

# CNT 4714: Enterprise Computing Summer 2014

## Introduction To Servlet Technology – Part 1

Instructor :      Dr. Mark Llewellyn  
                         markl@cs.ucf.edu  
                         HEC 236, 4078-823-2790  
                         <http://www.cs.ucf.edu/courses/cnt4714/sum2014>

Department of Electrical Engineering and Computer Science  
Computer Science Division  
University of Central Florida



# Client-Server Relationship Revisited

- In a client-server relationship, the client requests that some action be performed and the server performs the action and responds to the client.
- This **request-response model** of communication is the foundation for the highest-level view of networking in Java – **servlets** and **JavaServer Pages (JSP)**.
- A **servlet** extends the functionality of a server, such as a Web server that serves Web pages to a user's browser using the HTTP protocol. A servlet can almost be thought of as an applet that runs on the server side--without a face. Java servlets make many Web applications possible.
- Packages `javax.servlet` and `javax.servlet.http` provide the classes and interfaces to define servlets. Packages `javax.servlet.jsp` and `javax.servlet.jsp.tagext` provide the classes and interfaces that extend the servlet capabilities for JSP.



# Client-Server Relationship Revisited (cont.)

- Using special syntax, JSP allows Web-page implementers to create pages that encapsulate Java functionality and even to write scriptlets of actual Java code directly into the page.
- A common implementation of the request-response model is between Web browsers and Web servers. When a user selects a Web site to browse through the browser (the client application), a request is sent to the appropriate Web server (the server application). The server normally responds to the client by sending the appropriate HTML Web page.
- Servlets are effective for developing Web-based solutions that help provide secure access to a Web site, interact with databases on behalf of a client, dynamically generate custom HTML documents to be displayed by browsers and maintain unique session information for each client.



# Static and Dynamic Web Content

- Consider how a web page is displayed by a browser.
  - Typically, the web page is created using HTML and stored as a file on the web server. A user enters a URL for the file from a web browser. The browser contacts the web server and requests the file. The server finds the file and returns it to the browser. The browser then displays the file for the user.
- Static information is stored in HTML files. The HTML files can be updated, but at any given time, every request for the same file returns exactly the same content. The contents do not change regardless of who requested the file.



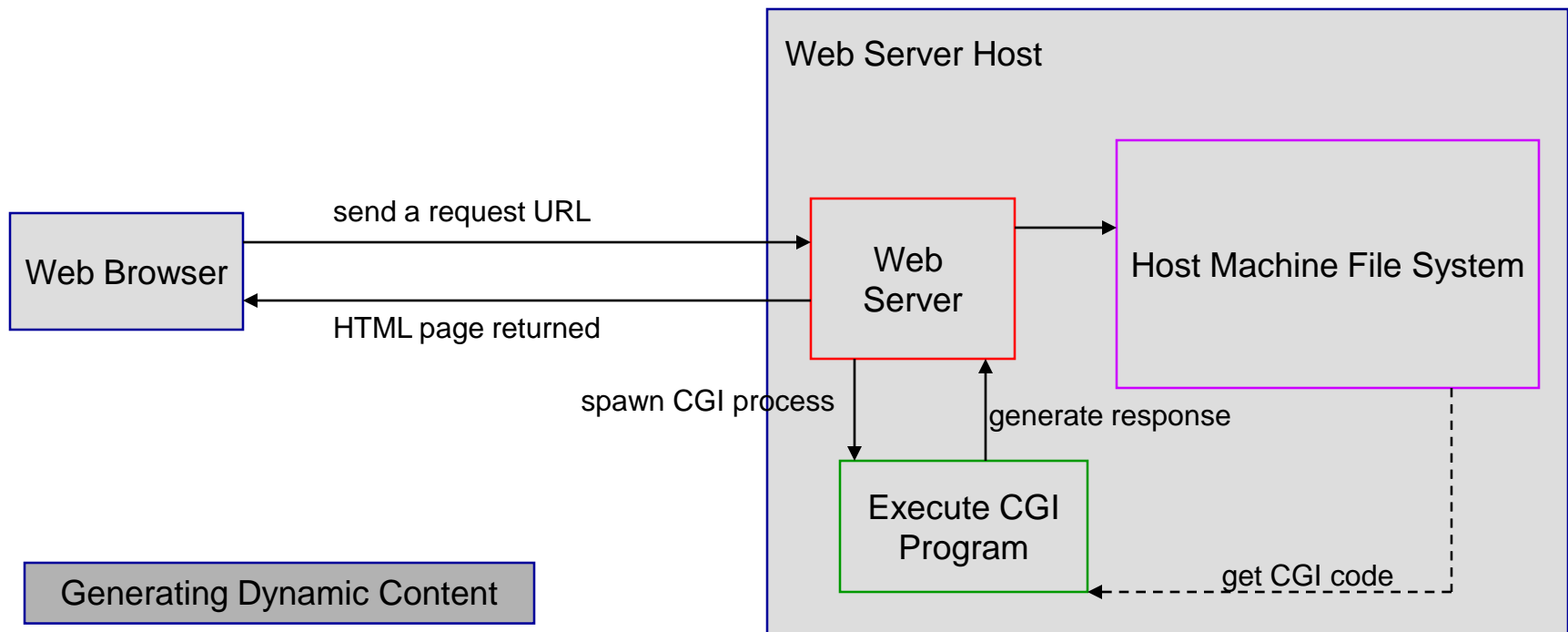
# Static and Dynamic Web Content (cont.)

- Not all information, however, is static in nature. Often HTML pages need to generate information dynamically.
- Dynamic web pages are generated by web server. The web server will execute certain programs to process user requests from browsers in order to produce a customized response.
- The **Common Gateway Interface (CGI)** was proposed to generate dynamic web content. The interface provides a standard framework for web servers to interact with external program known as **CGI programs**.



# CGI Programming

- When a web server receives a request from a browser it passes it to the CGI program. The CGI program processes the request and generates a response at runtime. CGI programs can be written in any language, but Perl is the most popular choice.



# The GET and POST Methods

- The two most common HTTP requests, also known as methods, are GET and POST.
- The web browser issues a request using a URL or an HTML form to trigger the web server to execute a CGI program. (We'll deal with forms later.) When issuing a CGI request directly from a URL, the GET method is used.
- This form of a URL is known as a **query string**. The URL query string consists of the location of the CGI program, parameters, and their values.
- When issuing a request from an HTML form, either a GET or POST method can be used.



# The GET and POST Methods (cont.)

- The form explicitly specifies which of the two is used.
- If the GET method is used, the data in the form are appended to the request string as if they were submitted using a URL.
- If the POST method is used, the data in the form are packaged as part of the request file. The server program obtains the data by reading the file.

The GET and POST methods both send requests to the web server. The POST method always triggers the execution of the corresponding CGI program. The GET method may not cause the CGI program to be executed if the previous same request is cached in the web browser. Browsers often cache web pages so that the same request can be quickly responded to without contacting the web server. The browser checks the request sent through the GET method as a URL query string. If the results for the exact same URL are cached on a disk, then the previous web page for the URL may be displayed. To ensure that a new web page is always displayed, use the POST method.





# From CGI To Java Servlets

- CGI provides a relatively simple approach for creating dynamic web applications that accept a user request, process it on the server side, and return responses to the user's browser.
- However, CGI is extremely slow when handling a large number of requests simultaneously, because the web server must spawn a process for executing each CGI program.
- Java servlets were developed to remedy the performance problem of CGI programs. Java servlets are basically Java programs that behave like CGI programs.



# Java Servlets

- Java servlets are executed upon request from a web browser.
- All servlets execute inside a **servlet container**, also referred to as a **servlet server** or a **servlet engine**.
- A servlet container is a single process that runs a JVM (Java Virtual Machine). The JVM creates a thread to handle each servlet (recall that threads have considerably less overhead than full-blown processes). All the threads share the same memory allocated to the JVM. Since the JVM persists beyond the lifecycle of a single servlet execution, servlets can share objects already created in the JVM.
  - For example, if multiple servlets access the same database, they can share the connection object.



# Thin Clients

- Servlets are the ideal solution for database-intensive applications that communicate with **thin clients**.
  - **Thin clients** are applications that provide presentation but do not process data, thus requiring few computing resources.
- The server is responsible for database access. Clients connect to the server using standard protocols available on most client platforms. The presentation-logic code for generating dynamic content can be written once and reside on the server for access by clients, to allow programmers to create efficient thin clients.



# Apache Tomcat Server

- Sun Microsystems, through the Java Community Process is responsible for the development of the servlet and JSP specifications.
- To run Java servlets, you need a [servlet container](#). While many servlet containers are available, the reference implementation of both these standards developed by the Apache Software Foundation ([www.apache.org](http://www.apache.org)) is known as Tomcat.
- Tomcat was developed as part of the Jakarta Project. The Jakarta Project contains many subprojects designed to help commercial server-side developers.
- Tomcat became a top-level project at Apache in early October 2005.
- Tomcat is the official reference implementation of the JSP and servlet standards. Tomcat can be used standalone as a web server or plugged into a web server like Apache, IIS (Internet Information Services), etc.. The current stable implementation is Tomcat 7.0.54 (as of May 22, 2014).



# Servlet Overview and Architecture

- The Internet offers many protocols. The HTTP (Hypertext Transfer Protocol) that forms the basis of the WWW uses URLs (Uniform Resource Locators) to locate resources on the Internet.
- URLs can represent files or directories and can represent complex tasks such as database lookups and Internet searches.
- JSP technology, basically an extension of servlet technology, simplifies the process of creating pages by separating presentation from content.
- Typically, JSPs are used when most of the content sent to the client is static text and markup, and only a small portion of the content is generated dynamically with Java code.



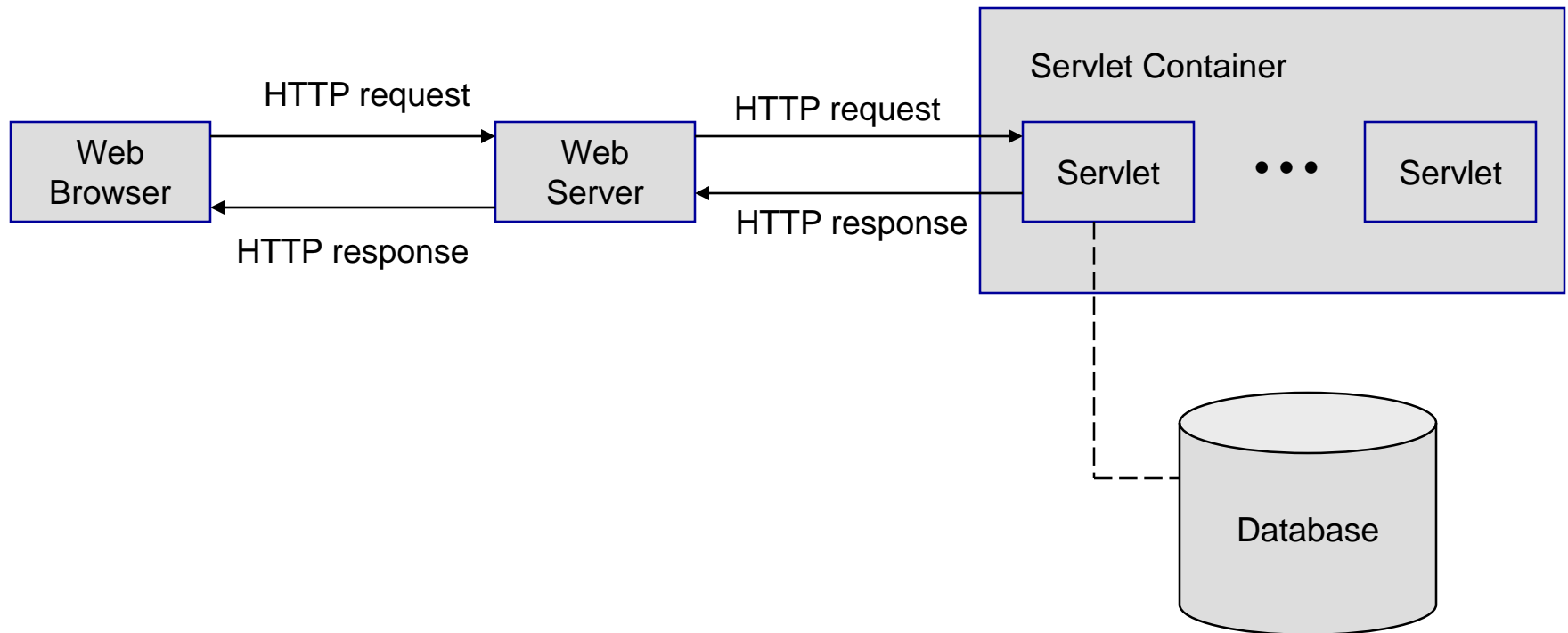
# Servlet Overview and Architecture (cont.)

