**1. Apache Tomcat HTTP Server**

*Apache Tomcat* is a Java-capable HTTP server, which could execute special Java programs known as "Java Servlet" and "Java Server Pages (JSP)". Tomcat is an *open-source* project, under the "Apache Software Foundation" (which also provides the most use, open-source, industrial-strength Apache HTTP Server). The mother site for Tomcat is [http://tomcat.apache.org](http://tomcat.apache.org/). Alternatively, you can find tomcat via the Apache mother site @ <http://www.apache.org>.

Tomcat was originally written by James Duncan Davison (then working in Sun), in 1998, based on an earlier Sun's server called Java Web Server (JWS). It began at version 3.0 after JSWDK 2.1 it replaced. Sun subsequently made Tomcat open-source and gave it to Apache.

The various Tomcat releases are:

1. Tomcat 3.x (1999): RI for Servlet 2.2 and JSP 1.1.
2. Tomcat 4.x (2001): RI for Servlet 2.3 and JSP 1.2.
3. Tomcat 5.x (2002): RI for Servlet 2.4 and JSP 2.0.
4. Tomcat 6.x (2006): RI for Servlet 2.5 and JSP 2.1.
5. Tomcat 7.x (2010): RI for Servlet 3.0, JSP 2.2 and EL 2.2.
6. Tomcat 8.x (2013): RI for Servlet 3.1, JSP 2.3, EL 3.0 and Java WebSocket 1.0.

Tomcat is an HTTP application runs over TCP/IP. In other words, the Tomcat server runs on a specific TCP port from a specific IP address. The default TCP port number for HTTP protocol is 80, which is used for the *production* HTTP server. For *test* HTTP server, you can choose any unused port number between 1024 and 65535.

### 2.  How to Install Tomcat 8 and Get Started with Java Servlet Programming

#### 2.1  STEP 1: Download and Install Tomcat

**NOTE**: At the time of writing, Tomcat 9 is at the alpha stage, not stable release. We shall install Tomcat 8.5.11.

For Windows

1. Goto [http://tomcat.apache.org](http://tomcat.apache.org/) ⇒ Under "Tomcat 8.5.{xx} Released" (where {xx} is the latest upgrade number) ⇒ Downloads ⇒ Under "8.5.{xx}" ⇒ Binary Distributions ⇒ Core ⇒ "**ZIP**" package (e.g., "apache-tomcat-8.5.{xx}.**zip**", about 9 MB).
2. Create your project directory, say "d:\myProject" or "c:\myProject". UNZIP the downloaded file into your project directory. Tomcat will be unzipped into directory "d:\myProject\apache-tomcat-8.0.{xx}".
3. For **ease of use**, we shall shorten and rename this directory to "d:\myProject\tomcat".

**Take note of Your Tomcat Installed Directory**. Hereafter, I shall refer to the Tomcat installed directory as <TOMCAT\_HOME>.

**Take note of Your Tomcat Installed Directory**. Hereafter, I shall refer to the Tomcat installed directory as <TOMCAT\_HOME>.

#### 2.2  STEP 2: Create an Environment Variable JAVA\_HOME

(For Windows)

You need to create an environment variable called "JAVA\_HOME" and set it to your JDK installed directory.

1. First, find your JDK installed directory. The default is "c:\Program Files\Java\jdk1.8.0\_{xx}", where {xx} is the upgrade number. Take note of your JDK installed directory.
2. To set the environment variable JAVA\_HOME in Windows 7/8/10: Start "Control Panel" ⇒ System and Security (Optional) ⇒ System ⇒ Advanced system settings ⇒ Switch to "Advanced" tab ⇒ Environment Variables ⇒ System Variables ⇒ "New" ⇒ In "Variable Name", enter "JAVA\_HOME" ⇒ In "Variable Value", enter your JDK installed directory as noted in Step 1.
3. To verify, **RE-START** a CMD shell (restart needed to refresh the environment) and issue:
4. **SET JAVA\_HOME**

JAVA\_HOME=c:\Program Files\Java\jdk1.8.0\_{xx} <== Verify that this is YOUR JDK installed directory

#### 2.3  STEP 3: Configure Tomcat Server

The Tomcat configuration files are located in the "conf" sub-directory of your Tomcat installed directory, e.g. "d:\myProject\tomcat\conf" (for Windows) or "/Applications/tomcat/conf" (for Mac OS). There are 4 configuration XML files:

1. server.xml
2. web.xml
3. context.xml
4. tomcat-users.xml

Make a BACKUP of the configuration files before you proceed!!!

