CNT 4714: Enterprise Computing Spring 2011

Introduction to Servlet Technology– Part 2

Instructor : Dr. Mark Llewellyn
markl@cs.ucf.edu
HEC 236, 407-823-2790
http://www.cs.ucf.edu/courses/cnt4174/spr2011

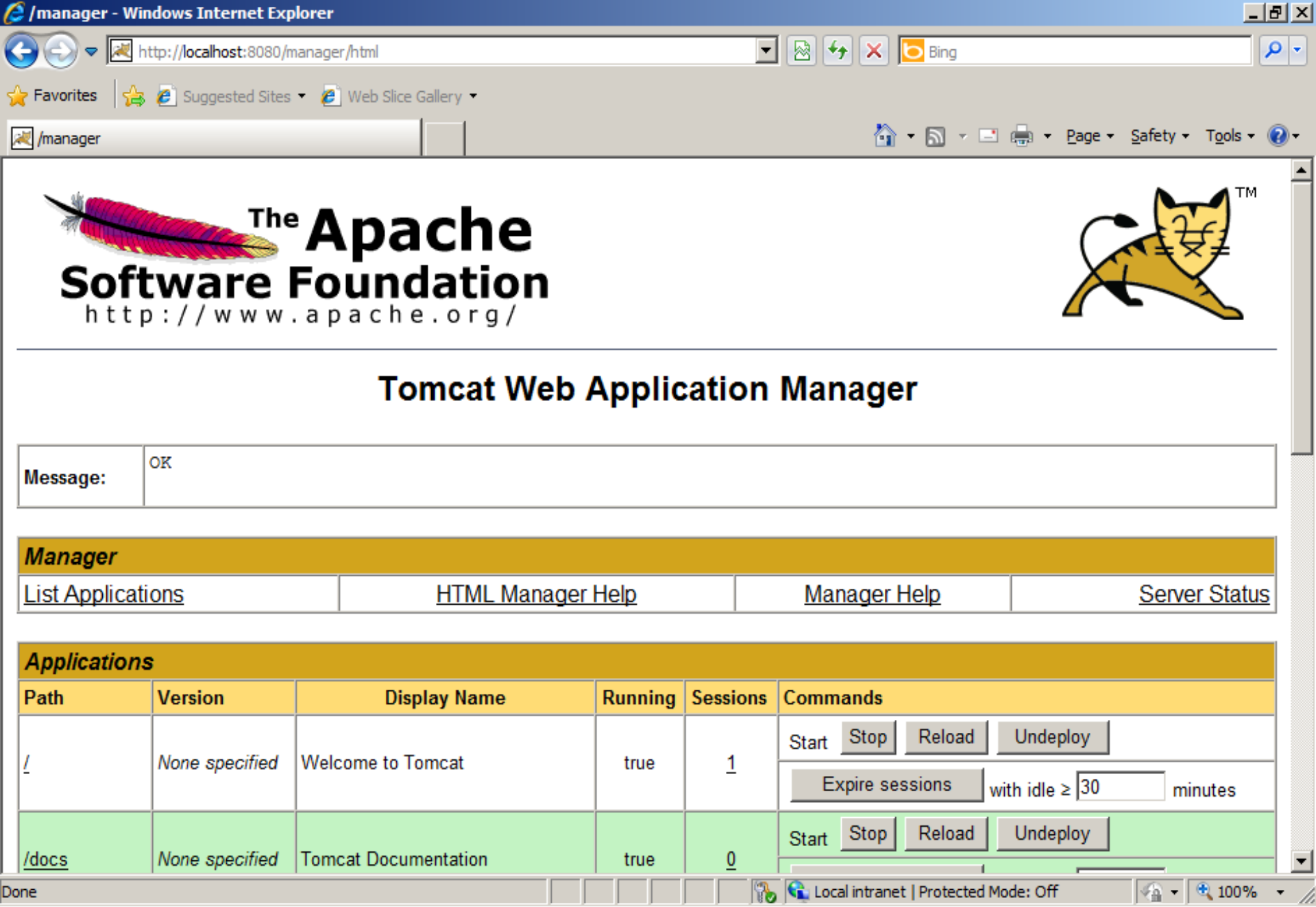
Department of Electrical Engineering and Computer Science
University of Central Florida

More Tomcat Details

- If your system does not recognize "localhost", enter http://127.0.0.1:8080 instead of http://localhost:8080. Address 127.0.0.1 basically means "this machine" which is the same as localhost.

- From the Tomcat homepage you can also act as the server administrator and manager. While we won't need to do anything on the administrator side (you must have set the host manager application during the installation process, (see page 27 of Servlets Part 1)), it is interesting to go into the manager side of things and look at the server from the server's point of view. It may also be necessary to reload applications occasionally (more on this later), which can be done from the manager application. See page 3 for an example.

- Checking the status of the server can also be accomplished from the Tomcat homepage. See page 4 for a sample.


/manager - Windows Internet Explorer
http://localhost:8080/manager/html
Bing
Favorites
Suggested Sites
Web Slice Gallery
/manager
Page
Safety
Tools
The Apache Software Foundation
http://www.apache.org/
TM
Tomcat Web Application Manager
Message:
OK
Manager
List Applications
HTML Manager Help
Manager Help
Server Status
Applications
Path
Version
Display Name
Running
Sessions
Commands
/
None specified
Welcome to Tomcat
true
1
Start
Stop
Reload
Undeploy
Expire sessions
with idle ≥
30
minutes
/docs
None specified
Tomcat Documentation
true
0
Start
Stop
Reload
Undeploy
Done
Local intranet | Protected Mode: Off
100%

/manager - Windows Internet Explorer

http://localhost:8080/manager/status

Bing

Favorites | Suggested Sites | Web Slice Gallery

/manager

Page | Safety | Tools

The Apache Software Foundation

http://www.apache.org/

Server Status

Manager			
List Applications	HTML Manager Help	Manager Help	Complete Server Status

Server Information					
Tomcat Version	JVM Version	JVM Vendor	OS Name	OS Version	OS Architecture
Apache Tomcat/7.0.8	1.6.0_23-b05	Sun Microsystems Inc.	Windows Server 2008	6.0	x86

JVM

Free memory: 3.96 MB Total memory: 7.87 MB Max memory: 123.75 MB

"ajp-bio-8009"

Done

Local intranet | Protected Mode: Off

100%

A Tour of Tomcat

- Before we look into creating our own servlets, we need to look more closely at Tomcat. This will help you better understand how web applications are developed and deployed.

- The directory structure within Tomcat looks like the one shown on the next page. It contains, among other things, seven directories named, `bin`, `conf`, `logs`, `lib`, `webapps`, `work`, and `temp`.

