Tutorial: Installing Tomcat 7 and Using it with Eclipse

This tutorial covers Tomcat 7, which supports the servlet 3.0 and JSP 2.2 specs. This means that you can also run servlet/JSP or JSF apps that support the latest versions. I recommend Tomcat 7 over Tomcat 6 for *all* apps, since Tomcat 7 also supports the older servlet 2.5 and JSP 2.1 specs. But, if you really need to use Tomcat 6, please see the tutorial on Eclipse with Tomcat 6. It takes only a short time to download Eclipse and learn the bare bones basics of using it to build Web apps and deploy them to Tomcat, and all the information you need to do this is described in this tutorial. This time will be very quickly recouped by the savings in development, debugging, and deployment times. To get started with Web apps in Eclipse, you only need to know a *very* small number of Eclipse features. You can gradually learn the advanced Eclipse capabilities at your leisure. Also note that if you print this page, the entire contents (including the expanded contents of all the accordion tabs below) will be printed.

If you find these free tutorials helpful, we would appreciate it if you would link to us. Send corrections or feedback on any tutorial to hall@coreservlets.com.

Quick Start

Here is a quick summary of basic use. *This section is enough for most developers*; however, if you want more details, expand the following sections.

1. **Install Java.** Download for Windows, MacOS, Linux, and Solaris from http://www.oracle.com/technetwork/java/javase/downloads/. As of early 2013, I used JDK 1.7.0_10 and 1.6.0_35 (the latest versions at the time), but any Java 6 or 7 version will work. Servlet 3.0 containers (of which Tomcat 7 is one) require Java 6+, and will not work with Java 5. You want the full JDK (with compiler), not just the JRE (for running existing apps). Accept all defaults when installing.
2. **Unzip Tomcat.** Unzip tomcat-7.0.34-preconfigured.zip into the location of your choice. I use the top level of the C drive, which results in C:\apache-tomcat-7.0.34. I modified the port so Tomcat runs on 80 instead of 8080, and made two other small changes: enabling directory listings, and making the server automatically restart when faces-config.xml or struts-config.xml changes. These changes are useful during development, but only the port number change should be set for deployment servers. Alternatively, you can download any Tomcat 7 version from the Tomcat Web site and then copy context.xml, server.xml, and web.xml into install_dir/conf. These files are annotated with comments on what modifications were done to change the port to 80 and to enable directory listings and automatic server restarts.
3. **Install and start Eclipse.** Download from http://www.eclipse.org/downloads/. Choose "Eclipse IDE for Java EE Developers", download, and unzip. As of early 2013, the latest version is Eclipse 3.8 (Juno), and is what I use as of the latest update of this tutorial. However, these instructions were also tested with Eclipse 3.7 and 3.6. Double click on eclipse.exe in the eclipse folder, then click on "Workbench".
4. **Tell Eclipse about Tomcat.** Click on Servers tab at bottom. R-click, New, Server, Apache, Tomcat v7.0, navigate to Tomcat 7 installation folder (e.g., C:\apache-tomcat-7.0.34), OK. If you don't see Servers tab, add the tab via Window, Show View, Servers.
5. **Run Tomcat.** Click on Servers tab at bottom. R-click on Tomcat v7.0, choose "Start". Open http://localhost/ in a browser (or http://localhost:8080/ if you downloaded Tomcat instead of using the preconfigured version, and then failed to change the port from 8080 to 80). Either way, you will see a 404 error message, but at least the message comes from Tomcat. Then, copy the ROOT app as described in the next section, come back, and reload http://localhost/ (or http://localhost:8080/ if using the unmodified version from the Tomcat download site). You should now see a friendly Tomcat welcome page. If you get a "port 80 is already in use" message, go to the Windows Control Panel, Services, and stop the other server (probably IIS). Also, on Linux and Solaris, you need admin privileges to start servers on low-numbered ports like 80. So, if you use the preconfigured Tomcat version but do *not* want port 80, double click on Tomcat at the bottom and change the port from 80 to something else (see "HTTP/1.1" in the "Ports" section on the right). But port 80 is nicer so that you can use URLs of the form http://localhost/app/blah instead of http://localhost:8080/app/blah, so using 80 should be your first choice.
6. **Copy the ROOT (default) Web app into Eclipse.** Eclipse forgets to copy the default apps (ROOT, examples, docs, etc.) when it creates a Tomcat folder inside the Eclipse workspace. Go to C:\apache-tomcat-7.0.34\webapps and copy the ROOT folder. Then go to your Eclipse workspace, go to the .metadata folder, and search for "wtpwebapps". You should find something like C:\your-eclipse-workspace-location\.metadata\.plugins\org.eclipse.wst.server.core\tmp0\wtpwebapps (or .../tmp**1**/wtpwebapps if you already had another server registered in Eclipse). Go to the wtpwebapps folder and paste ROOT (say "yes" if asked if you want to merge/replace folders/files). Then reload http://localhost/ to see the Tomcat welcome page.
7. **Import and test a sample Web App.** Grab test-app.zip, save it, and import it into Eclipse. Use File, Import, General, Existing Projects, Select archive file. Then click Browse and navigate to test-app.zip. Click on Servers tab at bottom. R-click on Tomcat v7.0 Server, choose "Add and Remove Projects". Choose test-app project. Start Tomcat, or restart if already running. Open http://localhost/test-app/ in browser. Note that this app uses the servlet 3.0 @WebServlet annotation to provide the URLs for the various servlets. See the source code for details.
8. **Create and test a new Web App.** File, New, Project, Web, Dynamic Web Project. Make sure that the Target runtime is "Apache Tomcat v7.0". Copy HTML, JSP, and servlet files from the test-app project into this new project. Deploy and test it as above.
9. **Tweak Eclipse preferences.** Window, Preferences, then many options re font sizes, import and indentation styles, and so forth. At a minimum, set the JDK location: Window, Preferences, Java, Installed JREs, and make sure that a JDK (not just JRE) is selected. If not, press Add, and then navigate to the base JDK folder. If you develop with servlets, you probably also want to suppress unnecessary warnings about Serializable classes for servlets. Window, Preferences, Java, Compiler, Errors/Warnings, change "Serializable class without ..." to "Ignore".
10. **Bookmark the servlet & JSP Javadocs.** Add the servlet 3.0, JSP 2.2, and EL 2.2 API to your bookmarks/favorites list.
11. **Learn details of developing Java-based Web apps.** If you got through the previous steps, Eclipse and Tomcat are all set up, and you are ready to start the fun part: learning how to develop real apps! For this, you have two main choices:
    - Servlet and JSP programming. This is a very widely used library and is used by thousands of major sites. However, it is a bit low-level by the standards of modern Web apps.
    - JSF 2 programming. JavaServer Faces version 2 is a higher-level and more powerful library for building Web apps. JSF is even better when you add in a rich component library like PrimeFaces. JSF is recommended over servlets/JSP for most new projects.