- Servlets are more commonly used when a small portion of the content sent to the client is static text or markup. In fact, some servlets do not produce content. Rather, they perform a task on behalf of the client, then invoke other servlets or JSPs to provide a response.
- Note that in most cases servlet and JSP technologies are interchangeable.
- The server that executes a servlet is referred to as the  **servlet container**  or  **servlet engine** .
- Servlets and JSP have become so popular that they are now supported directly or with third-party plug-ins by most major Web servers and application servers (servers that execute applications to generate dynamic Web pages in response to requests).



# Servlet Overview and Architecture (cont.)

- We'll look at servlets that implement the request-response model between clients and servers using the HTTP protocol. This architecture is shown in the diagram below.



# Servlet Overview and Architecture (cont.)

## Explanation of the architecture diagram on previous page

- A client application sends an HTTP request to the server.
- The servlet container receives the request and directs it to be processed by the appropriate servlet.
- The servlet does its processing, which may include interacting with a database or other server-side components, such as other servlets or JSPs.
- The servlet returns its results to the client – normally in the form of an HTML, XHTML, or XML document to display in a browser.





# Interface Servlet and the Servlet Lifecycle

- Architecturally speaking, all servlets must implement the `Servlet` interface of package `javax.servlet`.
- The methods of interface `Servlet` are invoked by the servlet container. This interface declares five methods which deal with the execution of a servlet. These methods are shown on the next page. For the details see: [www.java.sun.com/j2ee/1.4/docs/api/javax/servlet/Servlet.html](http://www.java.sun.com/j2ee/1.4/docs/api/javax/servlet/Servlet.html)
- A servlet's life cycle begins when the servlet container loads it into memory – normally, in response to the first request for the servlet.
- Before the servlet can handle that request, the container invokes the servlet's `init` method.



# Methods of the Servlet Interface

Method	Description
<code>destroy( )</code>	Called by the servlet container to indicate to a servlet that the servlet is being taken out of service.
<code>getServletConfig( )</code>	Returns a <code>ServletConfig</code> object, which contains initialization and startup parameters for this servlet.
<code>getServletInfo( )</code>	Returns information about the servlet, such as author, version, and copyright.
<code>init( )</code>	Called by the servlet container to indicate to a servlet that the servlet is being placed into service.
<code>service( )</code>	Called by the servlet container to allow the servlet to respond to a request.



# The Servlet Lifecycle

- After `init` completes execution, the servlet can respond to its first request.
- All requests are handled by the a servlet's `service` method, which receives the request, processes it and sends a response to the client.
- During the servlet's lifecycle, the method `service` is invoked once per request. Each new request is typically handled in a separate thread of execution (managed by the servlet container) in which method `service` executes.
- When the servlet container terminates the servlet (whenever the servlet needs more memory or when it is shutdown), the servlet's `destroy` method is invoked to release servlet resources.



# Setting Up Tomcat

- Tomcat is a fully functional implementation of servlets and JSPs. It includes a Web server, so it can be used as a standalone test container for servlets and JSPs.
- The current stable version is 7.0.54 available from [www.apache.org](http://www.apache.org). This version was declared stable on May 22, 2014.
  1. Select the Tomcat page from the menu on the left-hand side of the screen (its way down the page). As shown on page 21.
  2. Once in the Tomcat project, select Download Tomcat 7.0.54 from the left-hand side of the screen as shown on page 22.
  3. Once in the download binaries screen, select the option of your choice. This is shown on page 24.



Welcome to The Apache Software Foundation! - Windows Internet Explorer

http://www.apache.org/ tomcat

File Edit View Favorites Tools Help

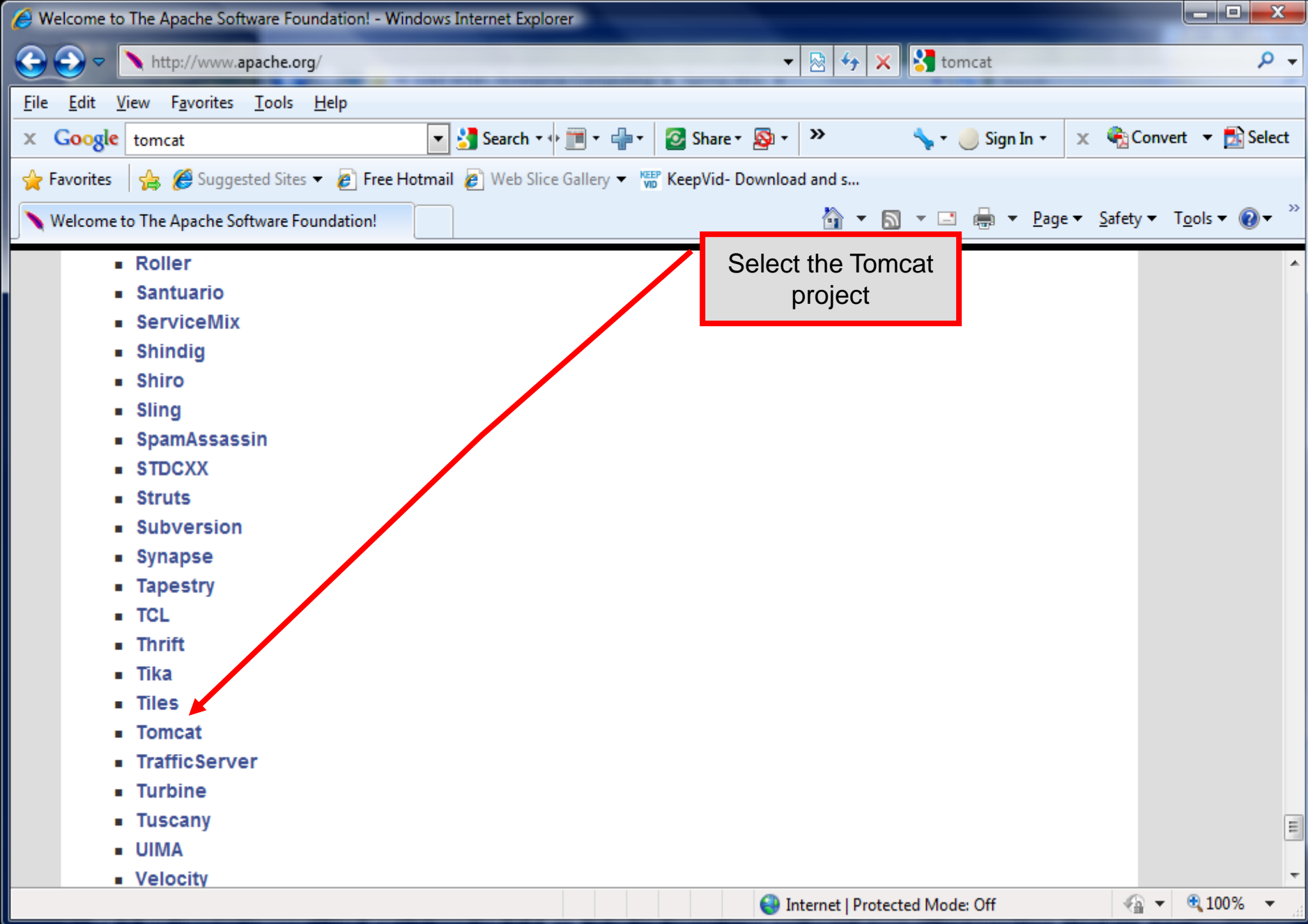
Google tomcat Search Share Sign In Convert Select

Welcome to The Apache Software Foundation!

- Roller
- Santuario
- ServiceMix
- Shindig
- Shiro
- Sling
- SpamAssassin
- STDCXX
- Struts
- Subversion
- Synapse
- Tapestry
- TCL
- Thrift
- Tika
- Tiles
- Tomcat
- TrafficServer
- Turbine
- Tuscany
- UIMA
- Velocity

Select the Tomcat project

Internet | Protected Mode: Off 100%





# Apache Tomcat

## Apache Tomcat

- Home
- Taglibs
- Maven Plugin

## Download

- Which version?
- Tomcat 8.0
- Tomcat 7.0
- Tomcat 6.0
- Tomcat Connectors
- Tomcat Native
- Archives

## Documentation

- Tomcat 8.0
- Tomcat 7.0
- Tomcat 6.0
- Tomcat Connectors
- Tomcat Native
- Wiki
- Migration Guide

## Problems?

- Security Reports
- Find help
- FAQ
- Mailing Lists
- Bug Database
- IRC

### Apache Tomcat

Apache Tomcat is an open source software implementation of the Java Servlet and JavaServer Pages technologies. The Java Servlet and JavaServer Pages specifications are developed under the [Java Community Process](#).

Apache Tomcat is developed in an open and participatory environment and released under the [Apache License version 2](#). Apache Tomcat is intended to be a collaboration of the best-of-breed developers from around the world. We invite you to participate in this open development project. To learn more about getting involved, [click here](#).

Apache Tomcat powers numerous large-scale, mission-critical web applications across a diverse range of industries and organizations. Some of these users and their stories are listed on the [PoweredBy](#) wiki page.

Apache Tomcat, Tomcat, Apache, the Apache feather, and the Apache Tomcat project logo are trademarks of the Apache Software Foundation.

### Tomcat 6.0.41 Released

2014-05-23

The Apache Tomcat Project is proud to announce the release of version 6.0.41 of Apache Tomcat. This release includes a number of bug fixes over Apache Tomcat 6.0.39. The notable changes include:

- Add support for using ecj-P20140317-1600.jar to use Java 8 syntax in JSPs
- Update native library to 1.1.30
- Various improvements to XML processing

[Download](#) | [ChangeLog for 6.0.41](#)

### Tomcat 7.0.54 Released

2014-05-22

The Apache Tomcat Project is proud to announce the release of version 7.0.54 of Apache Tomcat. This release contains a number of bug fixes and improvements compared to version 7.0.53. The notable changes since 7.0.53 include:

- Extend and improve memory leak protection and fix a few leaks that crept in during the various refactorings





# Apache Tomcat

## Apache Tomcat

- [Home](#)
- [Taglibs](#)
- [Maven Plugin](#)

## Download

- [Which version?](#)
- [Tomcat 8.0](#)
- [Tomcat 7.0](#)
- [Tomcat 6.0](#)
- [Tomcat Connectors](#)
- [Tomcat Native](#)
- [Archives](#)

## Documentation

- [Tomcat 8.0](#)
- [Tomcat 7.0](#)
- [Tomcat 6.0](#)
- [Tomcat Connectors](#)
- [Tomcat Native](#)
- [Wiki](#)
- [Migration Guide](#)

## Problems?

- [Security Reports](#)
- [Find help](#)

## Apache Tomcat Versions

Apache Tomcat is an open source software implementation of the Java Servlet and JavaServer Pages technologies. Different versions of Apache Tomcat are available for different versions of the Servlet and JSP specifications. The mapping between the specifications and the respective Apache Tomcat versions is:

Servlet Spec	JSP Spec	EL Spec	WebSocket Spec	Apache Tomcat version	Actual release revision	Support Java Versions
3.1	2.3	3.0	1.0	8.0.x	8.0.8 (beta)	7 and later
3.0	2.2	2.2	1.0	7.0.x	7.0.54	6 and later (WebSocket 1.0 requires 7 or later)
2.5	2.1	2.1	N/A	6.0.x	6.0.41	5 and later
2.4	2.0	N/A	N/A	5.5.x (archived)	5.5.36 (archived)	1.4 and later
2.3	1.2	N/A	N/A	4.1.x (archived)	4.1.40 (archived)	1.3 and later
2.2	1.1	N/A	N/A	3.3.x (archived)	3.3.2 (archived)	1.1 and later

Each version of Tomcat is supported for any stable Java release that meets the requirements of the final column in the table above.

Tomcat should also work on any Java early access build that meets the requirements of the final column in the table above. For example, users were successfully running Tomcat 8 on Java 8 many months before the first stable



**Problems?**

- Security Reports
- Find help
- FAQ
- Mailing Lists
- Bug Database
- IRC

**Get Involved**

- Overview
- SVN Repositories
- Buildbot
- Reviewboard
- Tools

**Media**

- Blog
- Twitter

**Misc**

- Who We Are
- Heritage
- Apache Home
- Resources
- Contact
- Legal
- Sponsorship
- Thanks

Other mirrors: <http://www.dsgnwrl.com/am/>

### 7.0.52

Please see the [README](#) file for packaging information. It explains what every distribution contains.

### Binary Distributions

- Core:
  - [zip \(pgp, md5\)](#)
  - [tar.gz \(pgp, md5\)](#)
  - [32-bit Windows zip \(pgp, md5\)](#)
  - [64-bit Windows zip \(pgp, md5\)](#)
  - [64-bit Itanium Windows zip \(pgp, md5\)](#)
  - [32-bit/64-bit Windows Service Installer \(pgp, md5\)](#)
- Full documentation:
  - [tar.gz \(pgp, md5\)](#)
- Deployer:
  - [zip \(pgp, md5\)](#)
  - [tar.gz \(pgp, md5\)](#)
- Extras:
  - [JMX Remote jar \(pgp, md5\)](#)
  - [Web services jar \(pgp, md5\)](#)
  - [JULI adapters jar \(pgp, md5\)](#)
  - [JULI log4j jar \(pgp, md5\)](#)
- Embedded:
  - [tar.gz \(pgp, md5\)](#)

Select the download version you need. Select Windows Service Installer for a Windows installer version.

