CNT 4714: Enterprise Computing
Fall 2022

Introduction To Servlet Technology – Part 1
Window OS Version

Instructor : Dr. Mark Llewellyn ~~HEC 236, 407-823-2790~~
Email: mark.llewellyn@ucf.edu

Office Hours via Zoom: M: 4:00-5:00 pm T&Th: 12:30-1:30pm
Office Hours Zoom Details: Meeting ID: 978 0856 0322
Passcode: 241343

Q&A Sessions via Zoom: Meeting ID: 956 3567 4945
Passcode: 517419

Department of Computer Science
University of Central Florida

Client-Server Relationship Revisited

- In a client-server relationship, the client requests that some action be performed, 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 scriplets 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

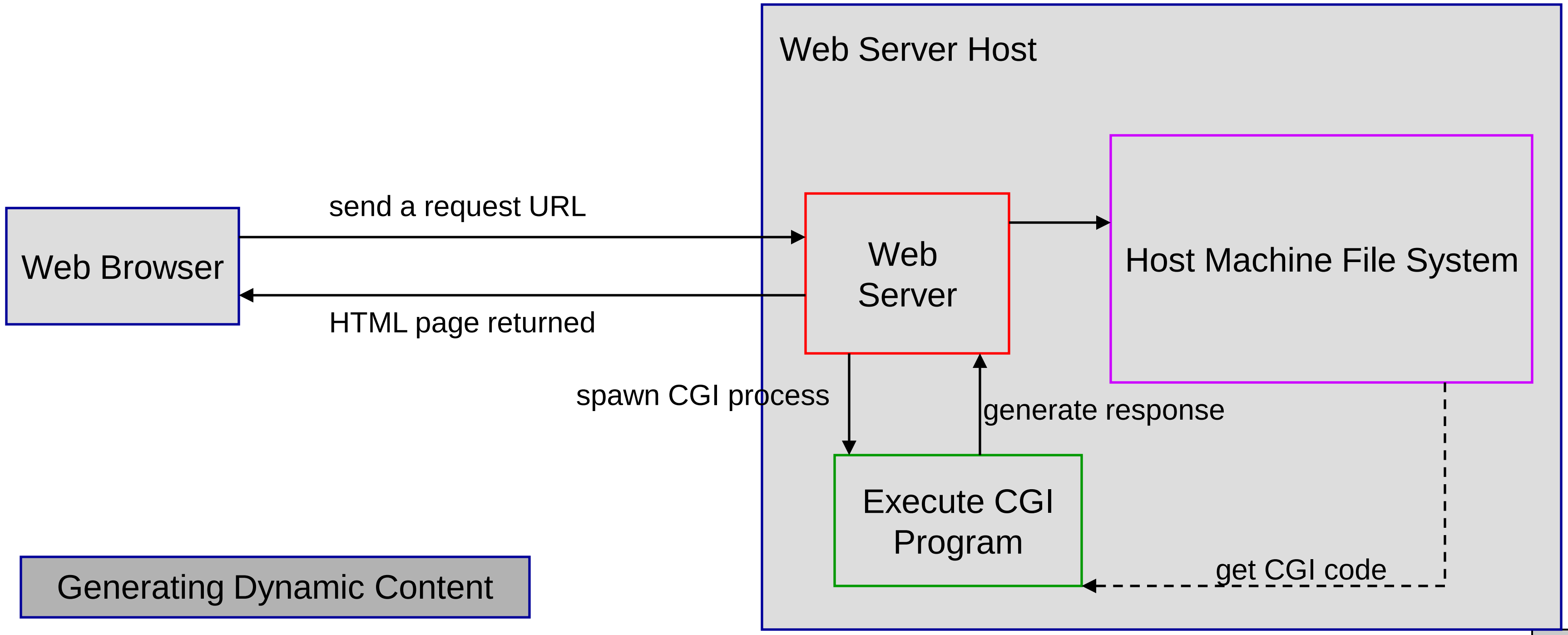
- Consider how a web page is displayed by a browser.
  - Typically, the web page is created using HTML and stored as a file on the web server. A user enters a URL for the file from a web browser. The browser contacts the web server and requests the file. The server finds the file and returns it to the 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) 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 10.1.1 (released on October 11, 2022).

Servlet Overview and Architecture

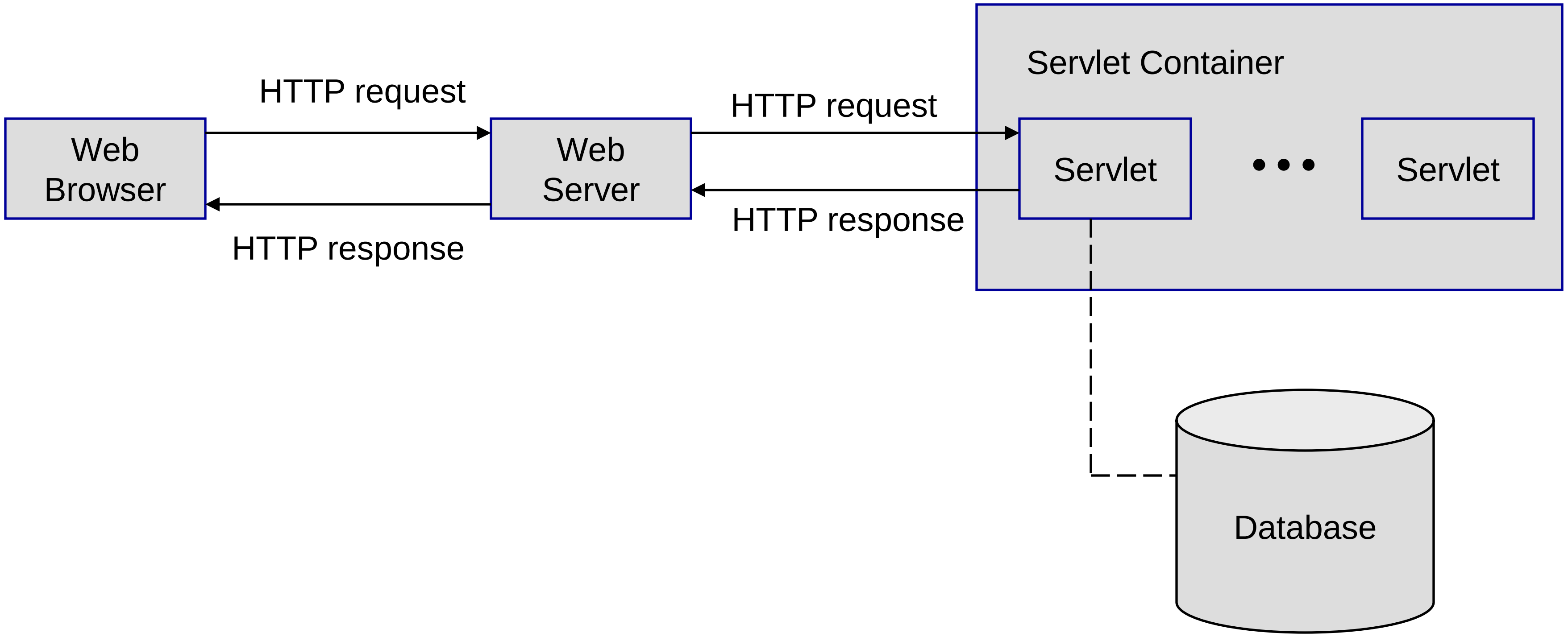
- The Internet offers many protocols. The HTTP (Hypertext Transfer Protocol) that forms the basis of the WWW uses URLs (Uniform Resource Locators) to locate resources on the Internet.
- URLs can represent files or directories and can represent complex tasks such as database lookups and Internet searches.
- JSP technology, basically an extension of servlet technology, simplifies the process of creating pages by separating presentation from content.
- Typically, JSPs are used when <u>most</u> 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.)

<u>Explanation of the architecture diagram on previous page</u>

- 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: https://javaee.github.io/javaee-spec/javadocs/
- 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
destroy( )	Called by the servlet container to indicate to a servlet that the servlet is being taken out of service.
getServletConfig( )	Returns a ServletConfig object, which contains initialization and startup parameters for this servlet.
getServletInfo( )	Returns information about the servlet, such as author, version, and copyright.
init( )	Called by the servlet container to indicate to a servlet that the servlet is being placed into service.
service( )	Called by the servlet container to allow the servlet to respond to a request.

The Servlet Lifecycle

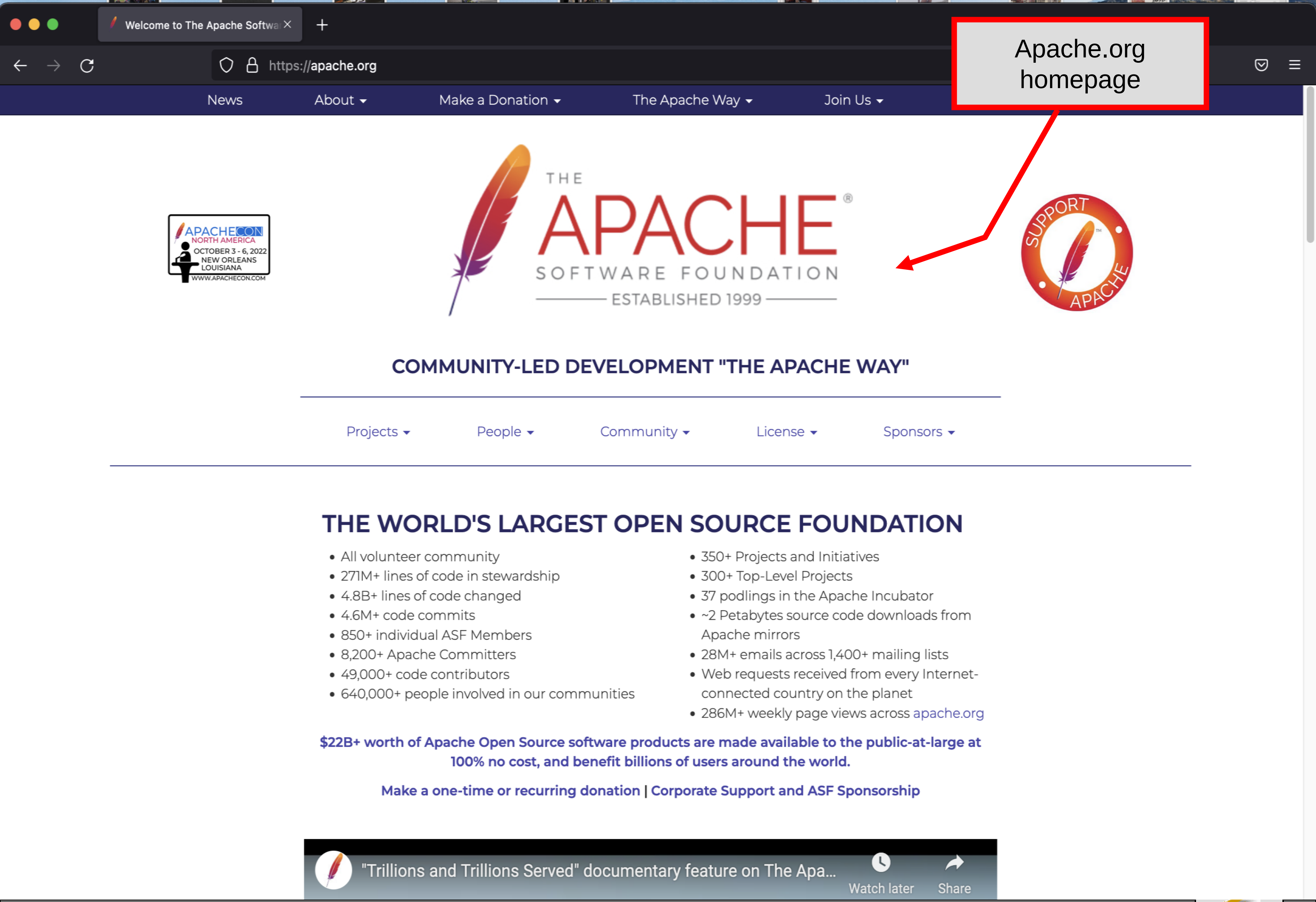
- After `init` completes execution, the servlet can respond to its first request.
- All requests are handled by the a servlet's `service` method, which receives the request, processes it and sends a response to the client.
- During the servlet's lifecycle, the method `service` is invoked once per request. 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 10.1.1 available from www.apache.org. This version was declared stable and released on October 11, 2022. This installation in on a host system running Windows 11 Pro OS.

1. Go to the apache.org home page as shown on page 21.
2. Select the Tomcat page from the menu on the bottom of the page (its way down the page). As shown on page 22.
3. Once in the Tomcat project, select Download Tomcat 10.1.1 from the screen as shown on page 23.
4. Once in the download binaries screen, select the option of your choice. This is shown on page 24.


Welcome to The Apache Softwa
https://apache.org
News
About
Make a Donation
The Apache Way
Join Us
Apache.org homepage
APACHECON NORTH AMERICA OCTOBER 3 - 6, 2022 NEW ORLEANS LOUISIANA WWW.APACHECON.COM
THE APACHE SOFTWARE FOUNDATION ESTABLISHED 1999
SUPPORT APACHE
Projects
People
Community
License
Sponsors
"Trillions and Trillions Served" documentary feature on The Apa...
Watch later
Share

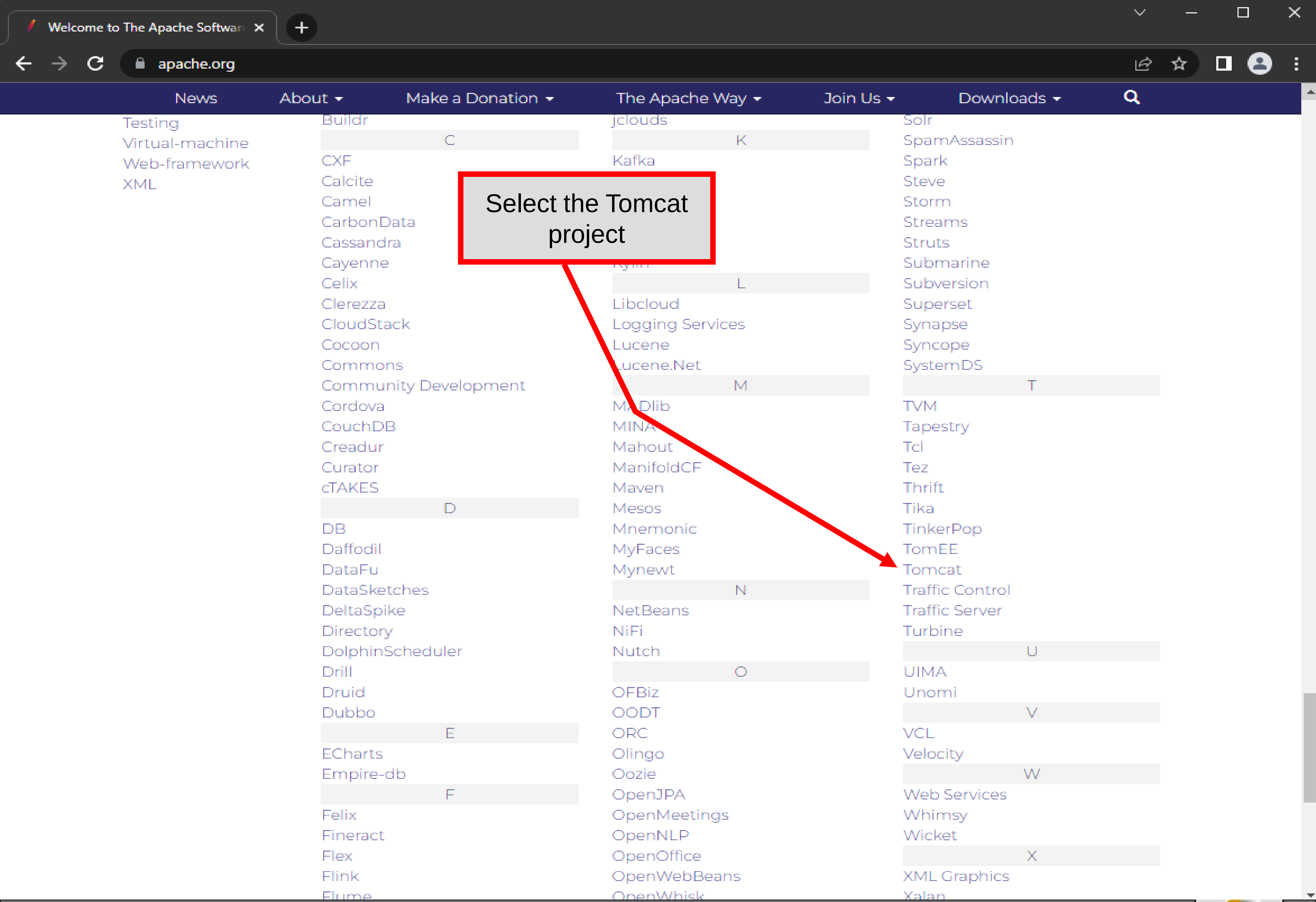
COMMUNITY-LED DEVELOPMENT "THE APACHE WAY"