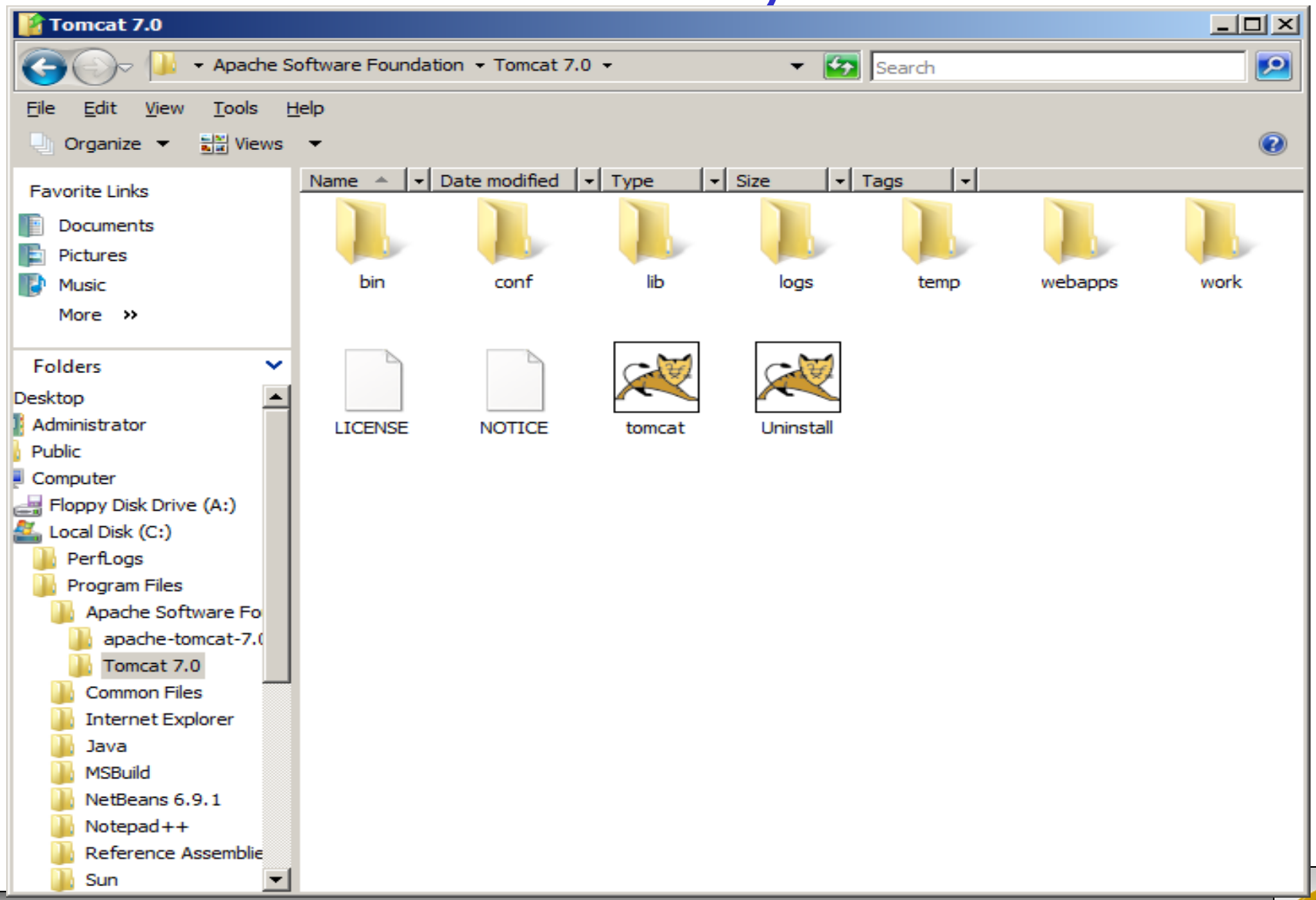
`bin`

- Directory `bin` contains scripts for starting and stopping Tomcat as well as some additional tools.

`conf`

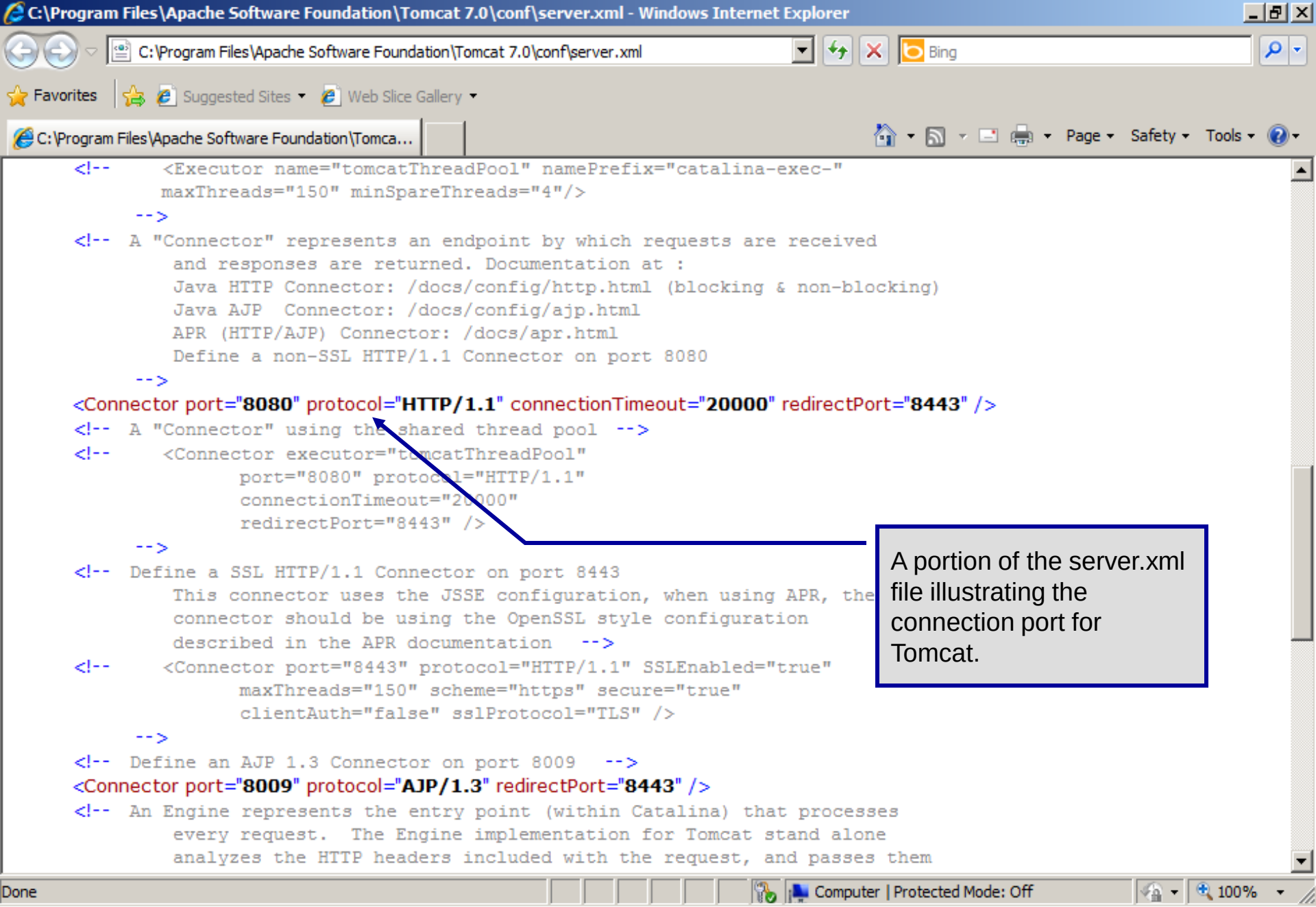
- Directory `conf` contains files used to configure Tomcat at the global level, although it is possible for each web application to override many of the values provided in this directory.