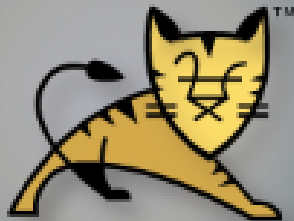
**NOTE:** There seems to be some issues with Tomcat 7.0.52 (and possibly 7.0.50) running under Windows 8. I am showing an install here of 7.0.52, but notice later on that I have reverted to 7.0.42 for the examples.







<http://tomcat.apache.org>



Apache Tomcat 7

## Welcome to the Apache Tomcat Setup Wizard

This wizard will guide you through the installation of Apache Tomcat.

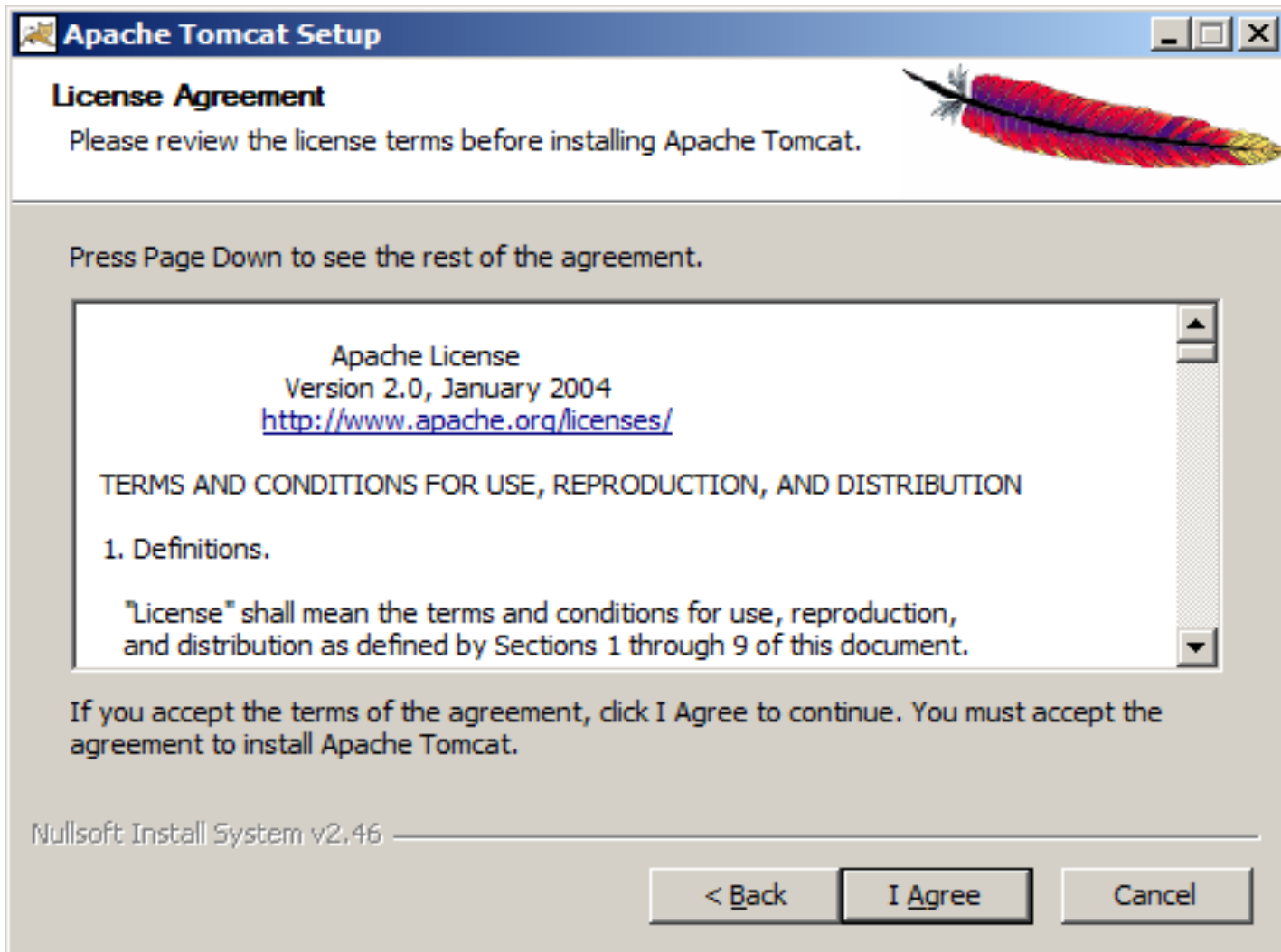
It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

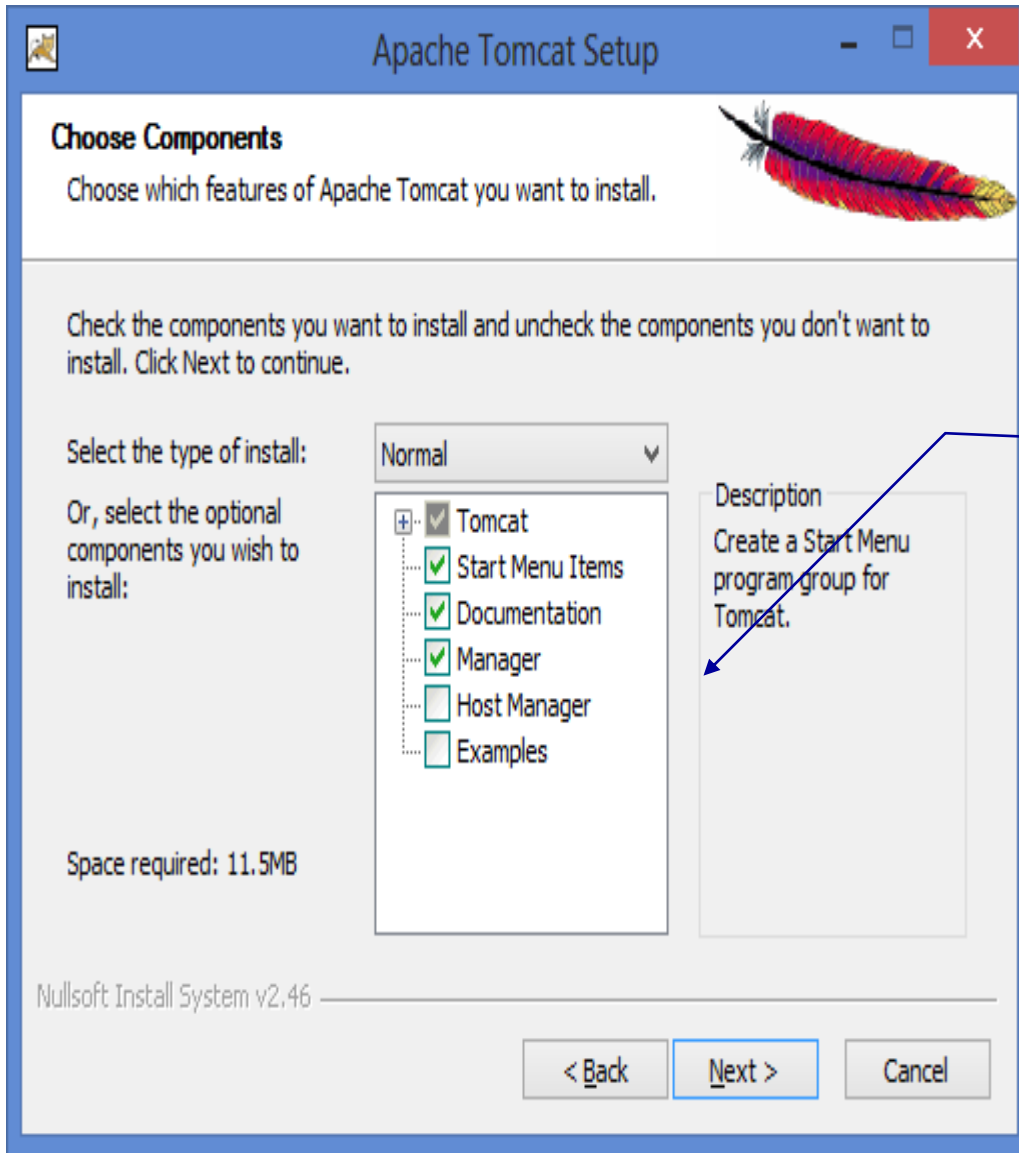
Click Next to continue.

Next >

Cancel







The host manager will not be selected in the default setting. Be sure to check its box if you will run multiple Tomcats. You can decide whether or not to include the examples.



**Configuration**  
Tomcat basic configuration.

Server Shutdown Port: 8005

HTTP/1.1 Connector Port: 8080

AJP/1.3 Connector Port: 8009

Windows Service Name: Tomcat7

Create shortcuts for all users:

Tomcat Administrator Login (optional)

User Name:

Password:

Roles: admin-gui,manager-gui

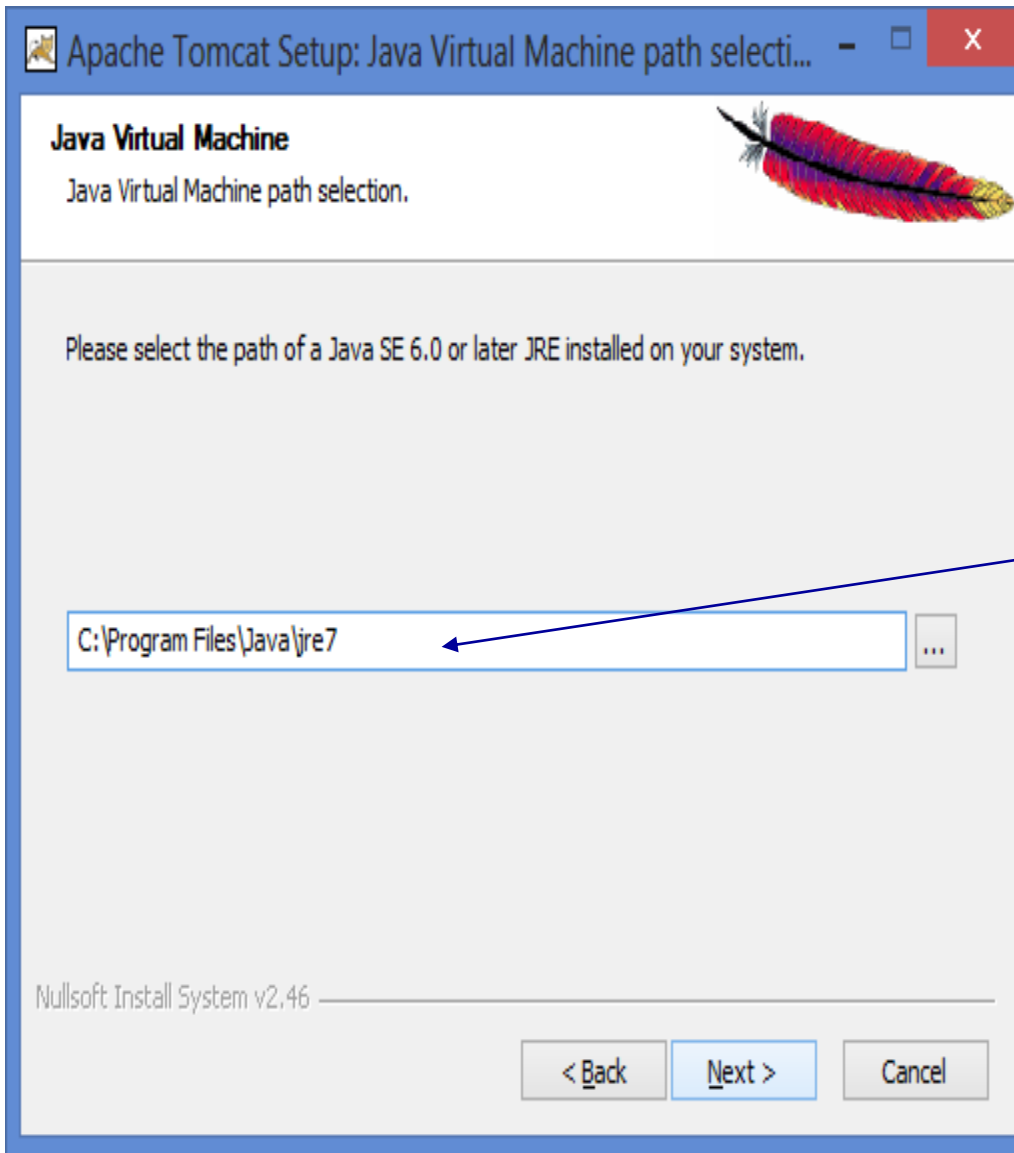
Nullsoft Install System v2.46

< Back   Next >   Cancel

Port 8080 is the default Tomcat connector port. Unless you have a conflict with this port, use this for Tomcat. When we setup Apache later, we'll put in on a different connector port so that both servers can be running simultaneously.