    Both servlets/JSP and JSF 2 work well on Tomcat 7.

Install Java

Download for Windows, MacOS, Linux, and Solaris from http://www.oracle.com/technetwork/java/javase/downloads/. Get the standard edition (Java SE), not the enterprise edition (Java EE). Be sure to get the JDK (which has a compiler), not just the JRE (which is for running preexisting applications). As of early 2013, I used JDK 1.7.0_10 and 1.6.0_35 (the latest versions at the time), but any Java 6 or 7 version will work. Servlet 3.0 containers (of which Tomcat 7 is one) require Java 6+, and will not work with Java 5. Once you have downloaded the installer, run it and accept all default settings. There is no need to set any environment variables, but if you already had an earlier version of Java installed and you had set the PATH variable (pointing at the bin subfolder) or the JAVA_HOME variable (pointing at the main installation folder), it is a good idea to update them to the new version.

Also, please note that if you use 1.6.0_21, you also need to fix the Eclipse PermGen space error or Eclipse will run out of memory and crash. That is the *only* Java version where you have this problem.

Unzip Tomcat

Unzip tomcat-7.0.34-preconfigured.zip into the location of your choice. I use the top level of the C drive, resulting in *C:\apache-tomcat-7.0.34\*. This preconfigured version of Tomcat has the following settings already in place.

- **The port is changed from 8080 to 80.** This lets you enter URLs of the form *http://localhost/...* instead of *http://localhost:8080/...*.
    - When you download Tomcat from the Apache site, the port is 8080 in case you already have another server running on port 80.
- **Directory listings are turned on.** If you type a URL ending in / and there is no welcome file, Tomcat shows a directory listing.
    - Directory listings were on by default in previous Tomcat versions, but are off in the current version. They are convenient during development so you can just click on files, but most developers disable them for deployed applications.
- **Tomcat monitors struts-config.xml and faces-config.xml.** Whenever either of these files changes, Tomcat reloads the Web application.