Tomcat Directory Structure

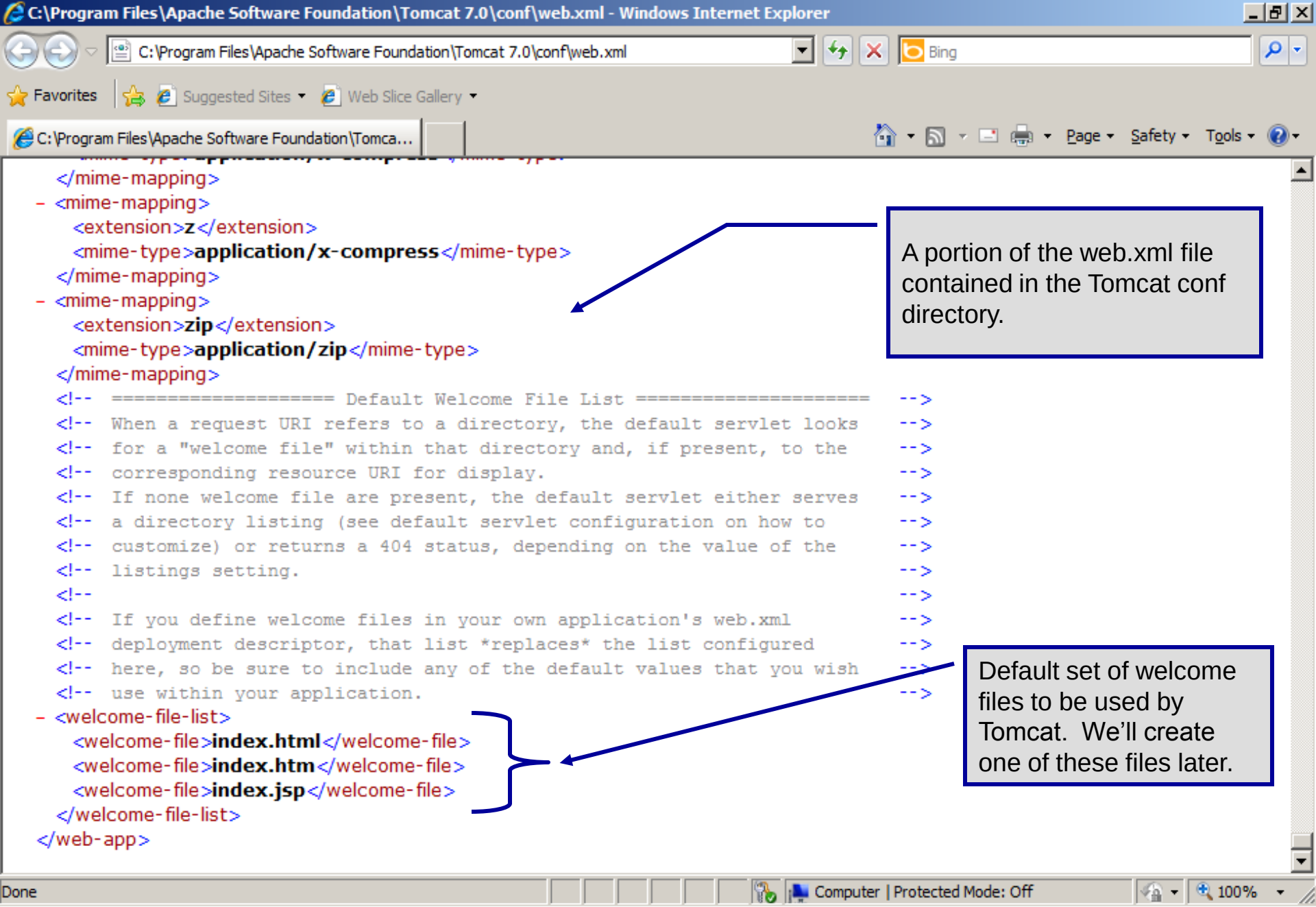


A Tour of Tomcat (cont.)

- The most important file inside the `conf` directory is `server.xml`, which tells Tomcat the set of services to run when it starts up as well as what port to listen to. This file also specifies the set of resources to make available to applications and a number of security parameters. A portion of this file (the part illustrating the non-SSL HTTP port) is shown on page 8.
- There is also a `web.xml` file in this directory, which establishes default values that may be overridden by values in each applications `web.xml` file. A portion of this file is shown on page 9.
- The file `jk2.properties` defines a set of properties that are used when Tomcat is installed as an application server in conjunction with an external web server such as Apache or IIS. In these notes we will assume that Tomcat is running in stand-alone mode, where it operates as both a web server and application server.



A portion of the server.xml file illustrating the connection port for Tomcat.


C:\Program Files\Apache Software Foundation\Tomcat 7.0\conf\web.xml - Windows Internet Explorer
C:\Program Files\Apache Software Foundation\Tomcat 7.0\conf\web.xml
Favorites
Suggested Sites
Web Slice Gallery
C:\Program Files\Apache Software Foundation\Tomca...
Page
Safety
Tools
</mime-mapping>
<mime-mapping>
<extension>z</extension>
<mime-type>application/x-compress</mime-type>
</mime-mapping>
<mime-mapping>
<extension>zip</extension>
<mime-type>application/zip</mime-type>
</mime-mapping>
<!-- ==================== Default Welcome File List ==================== -->
<!-- When a request URI refers to a directory, the default servlet looks -->
<!-- for a "welcome file" within that directory and, if present, to the -->
<!-- corresponding resource URI for display. -->
<!-- If none welcome file are present, the default servlet either serves -->
<!-- a directory listing (see default servlet configuration on how to -->
<!-- customize) or returns a 404 status, depending on the value of the -->
<!-- listings setting. -->
<!-- -->
<!-- If you define welcome files in your own application's web.xml -->
<!-- deployment descriptor, that list *replaces* the list configured -->
<!-- here, so be sure to include any of the default values that you wish -->
<!-- use within your application. -->
<welcome-file-list>
<welcome-file>index.html</welcome-file>
<welcome-file>index.htm</welcome-file>
<welcome-file>index.jsp</welcome-file>
</welcome-file-list>
</web-app>
A portion of the web.xml file contained in the Tomcat conf directory.
Default set of welcome files to be used by Tomcat. We'll create one of these files later.
Done
Computer | Protected Mode: Off
100%

A Tour of Tomcat (cont.)

logs

- The logs directory contains a n umber of log files created by Tomcat. The file `catalina.out` contains anything written to `System.out` and `System.err`, as well as information relevant to the server as a whole.

lib

- In previous versions of Tomcat, this directory was named `common` and contained three subdirectories – classes, lib, and endorsed – which contain code used by Tomcat. The newer versions of Tomcat, including 6.0.29, have condensed this into a single directory named `lib`. Any custom `.jar` files that may be needed throughout Tomcat, such as a JDBC driver, are placed in these directories.

A Tour of Tomcat (cont.)

webapps

- This directory contains all the web applications Tomcat is configured to run, one web application per subdirectory. We will be placing the web applications that we develop into subdirectories in this directory. We'll look in more detail at the structure of these subdirectories a bit later.

work

- This directory is used by Tomcat to hold servlets that are built from JSP pages. Users will typically not need anything in this directory.

temp

- This directory is used internally by Tomcat and can be ignored.

Servlet Interface

- The servlet packages define two abstract classes that implement interface `Servlet` – class `GenericServlet` (from the package `javax.servlet`) and class `HttpServlet` (from the package `javax.servlet.http`).
- These classes provide default implementations of some `Servlet` methods.
- Most servlets extend either `GenericServlet` or `HttpServlet` and override some or all of their methods.
- The `GenericServlet` is a protocol-independent servlet, while the `HttpServlet` uses the HTTP protocol to exchange information between the client and server.
- We're going to focus exclusively on the `HttpServlet` used on the Web.

