Inefficiency...

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/JSF 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 examples of all the sections 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 help@coreproducts.com.

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

1. Install Java. Download for Windows, Mac OS, 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.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 small changes: enabling directory listings, and making the server automatically restarts when facess-config.xml or struts-config.xml changes. These changes are useful during development, but only the port number change should be set for deployable servers.

3. 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 Tomcat Server tab, and the tab via Window, Show View, Servers.

4. Run Tomcat. Click on Servers tab at bottom. R-click 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 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) (using the unmodified version from the Tomcat download site). You should now see a friendly Tomcat welcome page.

5. 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 to your Eclipse workspace, over the meta-data folder, and search for “webapps” app. You should find something like C:\your-eclipse-workspace\your-tomcat-\webapps\ROOT (or, if you had another server registered in Eclipse). Go to the webapps 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.

6. 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 Tomcat v7.0, Server, choose Add and Remove Projects. Choose test-app project. Start Tomcat, or restart if already running. Open http://Localhost/testapp in browser. Note that this app uses the servlet 3.0 Servlet annotation to provide the URLs for the various servlets. See the source code for details.

7. Create a test and a new Web App. File, New, Project, 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. Your new project should now be up and running.

8. Tweak Eclipse preferences. Window, Preferences, then many options re font sizes, 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 deprecated classes for servlets, JSPs, and JSF (by unchecking “warn about deprecated classes for...” under “JSP/JavaServer Pages, Java, Compilers, Errors/Warnings, change “Serializable class without...” to “Ignore”).

9. Bookmark the servlet & JSP Javadoc. Add the servlet 3.0, JSP 2.2, and El 2.2 ADI to your bookmarks/favorites list.

10. Learn more about Java web apps. For this part, I have two main choices:
   - Book: Java and XML. 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.
   - Book: Java EE. 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/JSF for most new projects.

Both servlets/JSF 2 work well on Tomcat 7.
LinkedIn Study: “Developers spend 13 hrs / wk configuring their environment.”

Costs & Risks

**DEVELOPER CONFIGURATION**
- New Projects
- Changing Branches
- Updating Tools & Plug-ins
- Compile Lag
- Code Reviews
- Failed Build investigations
- Desktop Resource Thrashing
- VM Synchronization
- Merge Conflicts

**TEAM CHALLENGES**
- Unnecessary Hardware
- Tribal Knowledge
- BYOD Incompatibilities
- Environment Debt
- Time Tracking
- Hackathon / Classroom Setup
- Recruiting Overhead

**SECURITY RISKS**
- IP Security
- External Threat Vectors
Developer Environment

IDE
- Project
- SCM Client
- Code
- Editor
- Code Assistants
- Debugger
- Deployer
- Plug-Ins

Other
- Builder
  - Compiler
  - Unit Tester
  - Packager
- Runner
  - Frameworks
  - App Servers
  - Databases

Different lifecycles for installation, update, configuration, and sunset.
THE WORLD NEEDS

Developer Environment Clouds

Project
SCM Client
Code
Editor
Code Assistants
Builder
→ Compiler
→ Unit Tester
→ Packager
Runner
→ Frameworks
→ App Servers
→ Databases
Debugger
Deployer
Plug-Ins

Fast
Secure
Extensible
Instant

SAAS
On-Premise
SDK
How A Developer Environment Cloud WORKS

1. **Provision Instantly**
   - Install nothing. No download. No maintenance. Complete and comprehensive.
   - Create environments for every task.

2. **Use Existing IDE & ALM Tools**
   - Use your favorite desktop IDE or our browser IDE. Integrate existing ALM systems with the CLI & API.

3. **Clone, Embed, and Share**
   - Single click copying and sharing environments. Link and embed in other products. Restrict access with privacy.

4. **Scale Development**
   - Add CPU & RAM to environments. Basket environments to maximize density with minimum hardware. Connect identity & firewalled systems.

5. **Manage, Analyze, and Extend**
   - Quarantine IP, set feature restrictions, and track time. Automate internal processes with custom extensions.
FACTORIES FOR INSTANT: PROVISIONING

OLD WAY
Never ending steps, manually executed.
Failure and abandonment rates high.

NEW WAY: 1 CLICK
Shorter setup cycles. Predictable configuration.
Testable and part of continuous delivery processes.

http://codenvy.com/factory
Fast
- Install nothing, cached client
- Separation processing
- Multi-user development
- Multi-user processing
- Builder providers
- Centralized dependency mgmt
- Automated, cached dependencies
- Docker runner recipes
- Pre-configured, cached runners
- Factories for instant access
- Queues for multi-processing
- Anticipatory elasticity

Secure
- Public / private projects
- Quarantine environments
- Tokened access to services
- Secure protocols
- On premises installation
- Behavioral restrictions
- Multiple workspaces

Extensible
- Add / remove plug-ins
- Client- and server- extensions
- Themes
- Editor & key binding choice
- Swap builder & runners
- Customize packaged environments
- Custom RAM allocations
- Author custom extensions
- Extension tooling

1/3 the time & cost of VDI, desktop DIY, cloud IDEs, or Vagrant
Compliance violations reduced by 90%
Who are your Devis?

1. **The Celebrity**
   The engineer we try to hire ... because he has vision and communicates it brilliantly.

2. **The Power Dev**
   The engineer with near magical abilities. We love and fear them.

3. **The Contributor Dev**
   Everyone else who touches code: engineers, QA, docs, operations, design, and PM. They are the heart of a project.
The Sync Problem

Power Developer
- Code, Test, IDE, Plug-In, & Artifact Creation

DevOps
- Code Repo
- Integration
- Environments
- Automated

Sync

Contributor Developer
- Design, QA, Support, Docs, Partners, Mgrs
- Not Automated
The Codenvy Cloud