This saves you from restarting the server when you change these files.
- If you do not use Struts or JSF, this change will not be beneficial to you. But it does not hurt either way.

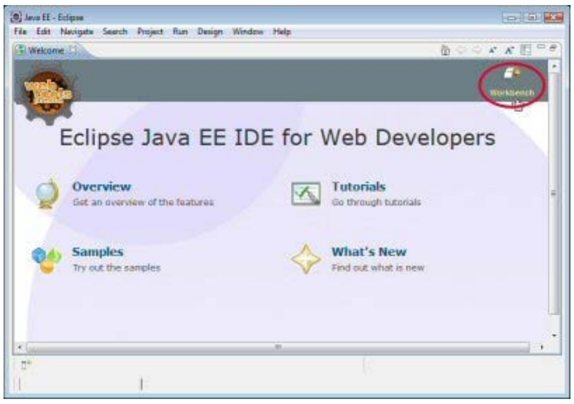
Alternatively, you can download any Tomcat 7 version from the Tomcat Web site and then copy context.xml, server.xml, and web.xml into install_dir/conf. These files are annotated with comments on what modifications were done to change the port to 80 and to enable directory listings and automatic server restarts.

Install and Start Eclipse

Go to http://www.eclipse.org/downloads/. Choose "Eclipse IDE for Java EE Developers", download, and unzip. As of early 2013, the latest version is Eclipse 3.8 (Juno), and is what I use as of the latest update of this tutorial. However, these instructions were also tested with Eclipse 3.8 (Indigo) and 3.6 (Helios). Eclipse 3.5 and earlier do not have adapters for Tomcat 7, so cannot be used. There is no real installer, so unzipping it is all you need to do. I normally unzip into the top level of the C drive, resulting in C:\eclipse, but any location is fine.

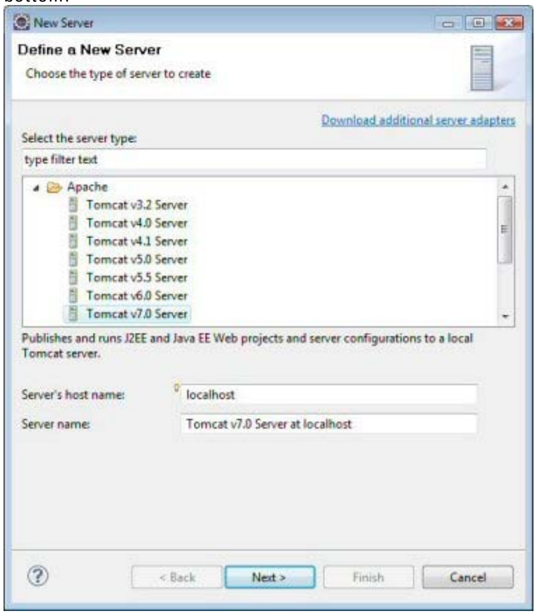
Start Eclipse by going to the "bin" folder and double-clicking on eclipse.exe. I usually make a shortcut on the desktop by R-clicking on eclipse.exe, selecting Copy, then going to the desktop, R-clicking, and doing Paste Shortcut.

After you start Eclipse, select "Workbench" as shown on the image to the right.



Tell Eclipse about Tomcat

First, start Eclipse and go to the Workbench as shown in the previous section. Then, click on Servers tab at bottom. (If you don't see Servers tab, add the tab via Window, Show View, Servers.) R-click on Servers tab, New, Server, Apache, Tomcat v7.0, navigate to the folder where you unzipped Tomcat (e.g., *C:\apache-tomcat-7.0.34\*), OK. You should now see "Tomcat v7.0 Server at localhost" listed under the Servers tab at the bottom.


New Server

Define a New Server

Choose the type of server to create

Download additional server adapters

Select the server type:

type filter text

- Apache
  - Tomcat v3.2 Server
  - Tomcat v4.0 Server
  - Tomcat v4.1 Server
  - Tomcat v5.0 Server
  - Tomcat v5.5 Server
  - Tomcat v6.0 Server
  - Tomcat v7.0 Server

Publishes and runs J2EE and Java EE Web projects and server configurations to a local Tomcat server.