Servlet Interface (cont.)

- `HttpServlet` defines enhanced processing capabilities for services that extend a Web server's functionality.
- The key method in every servlet is `service`, which accepts both a `ServletRequest` object and a `ServletResponse` object. These object provide access to input and output streams that allow the servlet to read data from and send data to the client.
- If a problem occurs during the execution of a servlet, either `ServletExceptions` or `IOExceptions` are thrown to indicate the problem.

HTTPServlet Class

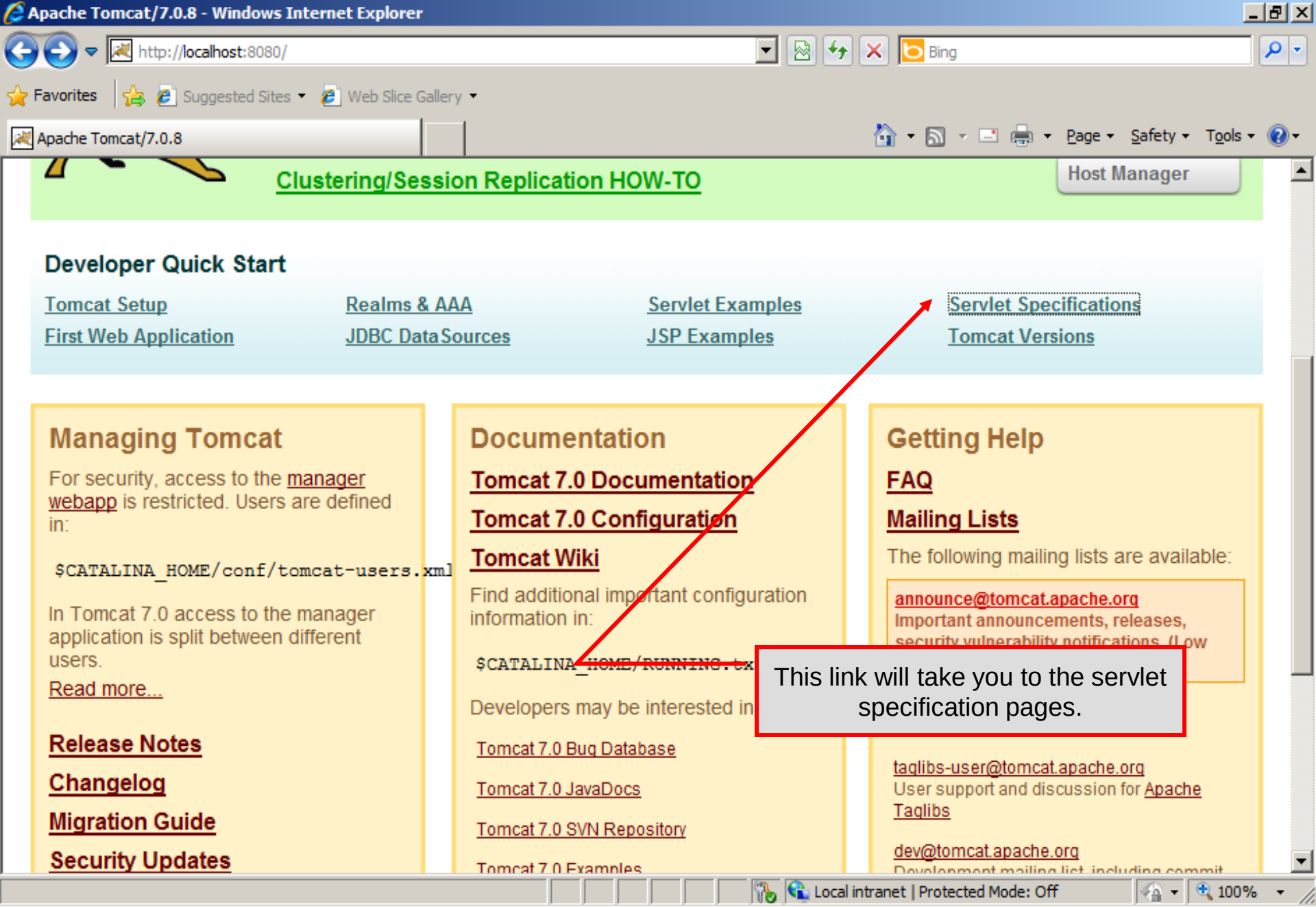
- Servlets typically extend class `HttpServlet`, which overrides method `service` to distinguish between the various requests received from a client web browser.
- The two most common HTTP request types (also known as request methods) are `get` and `post`. (See also Servlets – Part 1 notes.)
  - A get request retrieves information from a server. Typically, an HTML document or image.
  - A post request sends data to a server. Typically, post requests are used to pass user input to a data-handling process, store or update data on a server, or post a message to a news group or discussion forum.
- Class `HttpServlet` defines methods `doGet` and `doPost` to respond to get and post requests from a client.

HTTPServlet Class (cont.)

- Methods `doGet` and `doPost` are invoked by method `service`, which is invoked by the servlet container when a request arrives at the server.
- Method service first determines the request type, the invokes the appropriate method for handling such a request.
- In addition to methods `doGet` and `doPost`, the following methods are defined in class `HttpServlet`:
  - `doDelete` (typically deletes a file from the server)
  - `doHead` (client wants only response headers no entire body)
  - `doOptions` (returns HTTP options supported by server)
  - `doPut` (typically stores a file on the server)
  - `doTrace` (for debugging purposes)

HTTPServletRequest Interface

- Every invocation of `doGet` or `doPost` for an `HttpServlet` receives an object that implements interface `HttpServletRequest`.
- The servlet container creates an `HttpServletRequest` object and passes it to the servlet's `service` method, which in turn, passes it to `doGet` or `doPost`.
- This object contains the clients' request and provides methods that enable the servlet to process the request.
- The full list of HttpServletRequest methods is available at: www.java.sun.com/j2ee/1.4/docs/api/index.html, however, a few of the more common ones are shown on page 19. (Note: you can also get to them from Tomcat, see next page.)


Apache Tomcat/7.0.8 - Windows Internet Explorer
http://localhost:8080/
Bing
Favorites
Suggested Sites
Web Slice Gallery
Apache Tomcat/7.0.8
Page
Safety
Tools
Clustering/Session Replication HOW-TO
Host Manager
Developer Quick Start
Tomcat Setup
First Web Application
Realms & AAA
JDBC DataSources
Servlet Examples
JSP Examples
Servlet Specifications
Tomcat Versions
Managing Tomcat
For security, access to the manager webapp is restricted. Users are defined in:
$CATALINA_HOME/conf/tomcat-users.xml
In Tomcat 7.0 access to the manager application is split between different users.
Read more...
Release Notes
Changelog
Migration Guide
Security Updates
Documentation
Tomcat 7.0 Documentation
Tomcat 7.0 Configuration
Tomcat Wiki
Find additional important configuration information in:
$CATALINA_HOME/RUNNING.tx
Developers may be interested in
Tomcat 7.0 Bug Database
Tomcat 7.0 JavaDocs
Tomcat 7.0 SVN Repository
Tomcat 7.0 Examples
Getting Help
FAQ
Mailing Lists
The following mailing lists are available:
announce@tomcat.apache.org
Important announcements, releases,
taglibs-user@tomcat.apache.org
User support and discussion for Apache Taglibs
dev@tomcat.apache.org
This link will take you to the servlet specification pages.
Local intranet | Protected Mode: Off
100%

HTTPServletRequest Methods

- `Cookie[] getCookies()` – returns an array of Cookie objects stored on the client by the server. Cookies are used to uniquely identify clients to the server.
- `String getLocalName()` – gets the host name on which the request was received.
- `String getLocalAddr()` – gets the IP address on which the request was received.
- `int getLocalPort()` – gets the IP port number on which the request was received.
- `String getParameter( String name)` – gets the value of a parameter set to the servlet as part of a `get` or `post` request.

