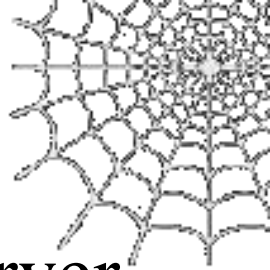


# Tomcat

*Servers allow web sites to do things, not just display things*

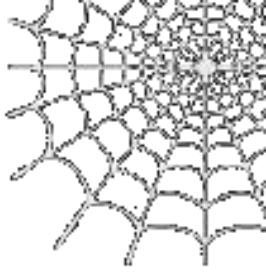
# Web Servers



To create a dynamic web site, you have to choose a web server, which amounts to choosing a framework with a scripting or development language for writing the pages, and we will look at just a few:

<u>CGI</u>	Perl or C
<u>ASP</u>	VBScript or C#
<u>LAMP</u>	PHP
<u>Zope</u>	Python
<u>Rails</u>	Ruby
<u>JSP</u>	Java, <u>our choice</u>

# SysAdmin

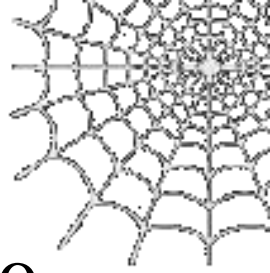


SysAdmin is short for system administrator skills, and *all* web-server related software assumes them

You *need* a particular attitude (and type of experience) to install, configure, run and troubleshoot servers

Troubleshooting is like debugging, but *harder*, there are many more systems that could be at fault, including ones outside your control and with no source code, and the pressure to fix things is high

# Strategy



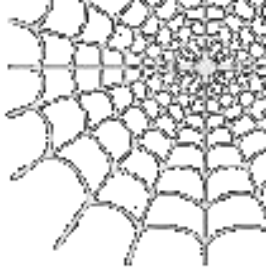
As with everything else we have done, we need a strategy to deal with this problem

First: every time you install, configure or customise, do it in small steps (each dealing with only one issue)

Second: make sure each step is reversible (e.g. make a backup of each file you change)

Third: keep a personal log of every step you take and everything you find out to help next time (but remember it still won't be a foolproof recipe)

# Tomcat



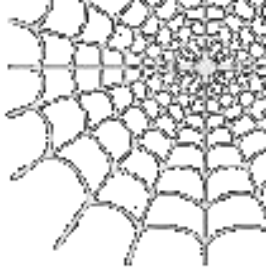
Web servers are not glamorous, so there is no particularly outstanding *proprietary* web server

The most popular open source web server is "Apache", the C-based web server from Apache

We will concentrate on Tomcat, Apache's newer Java-based server (started by Sun), open source, platform independent

Tomcat supports the JSP standard for working with Java, and we will only use standard facilities

# Tomcat Features



Tomcat allows standard embedded Java in web pages

It supports a standard way of writing ordinary Java classes, and integrating them with a web site

It also provides standard embedded scripting, based on XML tags and insertions, it can be used with JSTL, a standard library of programmed tags, including database querying tags, and it supports a standard way of writing custom tags, to share fragments between pages, but we will ignore all these in favour of using Java directly

# Installing Tomcat



When you first install Tomcat, you can do it on a workstation or on your own PC as an ordinary user

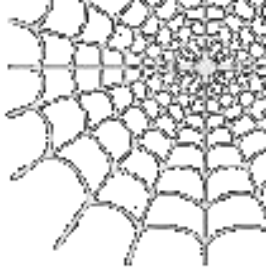
Later, of course, you need to find a permanently running, permanently connected computer to host Tomcat, and maintain a service as an administrator

It seems easy to install, but don't let that fool you, it is still system software and there are still pitfalls:

<http://tomcat.apache.org/tomcat-6.0-doc/>

<http://www.coreservlets.com/Apache-Tomcat-Tutorial/>

# Java



The first pitfall for beginners is that Tomcat relies on Java, which must be installed *properly* first

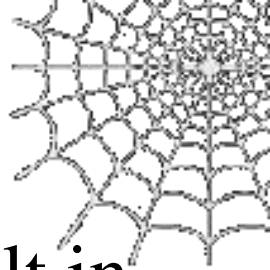
You can use the JDK, either J2SE (Standard) or J2EE (Enterprise) Edition, where J2EE adds web-related libraries

