader;

accept defaults for Network Configuration for `eth0`;

accept defaults for Hostname Configuration (DHCP);

disable Firewall and with the warning screen select proceed;

disable Security Enhanced Linux;

set Time Zone;

enter root password.

Following this begins the core installation. Package selection screen that eventually appears should show the server packages selections checked;

accept this configuration;

note that

`/root/install.log`

contains the particulars just chosen in these steps. Time to move on to the next round of performing the configuration;

select continue at the next screen prompt;

system will proceed to install packages and prompt for necessary CDROMs.

Testing Fedora Linux for Connectivity

At the console enter:

`ifconfig`

to obtain the IP of the system.

Using SSH (OpenSSH for windows) as recommended, standard calling nomenclature is:

ssh root@67.171.24.169

respond to the prompt for password in the conventional way.

The following is not strictly necessary, but is offered here as a sidebar:

For secure ftp services, start sftp-server through the ssh connection with the call:

`/usr/lib/ssh/sftp-server`

From the client side (remote system):

open a secure ftp session with server type explicitly set to SFTP using SSH2.

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

Visit:

<http://www.apachefriends.org/en/xampp-linux.html>

to obtain a more complete server package called XAMPP (distribution file xampp-linux-1.4.16.tar.gz) that contains an easily installed server complete with MySQL, PHP, and other features that are already integrated.

Consult the Apache Appendix for details of installation and configuration (relatively straightforward and easy going). However, it must be observed that this addition created numerous security holes where the default login and password values would be available to anyone with similar access to the XAMPP package.

create the directory /download and FTP the XAMPP installation file into it
cd /download

Install XAMPP with the call:

```
tar xvfz xampp-linux-1.4.16.tar.gz -C /opt
```

Testing XAMPP

In the remote shell, call:

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

point a browser at:

```
http://24.19.47.204/xampp/
```

and confirm the splash page appears.

In the remote shell, call:

```
/opt/lampp/lampp stop
```

Configuring XAMPP

Use the following command switch "back" to PHP 4.3.x:

```
/opt/lampp/lampp php4
```

This will also start the Apache server, but not the servers beneath it like MySQL or ProFTPD.

The following is included for reference:

And with the following command you can switch back to PHP 5.0.x:

```
/opt/lampp/lampp php5
```

To find which version of PHP is in use simply use phpinfo() or call this command:

```
/opt/lampp/lampp phpstatus
```

Depending upon the steps above, shut down the ftp server

```
/opt/lampp/lampp stopftp                      Stops the ProFTPD server
```

Then proceed with security:

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

for XAMPP page security

ID: **lampp** (there is no option to change this)

PW: *********

the system will prompt about the availability of MySQL over the net, do you want to turn it off?

yes

for MySQL & phpMy security

ID: **pma** (there is no option to change this)

PW: ********* (appears to be identical to user root - mySQL)

ID: **root** (there is no option to change this)

PW: ********* (appears to be identical to user pma - phpMyAdmin)

for ProFTP security (when running) will prompt for password change, enter:

yes

PW: *********

Setting XAMPP Bootstrap

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.

First, find out your default runlevel.

In the remote shell, simply type:

```
egrep :initdefault: /etc/inittab
```

You should see:

id:3:initdefault: shows up for Fedora Linux

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/rc3.d for Fedora Linux
```

Now carry out the actual configuration by typing:

```
ln -s /opt/lampp/lampp S99lampp
```

```
ln -s /opt/lampp/lampp K01lampp
```

If there is any interference from another Apache installation, reference:

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
```

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

To reboot after configuration changes:

reboot

Now XAMPP should start and stop automatically if you boot or shutdown your machine. This should be tested after booting by pointing browser at <http://24.19.47.204/xampp/>

Introducing Java for a JSP Server

Java is a networking language that is secure and feature rich for supporting Java Server Pages, JSP.

Installing Java

Consult Java Appendix below.

From:

<http://java.sun.com/j2se/1.4.2/download.html>

Download `j2sdk-1_4_2_09-linux-i586-rpm.bin` to `/download`

Using FileZilla, change the file attributes to allow for execution.

Within the `/download` directory call:

```
j2sdk-1_4_2_09-linux-i586-rpm.bin -localinstall
```

then

```
rpm - i j2sdk-1_4_2_09-linux-i586.rpm
```

Configuring Java

Set the environmental variable:

```
JAVA_HOME=/usr/java/j2sdk1.4.2_09/
```

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 either the LINUX, or 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: `Test.java`

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

```
}
```

Compile with:

```
/usr/java/j2sdk1.4.2_09/bin/javac Test.java
```

The result of the compile is the file: `Test.class`

Run:

```
/usr/java/j2sdk1.4.2_09/bin/java Test
```

```
Hello world
```

Introducing GNU m4

GNU m4 is an implementation of the traditional Unix macro processor.