Server's host name: localhost

Server name: Tomcat v7.0 Server at localhost

< Back | Next > | Finish | Cancel

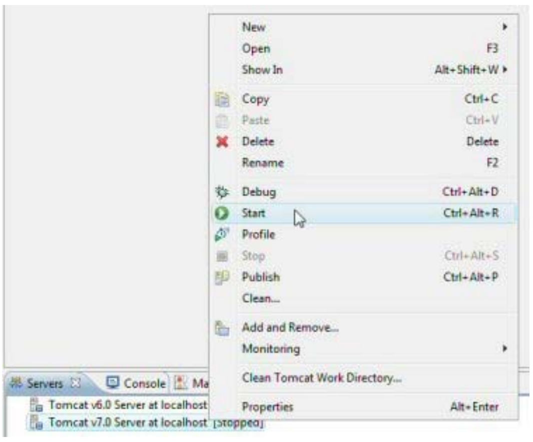
Run Tomcat

Click on Servers tab at bottom. R-click on Tomcat v7.0, choose "Start". Open http://localhost/ in a browser (or http://localhost:8080/ if you downloaded Tomcat instead of using the preconfigured version, and then failed to change the port from 8080 to 80). Either way, you will see a 404 error message, but at least the message comes from Tomcat. Then, copy the ROOT app as described in the next section, come back, and reload http://localhost/ (or http://localhost:8080/ if using the unmodified version from the Tomcat download site). You should now see a friendly Tomcat welcome page.

If you fail to copy the ROOT files as mentioned below, http://localhost/ will result in a 404 error. But, the error page comes from Tomcat, so it shows that Tomcat is running. The problem is that Eclipse doesn't copy the default Web application to the Eclipse/Tomcat folder. However, if you copy the ROOT files as described in the next section, http://localhost/ will give the nice friendly "Welcome to Tomcat" page.

If you get a "port 80 is already in use" message, that means you have another server already using port 80 (probably IIS). To stop the other server, go to the Control Panel, Services, and stop it from there. Or, double click on Tomcat at the bottom of the Eclipse window and change the HTTP/1.1 port (see the upper right) from 80 to something else. But using port 80 is nicer, so it is better to stop the other server instead of changing the Tomcat port.

It is also easy to start and stop Tomcat manually, without using Eclipse. Just go to the "bin" folder of the Tomcat installation directory (e.g., C:\apache-tomcat-7.0.34\bin) and double click on startup.bat and shutdown.bat. You can also manually deploy WAR files to the "webapps" folder of the Tomcat installation directory. However, doing everything in Eclipse is *much* more convenient because Eclipse automatically sends over changes as you modify files.



Copy the ROOT (Default) Web App into Eclipse.

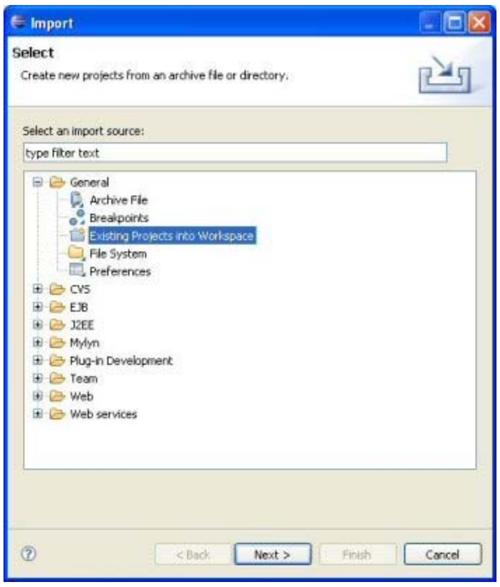
Eclipse forgets to copy the default apps (ROOT, examples, etc.) when it creates a Tomcat folder inside the Eclipse workspace. Go to C:\apache-tomcat-7.0.34\webapps, R-click on the ROOT folder and copy it. Then go to your Eclipse workspace, go to the .metadata folder, and search for "wtpwebapps". You should find something like *your-eclipse-workspace*\.metadata\.plugins\org.eclipse.wst.server.core\tmp0\wtpwebapps (or .../tmp**1**/wtpwebapps if you already had another server registered in Eclipse). Go to the wtpwebapps folder, R-click, and paste ROOT (say "yes" if asked if you want to merge/replace folders/files). Then reload http://localhost/ to see the Tomcat welcome page.