For programming, you may want to bookmark both the J2SE and J2EE documentation

<http://java.sun.com/javase/6/docs/api/>

<http://java.sun.com/javaee/5/docs/api/>

# Java Strategies



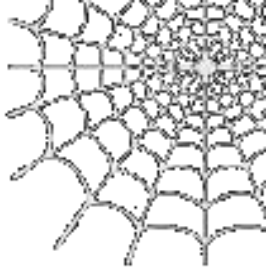
Tomcat has the extra library classes it needs from J2EE built in, so there are two strategies possible

The first is to install J2SE and, for compiling support classes, link with Tomcat's J2EE classes (preferred, but you have to know how!)

The second is to install J2EE (but the extra libraries in it may not match Tomcat's versions!)

Another consideration is that upgrade of a public JDK, e.g. adding extension libraries, can crash Tomcat, so to play safe, give Tomcat its own copy of Java

# Environment variables

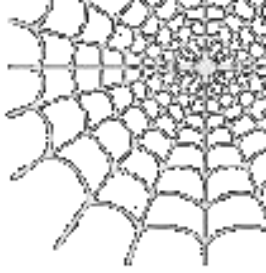


Correct installation of Java involves environment variables, which are horrid but still standard and reasonably portable between platforms

For Java, it is the PATH variable you have to change, and how you do it is different on every platform, but there are instructions in the Java install guide

The PATH is a list, separated by `:` (Unix) or `;` (Win), of full pathnames of directories where the platform looks for commands, and the Java bin directory adds `java`, `javac`...

# Java tests



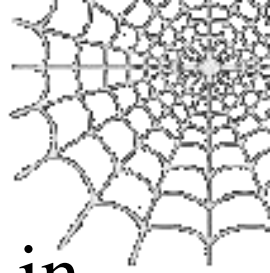
A typical platform has several Java facilities, so if you get installation wrong, things may not work

To test that everything is OK, get a command/console/terminal/shell/prompt/DOS/cmd window, change to any directory outside the Java installation, and try:

```
javac -version  
java -version
```

If a version is wrong, try putting the Java bin directory earlier in the PATH environment variable

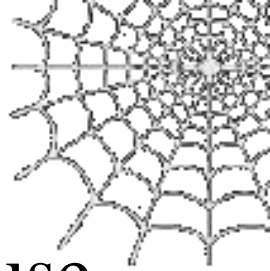
# Tomcat directory structure



Customisation is by hand-editing of scripts and config files in the tomcat installation directory

tomcat	(or whatever)
bin	startup/shutdown scripts
conf/server.xml	main config file
logs	log files for troubleshooting
webapps	sub-sites & applications
ROOT	main site
index.jsp	home page
WEB-INF	site-specific unpublished stuff
web.xml	config file for this (sub-)site

# Installing on Windows



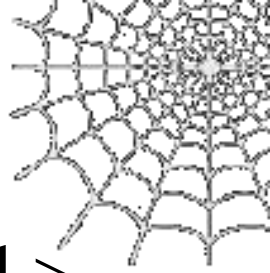
Choose "core" and manually install a 32 or 64 bit version (use Control-Panel/System to find out which you have), or use an installer which sets up a service

If manual, use tomcat/bin/startup.bat & shutdown.bat which call catalina.bat, which documents environment variables, best added near the top:

```
CATALINA_HOME=../tomcat  
JAVA_HOME=../jdk
```

You can also set up a service manually

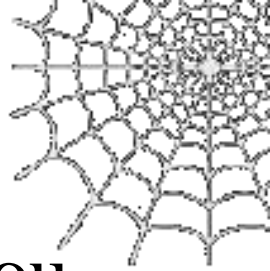
# Windows Services



To view/configure Windows services, enable Control Panel > Taskbar... > Start Menu > Customize > Advanced > Start menu items > System Administrative Tools, which adds a folder to the start menu.