THE WORLD'S LARGEST OPEN SOURCE FOUNDATION

- All volunteer community
- 271M+ lines of code in stewardship
- 4.8B+ lines of code changed
- 4.6M+ code commits
- 850+ individual ASF Members
- 8,200+ Apache Committers
- 49,000+ code contributors
- 640,000+ people involved in our communities
- 350+ Projects and Initiatives
- 300+ Top-Level Projects
- 37 podlings in the Apache Incubator
- ~2 Petabytes source code downloads from Apache mirrors
- 28M+ emails across 1,400+ mailing lists
- Web requests received from every Internet-connected country on the planet
- 286M+ weekly page views across apache.org

$22B+ worth of Apache Open Source software products are made available to the public-at-large at 100% no cost, and benefit billions of users around the world.

Make a one-time or recurring donation | Corporate Support and ASF Sponsorship



Welcome to The Apache Software
apache.org

News | About | Make a Donation | The Apache Way | Join Us | Downloads

Testing
Virtual-machine
Web-framework
XML

Buildr

C

CXF
Calcite
Camel
CarbonData
Cassandra
Cayenne
Celix
Clerezza
CloudStack
Cocoon
Commons
Community Development
Cordova
CouchDB
Creadur
Curator
cTAKES

D

DB
Daffodil
DataFu
DataSketches
DeltaSpike
Directory
DolphinScheduler
Drill
Druid
Dubbo

E

ECharts
Empire-db

F

Felix
Fineract
Flex
Flink
Flume

jclouds

K

Kafka

L

Libcloud
Logging Services
Lucene
Lucene.Net

M

MADlib
MINA
Mahout
ManifoldCF
Maven
Mesos
Mnemonic
MyFaces
Mynewt

N

NetBeans
NiFi
Nutch

O

OFBiz
OODT
ORC
Olingo
Oozie
OpenJPA
OpenMeetings
OpenNLP
OpenOffice
OpenWebBeans
OpenWhisk

Solr
SpamAssassin
Spark
Steve
Storm
Streams
Struts
Submarine
Subversion
Superset
Synapse
Syncope
SystemDS

T

TVM
Tapestry
Tcl
Tez
Thrift
Tika
TinkerPop
TomEE
Tomcat
Traffic Control
Traffic Server
Turbine

U

UIMA
Unomi

V

VCL
Velocity

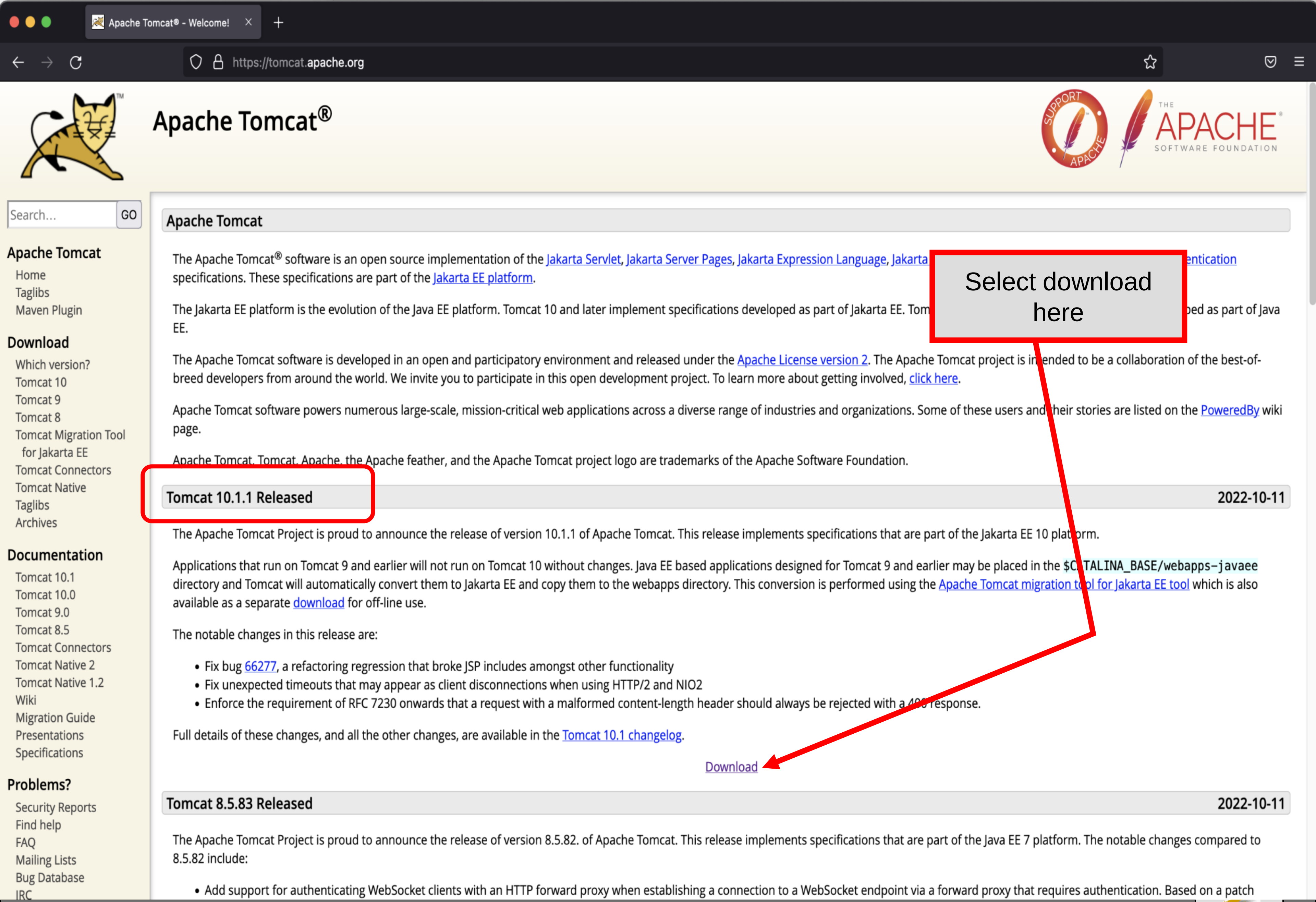
W

Web Services
Whimsy
Wicket

X

XML Graphics
Xalan

Select the Tomcat project



Apache Tomcat®

Apache Tomcat
- Home
- Taglibs
- Maven Plugin

Download
- Which version?
- Tomcat 10
- Tomcat 9
- Tomcat 8
- Tomcat Migration Tool for Jakarta EE
- Tomcat Connectors
- Tomcat Native
- Taglibs
- Archives

Documentation
- Tomcat 10.1
- Tomcat 10.0
- Tomcat 9.0
- Tomcat 8.5
- Tomcat Connectors
- Tomcat Native 2
- Tomcat Native 1.2
- Wiki
- Migration Guide
- Presentations
- Specifications

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

Apache Tomcat

The Apache Tomcat® software is an open source implementation of the Jakarta Servlet, Jakarta Server Pages, Jakarta Expression Language, Jakarta entication specifications. These specifications are part of the Jakarta EE platform.

The Jakarta EE platform is the evolution of the Java EE platform. Tomcat 10 and later implement specifications developed as part of Jakarta EE. Tom ped as part of Java EE.

The Apache Tomcat software is developed in an open and participatory environment and released under the Apache License version 2. The Apache Tomcat project 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 software 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 10.1.1 Released 2022-10-11

The Apache Tomcat Project is proud to announce the release of version 10.1.1 of Apache Tomcat. This release implements specifications that are part of the Jakarta EE 10 platform.

Applications that run on Tomcat 9 and earlier will not run on Tomcat 10 without changes. Java EE based applications designed for Tomcat 9 and earlier may be placed in the `$CATALINA_BASE/webapps-javaee` directory and Tomcat will automatically convert them to Jakarta EE and copy them to the webapps directory. This conversion is performed using the Apache Tomcat migration tool for Jakarta EE tool which is also available as a separate download for off-line use.

The notable changes in this release are:

- Fix bug 66277, a refactoring regression that broke JSP includes amongst other functionality
- Fix unexpected timeouts that may appear as client disconnections when using HTTP/2 and NIO2
- Enforce the requirement of RFC 7230 onwards that a request with a malformed content-length header should always be rejected with a 400 response.

Full details of these changes, and all the other changes, are available in the Tomcat 10.1 changelog.

Download

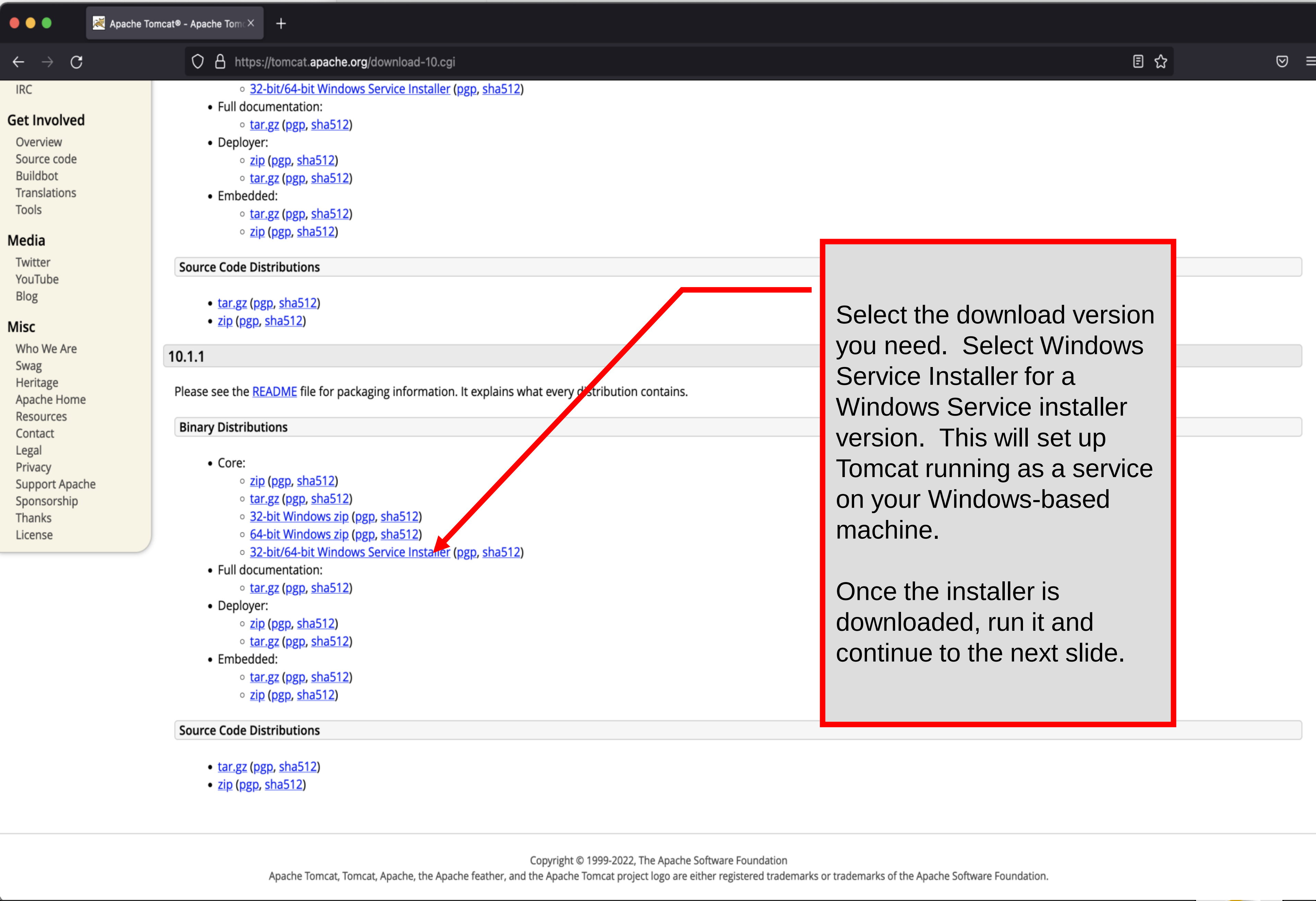
Tomcat 8.5.83 Released 2022-10-11

The Apache Tomcat Project is proud to announce the release of version 8.5.82. of Apache Tomcat. This release implements specifications that are part of the Java EE 7 platform. The notable changes compared to 8.5.82 include:

- Add support for authenticating WebSocket clients with an HTTP forward proxy when establishing a connection to a WebSocket endpoint via a forward proxy that requires authentication. Based on a patch

Apache Tomcat® - Welcome!
https://tomcat.apache.org
Search...
GO
SUPPORT APACHE
THE APACHE SOFTWARE FOUNDATION

Select download here


Apache Tomcat® - Apache Tom
https://tomcat.apache.org/download-10.cgi

IRC

Get Involved
- Overview
- Source code
- Buildbot
- Translations
- Tools

Media
- Twitter
- YouTube
- Blog

Misc
- Who We Are
- Swag
- Heritage
- Apache Home
- Resources
- Contact
- Legal
- Privacy
- Support Apache
- Sponsorship
- Thanks
- License

  - 32-bit/64-bit Windows Service Installer (pgp, sha512)
- Full documentation:
  - tar.gz (pgp, sha512)
- Deployer:
  - zip (pgp, sha512)
  - tar.gz (pgp, sha512)
- Embedded:
  - tar.gz (pgp, sha512)
  - zip (pgp, sha512)

Source Code Distributions

- tar.gz (pgp, sha512)
- zip (pgp, sha512)

10.1.1

Please see the README file for packaging information. It explains what every distribution contains.