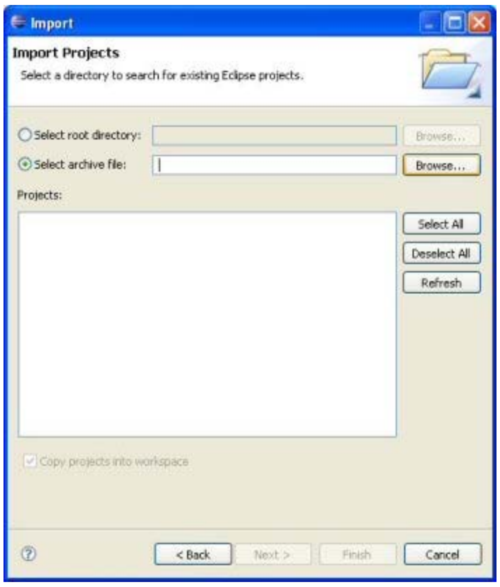
Import and Test a Sample App

Grab test-app.zip and save it. Import it into Eclipse with File, Import, General, Existing Projects, Select archive file. Then click Browse and navigate to test-app.zip. Note that this app uses the @WebServlet annotation to give URL patterns to servlets, and will only run in Tomcat 7 or other servlet 3.0 containers. It will *not* run in Tomcat 6.

Click on Servers tab at bottom. R-click on Tomcat v7.0 Server, choose "Add and Remove Projects". Choose test-app. Start Tomcat, or restart it if already running (R-click on Tomcat and choose either "Start" or "Restart"). Try the following URLs in a browser:

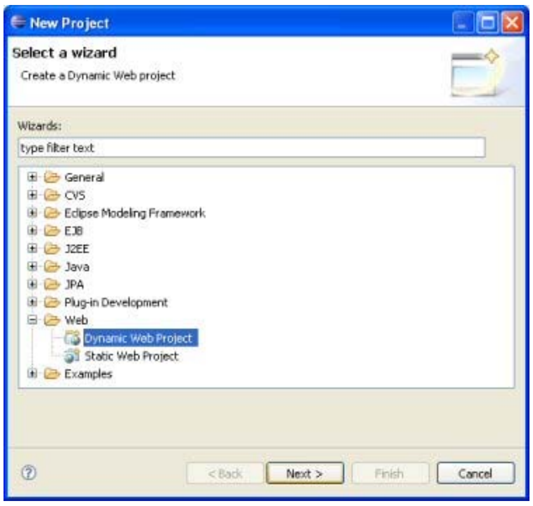
- **http://localhost/test-app/** Welcome page showing links to pages and servlets below.
- **http://localhost/test-app/hello.html** Simple HTML page.
- **http://localhost/test-app/hello.jsp** Simple JSP page.
- **http://localhost/test-app/hello** The HelloWorld servlet that generates plain text.
- **http://localhost/test-app/test1** The TestServlet that generates HTML, accessed via the URL given in the @WebServlet annotation in the Java source code.
- **http://localhost/test-app/test2** The TestServlet, accessed via the URL given in the web.xml file.

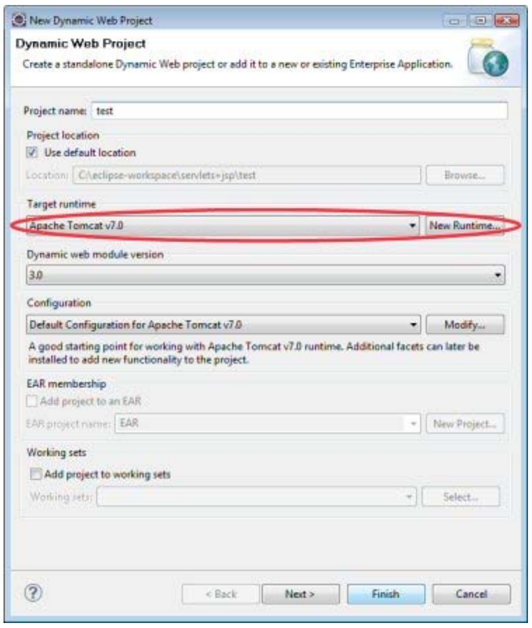




Create a New Web App in Eclipse

- **Make empty project.**
  - File, New, Project, Web, Dynamic Web Project. Eclipse remembers the recent project types, so once you do this once, you can just do File, New, Dynamic Web Project.
  - For "Target Runtime", choose "Apache Tomcat v7.0"
  - Give it a name (e.g., "test").
  - Accept all other defaults.