HTTPServletResponse Interface

- Every invocation of `doGet` or `doPost` for an `HttpServlet` receives an object that implements interface `HttpServletResponse`.
- The servlet container creates an `HttpServletResponse` object and passes it to the servlet's `service` method, which in turn, passes it to `doGet` or `doPost`.
- This object provides methods that enable the servlet to formulate the response to the client.
- The full list of HttpServletRequest methods is available at: www.java.sun.com/j2ee/1.4/docs/api/index.html, however, a few of the more common ones are shown on the next page. (Also accessible from Tomcat.)

HTTPServletResponse Methods

- `void addCookie (Cookie cookie)` – adds a Cookie to the header of the response to the client.
- `ServletOutputStream getOutputStream()` – gets a byte-based output stream for sending binary data to the client.
- `PrintWriter getWriter()` – gets a character-based output stream for sending text data (typically HTML formatted text) to the client.
- `void SetContentType (String type)` – specifies the content type of the response to the browser to assist in displaying the data.
- `void getContentType()` – gets the content type of the response.

Handling HTTP `get` Requests

- The primary purpose of an HTTP `get` request is to retrieve the contents of a specified URL, which is typically an HTML or XHTML document.
- Before we look at a complete implementation of a servlet execution, let's examine the Java code that is required for a basic servlet.
- Shown on the next page is a servlet that responds to an HTTP `get` request. This is a simple welcome servlet and is about as simple a servlet as is possible.
- Note: Tomcat will look for an `index.html`, or `welcome.html` files to run as a default "home page". At this point we haven't set one up so the initial screen for our web application will not be too pretty.

WelcomeServlet.java - Notepad

File Edit Format View Help

```
// A simple servlet to process get requests.

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class WelcomeServlet extends HttpServlet {
   // process "get" requests from clients
   protected void doGet( HttpServletRequest request,
      HttpServletResponse response )
         throws ServletException, IOException
   {
      response.setContentType( "text/html" );
      PrintWriter out = response.getWriter();
      // send XHTML page to client
      // start XHTML document
               out.println( "<?xml version = \"1.0\"?>" );
               out.println( "<!DOCTYPE html PUBLIC \"-//W3C//DTD " +
         "XHTML 1.0 Strict//EN\" \"http://www.w3.org" +
         "/TR/xhtml1/DTD/xhtml1-strict.dtd\">" );
               out.println(
         "<html xmlns = \"http://www.w3.org/1999/xhtml\">" );
               // head section of document
               out.println( "<head>" );
               out.println( "<title>Welcome to Servlets!</title>" );
               out.println( "<style type='text/css'> " );
               out.println( "<!-- " );
               out.println( " body {color:black; background-color:white;} " );
               out.println( " h1 {font-size:32pt;} " );
               out.println( " --> " );
               out.println( "</style>" );
            out.println( "</head>" );
      // body section of document
      //out.println( "<body background=purpleflowers.png lang=EN-US link=blue vlink=blue>");
               out.println( "<body lang=EN-US link=Blue vlink=Blue>" );
               out.println( "<h1> Hello!!</h1>");
               out.println( "<h1> Welcome To The Exciting World Of Servlet Technology!</h1>" );
               out.println( "</body>" );
      // end XHTML document
               out.println( "</html>" );
               out.close();  // close stream to complete the page
   }
}
```

Class name WelcomeServlet

doGet handles the HTTP get request – override method

Set MIME content type

XHTML document returned to the client

End the XHTML document generated by the servlet.

Handling HTTP `get` Requests (cont.)

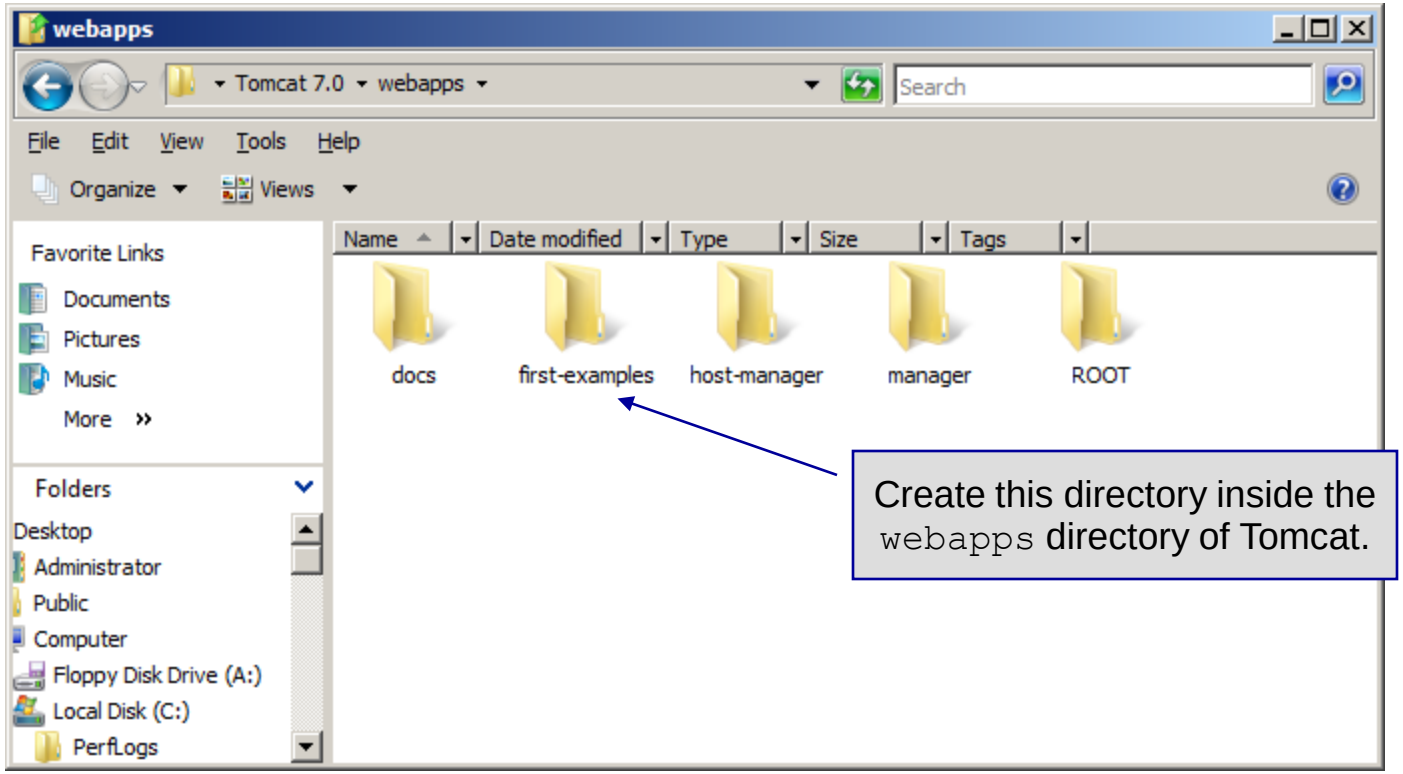
- The servlet creates an XHTML document containing the text "Hello! Welcome to the Exciting World of Servlet Technology!"
- This text is the response to the client and is sent through the `PrintWriter` object obtained from the `HttpServletRepsonse` object.
- The response object's `setContentType` method is used to specify the type of data to be sent as the response to the client. In this case it is defined as text/html, we'll look at other types later. In this case the browser knows that it must read the XHTML tags and format the document accordingly.
- The content type is also known as the MIME (Multipurpose Internet Mail Extension) type of the data.