Binary Distributions

- Core:
  - zip (pgp, sha512)
  - tar.gz (pgp, sha512)
  - 32-bit Windows zip (pgp, sha512)
  - 64-bit Windows zip (pgp, sha512)
  - 32-bit/64-bit Windows Service Installer (pgp, sha512)
- Full documentation:
  - tar.gz (pgp, sha512)
- Deployer:
  - zip (pgp, sha512)
  - tar.gz (pgp, sha512)
- Embedded:
  - tar.gz (pgp, sha512)
  - zip (pgp, sha512)

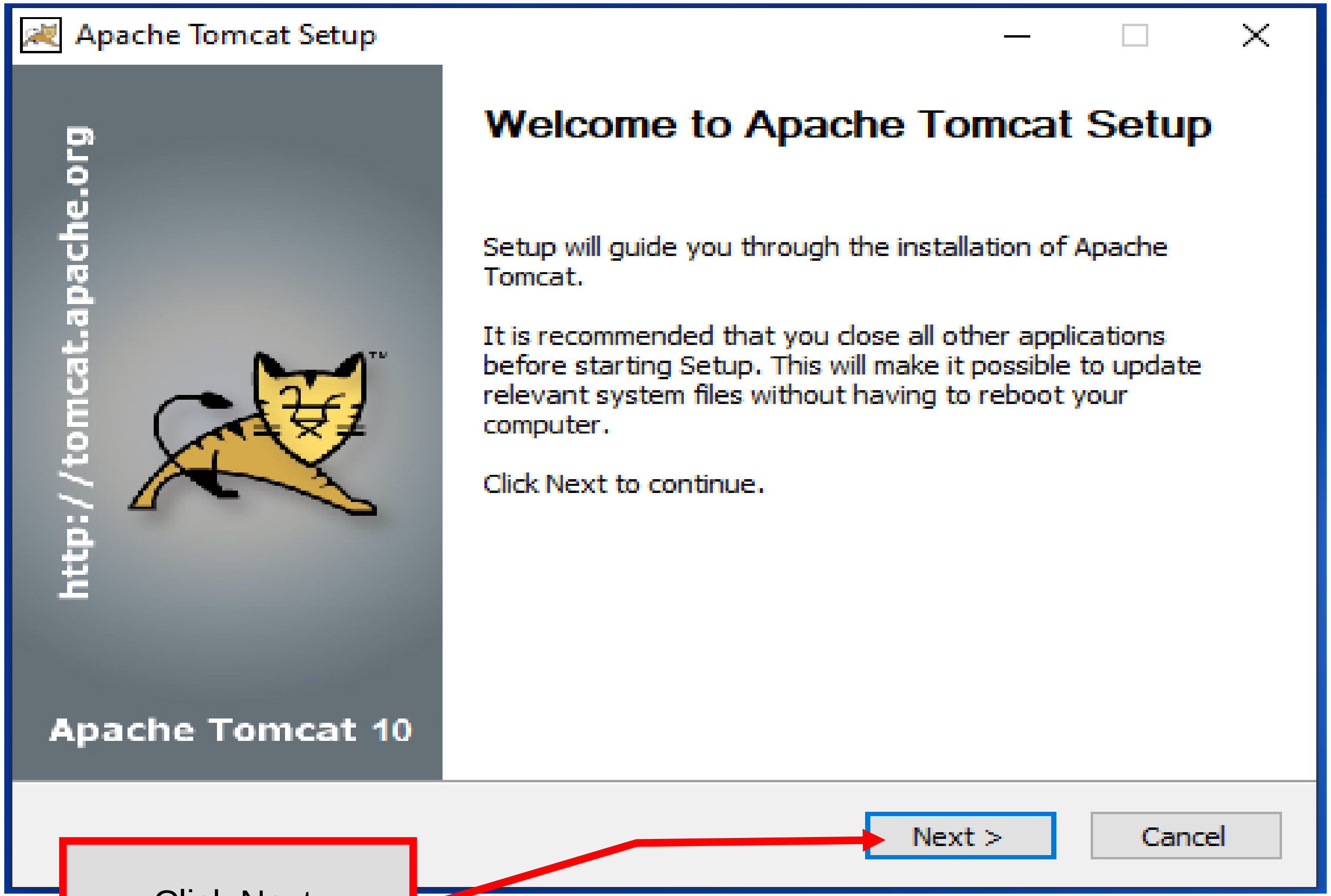
Source Code Distributions

- tar.gz (pgp, sha512)
- zip (pgp, sha512)

Copyright © 1999-2022, The Apache Software Foundation
Apache Tomcat, Tomcat, Apache, the Apache feather, and the Apache Tomcat project logo are either registered trademarks or trademarks of the Apache Software Foundation.

Select the download version you need. Select Windows Service Installer for a Windows Service installer version. This will set up Tomcat running as a service on your Windows-based machine.

Once the installer is downloaded, run it and continue to the next slide.



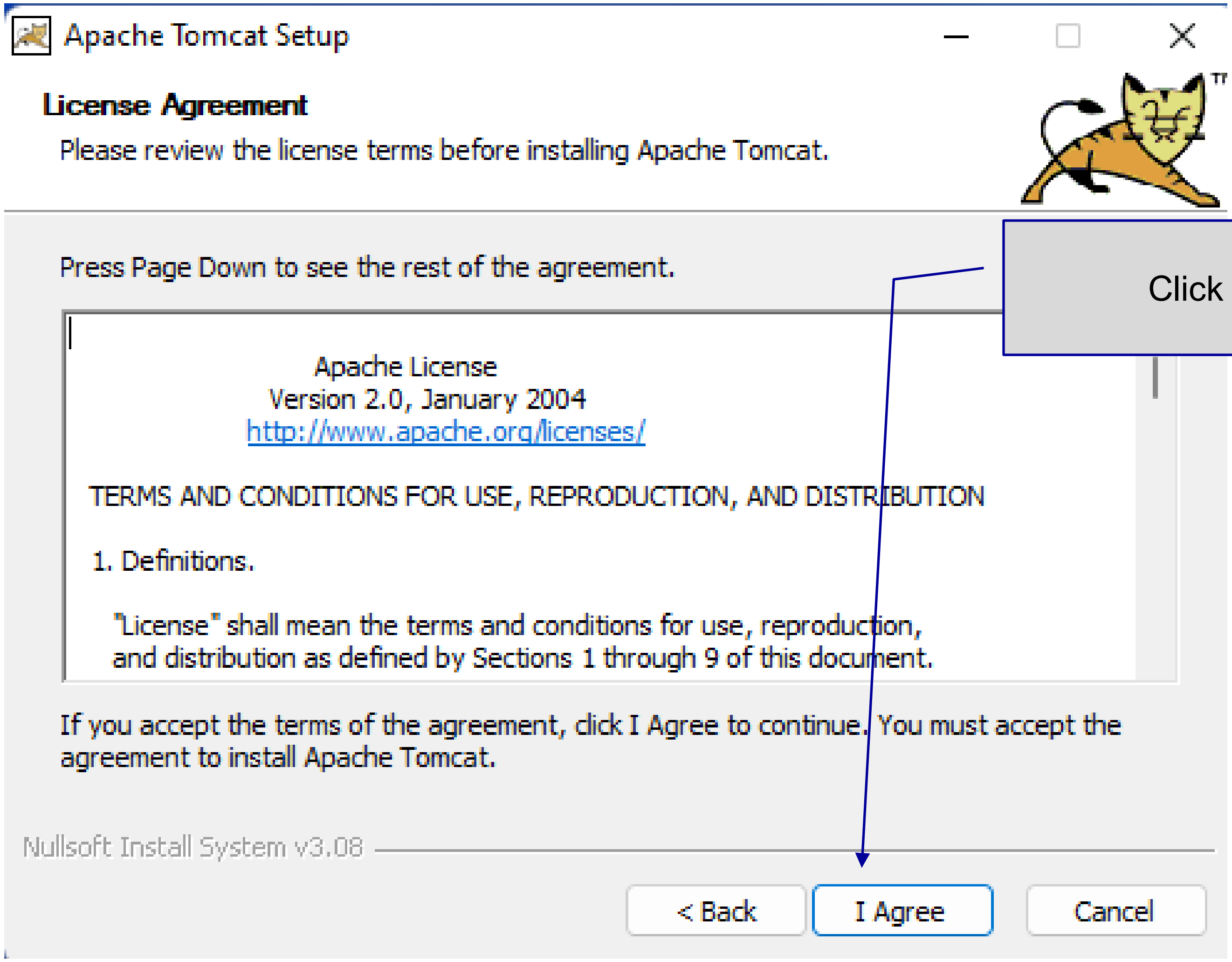
Welcome to Apache Tomcat Setup

Setup 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.

Click Next.



License Agreement

Please review the license terms before installing Apache Tomcat.

Press Page Down to see the rest of the agreement.

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

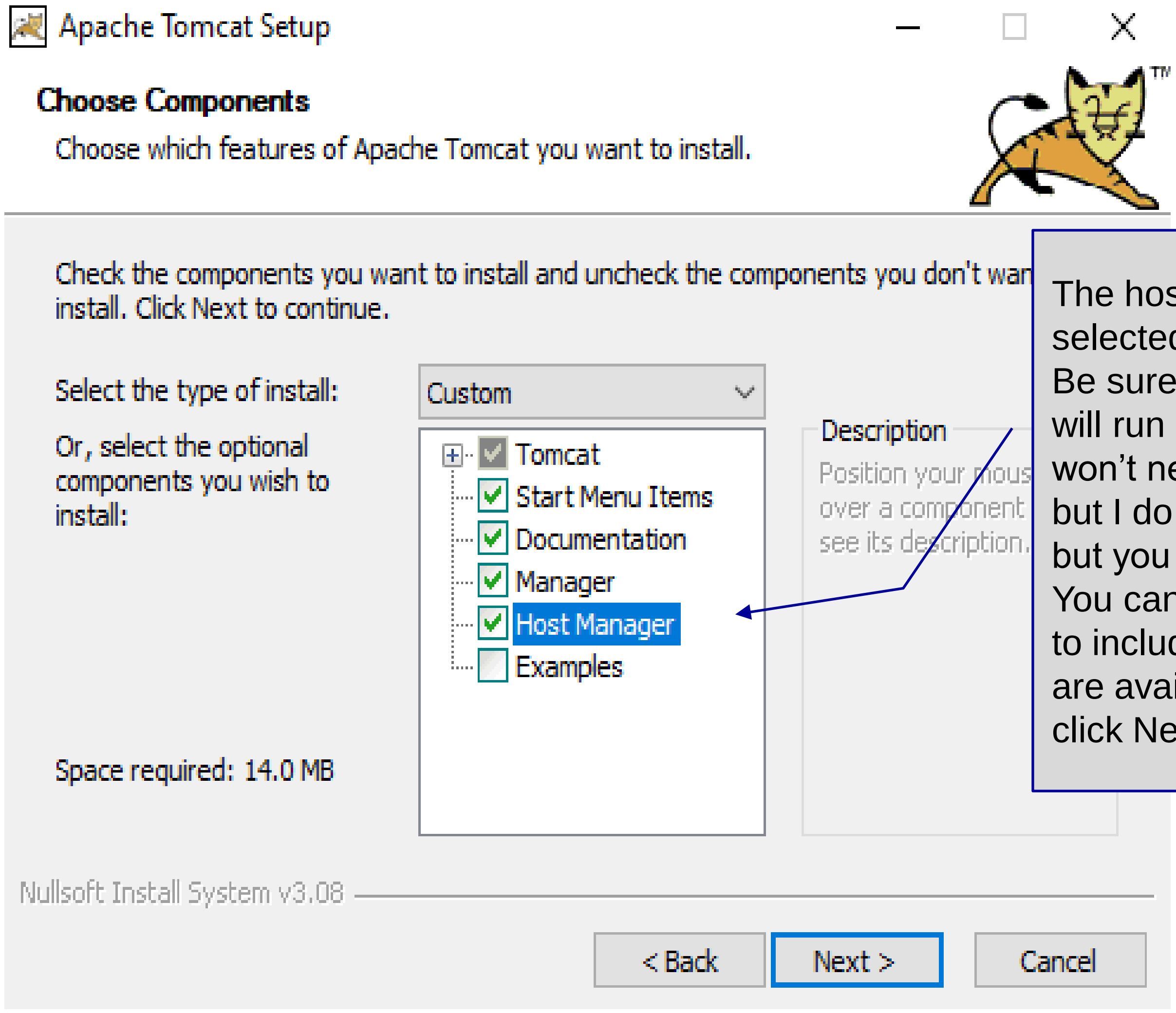
1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Apache Tomcat.

Nullsoft Install System v3.08

Click “I Agree”



Apache Tomcat Setup

Choose Components

Choose which features of Apache Tomcat you want to install.

Check the components you want to install and uncheck the components you don't wan install. Click Next to continue.

Select the type of install: Custom

Or, select the optional components you wish to install:

- [✓] Tomcat
- [✓] Start Menu Items
- [✓] Documentation
- [✓] Manager
- [✓] Host Manager
- [ ] Examples

Description

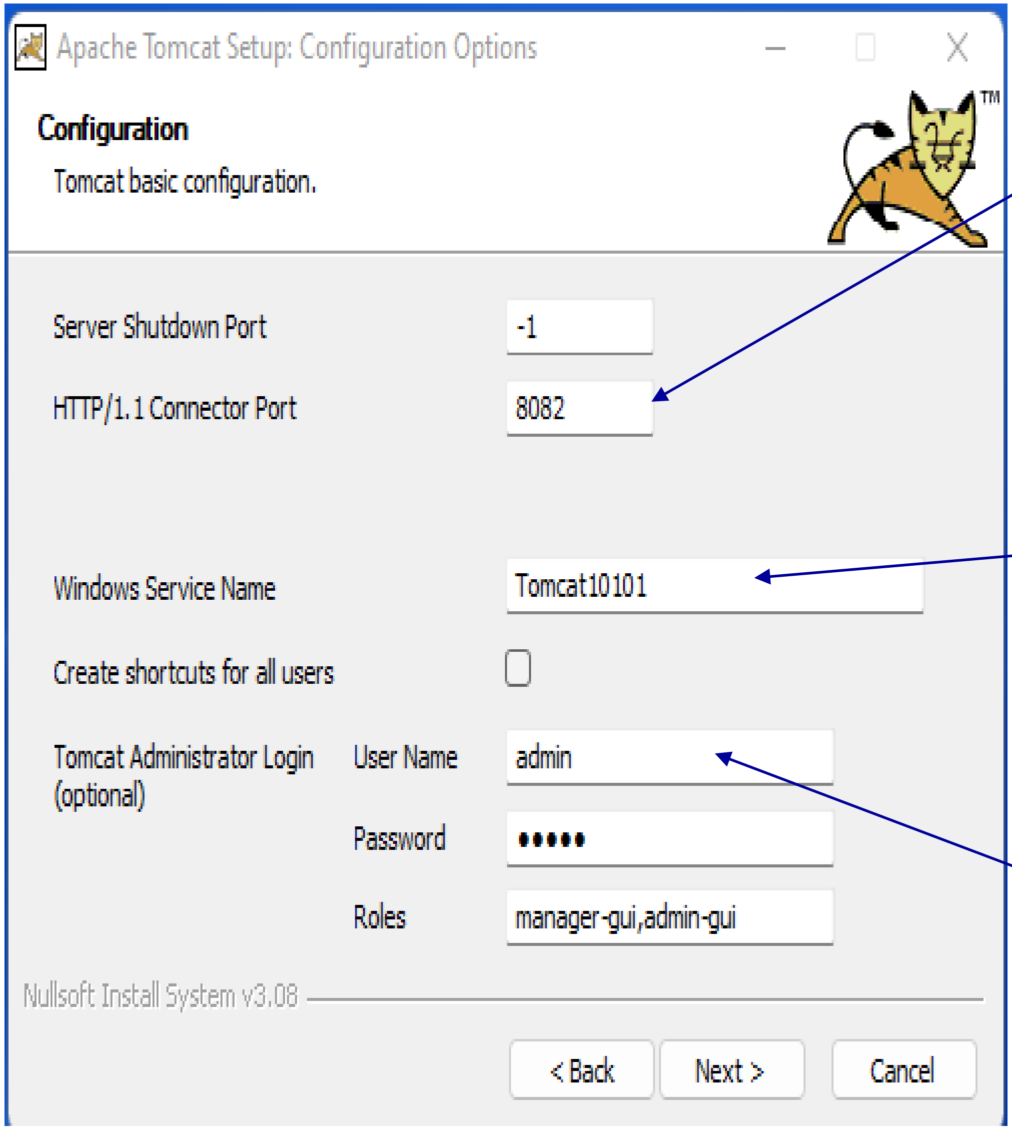
Position your mous over a component see its description.