Project name: test

Project location
☑ Use default location
Location: C:\eclipse-workspace\servlets+jsp\test

Target runtime: Apache Tomcat v7.0

Dynamic web module version: 3.0

Configuration: Default Configuration for Apache Tomcat v7.0

A good starting point for working with Apache Tomcat v7.0 runtime. Additional facets can later be installed to add new functionality to the project.

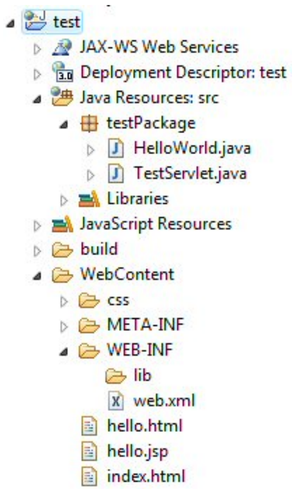
EAR membership
☐ Add project to an EAR
EAR project name: EAR

Working sets
☐ Add project to working sets
Working sets:

Add Code to New Apps in Eclipse

Here is a quick summary of the most commonly used folders in Dynamic Web Projects in Eclipse.

- **WebContent.**
  Regular Web files (HTML, JavaScript, CSS, JSP, images, etc.)
- **WebContent/some-subdirectory**
  Web files in subdirectory.
- **WebContent/WEB-INF**
  web.xml. This deployment descriptor be used for servlet mappings and many other tasks. However, this file can be *completely* omitted in servlet 3.0 apps, since servlet mappings can be done via the @WebServlet annotation in the Java source code.
- **WebContent/WEB-INF/lib**
  JAR files specific to application.
- **src/testPackage**
  Java code in testPackage package. Make a package by R-clicking on "Java Resources: src" and doing New, package. Always make packages: use of the default package is strongly discouraged in Web apps.
- **Note:**
  You can cut/paste or drag/drop existing files into the appropriate locations.



Test New Apps in Eclipse

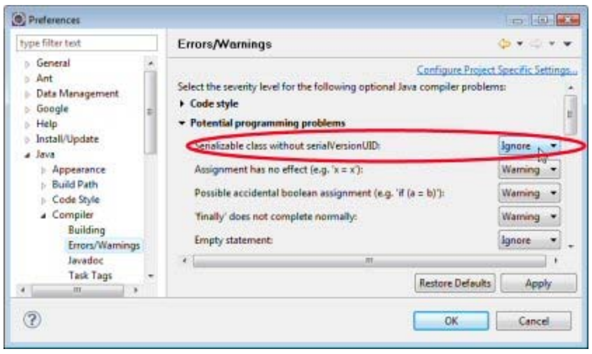
Follow same procedure as given in example above with "test-app" app: Click on Servers tab at bottom. R-click on Tomcat v7.0 Server, choose "Add and Remove Projects". Choose app. Start Tomcat, or restart if already running. Open http://localhost/*appName*/ in browser. You must restart the server after adding a new project. To do so, R-click server and choose "Restart".

For details on writing servlets, giving them custom URLs, using JSP, applying the MVC architecture, creating JSP custom tag libraries, and many other servlet and JSP topics, please see the servlet and JSP programming tutorial. You might also consider JSF 2, a higher-level and much more powerful library that is a standard part of Java EE 6, but that also runs fine in Tomcat and other servlet/JSP containers. For details, please see the JSF 2 programming tutorial.

Adjust Eclipse Preferences

There are many customizations you can do in Eclipse. You can adjust the font size, colors, indentation styles, import statement formats, and much much more. Go to Window, Preferences and browse around.

At a minimum, set the JDK location: Window, Preferences, Java, Installed JREs, and make sure that a JDK (not just JRE) is selected. If not, press Add, and then navigate to the base JDK folder. Another customization you almost always want to do is to suppress the unneeded warning re serialVersionUID on Serializable classes. The HttpServlet class is already Serializable, but nobody actually sends instances across the network or writes them to disk, and certainly almost nobody puts in a serialVersionUID field in servlets. But, if Eclipse marks every servlet with a warning, you get in the very bad habit of ignoring warnings. Most of the warnings are actually useful; this is a rare exception. So, go to Window, Preferences, Java, Compiler, Errors/Warnings, expand "Potential programming problems", and change "Serializable class without serialVersionUID" to "Ignore".



