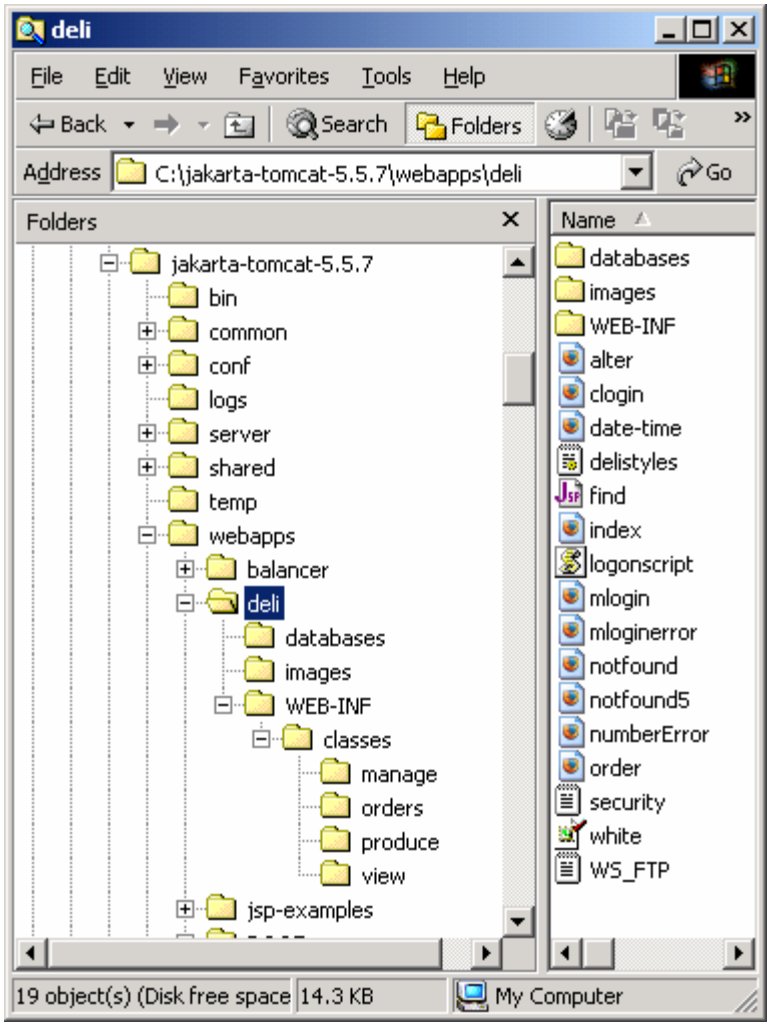
Configuring Tomcat for a Web Application

In order to configure Tomcat for a web application, files must be put into the proper places and the web.xml file should be edited to tell Tomcat where the servlet files are. You can create a new subfolder under the webapps folder. This folder can be the *root* of your application. Under this folder you will need a WEB-INF folder to contain the web.xml file and under that a classes file to hold the servlet class files.

As an example, consider a web application that concerns a delicatessen. The folder can be called deli, and the files can be stored in it. See the directory picture from Windows Explorer below.



From this you can see that the classes folder contains subfolders, manage, orders, produce, and view that are the names of the package files for the servlets. Anything in the deli folder is available to clients on the web, but any files in the WEB-INF folder and below are not. This provides a level of security for your development files.

Also note that in the grocery folder there are several html files, including one called index.html. It is the main web file for the application. If the web.xml file contains a welcome-file-list, it can be accessed using the URL

http://localhost/deli or http://localhost:8080/deli.

Other html and xml files can also be stored in this folder.

Web.xml file

The web.xml file is the *web application deployment descriptor*. It contains the information needed for the server to load and initialize the web application. It lists the servlets that are in the application and also contains patterns for finding them. The first thing in the file is the information about the XML Schema used to define the web.xml document. Because this is part of Tomcat, the xml elements are determined and must be written exactly as given.