Creating a Web Application

- One of the fundamental ideas behind Tomcat is that of a web application.
- A web application is a collection of pages, code, and configurations that is treated as a unit.
- Normally a web application will map to a particular URL, so URLs such as http://somesite.com/app1 and http://somesite.com/app2 will invoke different web applications called `app1` and `app2` respectively.
- Tomcat can contain an arbitrary number of web applications simultaneously.
- While web applications can be extremely complex, we'll start out with a minimal web application and build from there.

Creating a Web Application (cont.)

- The most basic web application in Tomcat will require the creation of a directory inside the `webapps` directory to hold the web application. For this first example, we'll create a subdirectory called `first-examples`.



Creating a Web Application (cont.)

- Within the `first-examples` directory we need to create a directory that will hold the configuration and all of the resources for the web application. This directory must be called `WEB-INF`.
- The most important and only required element in `WEB-INF` is the file `web.xml`. The `web.xml` file controls everything specific to the current web application. We'll look at this file in more detail later as we add to it, but for now we'll look only at the components of this file that are essential for a very simple web application.
- The next page illustrates our initial `web.xml` file.

C:\Program Files\Apache Software Foundation\Tomcat 7.0\webapps\first-examples\WEB-INF\web.xml - Notepad++

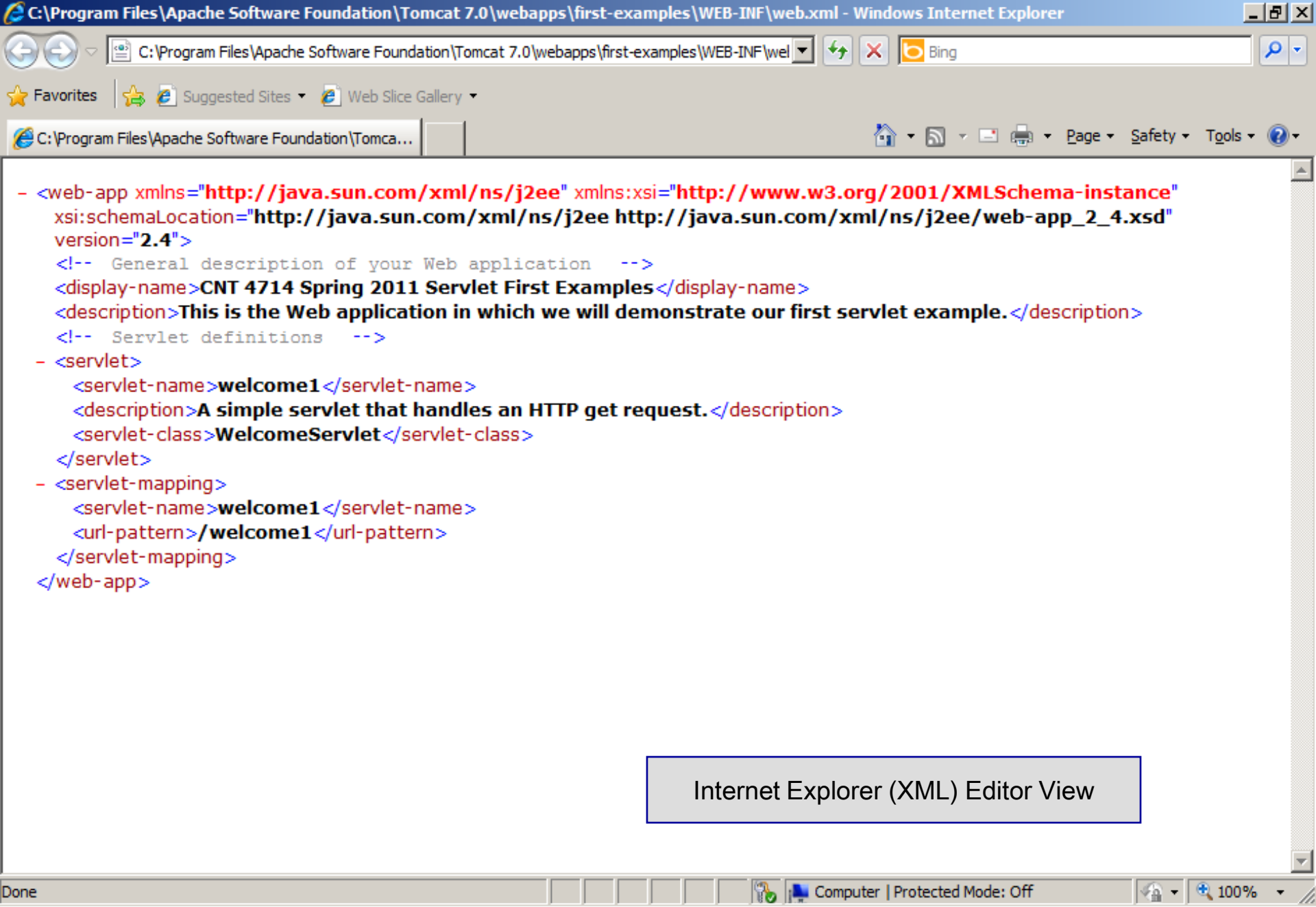
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?

web.xml

Notepad++ View

```
1  <web-app xmlns="http://java.sun.com/xml/ns/j2ee"
2     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
4        http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
5     version="2.4">
6
7     <!-- General description of your Web application -->
8     <display-name>
9        CNT 4714 Spring 2011 Servlet First Examples
10    </display-name>
11
12    <description>
13       This is the Web application in which we
14       will demonstrate our first servlet example.
15    </description>
16
17 <!-- Servlet definitions -->
18    <servlet>
19       <servlet-name>welcome1</servlet-name>
20       <description>
21          A simple servlet that handles an HTTP get request.
22       </description>
23       <servlet-class>
24          WelcomeServlet
25       </servlet-class>
26    </servlet>
27
28    <servlet-mapping>
```