Step 3(b) "conf\server.xml"

Locate the following lines (around Line 69) that define the HTTP connector, and change port="8080" to port="9999".

<!-- A "Connector" represents an endpoint by which requests are received

and responses are returned. Documentation at :

Java HTTP Connector: /docs/config/http.html (blocking & non-blocking)

Java AJP Connector: /docs/config/ajp.html

APR (HTTP/AJP) Connector: /docs/apr.html

**Define a non-SSL HTTP/1.1 Connector on port 8080**

-->

<Connector port="9999" protocol="HTTP/1.1"

connectionTimeout="20000"

redirectPort="8443" />

##### Step 3(b) "conf\web.xml" - **Enabling Directory Listing**

Again, use a programming text editor to open the configuration file "web.xml", under the "conf" sub-directory of Tomcat installed directory.

We shall enable directory listing by changing "listings" from "false" to "true" for the "default" servlet. This is handy for test system, but not for production system for security reasons.

Locate the following lines (around Line 103) that define the "default" servlet; and change the "listings" from "false" to "true".

<!-- The default servlet for all web applications, that serves static -->

<!-- resources. It processes all requests that are not mapped to other -->

<!-- servlets with servlet mappings. -->

<servlet>

<servlet-name>**default**</servlet-name>

<servlet-class>org.apache.catalina.servlets.DefaultServlet</servlet-class>

<init-param>

<param-name>debug</param-name>

<param-value>0</param-value>

</init-param>

<init-param>

<param-name>**listings**</param-name>

<param-value>**true**</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

##### Step 3(c) "conf\context.xml" - **Enabling Automatic Reload**

We shall add the attribute reloadable="true" to the <Context> element to enable automatic reload after code changes. Again, this is handy for test system but not for production, due to the overhead of detecting changes.

Locate the <Context> start element (around Line 19), and change it to <Context reloadable="true">.

<Context **reloadable="true"**>

......

......

</Context>

##### Step 3(d) (Optional) "conf\tomcat-users.xml"

Enable the Tomcat's manager by adding the highlighted lines, inside the <tomcat-users> elements:

<tomcat-users>

<role rolename="manager-gui"/>

<user username="manager" password="xxxx" roles="manager-gui"/>

</tomcat-users>

This enables the manager GUI app for managing Tomcat server.

#### 2.4  STEP 4: Start Tomcat Server

The Tomcat's executable programs and scripts are kept in the "bin" sub-directory of the Tomcat installed directory, e.g., "d:\myProject\tomcat\bin" (for Windows) or "/Applications/tomcat/bin" (for Mac OS).

##### Step 4(a) Start Server

For Windows

Launch a CMD shell. Set the current directory to "<TOMCAT\_HOME>\bin", and run "startup.bat" as follows:

// Change the current directory to Tomcat's "bin"

// Assume that Tomcat is installed in "d:\myProject\tomcat"

**d:** // Change the current drive

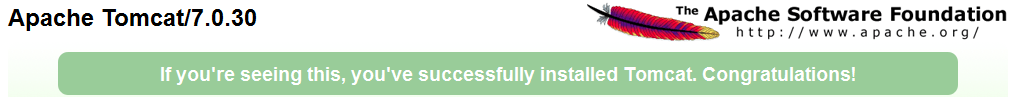
**cd \myProject\tomcat\bin** // Change Directory to YOUR Tomcat's "bin" directory

// Start Tomcat Server

**startup**

##### Step 4(b) Start a Client to Access the Server

Start a browser (as HTTP client). Issue URL "http://localhost:9999" to access the Tomcat server's welcome page. The hostname "localhost" (with IP address of 127.0.0.1) is meant for local loop-back testing inside the same machine. For users on the other machines over the net, they have to use the server's IP address or DNS domain name or hostname in the format of "http://serverHostnameOrIPAddress:9999".



Try issuing URL http://localhost:9999/examples to view the servlet and JSP examples. Try running some of the servlet examples.

(Optional) Try issuing URL http://localhost:9999/manager/html to run the Tomcat Web Manager. Enter the username and password configured earlier in tomcat-users.xml.