The elements in the file that we will use (in the order they should be included) are: display-name, description, context-param, servlet, servlet-mapping, welcome-file-list, and error-page. They must be contained between the tags, <web-app></web-app> and are each listed below with examples. We will add several more when we add security to the application.

display-name gives a name for the application that can be used by management tools.

```
<display-name>Deli Application</display-name>
```

description is used to store a description of the application.

```
<description>
    A delicatessen application that can access a database of fruits and vegetables. Clients will be
    able to display, find, delete, and add products, and also to change prices.
</description>
```

context-param is used to provide initialization parameters for the application.

```
<context-param>
    <param-name>Author</param-name>
    <param-value>Carol Wolf</param-value>
    <description>Pace University</description>
</context-param>
```

servlet contains information about the logical names and class names of all the servlets in the application. One example is shown here.

```
<servlet>
    <servlet-name>DisplayServlet</servlet-name>
    <servlet-class>produce.DisplayServlet</servlet-class>
    <load-on-startup>10</load-on-startup>
</servlet>
```

The servlet-name is the name of the java file and the servlet-class is the full name of the class file. Note that it includes the package name. The load-on-startup tag tells the server to load this file when the application is started and give it priority 10. If that is the lowest priority level, it will be loaded first. Servlets are loaded in the order of the load-on-startup number. The first two tags are required, but the third is optional.

servlet-mapping is used to create a pattern that can be used to reference the servlet. It provides a short-hand way to name the servlet.

```
<servlet-mapping>
    <servlet-name>DisplayServlet</servlet-name>
    <url-pattern>/display/*</url-pattern>
</servlet-mapping>
```

In the example above, whenever a request begins with display, this servlet will be started. So if in an html file we have <form method = "get" action="../deli/display">, the server will know to start the DisplayServlet program, and the servlet section above says it is in produce.DisplayServlet.

welcome-file-list is used to list the files that are to be used when the directory is called for.

```
<welcome-file-list>
    <welcome-file>index.html</welcome-file>
</welcome-file-list>
```

In this example, when the URL requested is http://localhost/deli, the index.html file in the deli folder will be sent to the client.

error-page provides a page to be used when the server encounters an error, either a file missing or an exception.

```
<error-page>
    <error-code>404</error-code>
    <location>/notfound.html</location>
</error-page>
```

This example says that when code 404, file not found, is returned, send the notfound.html page to the client.

There are other elements that can be part of web.xml. Some of these are used for filtering data that is sent back and forth to the client and for security. There will be more on security later.

index.html

The index file is the one that is sent to the client when the URL contains the name of the directory. You can use any name for it, but index.html is a common one. It should be listed in the welcome-file-list element. A simple example follows:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>
<head> <title>Grocery Store </title>
</head>

<body bgcolor = "cyan">
    <h3>Display the Data</h3>
    <form method = "get" action="../deli/display">
        <p><input type="submit" value="Send"></p>
    </form>
</body>
</html>
```

A complete example of a web application deployment descriptor follows. It is for the deli example described above.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation=
```

```
    "http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
    version="2.4">

    <display-name>Deli Application</display-name>

    <description>
        A delicatessen application that can access a database of grocery items.
        Clients will be able to display and find products.
    </description>

    <context-param>
        <param-name>Author</param-name>
        <param-value>Carol Wolf</param-value>
        <description>Pace University</description>
    </context-param>

    <!-- The list of servlets used in the application. -->
    <servlet>
        <servlet-name>DisplayServlet</servlet-name>
        <servlet-class>produce.DisplayServlet</servlet-class>
        <load-on-startup>10</load-on-startup>
    </servlet>
    <servlet>
        <servlet-name>FindServlet</servlet-name>
        <servlet-class>produce.FindServlet</servlet-class>
    </servlet>

    <!-- The mappings used for the servlets. -->
    <servlet-mapping>
        <servlet-name>DisplayServlet</servlet-name>
        <url-pattern>/display/*</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>FindServlet</servlet-name>
        <url-pattern>/find/*</url-pattern>
    </servlet-mapping>

    <!-- The home page for the application. -->
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
    </welcome-file-list>

    <!-- The page to be used when a “file not found” error occurs. -->
    <error-page>
        <error-code>404</error-code>
        <location>/notfound.html</location>
    </error-page>
</web-app>
```

Reference

1. Karl Moss, *Java Servlets Developer’s Guide*, McGraw-Hill/Osborne, 2002.