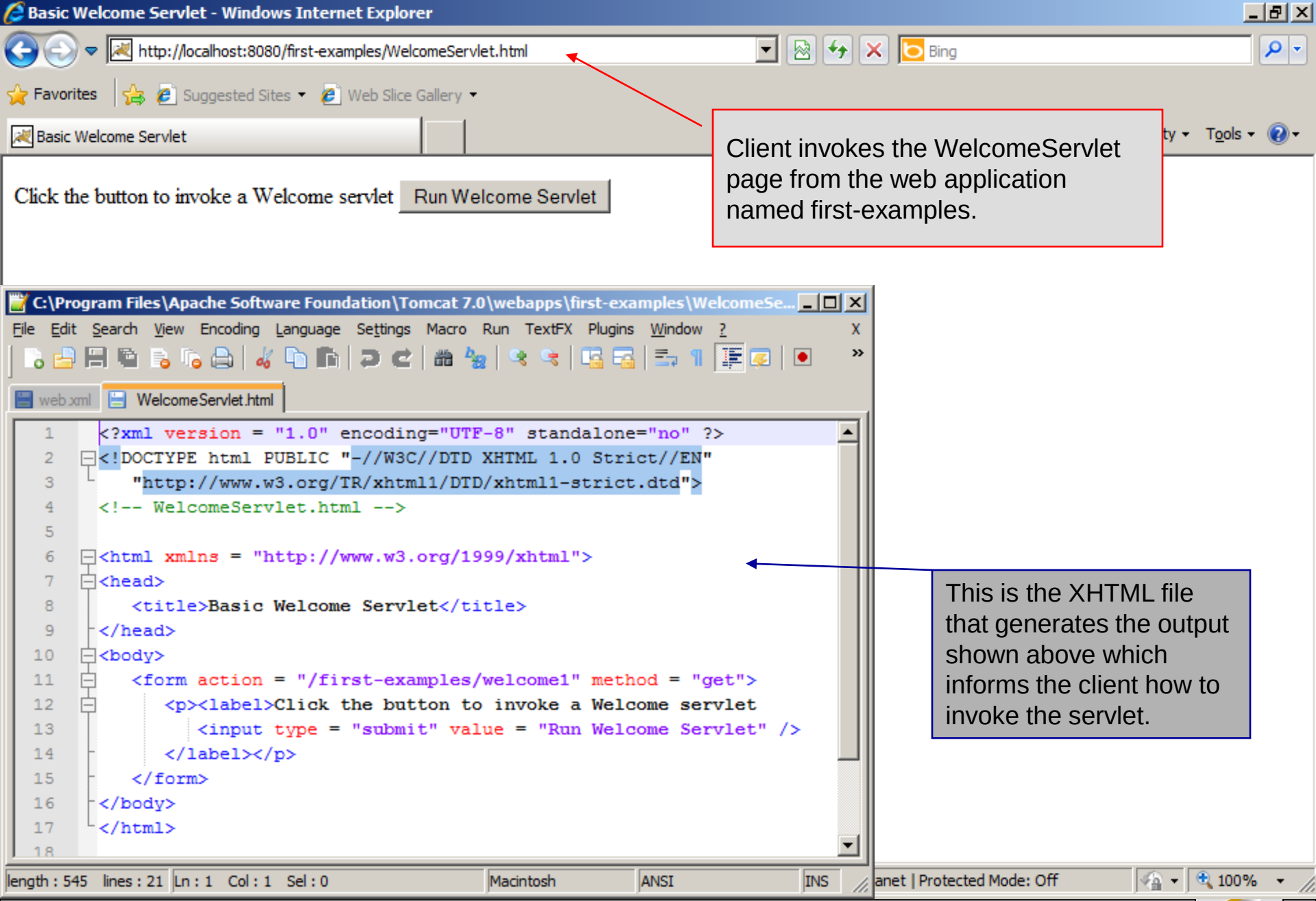
eXtensible Markup Language file | length : 966 lines : 35 | Ln : 1 Col : 1 Sel : 0 | Dos\Windows | ANSI | INS



Internet Explorer (XML) Editor View

Creating a Web Application (cont.)

- With these directories and files in place, Tomcat will be able to respond to a request for the page from a client at http://localhost:8080/first-examples/WelcomeServlet.html.
- Other HTML and JSP pages can be added at will, along with images, MP3 files, and just about anything else.
- Although what we have just seen is all that is required to create a minimal web application, much more is possible with a knowledge of how web applications are arranged and we will see this as we progress through this technology.
- The next few slides illustrate the execution of our simple web application (a welcome servlet).


Basic Welcome Servlet - Windows Internet Explorer
http://localhost:8080/first-examples/WelcomeServlet.html
Favorites
Suggested Sites
Web Slice Gallery
Basic Welcome Servlet
Tools
Click the button to invoke a Welcome servlet
Run Welcome Servlet
Client invokes the WelcomeServlet page from the web application named first-examples.
C:\Program Files\Apache Software Foundation\Tomcat 7.0\webapps\first-examples\WelcomeSe...
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?
web.xml
WelcomeServlet.html
1 <?xml version = "1.0" encoding="UTF-8" standalone="no" ?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4 <!-- WelcomeServlet.html -->
5
6 <html xmlns = "http://www.w3.org/1999/xhtml">
7 <head>
8 <title>Basic Welcome Servlet</title>
9 </head>
10 <body>
11 <form action = "/first-examples/welcome1" method = "get">
12 <label>Click the button to invoke a Welcome servlet
13 <input type = "submit" value = "Run Welcome Servlet" />
14 </label>
15 </form>
16 </body>
17 </html>
length : 545 lines : 21 Ln : 1 Col : 1 Sel : 0
Macintosh
ANSI
INS
anet | Protected Mode: Off
100%
This is the XHTML file that generates the output shown above which informs the client how to invoke the servlet.


Welcome to Servlets! - Windows Internet Explorer
http://localhost:8080/first-examples/welcome1
Bing
Favorites
Suggested Sites
Web Slice Gallery
Welcome to Servlets!
Page
Safety
Tools
Hello!!
Welcome To The Exciting World Of Servlet Technology!
Execution of the WelcomeServlet servlet
Done
Local intranet | Protected Mode: Off
100%

An XHTML Document

- The XHTML document shown on page 30 provides a `form` that invokes the servlet defined on page 27.
- The form's `action` attribute (/first-examples/welcome1) specifies the URL path that invokes the servlet.
- The form's method attribute indicates that the browser sends a get request to the server, which results in a call to the servlet's `doGet` method.
  - We'll look at how to set-up the URL's and deployment structure in the next set of notes.

Set-Up For First Web Application

- The exact set-up you need to use for setting up your web application in Tomcat is summarized on the next couple of pages.

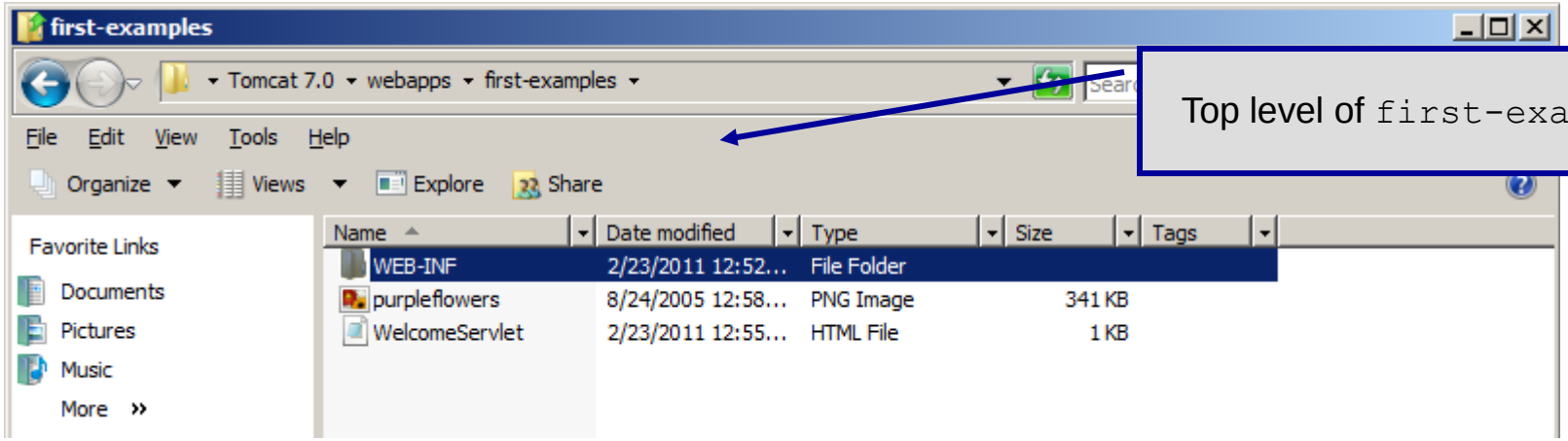
1. In the Tomcat `webapps` folder create a directory named `first-examples`.