Space required: 14.0 MB

Nullsoft Install System v3.08

< Back | 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 won’t need to for this class, but I do, so, I am checking it, but you don’t need to do so). You can decide whether or not to include the examples (they are available online. Then click Next.


Apache Tomcat Setup: Configuration Options

Configuration

Tomcat basic configuration.

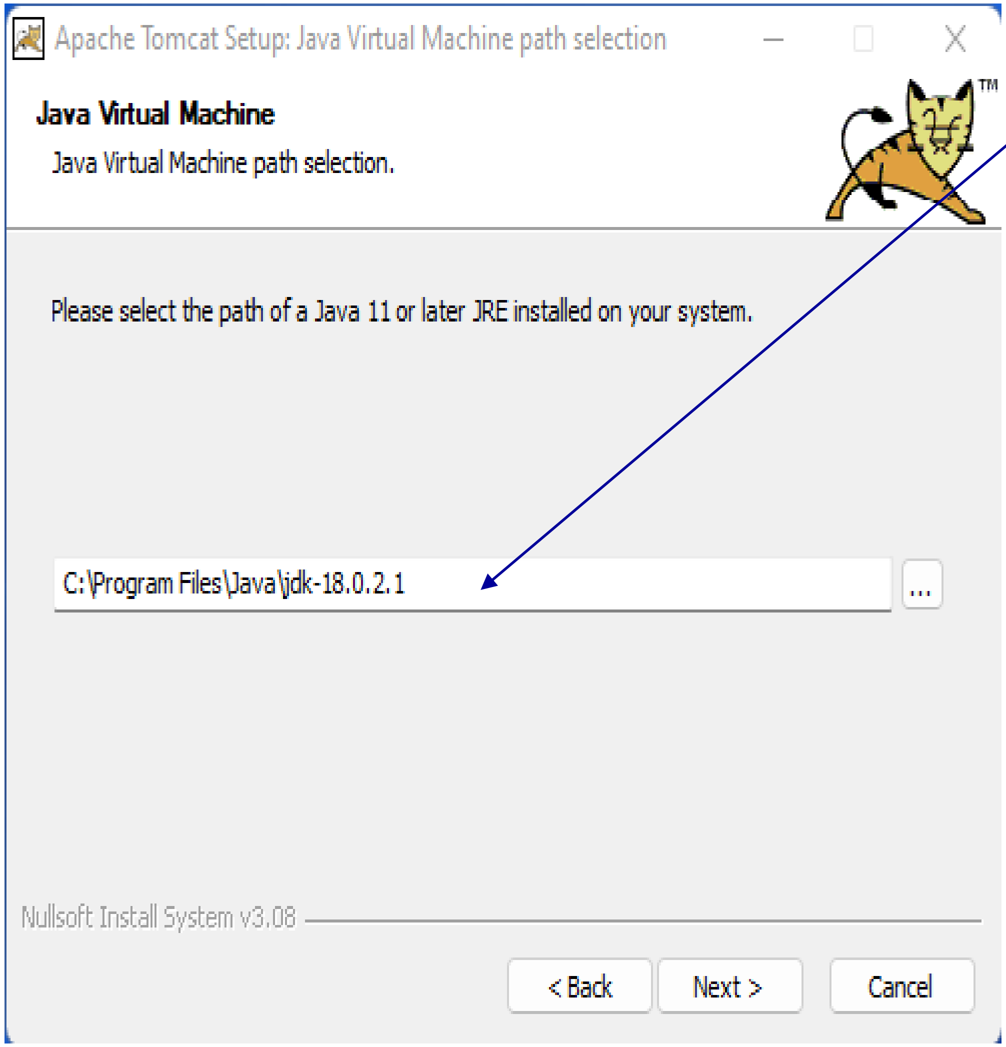
Server Shutdown Port		-1
HTTP/1.1 Connector Port		8082
Windows Service Name		Tomcat10101
Create shortcuts for all users		☐
Tomcat Administrator Login (optional)	User Name	admin
	Password	•••••
	Roles	manager-gui,admin-gui

Nullsoft Install System v3.08

Port 8080 is the default Tomcat connector port. Note that I am using port 8082 as I have multiple Tomcats running. When we setup Apache later, we'll put in on a different connector port so that both servers can be running simultaneously.

The default Service Name will be Tomcat10 – I've renamed mine because I have multiple Tomcat's running. You can leave yours as Tomcat10 if you'd like.

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.


Apache Tomcat Setup: Java Virtual Machine path selection

Java Virtual Machine

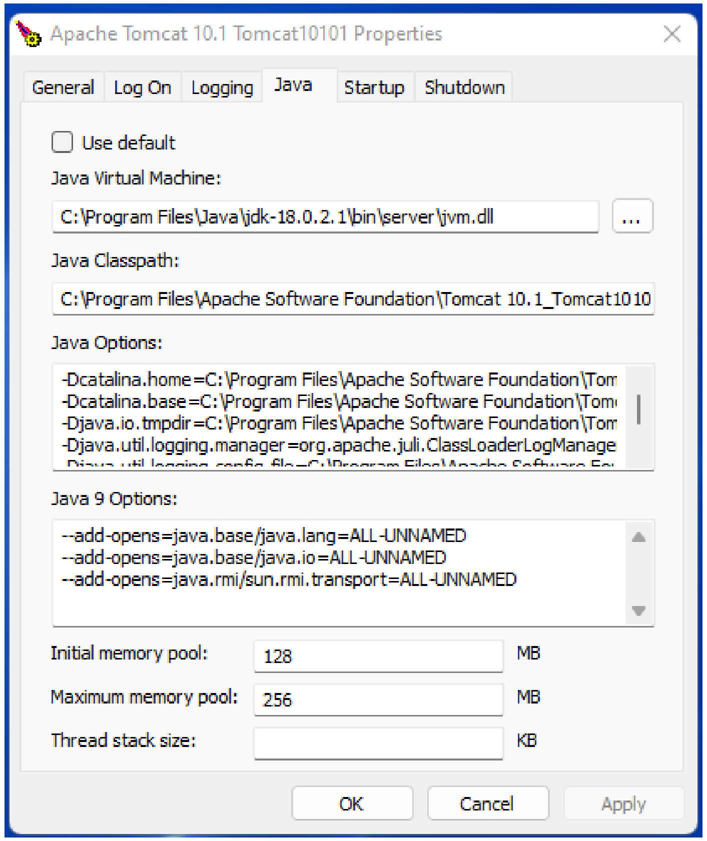
Java Virtual Machine path selection.

Please select the path of a Java 11 or later JRE installed on your system.

C:\Program Files\Java\jdk-18.0.2.1

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.

NOTE: Java SE 8.0 or later is required for Tomcat 10.X and above. (You can always reconfigure via the server configuration at a later time - see below.)


Apache Tomcat 10.1 Tomcat10101 Properties

☐ Use default

Java Virtual Machine:

C:\Program Files\Java\jdk-18.0.2.1\bin\server\jvm.dll

Java Classpath:

C:\Program Files\Apache Software Foundation\Tomcat 10.1_Tomcat1010

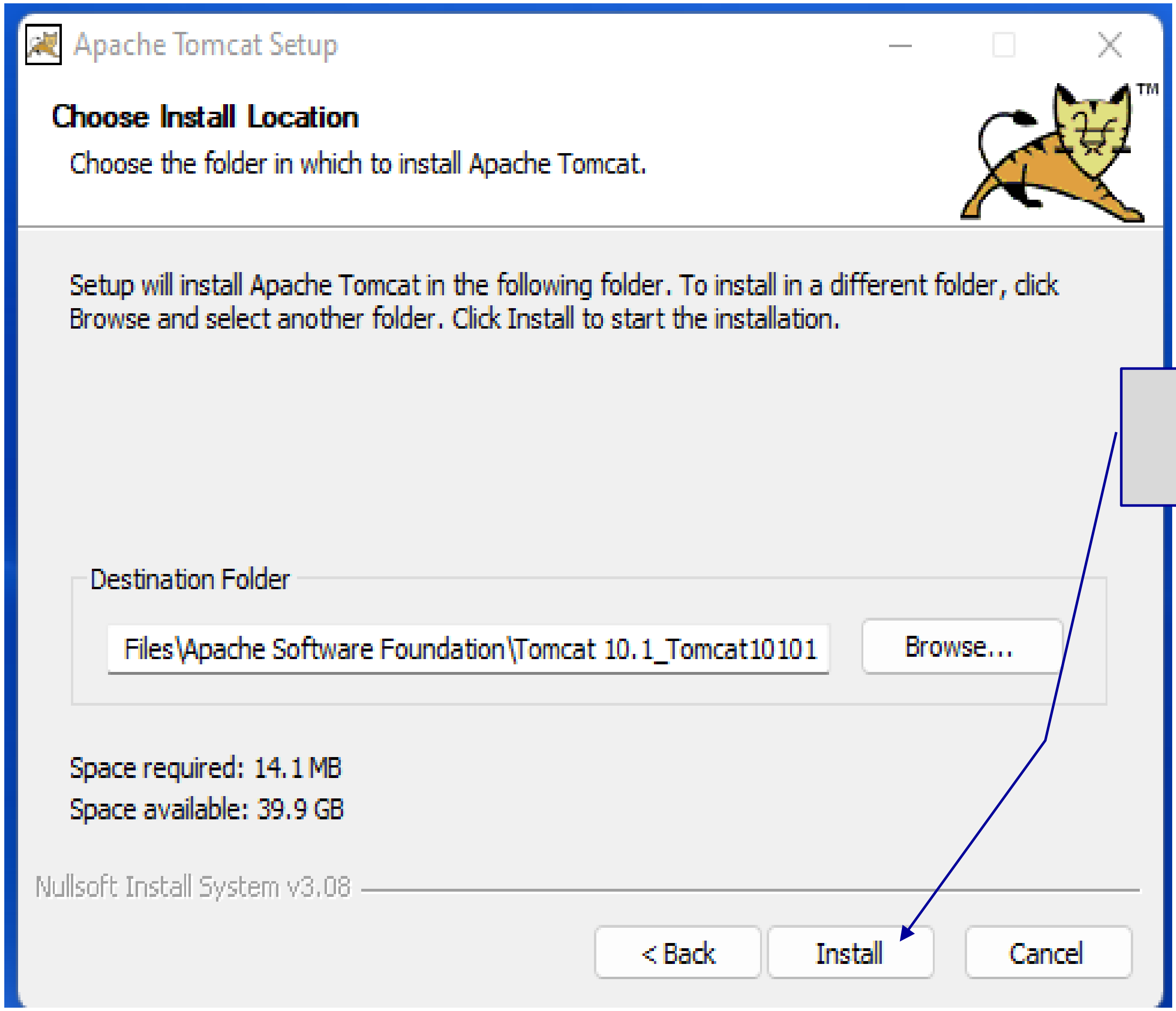
Java Options:

```
-Dcatalina.home=C:\Program Files\Apache Software Foundation\Tom
-Dcatalina.base=C:\Program Files\Apache Software Foundation\Tom
-Djava.io.tmpdir=C:\Program Files\Apache Software Foundation\Tom
-Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManage
```

Java 9 Options:

```
--add-opens=java.base/java.lang=ALL-UNNAMED
--add-opens=java.base/java.io=ALL-UNNAMED
--add-opens=java.rmi/sun.rmi.transport=ALL-UNNAMED
```

Initial memory pool:	128	MB
Maximum memory pool:	256	MB
Thread stack size:		KB



Apache Tomcat Setup

Choose Install Location

Choose the folder in which to install Apache Tomcat.