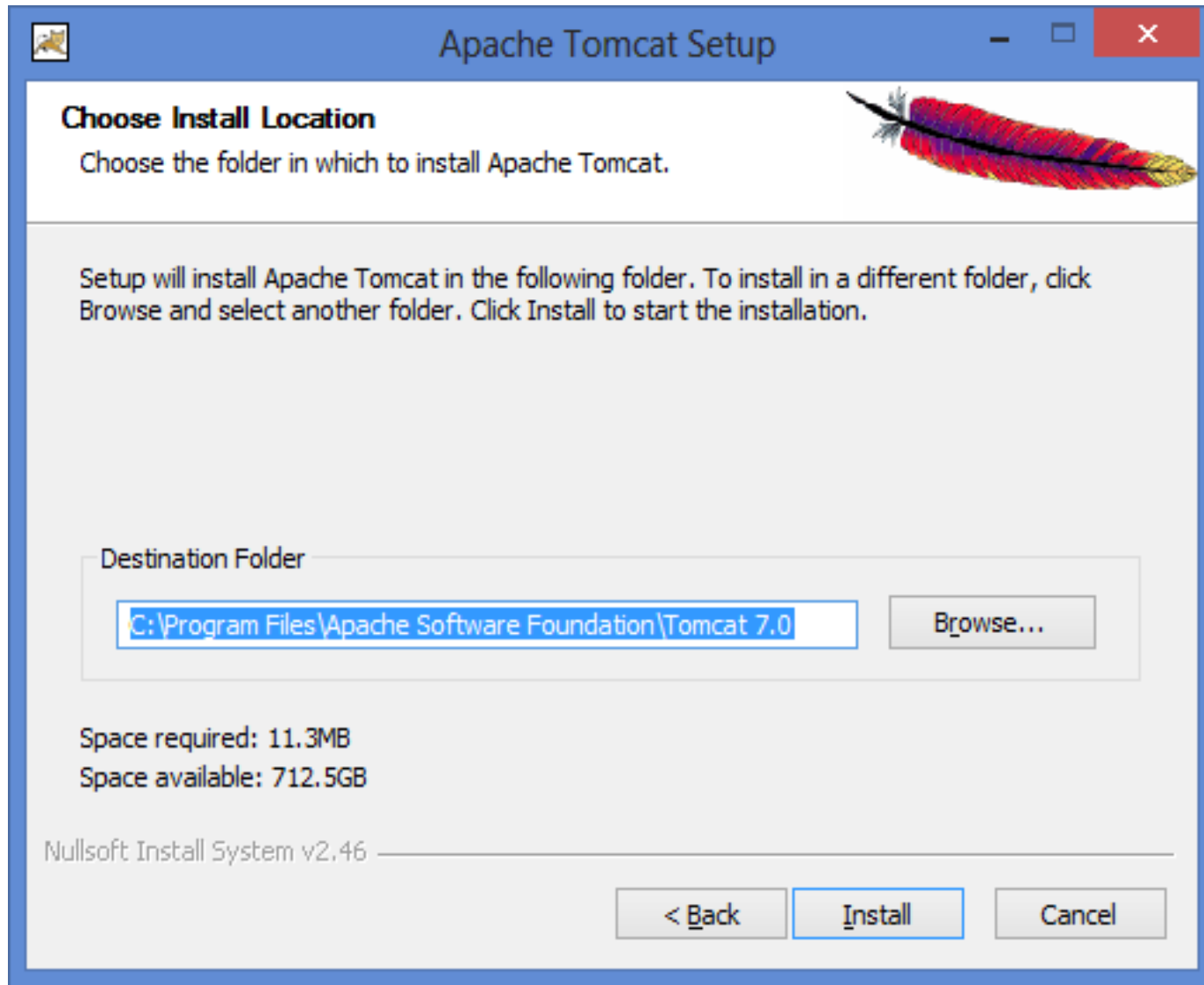
Set up a Tomcat Administrator login so that you can manage your Tomcat server more easily. This will be very important when you are deploying your servlets.

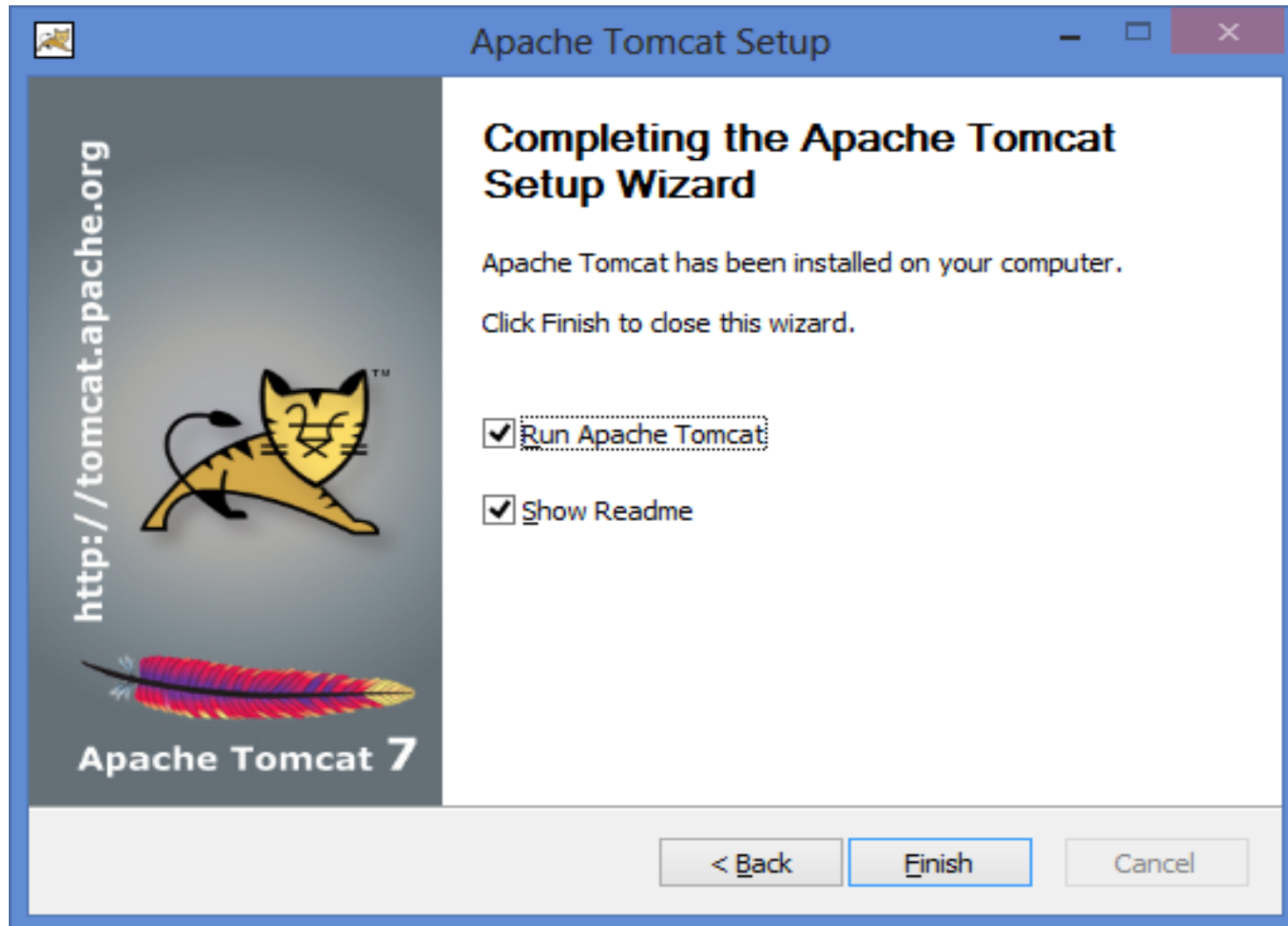




The installer should find the path to your Java `jre`. If you have more than one, be sure it is set to the one you want to use.







File Edit Format View Help

-----  
Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to You under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

-----

Apache Tomcat Version 7.0.54  
Release Notes

=====  
CONTENTS:  
=====

- \* Dependency Changes
- \* API Stability
- \* JNI Based Applications
- \* Bundled APIs
- \* Web application reloading and static fields in shared libraries
- \* Tomcat on Linux





# Setting Up Tomcat

- Once you've downloaded and installed Tomcat you're ready to run a demonstration test that will tell you if you've got everything set-up properly.

**NOTE:** During the install, Tomcat will ask you which TCP port Tomcat should run on (See page 28). To avoid any conflict with standard Web servers which default to TCP port 80, Tomcat is set to default to TCP port 8080. If you have any other service running on this port change the port number at this time to one on which no conflict will occur.

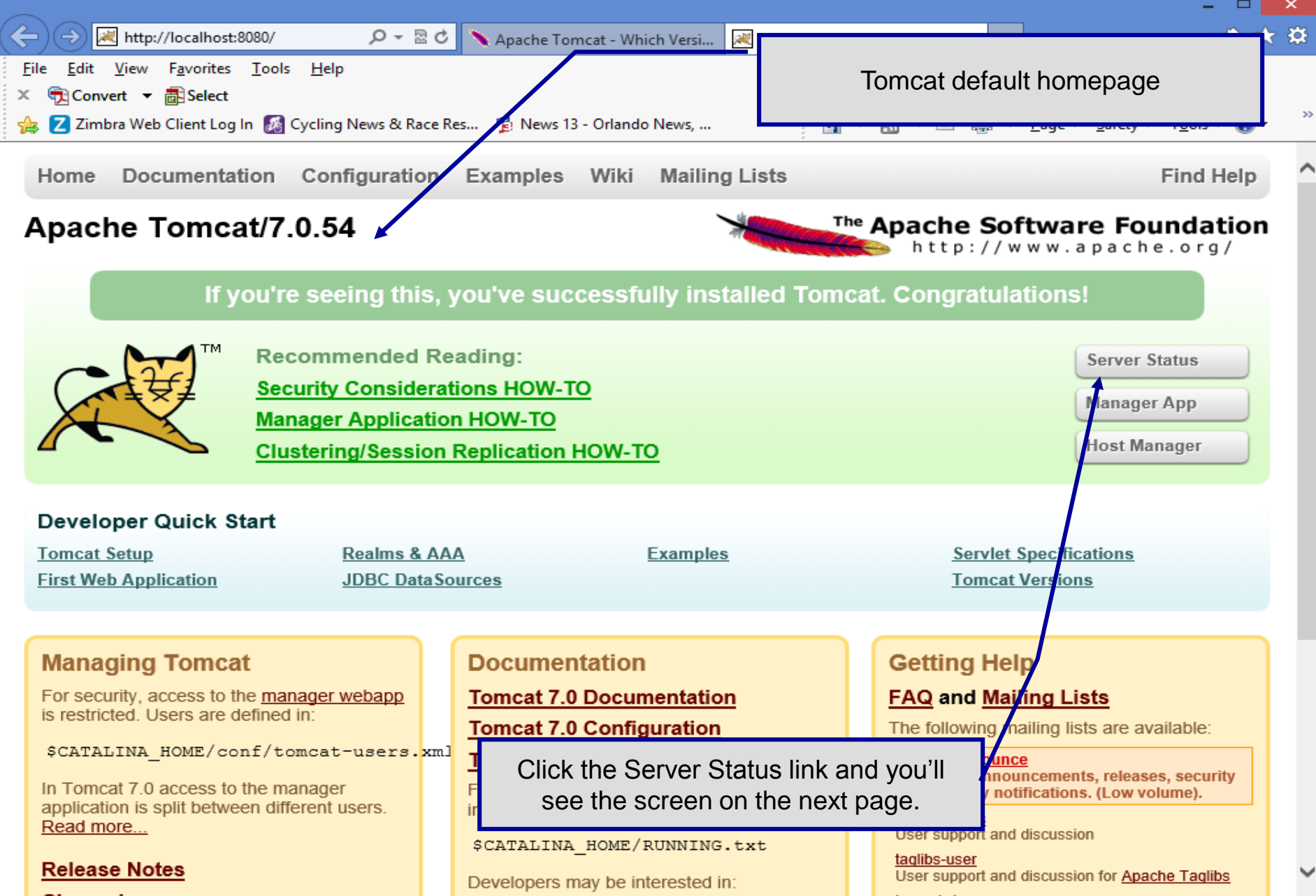
In all subsequent examples, I'm running Tomcat on TCP port 8080.



# Starting Up Tomcat

- Once Tomcat is installed, you need to start it as a service. On Windows machines, the current versions of Tomcat are installed as a service that will start when Windows starts. On Unix/Linux a `startup.sh` file is included so you just type `startup` (assuming you are in the `bin` directory where you located Tomcat).
  1. Start Tomcat running.
  2. Start your Web browser.
  3. Enter URL: <http://localhost:8080>
  4. You should see the screen on the following page if everything is set up ok.