##### Step 4(c) Shutdown Server

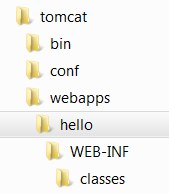
For Windows

You can shutdown the tomcat server by either:

1. Press Ctrl-C on the Tomcat console; OR
2. Run "<TOMCAT\_HOME>\bin\**shutdown.bat**" script. Open a new "cmd" and issue:
3. // Change the current directory to Tomcat's "bin"
4. **d:** // Change the current drive
5. **cd \myProject\tomcat\bin** // Change Directory to YOUR Tomcat's "bin" directory
7. // Shutdown the server
8. **shutdown**

#### 2.5  STEP 5: Develop and Deploy a WebApp

##### Step 5(a) Create the Directory Structure for your WebApp



First of all, choose a name for your webapp. Let's call it "hello". Goto Tomcat's "webapps" sub-directory. Create the following directory structure for you webapp "hello" (as illustrated):

1. Under Tomcat's "webapps", create your webapp root directory "hello" (i.e., "<TOMCAT\_HOME>\webapps\hello").
2. Under "hello", create a sub-directory "WEB-INF" (case sensitive, a "dash" not an underscore) (i.e., "<TOMCAT\_HOME>\webapps\hello\WEB-INF").
3. Under "WEB-INF", create a sub-sub-directory "classes" (case sensitive, plural) (i.e., "<TOMCAT\_HOME>\webapps\hello\WEB-INF\classes").

You need to keep your web resources (e.g., HTMLs, CSSs, images, scripts, servlets, JSPs) in the proper directories:

* "hello": The is called the context root (or document base directory) of your webapp. You should keep all your HTML files and resources visible to the web users (e.g., HTMLs, CSSs, images, scripts, JSPs) under this context root.
* "hello/WEB-INF": This directory, although under the context root, is not visible to the web users. This is where you keep your application's web descriptor file "web.xml".
* "hello/WEB-INF/classes": This is where you keep all the Java classes such as servlet class-files.

You should RE-START your Tomcat server to pick up the hello webapp. Check the Tomcat's console to confirm that "hello" application has been properly deployed:

......

INFO: Deploying web application directory D:\myProject\tomcat\webapps\**hello**

......

You can issue the following URL to access the web application "hello":

http://localhost:9999**/hello**

You should see the directory listing of the directory "<TOMCAT\_HOME>\webapps\hello", which shall be empty (provided you have enabled directory listing in web.xml earlier).

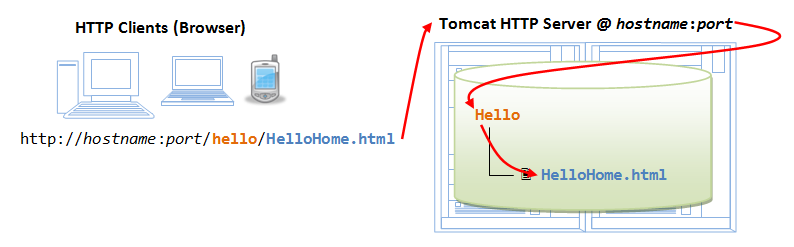
##### Step 5(b) Write a Welcome Page

Create the following HTML page and save as "HelloHome.html" in your application's root directory "hello".

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <html>  <head><title>My Home Page</title></head>  <body>  <h1>My Name is so and so. This is my HOME.</h1>  </body>  </html> |

You can browse this page by issuing this URL:

http://localhost:9999/hello/**HelloHome.html**



Alternatively, you can issue an URL to your web application root "hello":

http://localhost:9999/hello

The server will return the directory listing of your base directory. You can then click on "HelloHome.html".

Rename "HelloHome.html" to "index.html", and issue a directory request again:

http://localhost:9999/hello

Now, the server will redirect the directory request to "index.html", if the root directory contains an "index.html", instead of serving the directory listing.

You can check out the home page of your peers by issuing:

http://YourPeerHostnameOrIPAddress:9999/hello

http://YourPeerHostnameOrIPAddress:9999/hello/HelloHome.html

http://YourPeerHostnameOrIPAddress:9999/hello/index.html

with a valid "YourPeerHostnameOrIPAddress", provided that your peer has started his tomcat server and his firewall does not block your access. You can use command such as "ipconfig" (Windows), "ifconfig" (Mac OS and Unix) to find your IP address.

#### 2.6  STEP 6: Write a "Hello-world" Java Servlet

A servlet is Java program that runs inside a Java-capable HTTP Server, such as Apache Tomcat. A web user invokes a servlet by issuing an appropriate URL from a web browser (HTTP client).