Installing GNU m4

Visit,

<http://www.gnu.org/software/m4/m4.html>

and download `m4-1.4.3.tar.gz` into `/download`

Run (a variant of `tar -xvfz file.tar.gz -C /dir`):

```
gunzip m4-1.4.3.tar.gz
```

```
tar -xvf m4-1.4.3.tar -C /usr
```

Configuring GNU m4

Testing GNU m4

Introducing autoconf

Produces shell scripts that automatically configure source code.

Installing autoconf

Visit,

<http://directory.fsf.org/autoconf.html>

and download `autoconf-2.59.tar.gz` into `/download`

Run (a variant of `tar -xvfz file.tar.gz -C /dir`):

```
gunzip -xvfz autoconf-2.59.tar.gz
```

```
tar -xvf autoconf-2.59.tar -C /usr
```

Configuring autoconf

Testing autoconf

Introducing GCC, Gnu C Compiler

GCC is a compiler used to build Linux packages from distributed source. Its inclusion here is in anticipation of building portions of packages to be listed below.

Installing GCC

Visit,

<http://gcc.gnu.org/gcc-4.0/>

and download gcc-4.0.2.tar.gz into /download

Run:

```
tar -xvfz gcc-4.0.2.tar.gz -C /usr
```

Configuring GCC

Visit:

<http://gcc.gnu.org/install/configure.html>

Testing GCC

Introducing Tomcat, the JSP Server

Tomcat is an Apache JSP Server.

Installing Tomcat

Consult the Tomcat Appendix below.

Visit:

<http://tomcat.apache.org/download-55.cgi>

Download apache-tomcat-5.5.12.tar.gz to /download

cd to /download

Install Tomcat with the call:

```
tar -xvfz apache-tomcat-5.5.12.tar.gz -C /opt
```

Configuring Tomcat

Set environmental variable:

```
CATALINA_HOME=/opt/apache-tomcat-5.5.12/
```

```
cd $CATALINA_HOME/bin
```

```
tar -xvfz jsvc.tar.gz
```

```
cd jsvc-src
```

```
autoconf
```

reports *****BUG in Autoconf Ignored**

```
support/buildconf.sh
```

To proceed, we need to find and install gcc to do compilation in shell file that follows. Visit:

Enter:

```
PATH=$PATH:/usr/gcc-4.0.2/gcc
```

```
./configure --with-java=/usr/java/j2sdk1.4.2_09
```

```
./configure --with-java=/opt/j2sdk1.4.2_07
```

```
make
```

```
cp jsvc ..
```

no apparent function

```
cd ..
```

no apparent function

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 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

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).

Fedora Appendix

from:

<http://www.aarnet.edu.au/events/conferences/2004/apan-questnet/sipworkshop/serverbuild.html>

Linux Kernal using RedHat's Open Source "Fedora Core 1"

The following instructions are based on building a Unix server using typical PC Intel i386 based hardware.

IMPORTANT NOTE: Fedora Core 1 does not support Intel PCs using Serial ATA Harddisks, and recent testing shows that SER does not run on Fedora Core 2.

1. Follow the instructions at <http://fedora.redhat.com/download/>
 1. Download the three ISO images for Fedora Core 1 (not 2), also called "yarrow" from Red Hat.
<http://download.fedora.redhat.com/pub/fedora/linux/core/1/>
 2. Create three CDRoms using the three ISO images.
2. **Insert Fedora Core 1 CD 1 and boot from CD**
 1. Press Enter at boot screen to start setup
 2. When prompted to do the Media Test you can select Skip if you are confident your media is OK
 3. On the Fedora Core welcome screen, select Next
 4. Select English (English) as your language and press Next
 5. Select your keyboard type and press Next
 6. Select your mouse type and press Next
 7. If you have an existing Redhat Linux installation on the computer the Fedora installer will find this installation and ask you if you want to Upgrade the existing installation or perform a refresh installation of Fedora Core. If this is the case, select Install Fedora Core and press Next
3. We want to **do a Server installation** of Fedora Core so select Server and press Next
4. Select **Manual partition with Disk Druid** and press Next
 1. Delete any existing disk partitions so that your whole disk is Free Space
 2. For simplicity we will make **3 partitions, one swap partition, one boot partition and one big root partition**
 3. Create a **"Swap" partition to be 2 or 4 times the size of your available memory**
 1. Click New
 2. Select swap as the File System Type
 3. In the Size (MB) text field enter the size of your swap partition. As a guide, set the swap partition to be 2 or 4 times the size of your available memory. As an example, my computer has 256Meg

