



Examining Tomcat's Directories

In the previous chapter, you saw how to install Tomcat on various platforms using the binaries or the source as the fancy takes you. Now it's time to look at the directories that make up the Tomcat installation. You'll be introduced to the main configuration files and the structure of a web application. However, I'll leave the details on configuration until Chapter 4.

In this chapter you'll do the following:

- You'll examine the default Tomcat installation.
- You'll learn about the generic web application structure, both unpacked and packed.

Looking at CATALINA_HOME

The best place to start a discussion of Tomcat's directory structure is in the default installation directory, commonly called `CATALINA_HOME`. If you've installed Tomcat, then you'll have an environment variable pointing to this directory. Let's start by examining the `bin` directory and all the scripts it contains.

The `bin` Directory

The `bin` directory contains many scripts (in Windows they're called *batch files*, but the term *scripts* will do for the sake of brevity) for starting Tomcat in different modes and for stopping Tomcat, a number of utilities, and some Windows-specific executables. Previous versions of Tomcat (prior to Tomcat 5.5.x) came with several different scripts to start and stop Tomcat. With this release, these have been consolidated into the following executables.

The `tomcat6` Windows Executable

You can use the `tomcat6` executable to run the server if it's installed as an NT service. You can install Tomcat as a service when you first install Tomcat, as described in the previous chapter.

Note that the name of this executable must be the same as that of the service you want to start. So, if you install the service as `TomcatServ`, you must rename this file `TomcatServ.exe` if you want to use its services.

The tomcat6w Windows Executable

You can use the `tomcat6w` executable to run the Tomcat Properties box if Tomcat is installed as a service. Chapter 2 described this utility. You can use it to start and stop the service and set other options.

Note that the name of this executable must be the same as that of the service you want to start, with a `w` appended. So, if you install the service as `TomcatServ`, you must rename this file `TomcatServw.exe` if you want to use its services.

The conf Directory

The `conf` directory contains the following Tomcat configuration files:

- `catalina.policy` sets up the necessary permissions for Catalina when it's run within the context of a security manager.
- `catalina.properties` sets the locations of the various class loader directories. The defaults are the `common`, `server`, and `shared` directories and their subdirectories. The settings in this file determine which classes are available to all web applications and which classes are available to Tomcat. In other words, these settings configure the classpath of Tomcat and all web applications.
- `context.xml` is a file that sets the defaults for individual contexts.
- `logging.properties` is a file that manages the default logging levels for the Tomcat server itself.
- `server.xml` is the main configuration file for Tomcat and is discussed in detail in Chapter 4. You use it to configure everything from the shutdown command to logging, filtering, connections to other web servers, the port and host on which the server is running, and the location of each web application's files.
- `tomcat-users.xml` is the default user database for container-managed authentication. You can change the name and location of the file in `server.xml`. You'll see more on this mechanism in Chapter 4.
- `web.xml` is the default deployment descriptor for all web applications. Tomcat processes it before processing the `web.xml` files in the server's web applications.

The logs Directory

The `logs` directory is the default location for application log files.

You may have to schedule housekeeping tasks to ensure that the size of the `logs` directory doesn't grow out of hand.

The lib Directory

This directory contains all the various JAR files for Tomcat.

The temp Directory

Tomcat uses the `temp` directory for storing temporary files.

The webapps Directory

The webapps directory is the default location of Tomcat's web applications. You can change this location, and it's recommended that you do so, as you can then separate the application files that change relatively frequently from the server files that don't tend to change much. As a bonus, the installation directory for Tomcat can be kept as read/write for the administrator only, thus maintaining greater security—read/write access for other users need be provided only for the now separate webapps folder.

You can deploy web applications in webapps by placing them here, in both packaged and unpackaged formats, and they will be automatically deployed at the next server bootup. This is an alternative to the `conf/[Service_name]/[Host_name]` method and the various deployer (such as Ant and the Tomcat manager application) methods. These are discussed in later chapters.

The work Directory

The work directory is where Tomcat places the JSP code after it has been converted into servlet code. Once a JSP page has been visited, Tomcat also stores the compiled servlet here.

Understanding Web Application Structure

A web application is a collection of web resources, such as JSP pages, HTML pages, servlets, and configuration files, organized into a hierarchy as specified in the Servlet specification. You have two ways in which to organize a web application: packed and unpacked. The packed form is called a web archive (WAR) file, and the unpacked form is a collection of directories stored on the file system.

The unpackaged format is convenient for web application developers, as it allows them to replace individual files while the application is being developed and debugged.

However, in a deployment environment, it's often more convenient to provide a single file that can be automatically deployed. This reduces the deployment process to placing the file and setting up system resources. Tomcat can also automatically expand a web application once the server has booted. The automatic expansion of WAR files is configured in the `server.xml` file as part of the `<Host>` element that configures hosts.

Web Application Context

Each web application corresponds to a context component, as discussed in Chapter 1, and you assign a context path to each. The default context is called `ROOT` and corresponds to the name of the server with no other context information. For example, the `ROOT` web application on your local machine will correspond to `http://localhost:8080`. If you've configured Domain Name System (DNS) settings for your server, it may also be accessible from a location such as `www.companyname.com`.

Users access other web applications by requesting a context relative to the server. For example, users can access Tomcat's manager web application with the following URL: `http://localhost:8080/manager`.

Applications that you place in the webapps folder are named after the directory they're in. So, you can access the web application in the `tomcat-docs` directory with the following: `http://localhost:8080/tomcat-docs`. Each application on the server is known by its name, and users can access resources according to the remainder of the uniform resource locator (URL) after the web application's name.

This setup has a slight problem, however. If the ROOT web application contains a subdirectory that has the same name as a web application, and that web application and that subfolder have filenames in common, then the applications won't work as expected. For example, the following are two web applications that could cause confusion:

```
webapps/  
  ROOT/  
    tomcatBook/  
      index.html  
  tomcatBook/  
    index.html
```

In this case, `http://localhost:8080/tomcatBook` could map to both files and could cause confusion. Tomcat will display the `index.html` page from the `tomcatBook` web application and will ignore the folder in the ROOT web application. If your users are expecting the ROOT version, then they will be disappointed.

The WEB-INF Directory

The Servlet specification sets out how you partition web applications into public and private areas. You store the private resources in a directory called `WEB-INF` in the root of the web application. This is where you store all the web-application-specific configuration files, application classes, and application-specific utilities. Users may only access these resources indirectly (for example, through servlet mappings).

`WEB-INF` has a number of specialized subdirectories where you store specific files, such as tag files and tag library descriptors (TLDs). These are defined in the appropriate specification, be it for servlets or JSP pages. You'll deal with them in detail in Chapter 5 when you configure a web application, but here's a quick rundown:

```
webAppX/  
  WEB-INF/  
    classes/  
    lib/  
    tags/
```

The `classes` and `lib` directory follow the usual pattern in Tomcat; you place class files in `classes` and JAR files in `lib`. `tags` is a special directory for tag files, which are a part of the JSP 2.0 specification.

The META-INF Directory

The `META-INF` directory is placed at the root of a web application when it's deployed as a WAR file. This is where you place tag TLDs and tag files, so that they can be found using a unique uniform resource indicator (URI). If no context XML files for this WAR file exist, then you can also place one in this directory. You'll find more details of this directory in Chapter 5.

Summary

This chapter outlined the contents of Tomcat's installation directory, its subdirectories, and the scripts they contain. This information is all you need to manage Tomcat's operation, from startup to shutdown.

The chapter also covered the structure of a web application, without going into the details of configuration. It has given you a familiarity with Tomcat's internals and prepared you for the coming chapters.