Setup will install Apache Tomcat in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

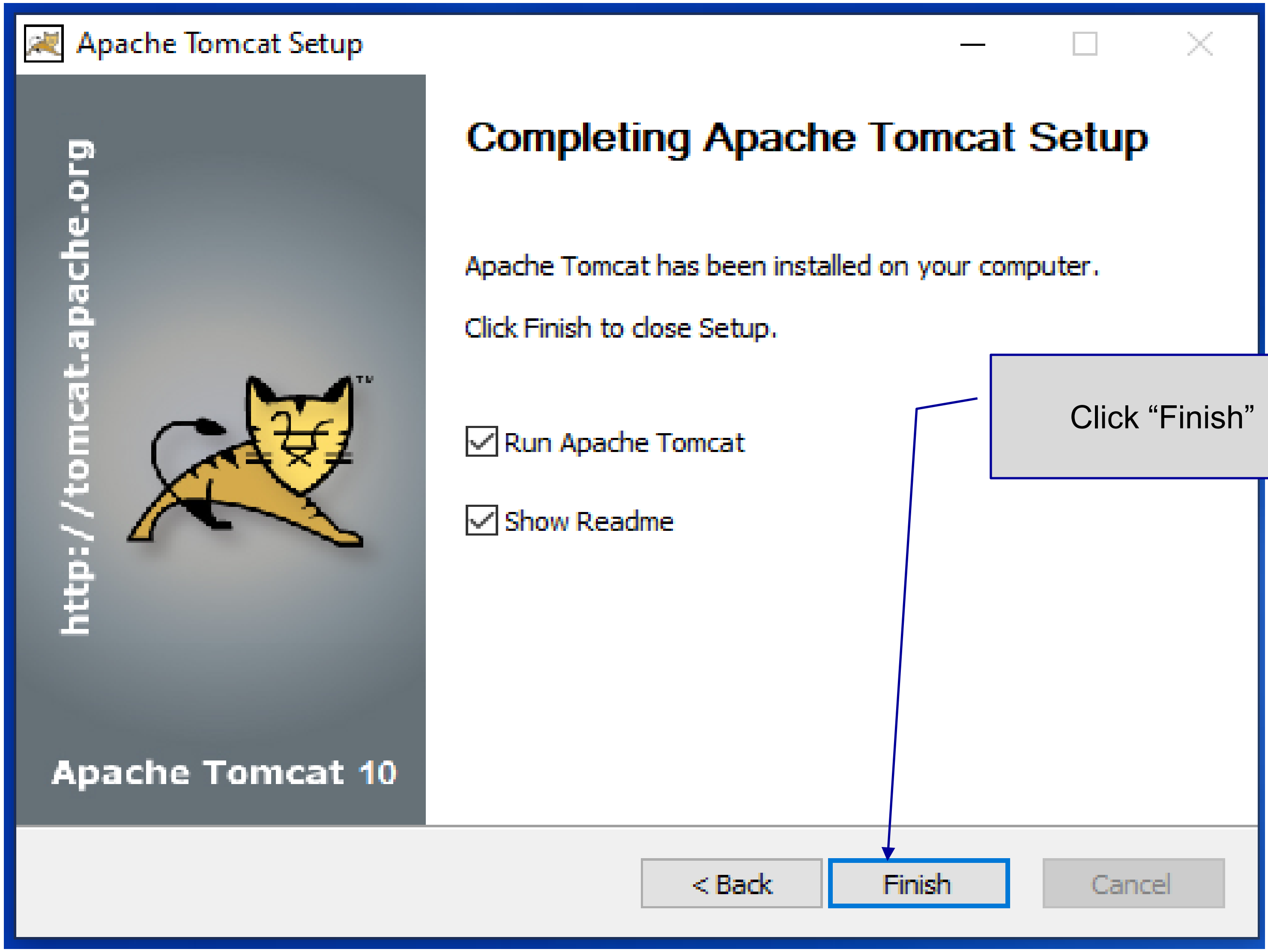
Destination Folder

Files\Apache Software Foundation\Tomcat 10.1_Tomcat10101

Space required: 14.1 MB
Space available: 39.9 GB

Nullsoft Install System v3.08

Click “Install”



Apache Tomcat Setup

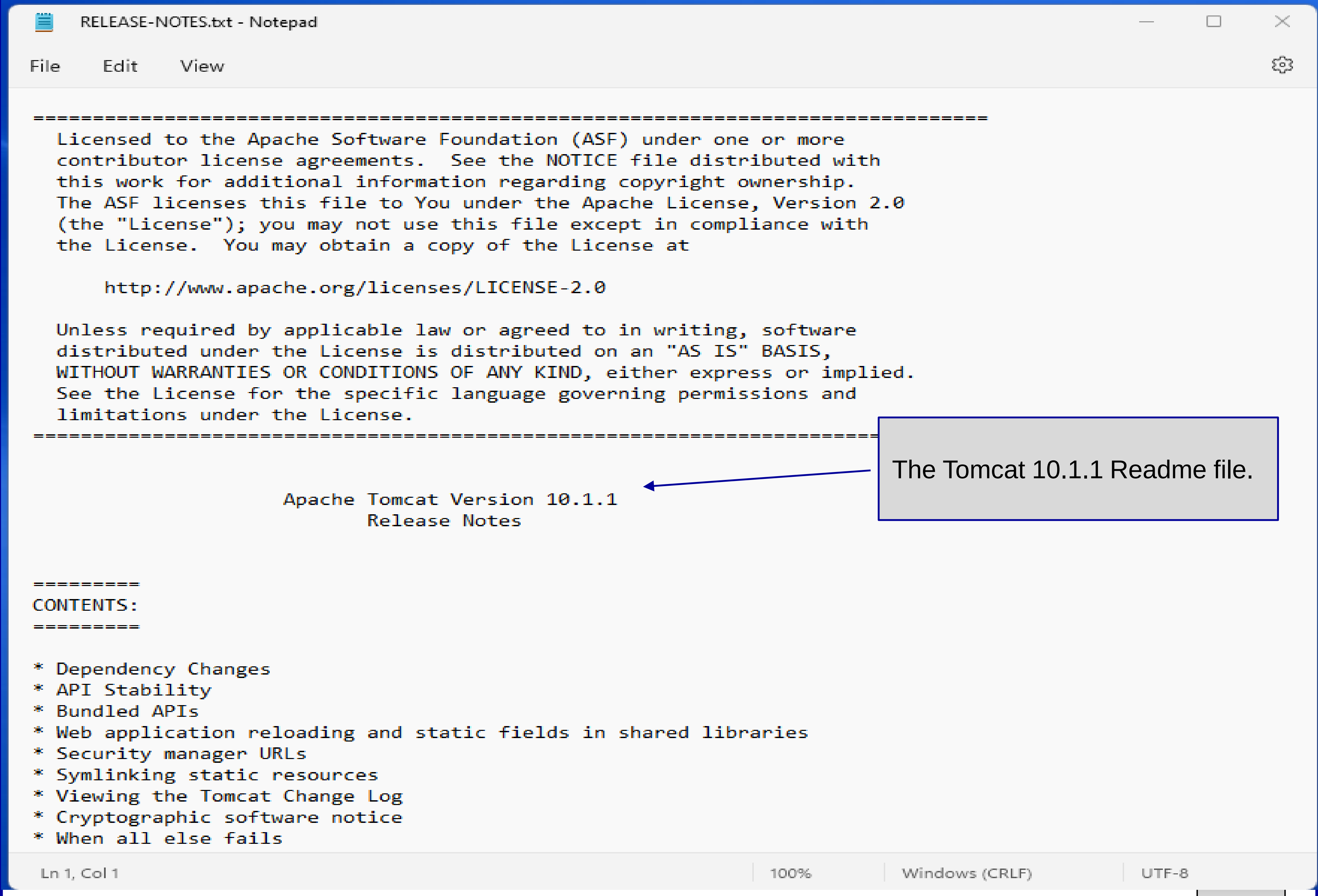
Completing Apache Tomcat Setup

Apache Tomcat has been installed on your computer.

Click Finish to close Setup.

- [✓] Run Apache Tomcat
- [✓] Show Readme

Click "Finish"



```
================================================================================
  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 10.1.1
                            Release Notes


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

* Dependency Changes
* API Stability
* Bundled APIs
* Web application reloading and static fields in shared libraries
* Security manager URLs
* Symlinking static resources
* Viewing the Tomcat Change Log
* Cryptographic software notice
* When all else fails
```

The Tomcat 10.1.1 Readme file.

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.

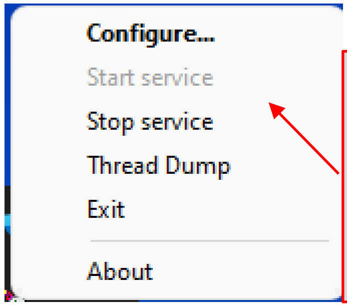
NOTE: During the install, Tomcat will ask you which TCP port Tomcat should run on (See page 28). To avoid any conflict with standard Web servers which default to TCP port 80, Tomcat is set to default to TCP port 8080. If you have any other service running on this port change the port number at this time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

In all subsequent examples, I'm running Tomcat on TCP port 8082.

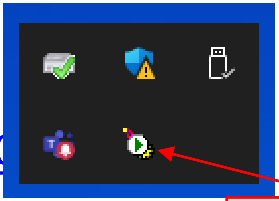
Starting Up Tomcat

- 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.) .

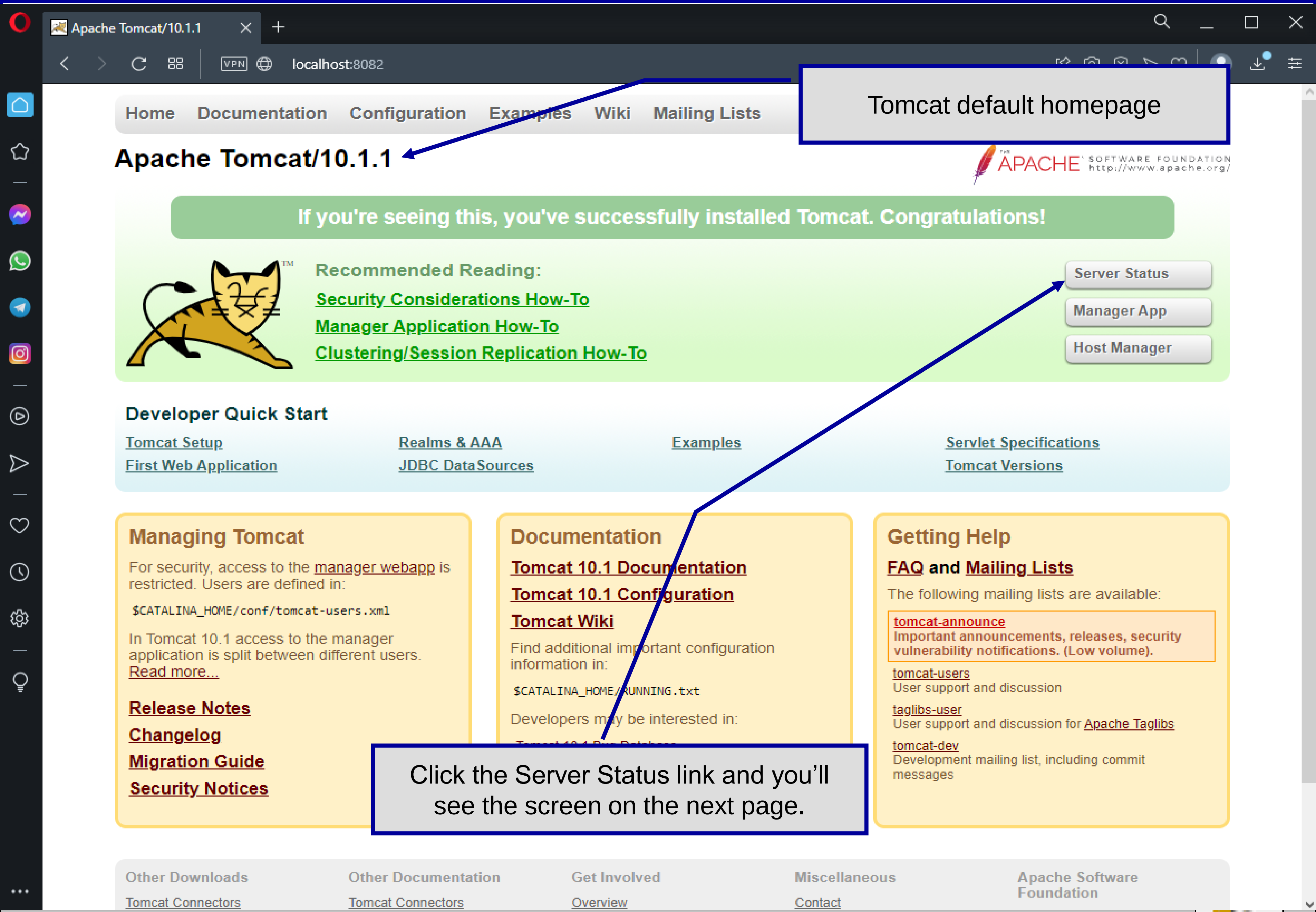
1. Start your Web browser.
2. Enter URL: http://localho
3. You should see the screen o e if everything is set up ok.



Right-click the Tomcat Monitor icon to see this list of options.



In your hidden icons list you will now have a Tomcat Monitor.



Apache Tomcat/10.1.1

localhost:8082

Home Documentation Configuration Examples Wiki Mailing Lists

Tomcat default homepage

Apache Tomcat/10.1.1

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 10.1 access to the manager application is split between different users. Read more...

Release Notes

Changelog

Migration Guide

Security Notices

Documentation

Tomcat 10.1 Documentation

Tomcat 10.1 Configuration

Tomcat Wiki

Find additional important configuration information in:

```
$CATALINA_HOME/RUNNING.txt
```

Developers may be interested in:

Getting Help

FAQ and Mailing Lists

The following mailing lists are available:

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

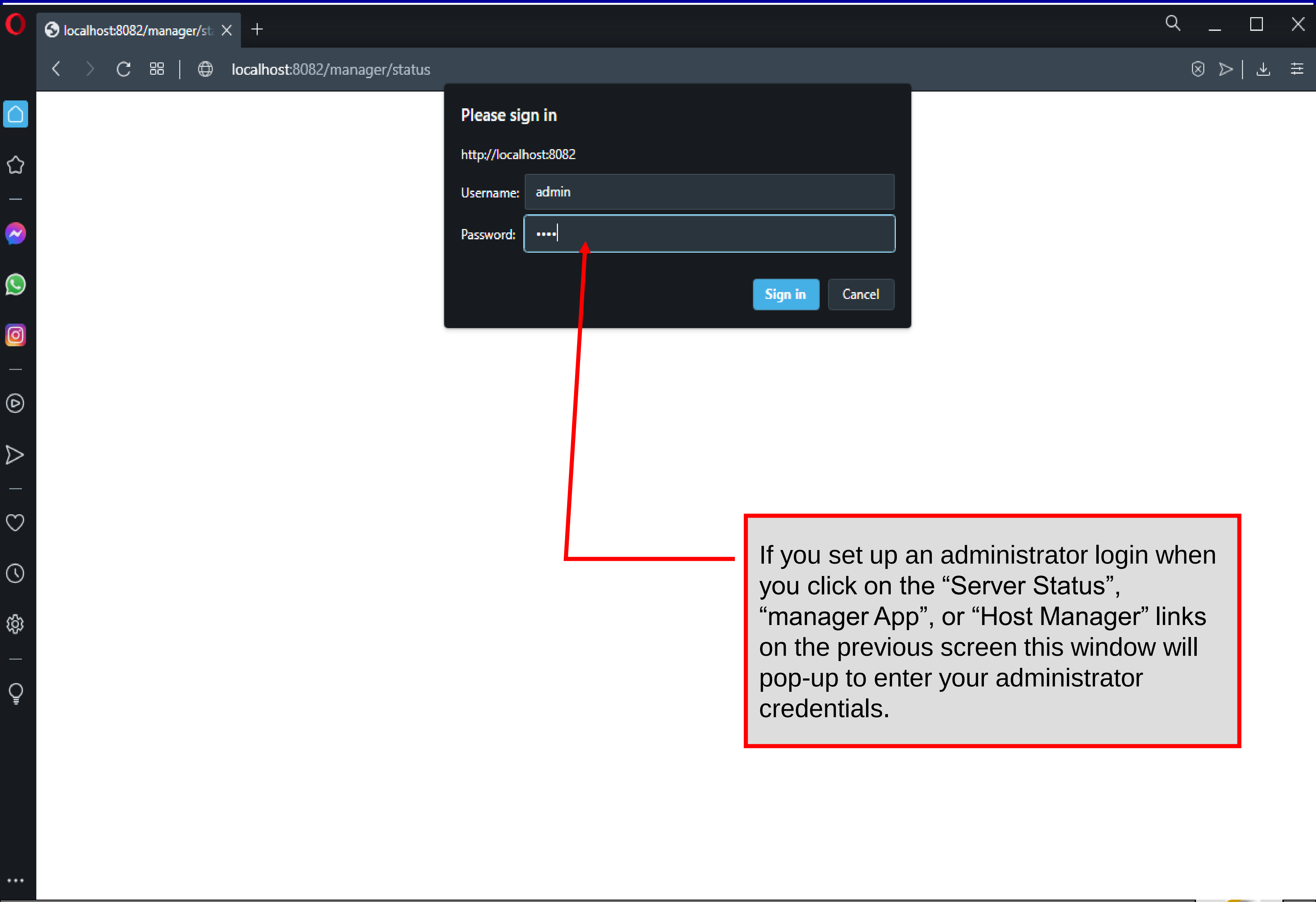
tomcat-users
User support and discussion