Select the severity level for the following optional Java compiler problems:

Serializable class without serialVersionUID:	Ignore
Assignment has no effect (e.g. 'x = x'):	Warning
Possible accidental boolean assignment (e.g. 'if (a = b)'):	Warning
'finally' does not complete normally:	Warning
Empty statement:	Ignore

Access the Servlet and JSP Javadocs

The single most useful reference for a servlet and JSP developer is the online API. Sadly, as of mid-2012, there was no online version of the servlet 3.0 and JSP 2.2 Javadocs, and browsing the massive Java EE 6 API is much more difficult because the servlet and JSP classes get lost in the crowd. So, I generated the API from the Tomcat 7 source code. You can access the result at http://docs.coreservlets.com/servlet-3.0-api/. And, if you haven't already done so long ago, bookmark the Java SE 7 API or Java SE 6 API.

More Information

Java

- Java SE 7 API
- Java SE 6 API
- Java SE Downloads for Windows, Linux, & Solaris
- Java SE Downloads for MacOS 10.6
- Java EE 6 API. Includes the servlet 3.0, JSP 2.2, and JSF 2.0 APIs.
- Java EE 6 Technologies. Includes servlet 3.0, JSP 2.2, JSTL 1.2, and JSF 2.0 specifications
- Java EE 5 API. Includes the servlet 2.5, JSP 2.1, and JSF 1.1 APIs.
- Java 6 & 7 Programming Tutorial
- Java 6 & 7 Training Courses
- Recommended Java Programming Books
- General Java Programming Resources

JSF (JavaServer Faces)

- JSF 2 Tutorial (with Eclipse)
- JSF 2.1 Documentation Home
- JSF 2.1 Java API
- JSF 2.1 Facelets Tags API
- JSF 2.1 Managed Beans Annotations API
- JSF 2.0 Java API
- JSF 2.0 Facelets Tags API
- JSF 2.0 Managed Beans Annotations API
- JSF 2 Specifications
- PrimeFaces Home Page
- RichFaces Home Page
- JSF 1.2 Java API
- JSF 1.2 Tag Library API
- Apache MyFaces Documentation
- JSF 2 Tutorial (plus PrimeFaces Intro)
- JSF 1 Tutorial.
- JSF 2.*x* Training Courses (optionally with PrimeFaces)
- Recommended JSF Books

Servlets & JSP

- Servlet 3.0, JSP 2.2, and EL 2.2 API. Supported by Tomcat 7.*x*.
- Tutorial on setting up Tomcat 7 and Integrating it with Eclipse
- Servlet 2.5 API. Supported by Tomcat 6.*x*.
- JSP 2.1 API. Supported by Tomcat 6.*x*.
- Servlet 2.4 API. Supported by Tomcat 5.*x* and 6.*x*.
- JSP 2.0 API. Supported by Tomcat 5.*x* and 6.*x*.
- Beginning/Intermediate Servlet and JSP Tutorial
- Advanced Servlet and JSP Tutorial
- Servlet and JSP Training Courses
- Recommended Servlet & JSP Books

Ajax, GWT, & JavaScript

- Top-Level GWT Docs Page
- GWT Java API
- JavaScript Reference: Core Objects
- JavaScript Reference: HTML DOM
- jQuery API
- jQuery UI Docs
- Script.aculo.us API
- Prototype API
- Dojo API
- Ext JS API
- YUI API
- Google Closure Home
- Ajax Tutorial (includes jQuery, Dojo, and other JavaScript Frameworks)
- GWT 2 Tutorial
- Ajax Training Courses (with jQuery, etc.)
- GWT 2 Training Courses
- Recommended Ajax-Related Books: Ajax, GWT, Core JavaScript, JavaScript Frameworks, HTML, XHTML, & CSS.

Spring, Hibernate, & JPA

- Spring 3.0 API
- Spring 3.0 Reference Manual
- Spring 2.5 API
- Spring 2.5 Reference Manual
- Top-Level Hibernate Docs Page
- JPA API
- Spring Framework Tutorial
- Spring Training Courses
- Hibernate and JPA Tutorial
- Hibernate Training Courses
- Recommended Spring Books
- Recommended Hibernate Books

Struts

- Struts 1.x Developer and User Guides
- Struts 1.x API
- Struts 2 Overview
- Struts 2 Developer's Guide
- Struts 2 API
- Struts Tutorial
- Recommended Struts Books