Tomcat default homepage

Apache Tomcat/7.0.54

If you're seeing this, you've successfully installed Tomcat. Congratulations!



Recommended Reading:  
[Security Considerations HOW-TO](#)  
[Manager Application HOW-TO](#)  
[Clustering/Session Replication HOW-TO](#)

Server Status  
Manager App  
Host Manager

Developer Quick Start

[Tomcat Setup](#)      [Realms & AAA](#)      [Examples](#)      [Servlet Specifications](#)  
[First Web Application](#)      [JDBC DataSources](#)      [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

Documentation

[Tomcat 7.0 Documentation](#)  
[Tomcat 7.0 Configuration](#)

`$CATALINA_HOME/RUNNING.txt`

Developers may be interested in:

Getting Help

[FAQ and Mailing Lists](#)

The following mailing lists are available:  
[announcements](#), [releases](#), [security](#), [notifications](#). (Low volume).  
User support and discussion  
[taglibs-user](#)  
User support and discussion for [Apache Taglibs](#)

Click the Server Status link and you'll see the screen on the next page.




If you set up an administrator login when you click on the these items this window will pop-up to enter your administrator credentials.

**Windows Security**

Internet Explorer

The server localhost at Tomcat Manager Application requires a username and password.

Warning: This server is requesting that your username and password be sent in an insecure manner (basic authentication without a secure connection).


 User name:

Password:

Remember my credentials

OK Cancel

# Apache Tomcat/7.0.42



If you're seeing this message, it means that your browser has blocked some elements from this page.

[Recommended Security Configuration](#)

[Manager Application](#)

[Clustering/](#)

### Developer Quick Start

[Tomcat Setup](#)

[First Web Application](#)

### Managing Tomcat

For security, access to the [manager webapp](#) is restricted. Users are defined in:

```
$CATALINA_HOME/conf/tomcat-users.xml
```

<http://localhost:8080/host-manager/html>

### Documentation

[Tomcat 7.0 Documentation](#)

[Tomcat 7.0 Configuration](#)

[Tomcat Wiki](#)

Find additional important configuration

### Getting Help

[FAQ and Mailing Lists](#)

The following mailing lists are available:

[announce@tomcat.apache.org](mailto:announce@tomcat.apache.org)  
Important announcements, releases, security vulnerability notifications. (Low volume).

Home Documentation Configuration Examples Wiki Mailing Lists Find Help

Apache Software Foundation  
<http://www.apache.org/>

### Congratulations!

[Server Status](#)

[Manager App](#)

[Host Manager](#)

[Servlet Specifications](#)

[Tomcat Versions](#)



Tomcat server status information page. Note server version is displayed here.



## Server Status

Manager			
<a href="#">List Applications</a>	<a href="#">HTML Manager Help</a>	<a href="#">Manager Help</a>	<a href="#">Complete Server Status</a>

Server Information							
Tomcat Version	JVM Version	JVM Vendor	OS Name	OS Version	OS Architecture	Hostname	IP Address
Apache Tomcat/7.0.54	1.7.0_25-b17	Oracle Corporation	Windows 8	6.2	amd64	HEC-236-G4TB8Y1	10.173.203.134

## JVM

Free memory: 81.11 MB Total memory: 122.68 MB Max memory: 227.56 MB

Memory Pool	Type	Initial	Total	Maximum	Used
PS Eden Space	Heap memory	32.00 MB	32.00 MB	74.68 MB	22.86 MB (30%)
PS Old Gen	Heap memory	85.37 MB	85.37 MB	170.68 MB	13.38 MB (7%)
PS Survivor Space	Heap memory	5.31 MB	5.31 MB	5.31 MB	5.31 MB (99%)
Code Cache	Non-heap memory	2.43 MB	2.43 MB	48.00 MB	1.47 MB (3%)
PS Perm Gen	Non-heap memory	20.75 MB	22.25 MB	82.00 MB	22.17 MB (27%)

"ajp-bio-8009"





Tomcat Web App Manager will be a very useful tool for you when deploying and developing servlets. Get to know it now.

## 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	0	Start <input type="button" value="Stop"/> <input type="button" value="Reload"/> <input type="button" value="Undeploy"/> <input type="button" value="Expire sessions"/> with idle ≥ <input type="text" value="30"/> minutes
/docs	None specified	Tomcat Documentation	true	0	Start <input type="button" value="Stop"/> <input type="button" value="Reload"/> <input type="button" value="Undeploy"/> <input type="button" value="Expire sessions"/> with idle ≥ <input type="text" value="30"/> minutes
/manager	None specified	Tomcat Manager Application	true	1	Start <input type="button" value="Stop"/> <input type="button" value="Reload"/> <input type="button" value="Undeploy"/> <input type="button" value="Expire sessions"/> with idle ≥ <input type="text" value="30"/> minutes

**Deploy**

Deploy directory or WAR file located on server

Context Path (required):



When the underlying code for a servlet is modified, you need to reload the servlet on the server in order for the clients to see the changes.

CAUTION! Undeploy removes all servlet files from the server!



## 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	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	None specified	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

### Deploy

Deploy directory or WAR file located on server

Context Path (required):





We won't be using the Tomcat Virtual Host Manager tool. This is a Tomcat tool that is useful when Tomcat is hosting several virtual servers.



### Tomcat Virtual Host Manager

Message: OK

Host Manager			
<a href="#">List Virtual Hosts</a>	<a href="#">HTML Host Manager Help (TODO)</a>	<a href="#">Host Manager Help (TODO)</a>	<a href="#">Server Status</a>

Host name		
Host name	Host aliases	Commands
localhost		Host Manager installed - commands disabled

**Add Virtual Host**

Host

Name:

Aliases:

App base:

AutoDeploy

DeployOnStartup

DeployXML

UseDocWAPs





# 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. You will need to do things on the administrator side (you must have set the host manager application during the installation process, (see page 27)), 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 38 for an example.
- Checking the status of the server can also be accomplished from the Tomcat homepage. See page 38 for a sample.





## Server Status

Manager			
<a href="#">List Applications</a>	<a href="#">HTML Manager Help</a>	<a href="#">Manager Help</a>	<a href="#">Complete Server Status</a>

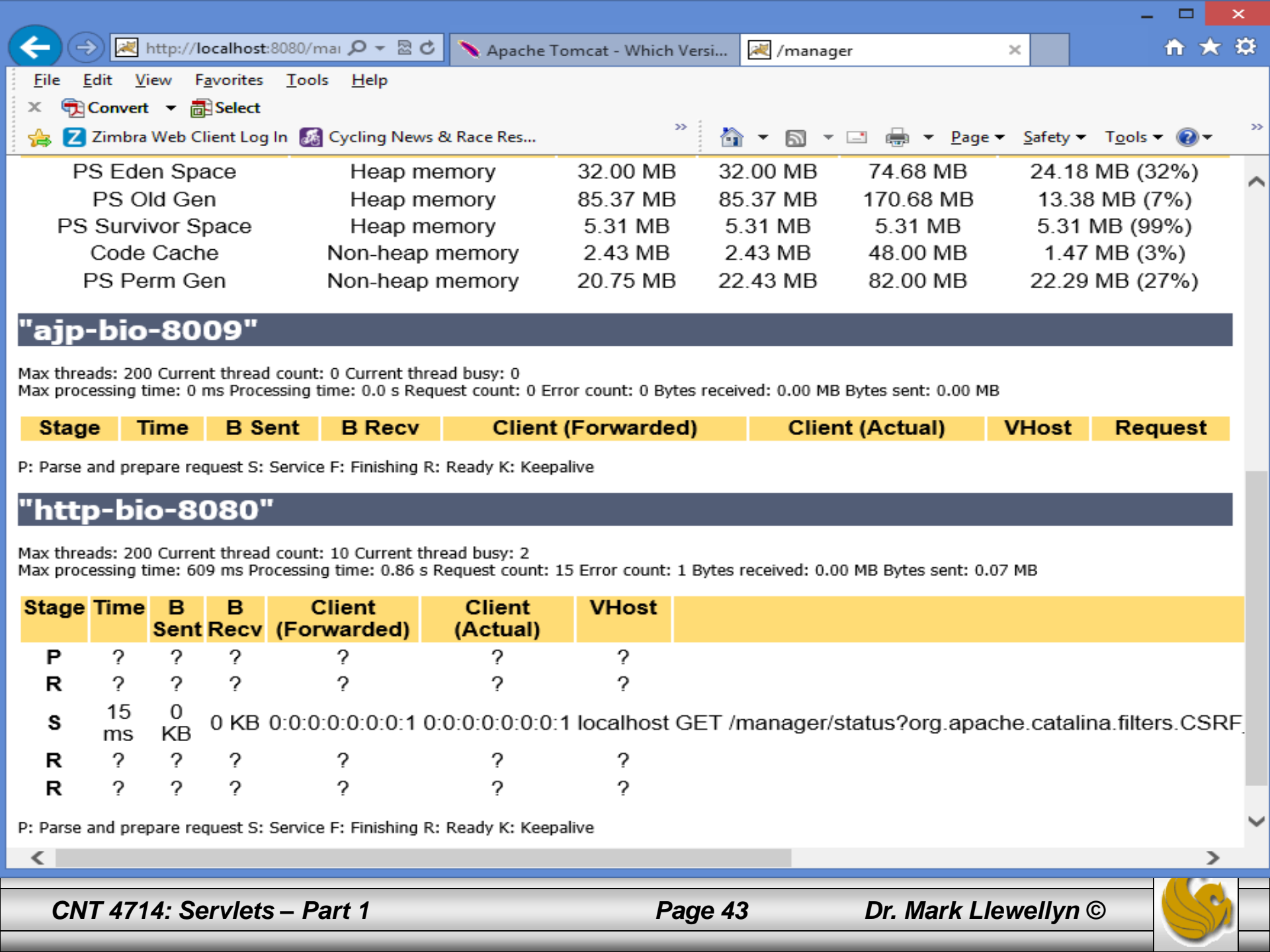
Server Information							
Tomcat Version	JVM Version	JVM Vendor	OS Name	OS Version	OS Architecture	Hostname	IP Address
Apache Tomcat/7.0.54	1.7.0_25-b17	Oracle Corporation	Windows 8	6.2	amd64	HEC-236-G4TB8Y1	10.173.203.134

## JVM

Free memory: 79.80 MB Total memory: 122.68 MB Max memory: 227.56 MB

Memory Pool	Type	Initial	Total	Maximum	Used
PS Eden Space	Heap memory	32.00 MB	32.00 MB	74.68 MB	24.18 MB (32%)
PS Old Gen	Heap memory	85.37 MB	85.37 MB	170.68 MB	13.38 MB (7%)
PS Survivor Space	Heap memory	5.31 MB	5.31 MB	5.31 MB	5.31 MB (99%)





PS Eden Space	Heap memory	32.00 MB	32.00 MB	74.68 MB	24.18 MB (32%)
PS Old Gen	Heap memory	85.37 MB	85.37 MB	170.68 MB	13.38 MB (7%)
PS Survivor Space	Heap memory	5.31 MB	5.31 MB	5.31 MB	5.31 MB (99%)
Code Cache	Non-heap memory	2.43 MB	2.43 MB	48.00 MB	1.47 MB (3%)
PS Perm Gen	Non-heap memory	20.75 MB	22.43 MB	82.00 MB	22.29 MB (27%)

## "ajp-bio-8009"

Max threads: 200 Current thread count: 0 Current thread busy: 0  
 Max processing time: 0 ms Processing time: 0.0 s Request count: 0 Error count: 0 Bytes received: 0.00 MB Bytes sent: 0.00 MB

Stage	Time	B Sent	B Recv	Client (Forwarded)	Client (Actual)	VHost	Request
-------	------	--------	--------	--------------------	-----------------	-------	---------

P: Parse and prepare request S: Service F: Finishing R: Ready K: Keepalive

## "http-bio-8080"

Max threads: 200 Current thread count: 10 Current thread busy: 2  
 Max processing time: 609 ms Processing time: 0.86 s Request count: 15 Error count: 1 Bytes received: 0.00 MB Bytes sent: 0.07 MB

Stage	Time	B Sent	B Recv	Client (Forwarded)	Client (Actual)	VHost	Request
P	?	?	?	?	?	?	
R	?	?	?	?	?	?	
S	15 ms	0 KB	0 KB	0:0:0:0:0:0:0:1	0:0:0:0:0:0:0:1	localhost GET /manager/status?org.apache.catalina.filters.CSRF...	
R	?	?	?	?	?	?	
R	?	?	?	?	?	?	