taglibs-user
User support and discussion for Apache Taglibs

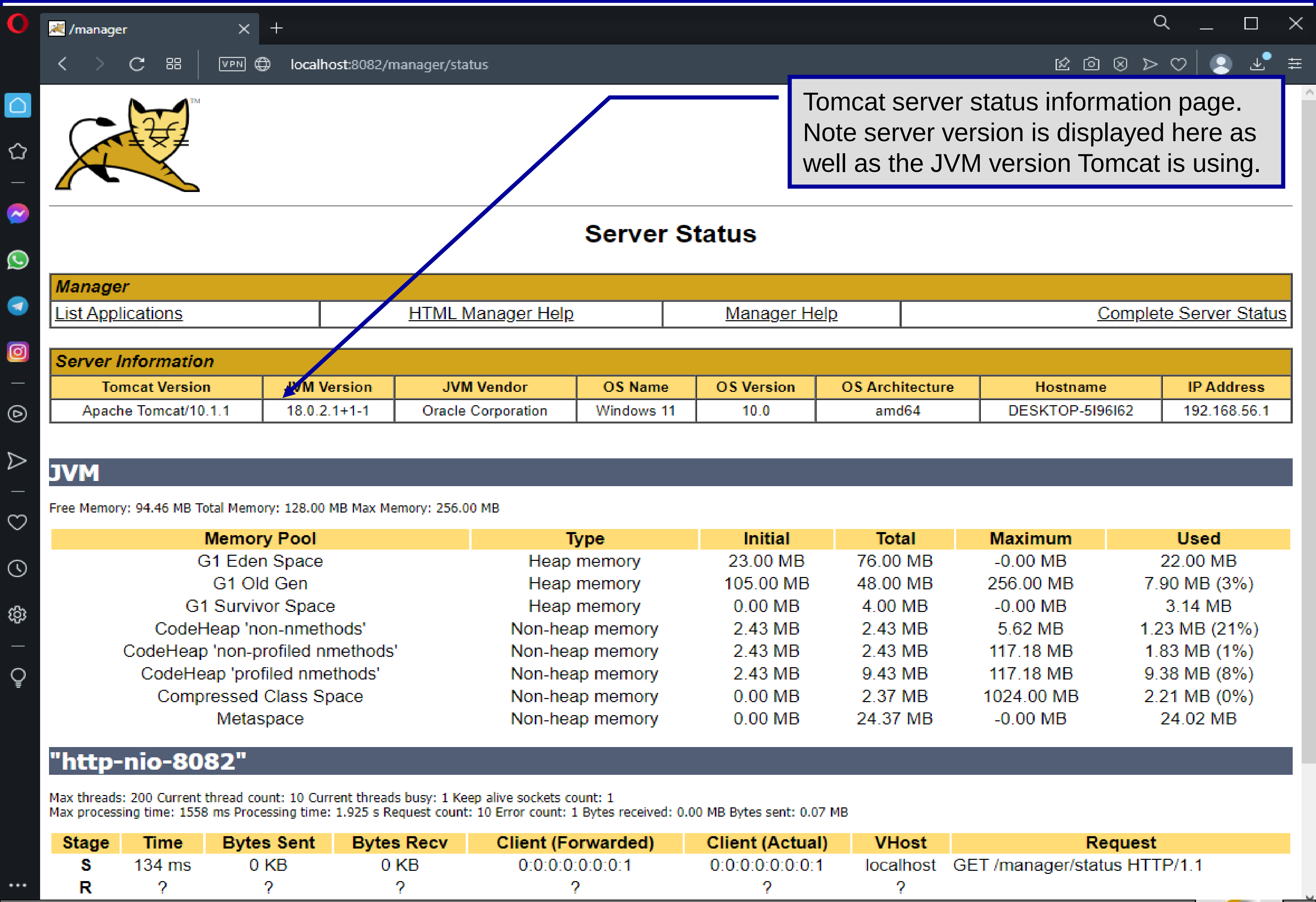
tomcat-dev
Development mailing list, including commit messages

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

Other Downloads	Other Documentation	Get Involved	Miscellaneous	Apache Software Foundation
Tomcat Connectors	Tomcat Connectors	Overview	Contact	



If you set up an administrator login when you click on the “Server Status”, “manager App”, or “Host Manager” links on the previous screen this window will pop-up to enter your administrator credentials.



Tomcat server status information page. Note server version is displayed here as well as the JVM version Tomcat is using.

localhost:8082/manager/status

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	Hostname	IP Address
Apache Tomcat/10.1.1	18.0.2.1+1-1	Oracle Corporation	Windows 11	10.0	amd64	DESKTOP-5I96I62	192.168.56.1

JVM

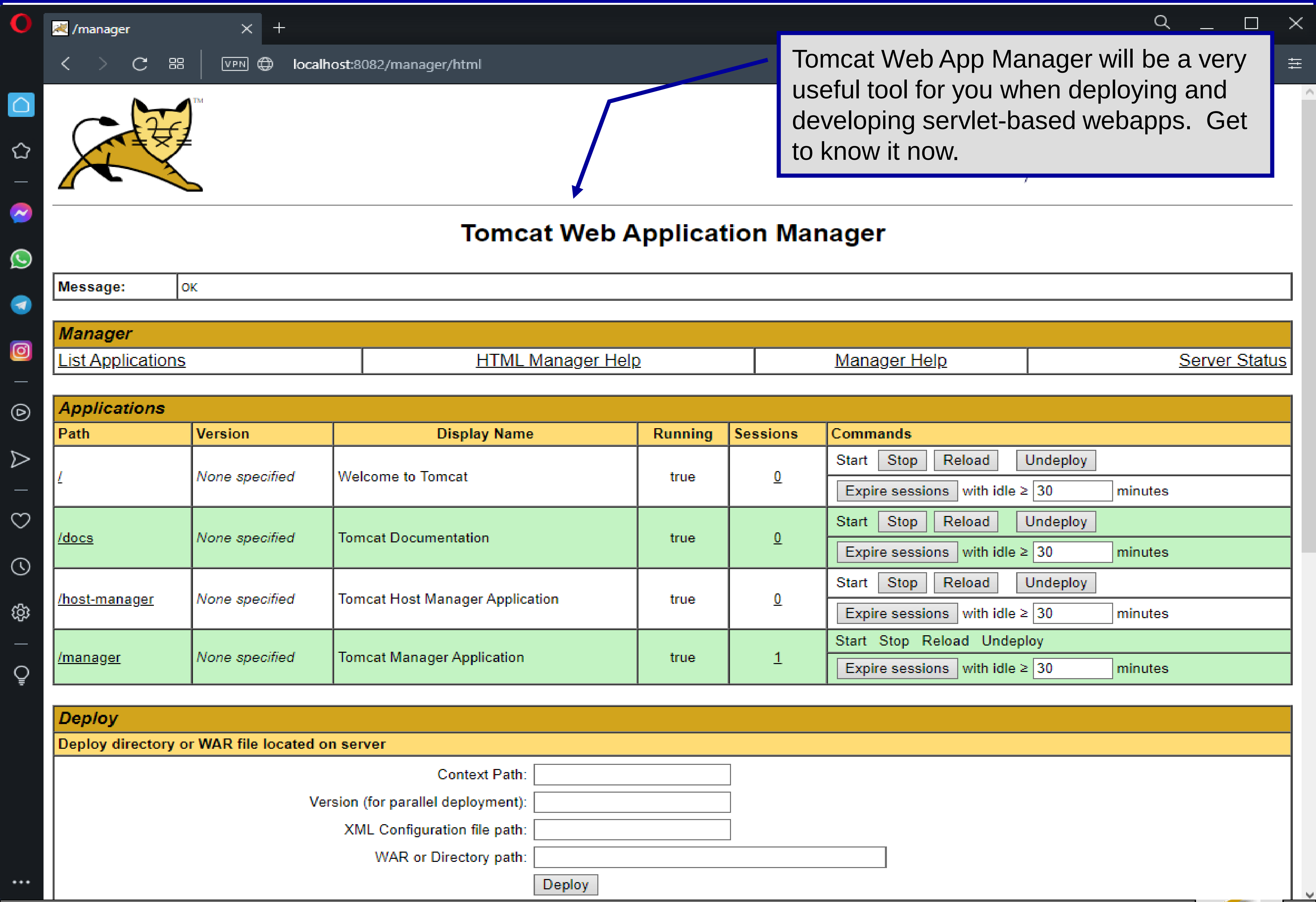
Free Memory: 94.46 MB Total Memory: 128.00 MB Max Memory: 256.00 MB

Memory Pool	Type	Initial	Total	Maximum	Used
G1 Eden Space	Heap memory	23.00 MB	76.00 MB	-0.00 MB	22.00 MB
G1 Old Gen	Heap memory	105.00 MB	48.00 MB	256.00 MB	7.90 MB (3%)
G1 Survivor Space	Heap memory	0.00 MB	4.00 MB	-0.00 MB	3.14 MB
CodeHeap 'non-nmethods'	Non-heap memory	2.43 MB	2.43 MB	5.62 MB	1.23 MB (21%)
CodeHeap 'non-profiled nmethods'	Non-heap memory	2.43 MB	2.43 MB	117.18 MB	1.83 MB (1%)
CodeHeap 'profiled nmethods'	Non-heap memory	2.43 MB	9.43 MB	117.18 MB	9.38 MB (8%)
Compressed Class Space	Non-heap memory	0.00 MB	2.37 MB	1024.00 MB	2.21 MB (0%)
Metaspace	Non-heap memory	0.00 MB	24.37 MB	-0.00 MB	24.02 MB

"http-nio-8082"

Max threads: 200 Current thread count: 10 Current threads busy: 1 Keep alive sockets count: 1
Max processing time: 1558 ms Processing time: 1.925 s Request count: 10 Error count: 1 Bytes received: 0.00 MB Bytes sent: 0.07 MB

Stage	Time	Bytes Sent	Bytes Recv	Client (Forwarded)	Client (Actual)	VHost	Request
S	134 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 HTTP/1.1
R	?	?	?	?	?	?	



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

localhost:8082/manager/html

Tomcat Web Application Manager

Message:	OK

Manager

List Applications	HTML Manager Help	Manager Help	Server Status

Applications

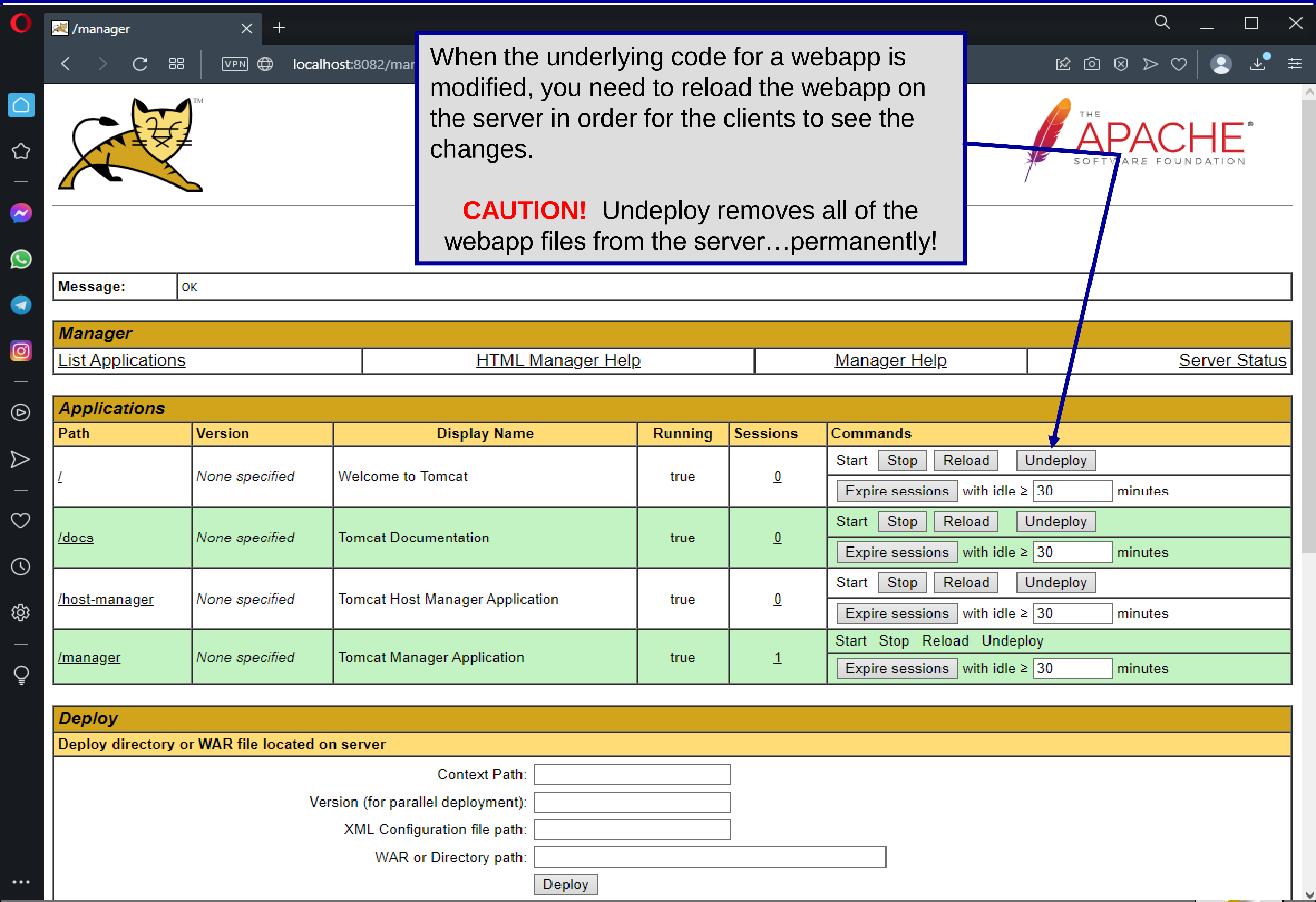
Path	Version	Display Name	Running	Sessions	Commands
/	*None specified*	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	*None specified*	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	*None specified*	Tomcat Host Manager Application	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:	
Version (for parallel deployment):	
XML Configuration file path:	
WAR or Directory path:	

Deploy



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

CAUTION! Undeploy removes all of the webapp files from the server…permanently!