Then go to Start>Program>Administrative Tools>Services and look for Apache Tomcat, and start/stop the service from there

# Installing on Unix



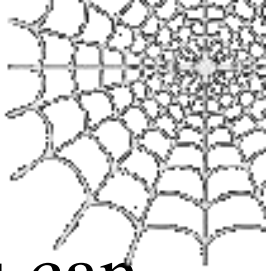
Unzip the distribution, and rename the result directory as you want, let's call it `tomcat` (aside: variations)

Run it using `tomcat/bin/startup.sh` & `shutdown.sh` which call `catalina.sh`, which documents environment variables, best added near the top:

```
CATALINA_HOME=../tomcat
JAVA_HOME=../jdk
```

Use `chmod +x *.sh` then run `startup.sh` to start, run it in the background to logoff and leave it running, call from an init script to have it started on reboot, ...

# Getting Started

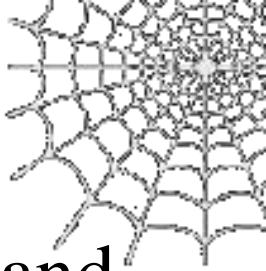


To test, visit `http://localhost:8080/` and then you can try creating some pages

You may find `tomcat/webapps/ROOT/index.jsp` *and* `.../index.html` so delete one, edit the other, then visit and refresh

In some older versions of Tomcat, nothing happens, because in `.../ROOT/WEB-INF/web.xml` there is an example of pre-compiling, which needs to be removed

# Configuring



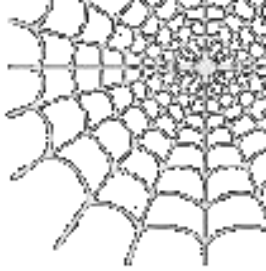
Follow the strategy: don't try to change two things at once, and backup a script or config file before changing it, so you can revert

Delete or rename `tomcat/logs/*` before restarting so you can see new log entries more easily

Check that shutting down Tomcat actually worked, or use Task Manager or `ps/kill` to get rid of the process

Backup `server.xml` and delete everything you can to simplify it; rummage in docs and google to find out what to do

# Ports



By default Tomcat uses these port numbers

8005 for control, used by shutdown.sh

8080 for http (not 80, in case no root access)

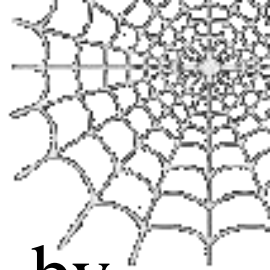
8443 for https (not 443, by default disabled)

8009 for connecting to other servers

This means you have to reconfigure if you want Tomcat to use standard ports, or if you want https

Also, two people cannot run Tomcat on the same computer without using different sets of port numbers (so don't run on snowy, only workstations)

# Changing Ports



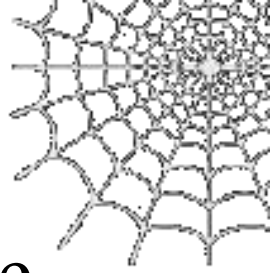
An important configuration is to switch to standard port 80, by editing `tomcat/conf/server.xml`

On Linux, you need root access for port 80 so Tomcat must run as root. On Windows, you may need to disable IIS or Skype.

For security, if the computer running Tomcat can also do other things, Tomcat should not run as root

A Java program can't change user, so a C program called `jsvc` is provided which opens port 80, switches user, then starts Tomcat

# Moving the site



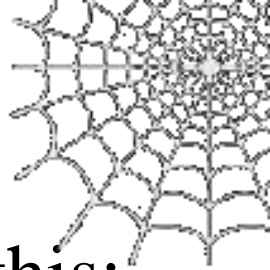
It is often inappropriate or inconvenient to have the web site files buried in `.../tomcat/webapps/ROOT`

Moving or defining subsites is done with XML `<Context>` elements in `server.xml` (or separate files in `tomcat/conf/Catalina/...`); if there isn't one, you may have to add a `<Context>` element like this:

```
<Context path="" docBase="C:\website"  
  debug="0" reloadable="true" />
```

The `docBase` is absolute, or relative to `webapps`

# Testing Java



To test embedded Java, write a web page `test.jsp` like this:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>Testing</title></head>
<body>
<p>Testing Java: <% out.print("OK"); %>.</p>
</body>
</html>
```

XHTML

The text between `<% . . . %>` is Java code

You should see Testing Java: OK. and nothing else