P: Parse and prepare request S: Service F: Finishing R: Ready K: Keepalive



# 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`, `lib`, `logs`, `temp`, `webapps`, and `work`.

## `bin`

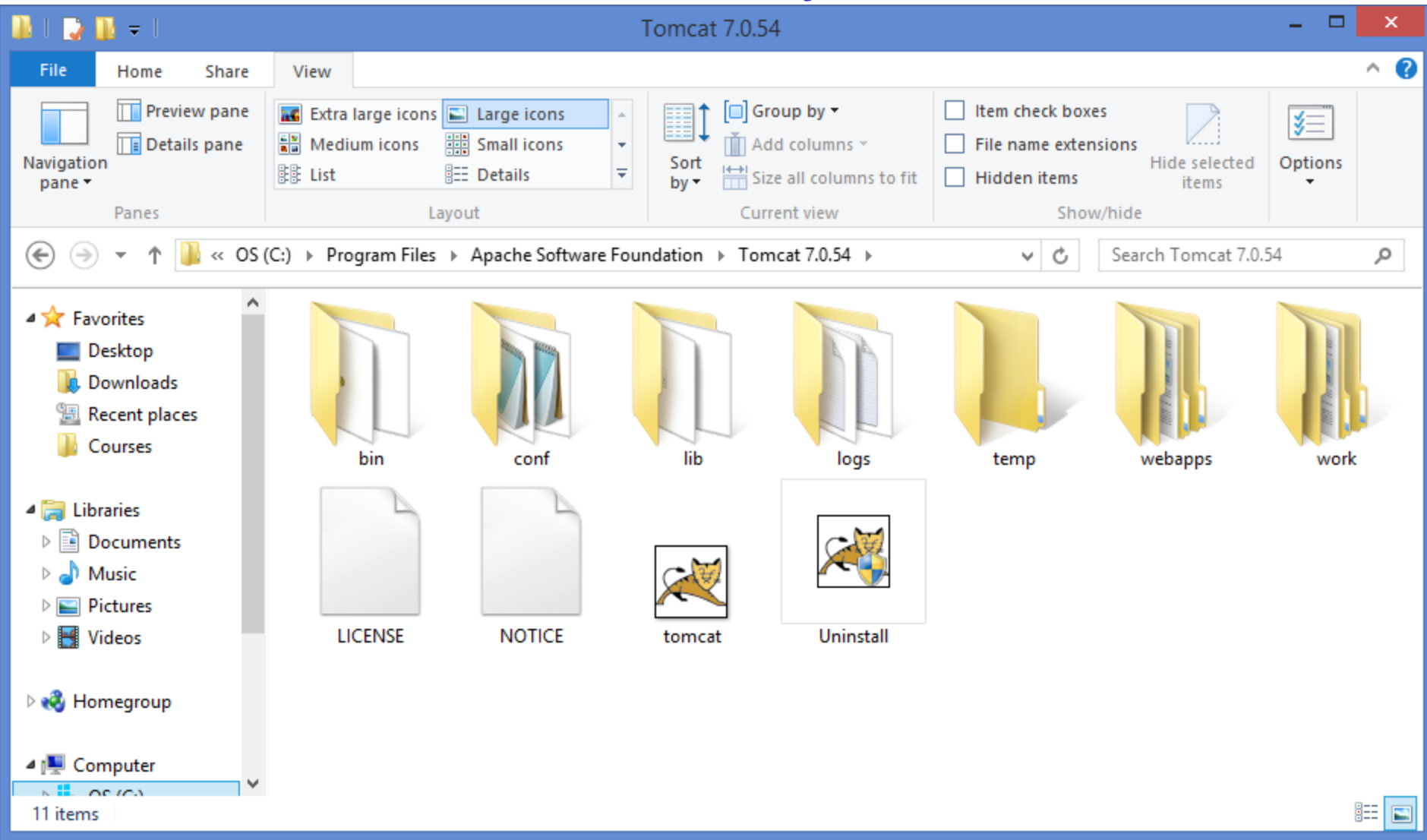
- Directory `bin` contains scripts for starting and stopping Tomcat as well as some additional tools.

## `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 47.
- 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 48.
- 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.



```
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
simpleXMLxhr.js notes.html notes.html server.xml
64 and responses are returned. Documentation at :
65 Java HTTP Connector: /docs/config/http.html (blocking & non-blocking)
66 Java AJP Connector: /docs/config/ajp.html
67 APR (HTTP/AJP) Connector: /docs/apr.html
68 Define a non-SSL HTTP/1.1 Connector on port 8080
69 -->
70 <Connector port="8080" protocol="HTTP/1.1"
71         connectionTimeout="20000"
72         redirectPort="8443" />
73 <!-- A "Connector" using the shared thread pool-->
74 <!--
75 <Connector executor="tomcatThreadPool"
76         port="8080" protocol="HTTP/1.1"
77         connectionTimeout="20000"
78         redirectPort="8443" />
79 -->
80 <!-- Define a SSL HTTP/1.1 Connector on port 8443
81 This connector uses the BIO implementation that requires the JSSE
82 style configuration. When using the APR/native implementation, the
83 OpenSSL style configuration is required as described in the APR/native
84 documentation -->
85 <!--
86 <Connector port="8443" protocol="org.apache.coyote.http11.Http11Protocol"
87         maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
88         clientAuth="false" sslProtocol="TLS" />
89 -->
```

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







```
4589     <extension>zmm</extension>
4590     <mime-type>application/vnd.handheld-entertainment+xml
4591 </mime-mapping>
4592
4593 <!-- ===== Default Welcome File List ===== -->
4594 <!-- When a request URI refers to a directory, the default servlet looks -->
4595 <!-- for a "welcome file" within that directory and, if present, to the -->
4596 <!-- corresponding resource URI for display. -->
4597 <!-- If no welcome files are present, the default servlet either serves a -->
4598 <!-- directory listing (see default servlet configuration on how to -->
4599 <!-- customize) or returns a 404 status, depending on the value of the -->
4600 <!-- listings setting. -->
4601 <!-- -->
4602 <!-- If you define welcome files in your own application's web.xml -->
4603 <!-- deployment descriptor, that list *replaces* the list configured -->
4604 <!-- here, so be sure to include any of the default values that you wish -->
4605 <!-- to use within your application. -->
4606
4607 <welcome-file-list>
4608     <welcome-file>index.html</welcome-file>
4609     <welcome-file>index.htm</welcome-file>
4610     <welcome-file>index.jsp</welcome-file>
4611 </welcome-file-list>
4612
4613 </web-app>
4614
```

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.





# A Tour of Tomcat (cont.)

## logs

- The logs directory contains a number 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, beginning with version 6.0.29, have condensed these into a single directory named `lib`. Any custom `.jar` files that may be needed throughout Tomcat, such as a JDBC driver, are placed in this directory.



# 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 objects 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, then 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](http://www.java.sun.com/j2ee/1.4/docs/api/index.html), however, a few of the more common ones are shown on page 18. (Note: you can also get to them from Tomcat, see next page.)



# Apache Tomcat/7.0.54



If you're seeing this, you've successfully installed Tomcat. Congratulations!



Re  
Sec  
Ma  
[Clustering/Session Replication HOW-TO](#)

This link will take you to the servlet specification pages.

- Server Status
- Manager App
- Host Manager

## Developer Quick Start

- [Tomcat Setup](#)
- [First Web Application](#)
- [Realms & AAA](#)
- [JDBC DataSources](#)
- [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

### Documentation

- [Tomcat 7.0 Documentation](#)
- [Tomcat 7.0 Configuration](#)
- [Tomcat Wiki](#)

Find additional important configuration information in:

### Getting Help

#### FAQ and Mailing Lists

The following mailing lists are available:

- [tomcat-announce](#)  
Important announcements, releases, security vulnerability notifications. (Low volume).





# 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](http://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 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.



\*C:\Program Files\Apache Software Foundation\Tomcat 7.0\webapps\first-example\WEB-INF\classes\WelcomeServlet.java - Notepad++

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

Class name WelcomeServlet

```
1 // A simple servlet to process get requests.
2
3 import javax.servlet.*;
4 import javax.servlet.http.*;
5 import java.io.*;
6
7 public class WelcomeServlet extends HttpServlet {
8     // process "get" requests from clients
9     protected void doGet( HttpServletRequest request,
10        HttpServletResponse response ) throws ServletException, IOException {
11
12         response.setContentType( "text/html" );
13         PrintWriter out = response.getWriter();
14         // send HTML5 page to client
15
16         // start HTML5 document
17         out.println( "<!DOCTYPE html>" );
18         out.println( "<html lang=\"en\">" );
19         out.println( "<meta charset=\"utf-8\">" );
20         // head section of document
21         out.println( "<head>" );
22         out.println( "<style type='text/css'>" );
23         out.println( "<!-- body{background-color:blue; color:white;} -->" );
24         out.println( "</style>" );
25         out.println( "<title>Welcome to Servlets!</title>" );
26         out.println( "</head>" );
27         // body section of document
28         out.println( "<body>" );
29         out.println( "<h1>Hello!!</h1>" );
30         out.println( "<h1>Welcome To The Exciting World Of Servlet Technology!</h1>" );
31         out.println( "</body>" );
32         // end HTML5 document
33
34         out.println( "</html>" );
35         out.close(); // close stream to complete the page
36     }
37 }
```

doGet handles the HTTP get request – override method

Set MIME content type

HTML document returned to the client

End the HTML document generated by the servlet.

Java source file length: 1332 lines: 39 Ln: 32 Col: 19 Sel: 0|0



# Handling HTTP `get` Requests (cont.)

- The servlet creates an HTML 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 `HttpServletResponse` 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.



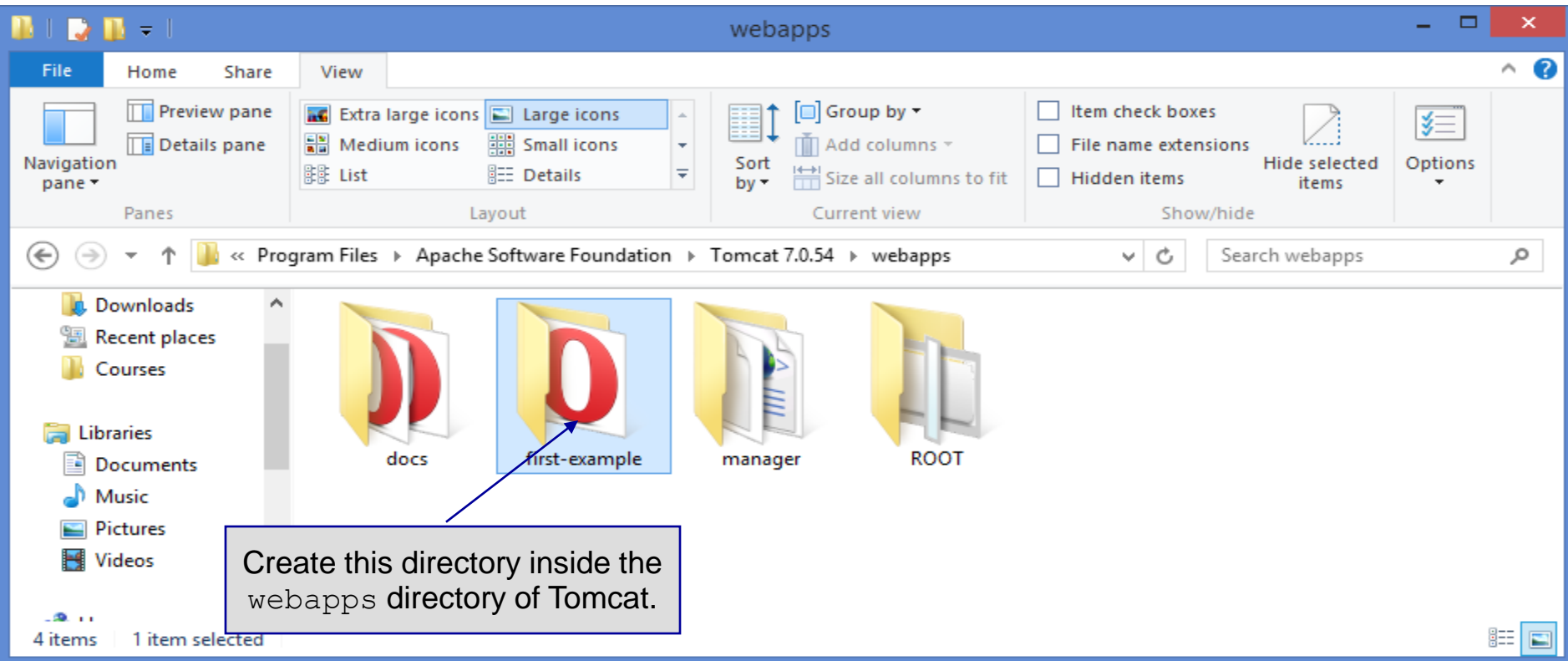
# Creating a Web Application

- 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-example`.





# Creating a Web Application (cont.)

- Within the `first-example` 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.



The web-app tag

Optional display-name tag.  
Used by administrator tools.

Optional description for reading the  
xml file.

Servlet declaration. Specifies the name of the  
servlet, the implementing class file, and any  
initialization parameters.