Message:	OK

Manager

List Applications	HTML Manager Help	Manager Help	Server Status

Applications

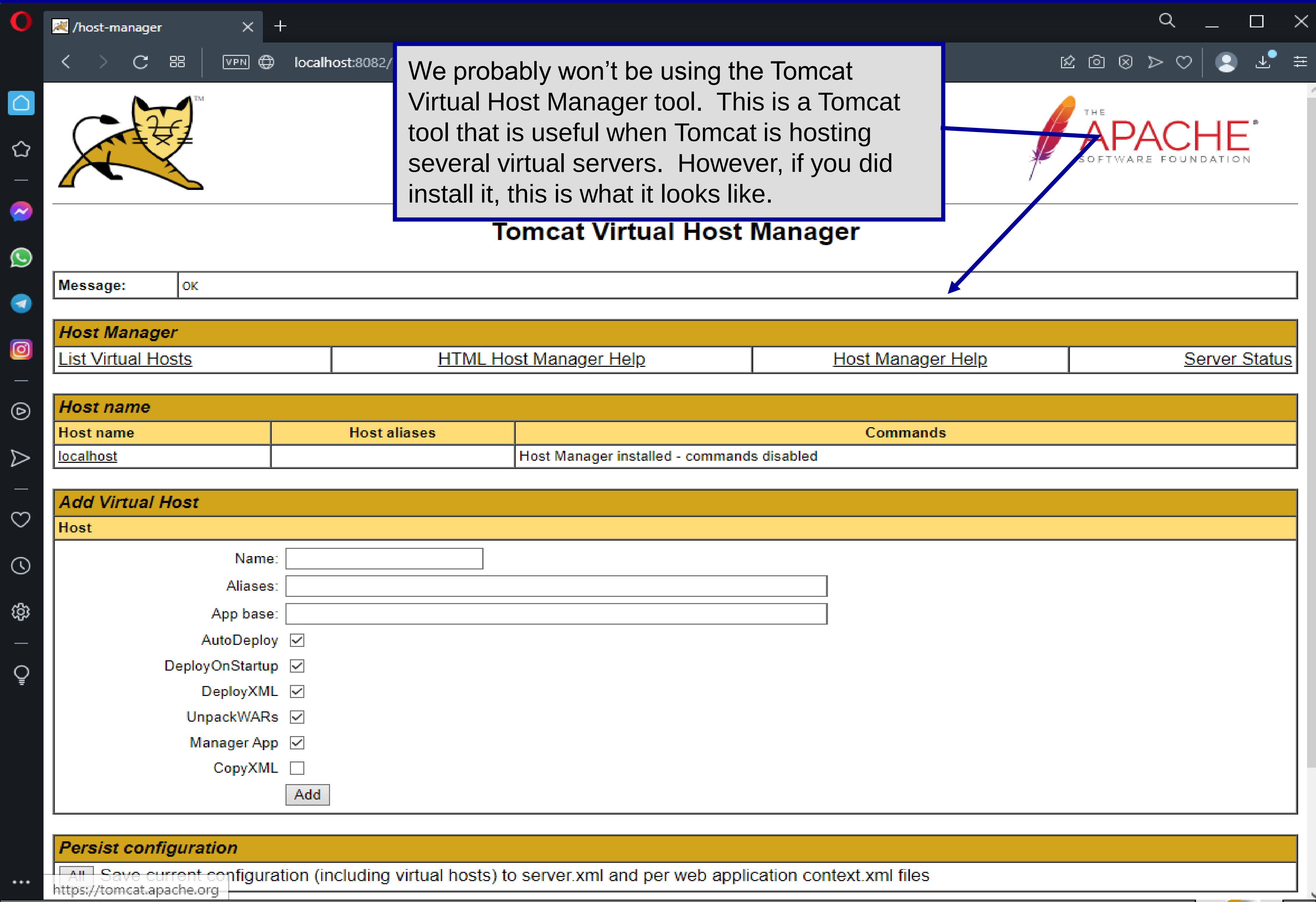
Path	Version	Display Name	Running	Sessions	Commands
/	*None specified*	Welcome to Tomcat	true	0	Start Stop Reload Undeploy
					Expire sessions with idle ≥ 30 minutes
/docs	*None specified*	Tomcat Documentation	true	0	Start Stop Reload Undeploy
					Expire sessions with idle ≥ 30 minutes
/host-manager	*None specified*	Tomcat Host Manager Application	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:	
Version (for parallel deployment):	
XML Configuration file path:	
WAR or Directory path:	

Deploy



We probably 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. However, if you did install it, this is what it looks like.

Tomcat Virtual Host Manager

Message:	OK

Host Manager			
List Virtual Hosts	HTML Host Manager Help	Host Manager Help	Server Status

Host name

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

Add Virtual Host

Host

Name:	
Aliases:	
App base:	
AutoDeploy	✓
DeployOnStartup	✓
DeployXML	✓
UnpackWARs	✓
Manager App	✓
CopyXML	☐

Add

Persist configuration

All Save current configuration (including virtual hosts) to server.xml and per web application context.xml files

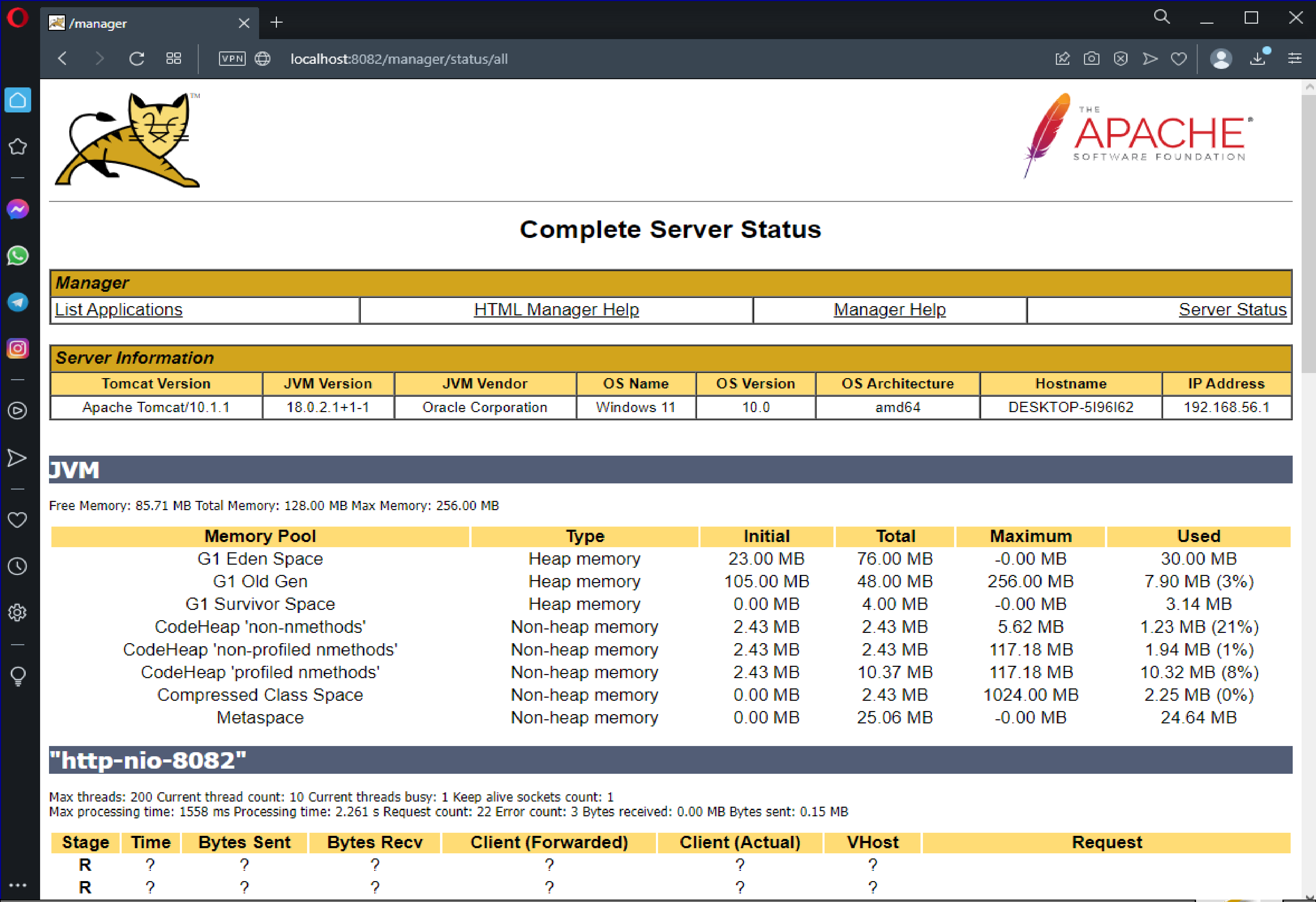
https://tomcat.apache.org

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, 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.

- Checking the status of the server can also be accomplished from the Tomcat homepage. See page 42-43 for a sample.



/manager

localhost:8082/manager/status/all

Complete Server Status

Manager			
List Applications	HTML Manager Help	Manager Help	Server Status

Server Information

Tomcat Version	JVM Version	JVM Vendor	OS Name	OS Version	OS Architecture	Hostname	IP Address
Apache Tomcat/10.1.1	18.0.2.1+1-1	Oracle Corporation	Windows 11	10.0	amd64	DESKTOP-5I96I62	192.168.56.1

JVM

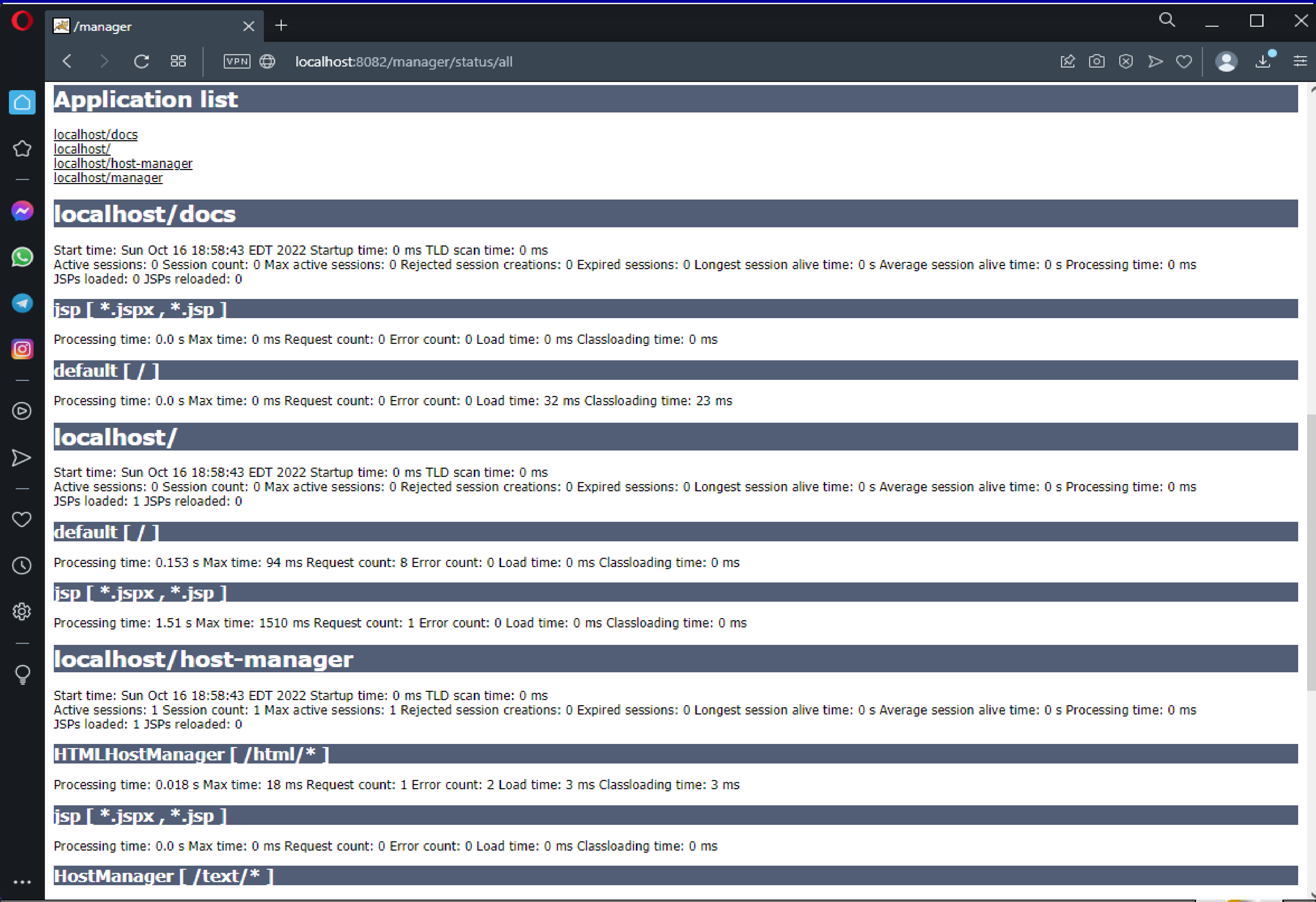
Free Memory: 85.71 MB Total Memory: 128.00 MB Max Memory: 256.00 MB

Memory Pool	Type	Initial	Total	Maximum	Used
G1 Eden Space	Heap memory	23.00 MB	76.00 MB	-0.00 MB	30.00 MB
G1 Old Gen	Heap memory	105.00 MB	48.00 MB	256.00 MB	7.90 MB (3%)
G1 Survivor Space	Heap memory	0.00 MB	4.00 MB	-0.00 MB	3.14 MB
CodeHeap 'non-nmethods'	Non-heap memory	2.43 MB	2.43 MB	5.62 MB	1.23 MB (21%)
CodeHeap 'non-profiled nmethods'	Non-heap memory	2.43 MB	2.43 MB	117.18 MB	1.94 MB (1%)
CodeHeap 'profiled nmethods'	Non-heap memory	2.43 MB	10.37 MB	117.18 MB	10.32 MB (8%)
Compressed Class Space	Non-heap memory	0.00 MB	2.43 MB	1024.00 MB	2.25 MB (0%)
Metaspace	Non-heap memory	0.00 MB	25.06 MB	-0.00 MB	24.64 MB

"http-nio-8082"

Max threads: 200 Current thread count: 10 Current threads busy: 1 Keep alive sockets count: 1
Max processing time: 1558 ms Processing time: 2.261 s Request count: 22 Error count: 3 Bytes received: 0.00 MB Bytes sent: 0.15 MB

Stage	Time	Bytes Sent	Bytes Recv	Client (Forwarded)	Client (Actual)	VHost	Request
R	?	?	?	?	?	?	
R	?	?	?	?	?	?	



Application list

localhost/docs
localhost/
localhost/host-manager
localhost/manager

localhost/docs

Start time: Sun Oct 16 18:58:43 EDT 2022 Startup time: 0 ms TLD scan time: 0 ms
Active sessions: 0 Session count: 0 Max active sessions: 0 Rejected session creations: 0 Expired sessions: 0 Longest session alive time: 0 s Average session alive time: 0 s Processing time: 0 ms
JSPs loaded: 0 JSPs reloaded: 0

jsp [*.jspx , *.jsp]

Processing time: 0.0 s Max time: 0 ms Request count: 0 Error count: 0 Load time: 0 ms Classloading time: 0 ms

default [/]

Processing time: 0.0 s Max time: 0 ms Request count: 0 Error count: 0 Load time: 32 ms Classloading time: 23 ms

localhost/

Start time: Sun Oct 16 18:58:43 EDT 2022 Startup time: 0 ms TLD scan time: 0 ms
Active sessions: 0 Session count: 0 Max active sessions: 0 Rejected session creations: 0 Expired sessions: 0 Longest session alive time: 0 s Average session alive time: 0 s Processing time: 0 ms
JSPs loaded: 1 JSPs reloaded: 0

default [/]

Processing time: 0.153 s Max time: 94 ms Request count: 8 Error count: 0 Load time: 0 ms Classloading time: 0 ms

jsp [*.jspx , *.jsp]

Processing time: 1.51 s Max time: 1510 ms Request count: 1 Error count: 0 Load time: 0 ms Classloading time: 0 ms

localhost/host-manager

Start time: Sun Oct 16 18:58:43 EDT 2022 Startup time: 0 ms TLD scan time: 0 ms
Active sessions: 1 Session count: 1 Max active sessions: 1 Rejected session creations: 0 Expired sessions: 0 Longest session alive time: 0 s Average session alive time: 0 s Processing time: 0 ms
JSPs loaded: 1 JSPs reloaded: 0

HTMLHostManager [/html/*]

Processing time: 0.018 s Max time: 18 ms Request count: 1 Error count: 2 Load time: 3 ms Classloading time: 3 ms

jsp [*.jspx , *.jsp]

Processing time: 0.0 s Max time: 0 ms Request count: 0 Error count: 0 Load time: 0 ms Classloading time: 0 ms

HostManager [/text/*]

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`.