2. In the top level of `first-examples` copy the `WelcomeServlet.html` file from the course code page.

3. In the top level of `first-examples` create a directory named `WEB-INF`.

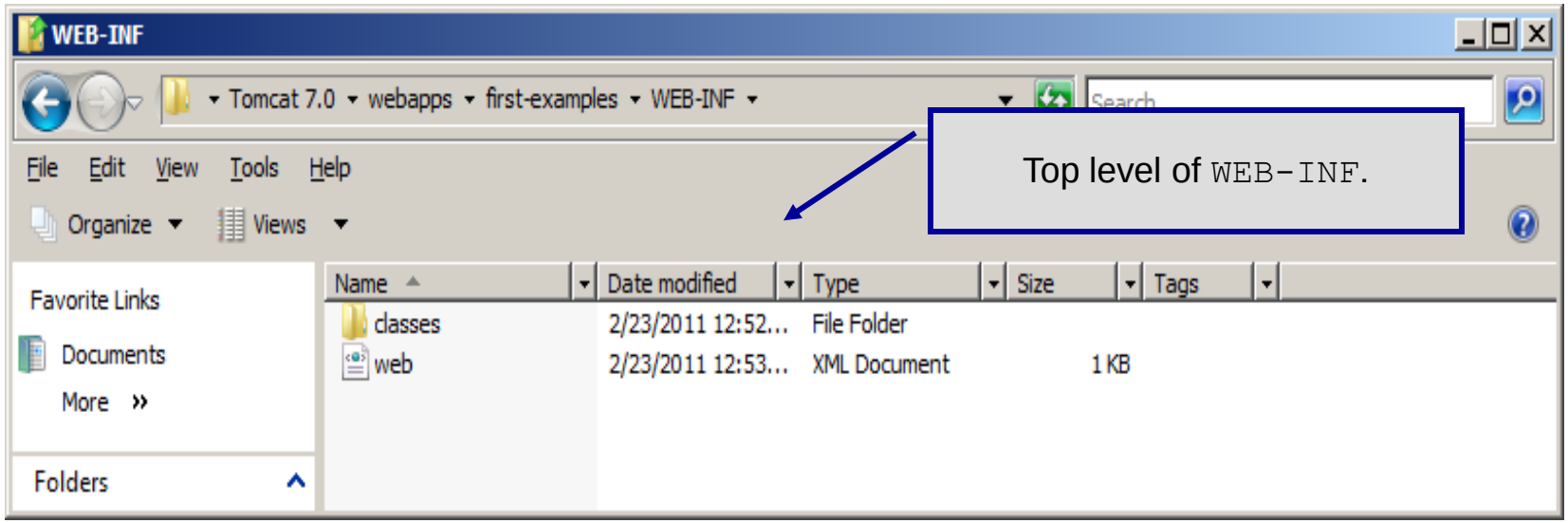
4. When steps 2 and 3 are complete the top level of first-examples should look like the picture at the top of the next page.

Set-Up For First Web Application (cont.)



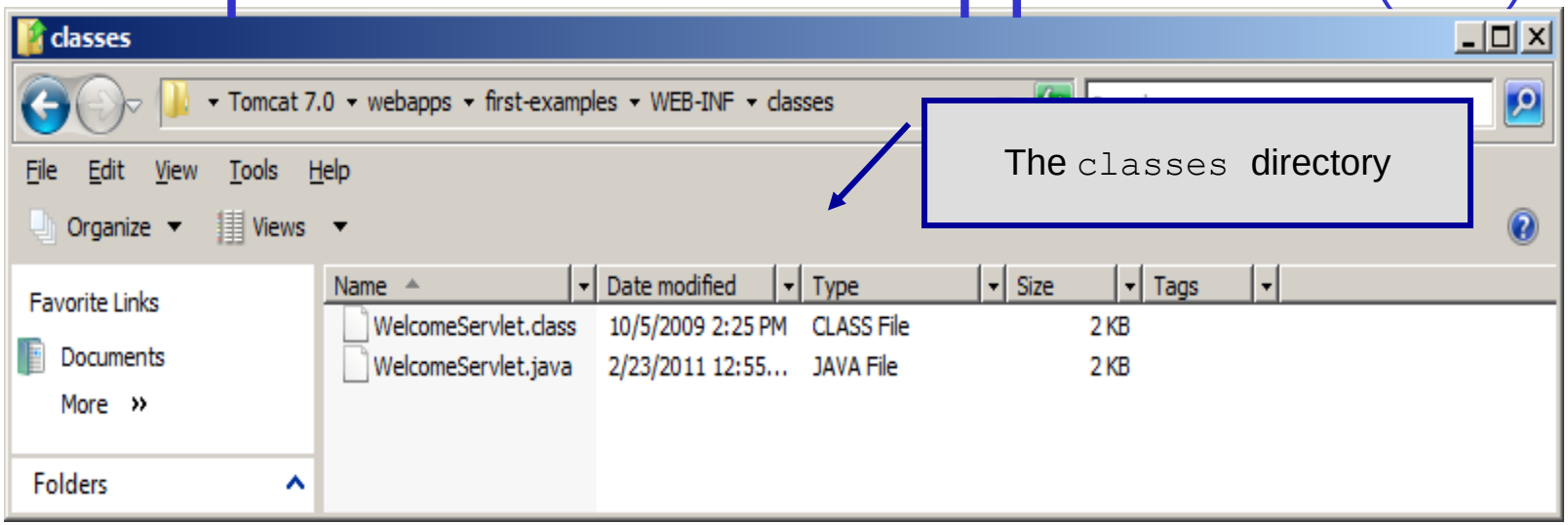
- Copy the `web.xml` configuration file from the course code page into the `WEB-INF` directory.
- At the top level of the `WEB-INF` directory create a directory named `classes`.
- When steps 5 and 6 are complete, the WEB-INF directory should look like the picture on the top of the next page.

Set-Up For First Web Application (cont.)



8. Copy the `WelcomeServlet.java` file from the course code page into the `classes` directory and compile it to produce the `WelcomeServlet.class` file which should also reside in the classes directory. (The `.java` file does not need to reside in this directory for a servlet, but it is handy to keep the source in the same place.)

Set-Up For First Web Application (cont.)



9. Once the `classes` directory looks like the one shown above. You are ready to invoke the servlet from a web browser. Start Tomcat and enter the URL http://localhost:8080/WelcomeServlet.html. Tomcat and the servlet will do the rest. If all goes well you should see the output that was shown on pages 30-31.