Servlet mapping associates a servlet name with a  
class of URLs. One servlet may be configured to  
handle multiple sets of URLs, however, only one  
servlet can handle any given URL.

```

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   <!-- General description of your Web application -->
7   <display-name>
8     CNT 4714 Summer 2014 Servlet First Example
9   </display-name>
10  <description>
11    This is the Web application in which we
12    will demonstrate our first servlet example.
13  </description>
14  <!-- Servlet definitions -->
15  <servlet>
16    <servlet-name>welcome1</servlet-name>
17    <description>
18      A simple servlet that handles an HTTP get request.
19    </description>
20    <servlet-class>
21      WelcomeServlet
22    </servlet-class>
23  </servlet>
24  <servlet-mapping>
25    <servlet-name>welcome1</servlet-name>
26    <url-pattern>/welcome1</url-pattern>
27  </servlet-mapping>
28 </web-app>

```



The image shows a web browser window with the Office XML Handler (IE10) Editor View. The address bar shows the file path: C:\Program Files\Apache Software Foundation\Tomcat\7.0.54. The browser's menu bar includes File, Edit, View, Favorites, Tools, and Help. The toolbar contains various icons for navigation and editing. The main content area displays XML code for a web application. The code is as follows:

```
<?xml version="1.0"?>
- <web-app version="2.4" xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns="http://java.sun.com/xml/ns/j2ee">
  <!-- General description of your Web application -->
  <display-name> CNT 4714 Summer 2014 Servlet First Example </display-name>
  <description> This is the Web application in which we will demonstrate our first servlet example.
  </description>
  <!-- Servlet definitions -->
  - <servlet>
    <servlet-name>welcome1</servlet-name>
    <description> A simple servlet that handles an HTTP get request. </description>
    <servlet-class> WelcomeServlet </servlet-class>
  </servlet>
  - <servlet-mapping>
    <servlet-name>welcome1</servlet-name>
    <url-pattern>/welcome1</url-pattern>
  </servlet-mapping>
</web-app>
```

Office XML Handler (IE10) 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-example/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).





The Tomcat Web Application Manager recognizes the new webapp.

## Tomcat Web Application

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	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	None specified	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/first-example	None specified	CNT 4714 Summer 2014 Servlet First Example	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes



# Tomcat/Java Configuration - The Servlet API

## **IMPORTANT !!**

- Your Tomcat installation includes the `servlet-api.jar` file. This file can be found in the `lib` folder in Tomcat. Copy this file into your `jdk/jre/lib/ext` folder to allow the java compiler access to the `javax.servlet` package.
- Note that your Java set-up may already have this installed depending on several things, so check your `jdk/jre/lib/ext` folder first.



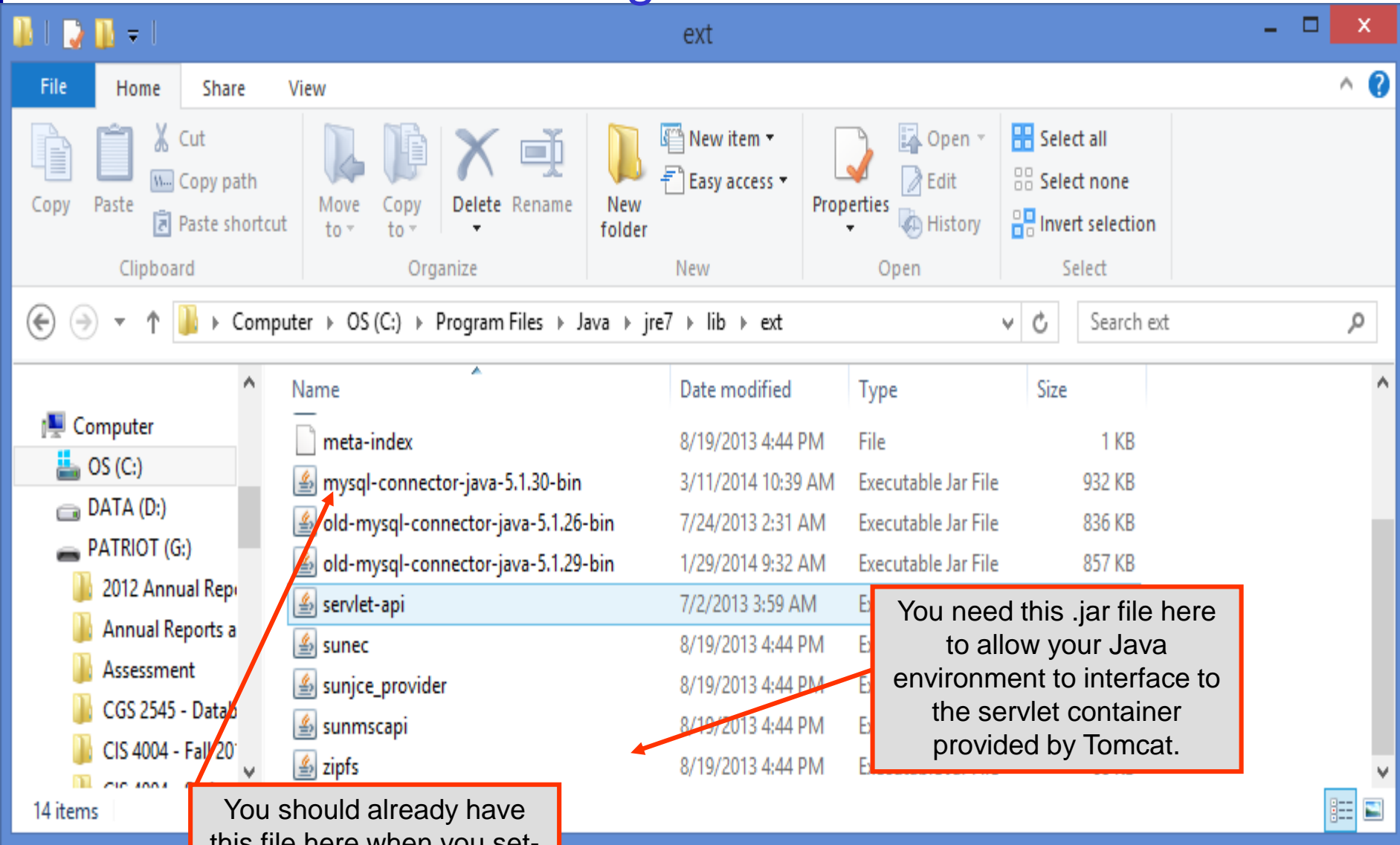
# Tomcat/Java Configuration - The Servlet API

This is the .jar file that you need to copy into your jre/lib/ext folder.

Name	Date modified	Type	Size
annotations-api	7/2/2013 3:59 AM	Executable Jar File	15 KB
catalina	7/2/2013 3:59 AM	Executable Jar File	1,545 KB
catalina-ant	7/2/2013 3:59 AM	Executable Jar File	53 KB
catalina-ha	7/2/2013 3:59 AM	Executable Jar File	132 KB
catalina-tribes	7/2/2013 3:59 AM	Executable Jar File	252 KB
ecj-4.2.2	7/2/2013 3:59 AM	Executable Jar File	1,760 KB
el-api	7/2/2013 3:59 AM	Executable Jar File	46 KB
jasper	7/2/2013 3:59 AM	Executable Jar File	586 KB
jasper-el	7/2/2013 3:59 AM	Executable Jar File	121 KB
jsp-api	7/2/2013 3:59 AM	Executable Jar File	87 KB
<b>servlet-api</b>	7/2/2013 3:59 AM	Executable Jar File	<b>174 KB</b>
tomcat-api	7/2/2013 3:59 AM	Executable Jar File	7 KB



# Tomcat/Java Configuration - The Servlet API



ext

File Home Share View

Cut Copy Paste Copy path Paste shortcut Move to Copy to Delete Rename New folder New item Easy access Properties Open Edit History Select all Select none Invert selection

Computer > OS (C:) > Program Files > Java > jre7 > lib > ext

Name	Date modified	Type	Size
meta-index	8/19/2013 4:44 PM	File	1 KB
mysql-connector-java-5.1.30-bin	3/11/2014 10:39 AM	Executable Jar File	932 KB
old-mysql-connector-java-5.1.26-bin	7/24/2013 2:31 AM	Executable Jar File	836 KB
old-mysql-connector-java-5.1.29-bin	1/29/2014 9:32 AM	Executable Jar File	857 KB
<b>servlet-api</b>	7/2/2013 3:59 AM	Executable Jar File	857 KB
sunec	8/19/2013 4:44 PM	Executable Jar File	857 KB
sunjce_provider	8/19/2013 4:44 PM	Executable Jar File	857 KB
sunmscapi	8/19/2013 4:44 PM	Executable Jar File	857 KB
zipfs	8/19/2013 4:44 PM	Executable Jar File	857 KB

Computer  
OS (C:)  
DATA (D:)  
PATRIOT (G:)  
2012 Annual Rep  
Annual Reports a  
Assessment  
CGS 2545 - Datab  
CIS 4004 - Fall 20  
CIS 4004 - Fall 20

14 items

You should already have this file here when you set-up MySQL and Java.

You need this .jar file here to allow your Java environment to interface to the servlet container provided by Tomcat.

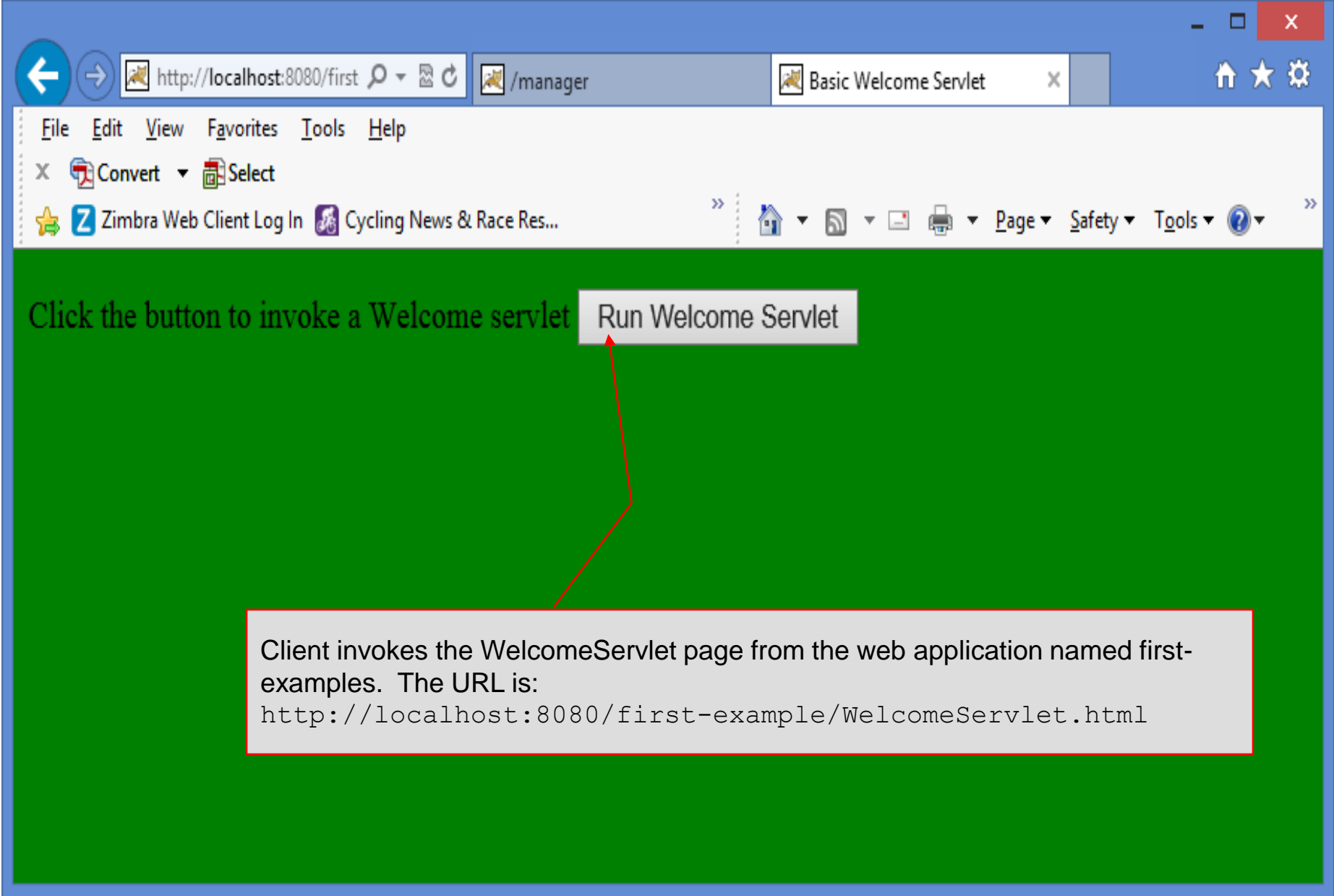




```
File Edit Search View Encoding Language Settings Macro Run Plugins Window ? X
notes.html Server.java Client.java WelcomeServlet.html
1 <!DOCTYPE html>
2 <html lang="en">
3 <!-- WelcomeServlet.html -->
4 <meta charset="utf-8">
5 <head>
6     <title>Basic Welcome Servlet</title>
7     <style type="text/css">
8         <!--
9             body{background-color: green;}
10        -->
11     </style>
12 </head>
13 <body>
14     <form action = "/first-example/welcome1" method = "get">
15     <p>
16         <label>Click the button to invoke a Welcome servlet
17         <input type = "submit" value = "Run Welcome Servlet" />
18     </label>
19     </p>
20     </form>
21 </body>
22 </html>
```