- RAM, so I will set a 1Gig swap
4. Under Additional Size Options select Fixed Size
 5. Ensure Force to be a primary partition is not selected and press OK
 4. Create a **"/boot" partition with a File System Type of ext3 with 100Mbytes**
 1. Click New
 2. In the Mount Point text field enter /boot to be an ext3 type of 100Meg.
 3. Select ext3 as the File System Type
 4. In the Size (MB) text field enter the size of your boot partition. This partition should be at least 50Meg, and a good size is usually 100Meg.
 5. Under Additional Size Options select Fixed Size
 6. Select Force to be a primary partition and press OK
 5. Create a **"/" root partition with the remained of the disk** (Optional would be to create a /var partition for the logs to go)
 1. Click New
 2. In the Mount Point text field enter /
 3. Select ext3 as the File System Type
 4. Under Additional Size Options select Fill to maximum allowable size to use the remainder of the disk
 5. Ensure Force to be a primary partition is not selected and press OK
 6. Now you have setup your partitions press Next
 5. Select GRUB as the boot loader and press Next
 6. Next you will need to setup the network device of your machine. The information you enter here will vary depending on your environment
 1. Select eth0 device and press Edit
 2. Deselect Configure using DHCP - you want to configure a fixed ip address
 3. Select Activate on boot
 4. Enter your IP address
 5. Enter your Netmask
 6. Press OK
 7. Set the hostname manually
 8. Enter Gateway IP address.
 9. Enter Primary DNS IP address - **For the Tutorial you may need to change it to simulate the SRV records, probably 169.222.239.2**
 10. Enter Secondary DNS IP address if available
 11. Enter Tertiary DNS IP address if available
 12. Press Next
 7. Select No firewall and press Next
 1. Press Proceed on the warning screen about a firewall, you can setup a proper firewall later on
 8. Select the Default language for your system and press Next
 9. Select your timezone and press Next
 10. Enter your desired Root Password and Confirm and press Next

11. This document assumes that this server will be dedicated to be your SIP server and so we will remove some packages that are not needed by your SIP server. For the purposes of the APAN SIP tutorial we will install Xwindows so you have a graphical interface with a browser to access the serweb tool. (Xwindows is not normally needed for your SIP server)
12. Under Desktops select **X Window System** and click Details (far right side of X Window System)
 1. Ensure only the following packages are installed (for a small installation although you can add more):
 - XFree86-twm**
 - Xfree86-xdm**
 - firstboot**
 - gdm**
 - rhgb**
 - xterm**
13. To install a decent Window Manager, under Desktops select your Window Manager of choice, either GNOME or KDE and install your required packages. For the purposes of the APAN SIP tutorial we will install we will install KDE.
14. Under Desktops select **KDE Desktop Environment** and click Details
 1. Ensure only the following packages are installed (for a small installation although you can add more):
 - kdeadmin**
 - kdenetwork**
 - kdeutils**
15. Under Applications select **Editors** and click Details
 1. Ensure only the following packages are installed (for a small installation although you can add more):
 - vim-enhanced**
 - emacs**
16. Under Applications select **Graphic Internet** and click Details
 1. Ensure only the following packages are installed (for a small installation although you can add more):
 - mozilla**
17. Under Applications deselect Text-based Internet
18. Under Servers click Details for **Server Configuration Tools**
 1. Ensure only the following packages are installed (for a small installation although you can add more):
 - Xfree86-xauth**
19. Under **Web Servers** click Details for
 1. Ensure only the following packages are installed (for a small installation although you can add more):
 - httpd-manual**
 - php**
 - php-mysql**
20. Under Servers deselect Windows File Server

21. Under Servers select **SQL Database Server** and click Details
 1. Ensure only the following packages are installed (for a small installation although you can add more):
mysql-server
22. Under Development select **Development Tools** and click Details
 1. Ensure only the following packages are installed (for a small installation although you can add more):
rpm-build
23. Under Development select **Network Servers** and click Details
 1. Ensure only the following packages are installed (for a small installation although you can add more):
freeradius
tftp-server
24. Under System deselect **Administration Tools**
25. Under System select **System Tools** and click Details
 1. Ensure only the following packages are installed (for a small installation although you can add more):
ethereal
nmap
screen
26. Under System deselect Printing Support
27. We are now ready to continue with the installation.
 1. Click Next
28. The installation will now check for dependencies, but there shouldn't be any for our installation as above
 1. Click Next again
29. The required media will be listed in a popup dialog box, click Continue
30. The installation will now proceed by formatting the required filesystems, transferring the install image to the hard drive, and then installing the selected packages.
31. You will be prompted to insert the required media at different times throughout the installation. When prompted, insert the required media and press OK.
32. When prompted to reboot, remove the installation media and press Reboot
33. You have now installed a system that will run as a SER SIP Server for the APAN SIP Tutorial

```
#
# ENABLE ALL THE FOLLOWING SERVICES TO START AUTOMATICLY ON
# RELOAD OF SERVER
Use the programme "ntsysv"
# enable these:
# http
```

```
# mysql  
# radiusd  
# tftp
```

Linux Grub Appendix

from:

<http://www.linuxgazette.com/issue64/kohli.html>

What is a boot loader?

