

# Tomcat

*Tomcat is the leading Java-based web server*

[Installing](#), [Configuring](#), [JSTL](#), [Testing](#)

---

## Tomcat Features

- Tomcat supports standard *embedded Java* within web pages
- It provides standard *embedded scripting* as an alternative, based on XML tags and insertions
- It can be used with *JSTL*, a standard library of programmed tags, including database querying
- It supports a standard way of writing *custom tags*, to share fragments between pages
- It supports a standard way of writing ordinary Java classes, and integrating them with a site

---

## Java

- The first pitfall for beginners is that Tomcat relies on Java, which must be installed *properly* first
  - You can use JRE or 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/>

---

## Environment variables

- Correct installation of Java involves environment variables, which are *horrid* but still standard and reasonably portable between platforms
- 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
- It is very easy to get this wrong, but it helps to understand: the PATH variable 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`...

# 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 Apache's newer Java-based server *Tomcat* (started by Sun), open source, platform independent

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

---

## 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 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 JRE/JDK, e.g. adding extension libraries, can crash Tomcat, so to play safe, give Tomcat its own copy

---

## 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 directory, 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 (or tool) editing of scripts and config files in the tomcat installation directory

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| WEB-INF    | site-specific unpublished stuff |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

## Installing on Windows XP

The Windows installer for Tomcat installs Tomcat as a service

To view/configure services, enable Control Panel > Taskbar... > Start Menu > Customize > Advanced > Start menu items > System Administrative Tools then Start>Program>Administrative Tools>Services and look for Apache Tomcat, and start/stop by hand

You can also configure whether or not the service starts running every time you boot

Test by visiting <http://localhost:8080/>

## Installing on Unix

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

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, ...

## Installing on Other Systems

On some unix/linux systems, Tomcat may get installed as a 'service' (daemon), then it may be run from a script such as `/etc/init.d/tomcat6`

However Tomcat is installed on your system, you will sooner or later **need** to customize the call which starts up Java

Therefore you **need** to work out which script to customize

## 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

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

Also: delete or rename `tomcat/logs/*` before restarting so you can see new entries more easily

Check that shutting down 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

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

## Changing Ports

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

On Linux, because  $80 < 1024$ , you need root access and Tomcat must then run as root. On Windows, you may need to disable IIS.

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

Tomcat must run as root, open port 80, then run as a different user, but Java can't do this, so a little C program called `jsvc` is provided which can, and `startup.sh` is replaced by a script which calls it

## Tandem Servers

It used to be common to run the two Apache servers in tandem, with the C-based server delivering HTML pages, and passing dynamic pages (.jsp) to Tomcat

Timing tests showed no measurable difference between this and Tomcat alone (version 4+), so it is best to avoid the complexity and run Tomcat alone

What the tests did show is that https (any server) is much slower than http (any server), and some pages suffer from either database or Java bottlenecks

## 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, and the documentation says these should no longer be in tomcat/conf/server.xml but in separate files in tomcat/conf/Catalina/..., but it turns out that for the main site, you still do have to add a <Context> element like this:

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

to the <Host> element of server.xml

## JSTL

Tomcat provides a general (Java-based) scripting facility for including data and for defining extra programmed XML tags

Many different kinds of tags have been proposed, and there is a system for creating your own, but it is easy to get distracted by all this

Recently, though, one particular collection of tags has been **standardised**, *JSTL = JSP Standard Tag Library*, but it isn't (yet) included with Tomcat by default

## Should you use JSTL?

JSTL does make some things easier, but there are complex issues, e.g. it makes it more difficult to integrate JSP pages with Java support classes

So, I suggest learning to do everything without JSTL first, then after you understand all the issues, decide whether JSTL is worth using for a particular project

For example, in a recent faculty-wide project, we decided against, because of our custom Java classes

## Getting Started

The first thing that needs to be done to install JSTL is to make sure that web.xml has version 2.5(+):

```
<?xml version="1.0" ...?>
<web-app xmlns="...j2ee"
  xmlns:xsi="...XMLSchema-instance"
  xsi:schemaLocation="...web-app_2_5.xsd"
  version="2.5">
...
</web-app>
```

The most recent version of Tomcat will have version 2.5 already, but for older versions, web.xml must be replaced

## Download

The best implementation of JSTL to download is probably the Jakarta Standard 1.1 Taglib, since other implementations are usually bundled up with stuff you don't want

Unzip it and copy *only* the two files jstl.jar and standard.jar into site/WEB-INF/lib

As with most changes to Tomcat, you may or may not need to restart Tomcat to see the change

You may also want to download the JSTL specification (and/or look at tutorial sites)

## 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>
```

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

You should see *Testing Java: OK*. and nothing else

## Testing Scripting

To test embedded scripting, try 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 scripting: ${"OK"}.</p>
</body>
</html>
```

The text between \${ ... } is scripting code which inserts a value

You should just see *Testing scripting: OK*.

# Testing JSTL

To test JSTL tags, write a web page like this:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %>
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>Testing</title></head>
<body>
<p>Testing JSTL: <c:out value="OK" />.</p>
</body>
</html>
```

The taglib elements import core and database tags

The `out` tag is a core tag which inserts a value

You should just see *Testing JSTL: OK*.