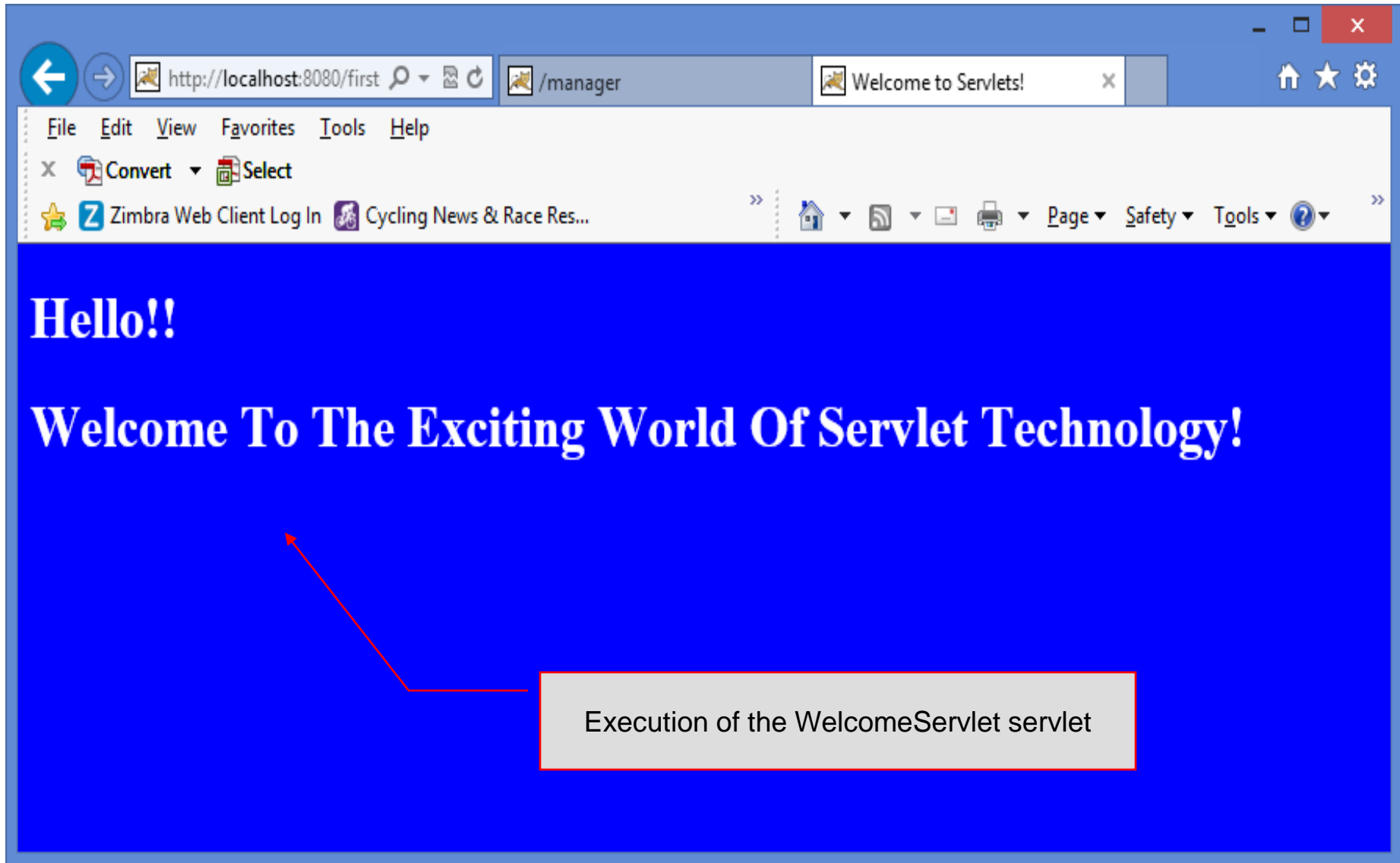
This is the HTML5 file that generates the output shown above which informs the client how to invoke the servlet.





Client invokes the WelcomeServlet page from the web application named first-examples. The URL is:  
`http://localhost:8080/first-example/WelcomeServlet.html`





# An XHTML Document

- The HTML5 document shown on page 73 provides a `form` that invokes the servlet defined on page 61.
- The form's `action` attribute (`/first-example/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.

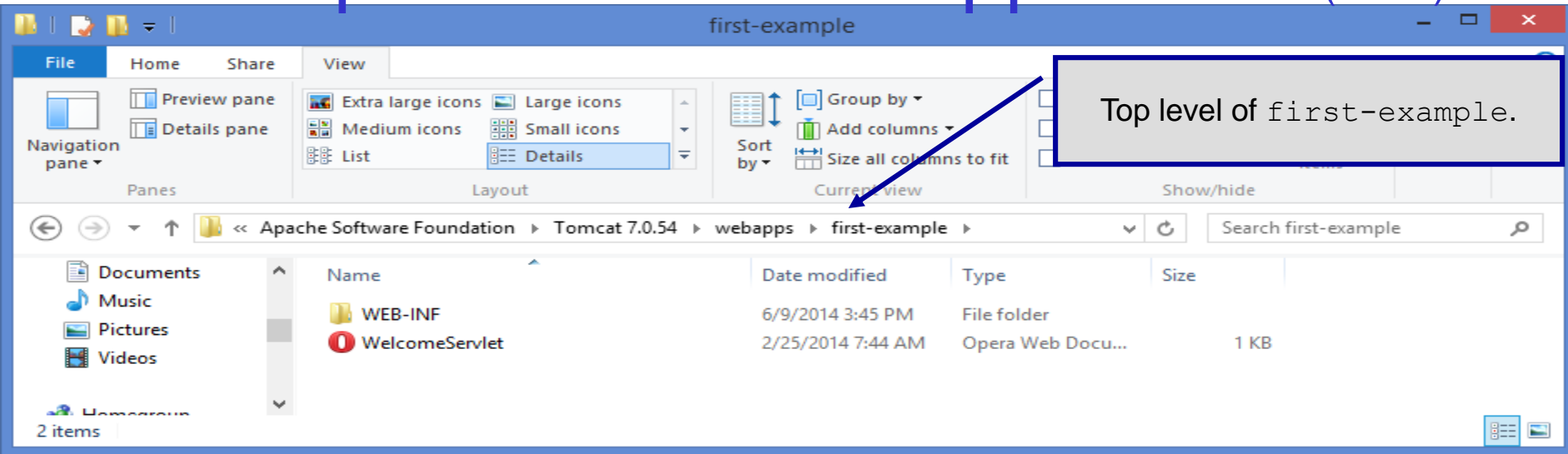


# Set-Up For First Web Application

- 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-example`.
  2. In the top level of `first-example` copy the `WelcomeServlet.html` file from the course code page.
  3. In the top level of `first-example` create a directory named `WEB-INF`.
  4. When steps 2 and 3 are complete the top level of `first-example` should look like the picture at the top of the next page.



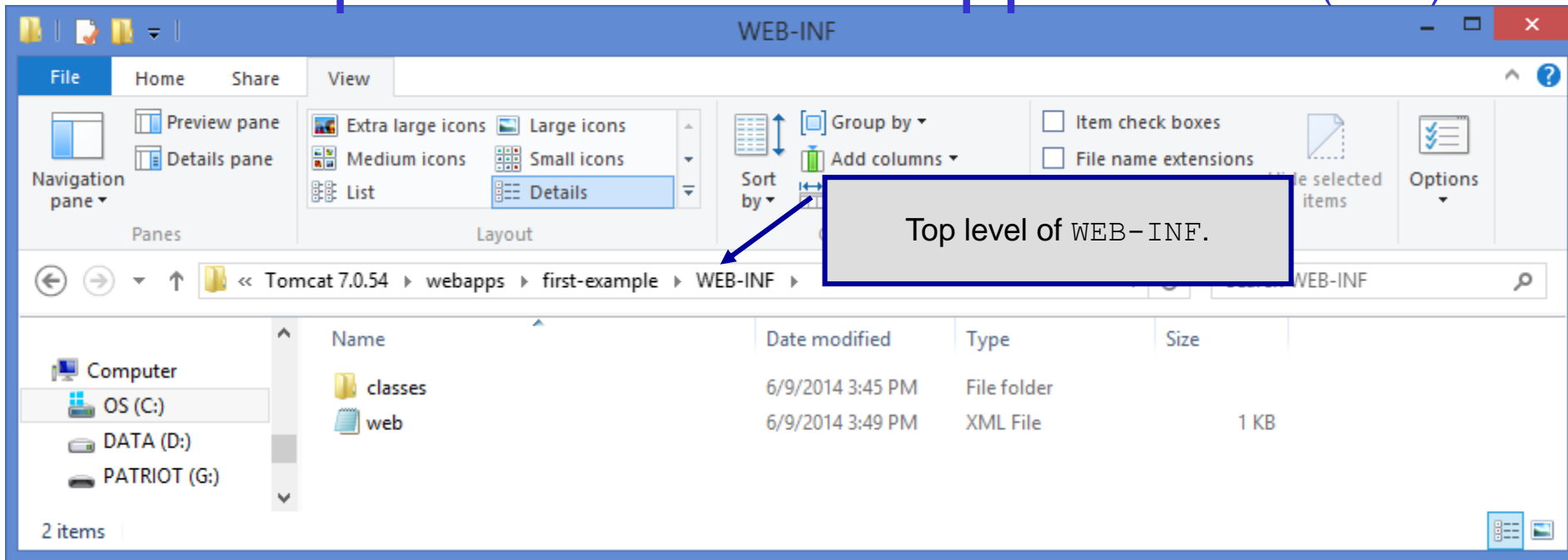
# Set-Up For First Web Application (cont.)



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



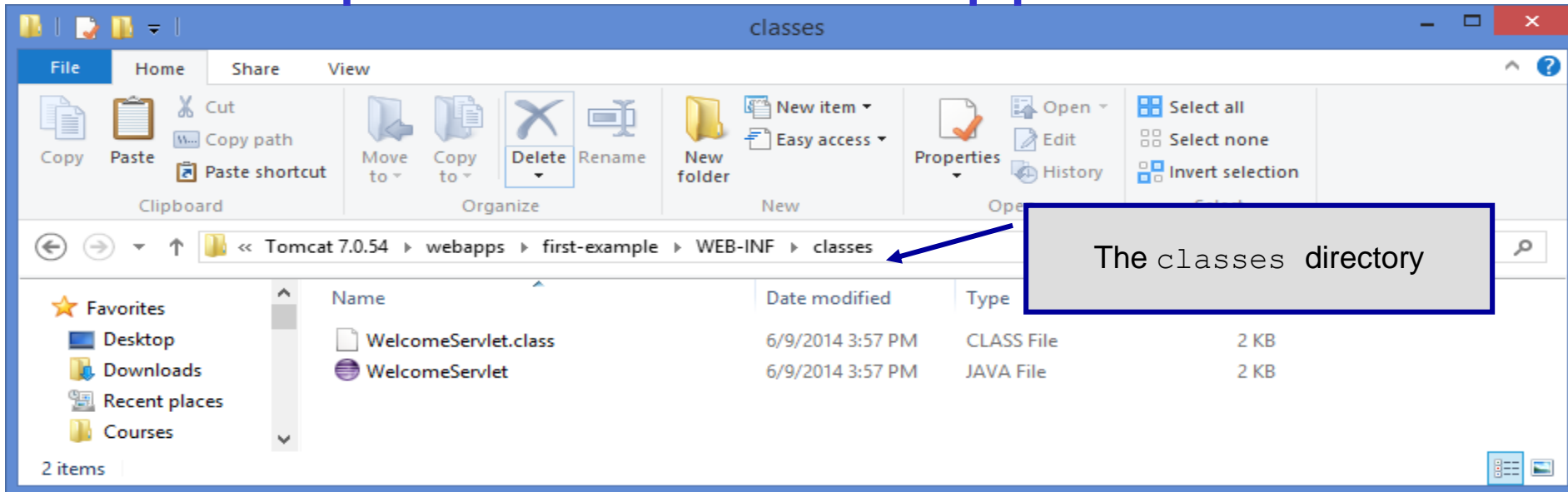
# Set-Up For First Web Application (cont.)



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/first-example/WelcomeServlet.html>. Tomcat and the servlet will do the rest. If all goes well you should see the output that was shown on pages 74-75.