A boot loader is a program that resides in the starting sectors of a disk, e.g., the MBR (Master Boot Record) of the hard disk. After testing the system during bootup, the BIOS (Basic Input/Output System) transfers control to the MBR if the system is set to be booted from there. Then the program residing in MBR gets executed. This program is called the boot loader. Its duty is to transfer control to the operating system, which will then proceed with the boot process.

There are a lot of boot loader programs available, including GNU GRUB (Grand Unified Boot Loader), Bootmanager, LILO (Linux LOader), NTLDR (boot loader for Windows NT systems), etc. I've chosen to discuss GNU GRUB and how to use it.

What is GRUB?

GRUB is a very powerful boot loader that can load a variety of operating systems such as Windows, DOS, Linux, GNU Hurd, *BSD, etc.

Currently LILO is the most popular boot loader, used by almost everyone with multiboot systems. But if you use LILO, you have to remember to rerun LILO every time you change your configuration or install a new kernel. Also, LILO has less flexibility than GRUB.

GRUB is another name for flexibility. Its latest release, 0.5.96.1, supports ext2 (a file system Linux uses), FAT16 and FAT32 (used by Win9x and ME), FFS (Fast File System used by *BSD UNIX), ReiserFS (a new journalling file system developed for Linux and integrated into Linux Kernel 2.4.1), and minix (an old file system developed for the MINIX OS, also used by earlier Linux). With GRUB, you can "see" into these file systems without even booting an operating system. For example, if you want to see the date and time stored in a text file and don't have time for the whole operating system to boot, you can use GRUB's shell (prompt "grub>") and type:

```
grub> cat (partition number)/home/god/filename.txt.
```

You'll have all your file system contents, including dates and times.

The best use of GRUB is that you can load any kernel on any partition right out of the box. For example, if you forget adding the newly compiled kernel to the list, you would

normally need to boot, add it to the list and then reboot to use it. But with GRUB, you can simply use the shell and load the desired kernel image.

I'll now explain the three primary steps to using GRUB: compilation, installation and configuration.

Refer to original for complete discussion.

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;

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.

XAMPP Directories and Commands

The following are the full command set for XAMPP Apache and components:

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>.

☞ What is where?

What is where? A big question of our existens, here are some answers! ;)

IMPORTANT FILES AND DIRECTORIES

File/Directory	Purpose
<code>/opt/lampp/bin/</code>	The XAMPP commands home. <code>/opt/lampp/bin/mysql</code> calls for example the MySQL monitor.
<code>/opt/lampp/htdocs/</code>	The Apache DocumentRoot directory.
<code>/opt/lampp/etc/httpd.conf</code>	The Apache configuration file.
<code>/opt/lampp/etc/my.cnf</code>	The MySQL configuration file.
<code>/opt/lampp/etc/php.ini</code>	The PHP configuration file.
<code>/opt/lampp/etc/proftpd.conf</code>	The ProFTPD configuration file. (since 0.9.5)
<code>/opt/lampp/phpmyadmin/config.inc.php</code>	The phpMyAdmin configuration file.

☞ Stopping XAMPP

To stop XAMPP simply call this command:

```
/opt/lampp/lampp stop
```

You should now see:

```
Stopping LAMPP 1.4.16...
LAMPP: Stopping Apache...
LAMPP: Stopping MySQL...
LAMPP stopped.
```

And XAMPP for Linux is stopped.

☞ Uninstall

To uninstall XAMPP just type in this command:

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


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

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.

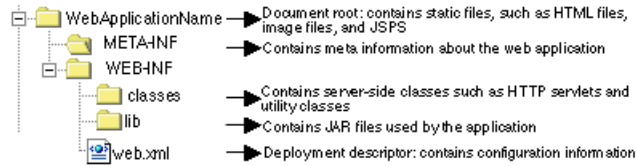


Figure 1. Web application directory structure

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.

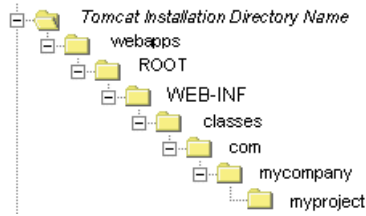


Figure 2. Package layout in Tomcat

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>` 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

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"/>
```

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 `PasswordUserDatabase` 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.renamet jar* (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.

JSP Syntax Appendix

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 Javabeans 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.ServletContext`

Config `Javax.servlet.http.ServletConfig`

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();
}
}

```