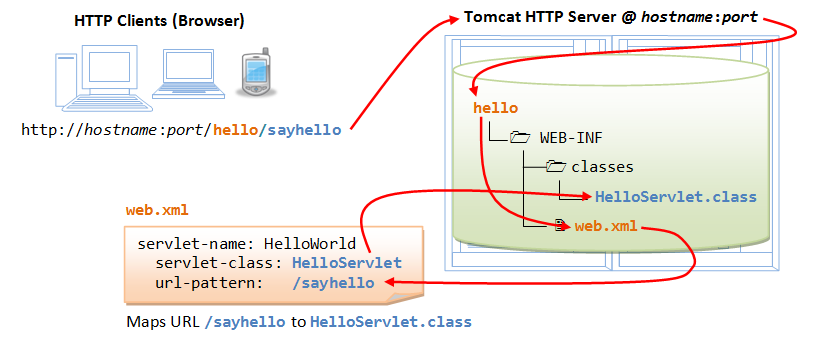
Before you proceed, I shall assume that you are familiar with Java Programming and have installed the followings:

1. JDK (Read "[How to install JDK and Get Started](https://www.ntu.edu.sg/home/ehchua/programming/howto/JDK_HowTo.html)").
2. A programming text editor, such as TextPad or Notepad++ (for Windows); jEdit or gEdit (for Mac OS) (Read "[Programming Text Editor](https://www.ntu.edu.sg/home/ehchua/programming/howto/CMD_Survival.html#editor)").

##### Step 6(a) Write a "Hello-world" Java Servlet

**A Java servlet is a Java program that runs inside a HTTP server. A web user invokes a servlet by issuing a URL from a browser (or HTTP client).**

**In this example, we are going to write a Java servlet called HelloServlet, which says "Hello, world!". We will then write a configuration such that web users can invoke this servlet by issuing URL http://hostname:port/hello/sayhello from their browser, as illustrated:**

****

**Write the following source codes called "HelloServlet.java" and save it under your application "classes" directory (i.e., "<TOMCAT\_HOME>\webapps\hello\WEB-INF\classes\HelloServlet.java"). This servlet says "Hello", echoes some request information, and prints a random number upon each request.**

|  |  |
| --- | --- |
|  |  |

##### **Step 6(b) Compiling the Servlet (DIFFICULT)**

We need the Servlet API to compile the servlet. Servlet API is NOT part of JDK. Nonetheless, Tomcat provides a copy in <TOMCAT\_HOME>/lib/servlet-api.jar. We need to include this JAR file in the compilation via the -cp (classpath) option.

(For Windows)

// Assume that Tomcat is installed in d:\myProject\tomcat

// Change directory to the source file

**d:**

**cd \myProject\tomcat\webapps\hello\WEB-INF\classes**

// Compile

**javac -cp .;d:\myProject\tomcat\lib\servlet-api.jar HelloServlet.java**

// Note: You need to enclose the jar file in double quotes if the path contains blank

// e.g., javac -cp .;"d:\Path To\tomcat\lib\servlet-api.jar" HelloServlet.java

**javac -cp .:/Applications/tomcat/lib/servlet-api.jar HelloServlet.java**

The output of the compilation is "HelloServlet.class". Browse the "classes" folder to make sure that it is created.

##### **Step 6(c) Configure Servlet's Request URL in "webapps\hello\WEB-INF\web.xml"**

A web user invokes a servlet, which is kept in the web server, by issuing a request URL from the browser. We need to configure this request URL for our HelloServlet.

Create the following configuration file called "**web.xml**", and save it under "**webapps\hello\WEB-INF**" (i.e., "<TOMCAT\_HOME>\webapps\hello\WEB-INF\web.xml").

|  |  |
| --- | --- |
|  |  |

In the above configuration, a servlet having a class file "HelloServlet.class" is mapped to request URL "/sayhello" (via an arbitrary servlet-name "HelloWorld"), under this web application "hello". In other words, the complete request URL for this servlet is "http://hostname:port**/hello/sayhello**".

This configuration file, saved under your webapp "hello", is applicable only to this particular webapp "hello".

RESTART your Tomcat server to refresh the "web.xml" file.

IMPORTANT: For EACH servlet, you need to write a pair of <servlet> and <servlet-mapping> elements with a common but arbitrary <servlet-name>. Take note that all the <servlet> elements MUST be grouped together and placed IN FRONT of the <servlet-mapping> elements.

##### **Step 6(d) Invoke the Servlet**

To run this servlet, start a browser, and issue the request URL configured earlier:

http://localhost:9999/**hello/sayhello**

You shall see the output of the servlet displayed in your web browser.

Refresh the browser, you shall see a new random number upon each refresh. In other word, the doGet() method of the servlet runs once per request.

only the output of the servlet (generated via the out.println() statements). The client has no access to the servlet source codes (which may contain confidential information).