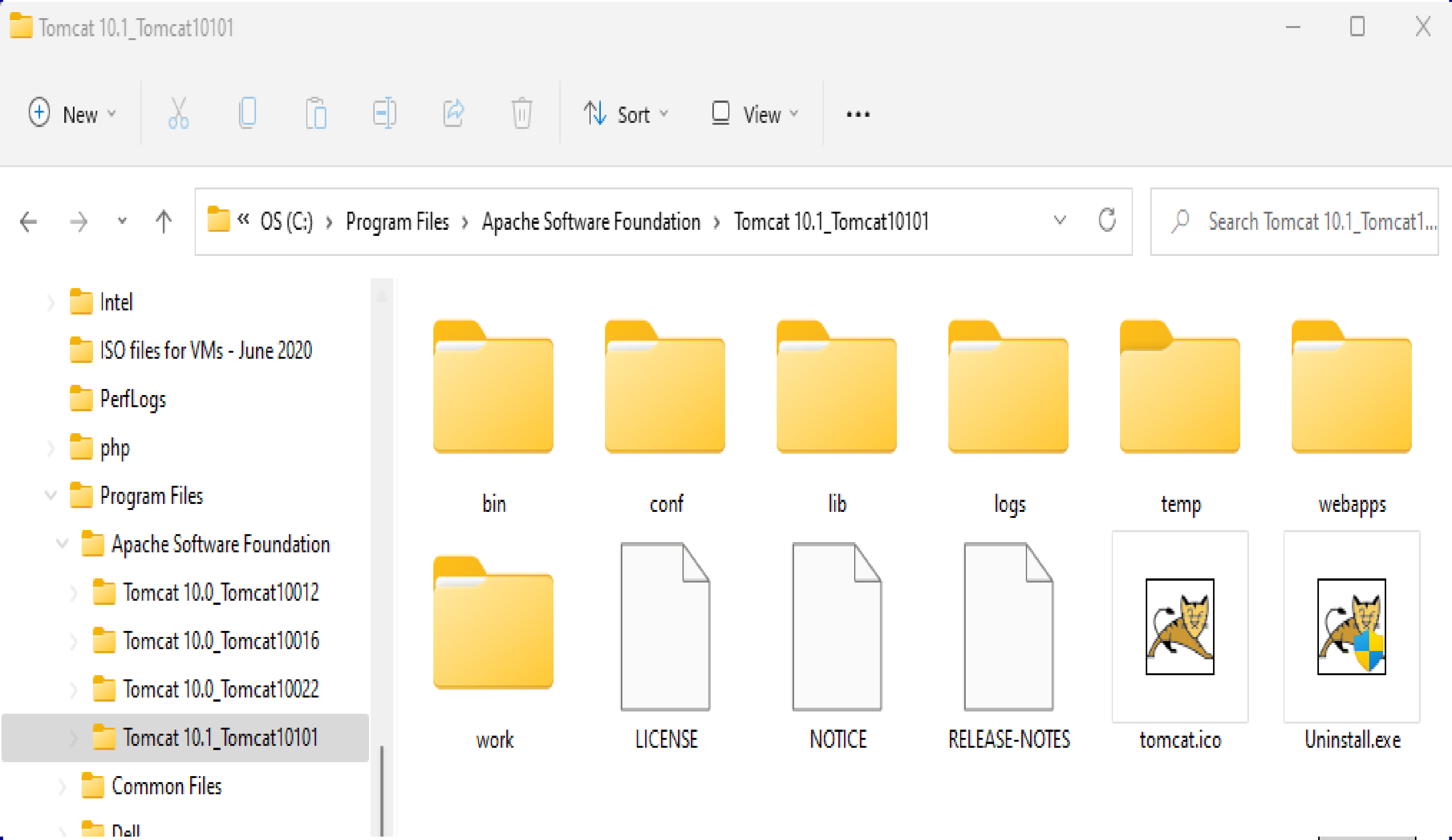
bin

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

conf

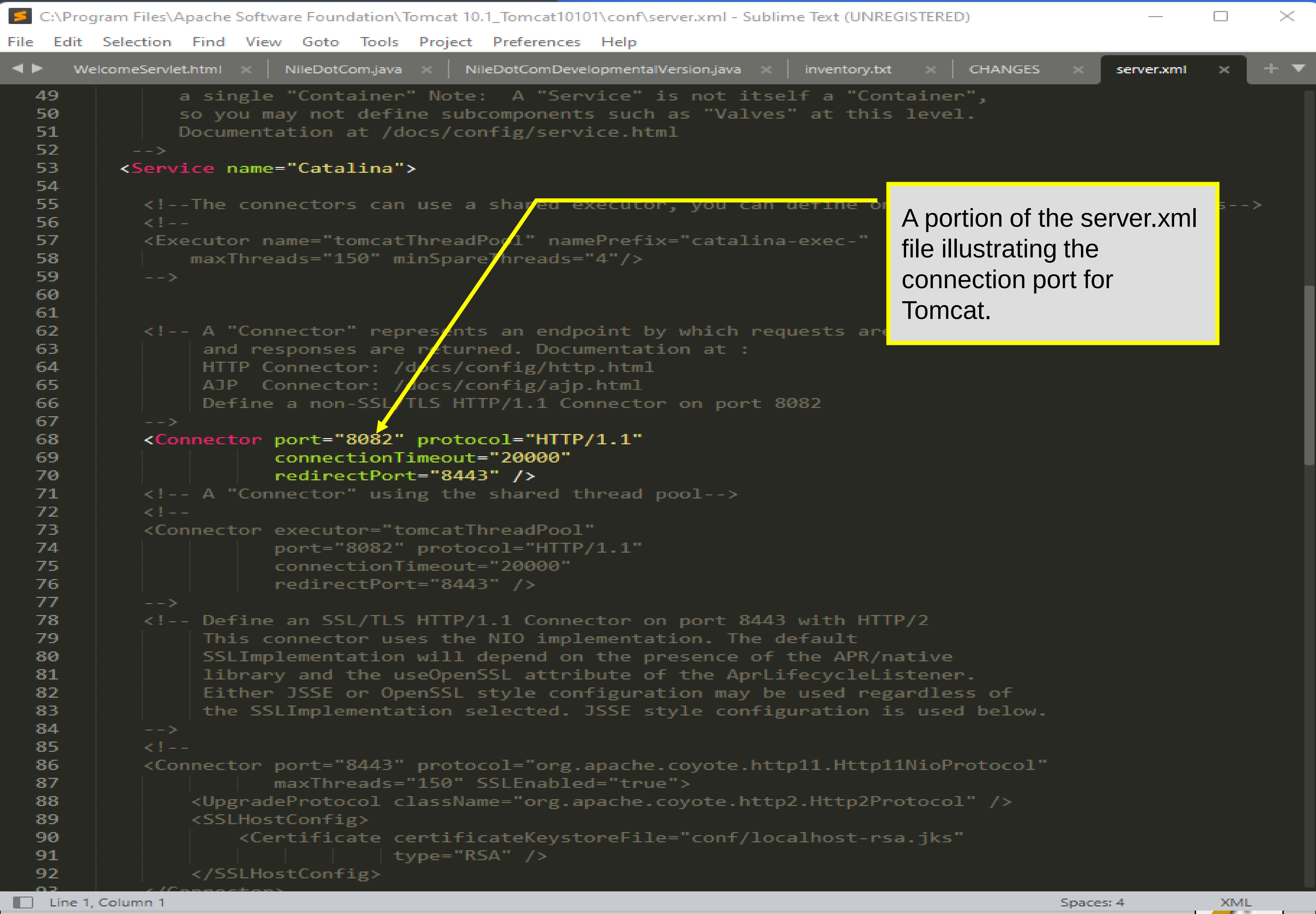
- Directory `conf` contains files used to configure Tomcat at the global level, although it is possible for each web application to override many of the values provided in this directory.

Tomcat Directory Structure



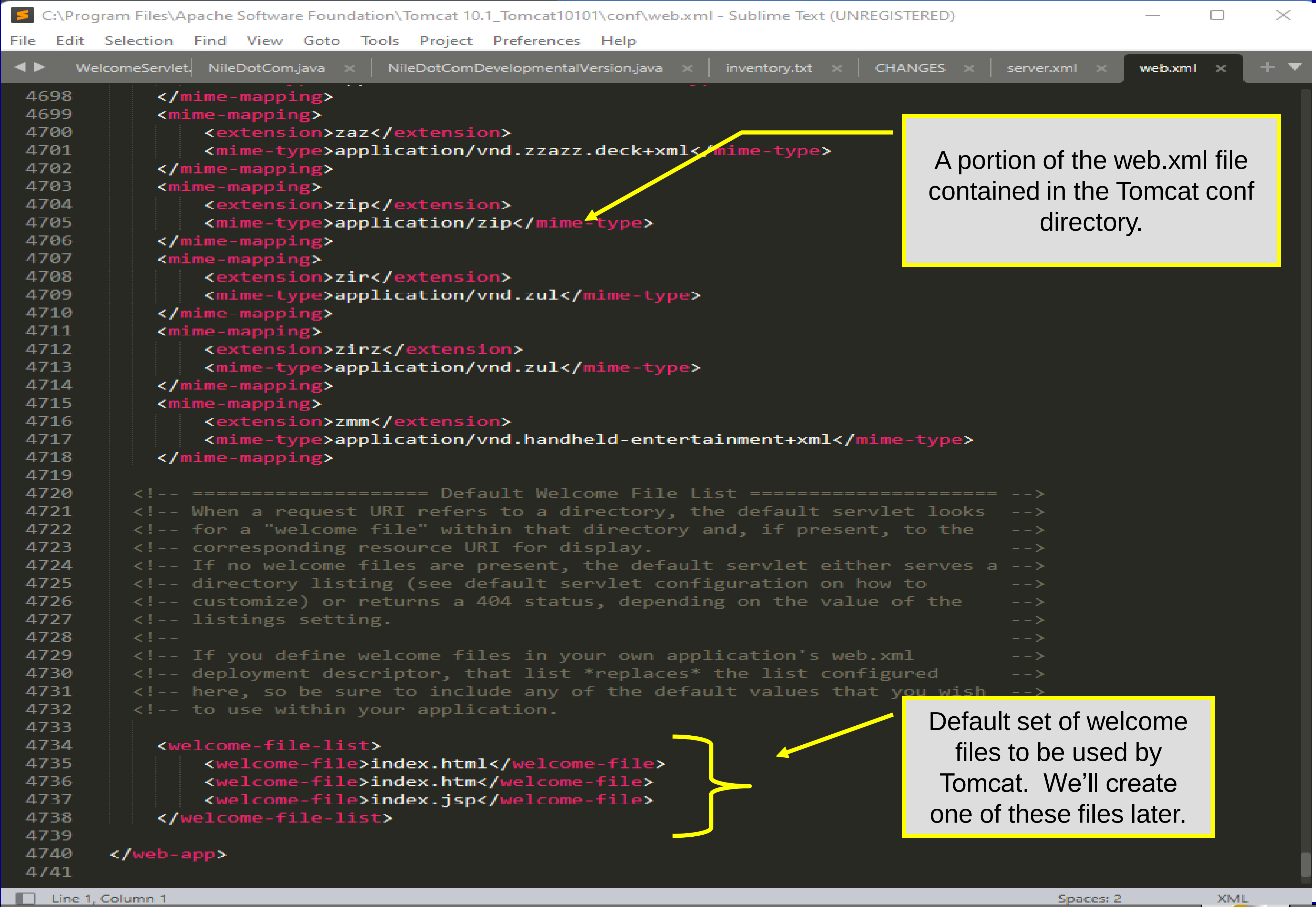
A Tour of Tomcat (cont.)

- 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.


C:\Program Files\Apache Software Foundation\Tomcat 10.1_Tomcat10101\conf\server.xml - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
WelcomeServlet.html
NileDotCom.java
NileDotComDevelopmentalVersion.java
inventory.txt
CHANGES
server.xml
Line 1, Column 1
Spaces: 4
XML

```
        a single "Container" Note:  A "Service" is not itself a "Container",
        so you may not define subcomponents such as "Valves" at this level.
        Documentation at /docs/config/service.html
    -->
  <Service name="Catalina">

    <!--The connectors can use a shared executor, you can define o                            s-->
    <!--
    <Executor name="tomcatThreadPool" namePrefix="catalina-exec-"
        maxThreads="150" minSpareThreads="4"/>
    -->


    <!-- A "Connector" represents an endpoint by which requests are
         and responses are returned. Documentation at :
         HTTP Connector: /docs/config/http.html
         AJP  Connector: /docs/config/ajp.html
         Define a non-SSL/TLS HTTP/1.1 Connector on port 8082
    -->
    <Connector port="8082" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443" />
    <!-- A "Connector" using the shared thread pool-->
    <!--
    <Connector executor="tomcatThreadPool"
               port="8082" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443" />
    -->
    <!-- Define an SSL/TLS HTTP/1.1 Connector on port 8443 with HTTP/2
         This connector uses the NIO implementation. The default
         SSLImplementation will depend on the presence of the APR/native
         library and the useOpenSSL attribute of the AprLifecycleListener.
         Either JSSE or OpenSSL style configuration may be used regardless of
         the SSLImplementation selected. JSSE style configuration is used below.
    -->
    <!--
    <Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
               maxThreads="150" SSLEnabled="true">
        <UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol" />
        <SSLHostConfig>
            <Certificate certificateKeystoreFile="conf/localhost-rsa.jks"
                         type="RSA" />
        </SSLHostConfig>
    </Connector>
```

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


C:\Program Files\Apache Software Foundation\Tomcat 10.1_Tomcat10101\conf\web.xml - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
WelcomeServlet NileDotCom.java NileDotComDevelopmentalVersion.java inventory.txt CHANGES server.xml web.xml
Line 1, Column 1
Spaces: 2
XML

```
    </mime-mapping>
    <mime-mapping>
        <extension>zaz</extension>
        <mime-type>application/vnd.zzazz.deck+xml</mime-type>
    </mime-mapping>
    <mime-mapping>
        <extension>zip</extension>
        <mime-type>application/zip</mime-type>
    </mime-mapping>
    <mime-mapping>
        <extension>zir</extension>
        <mime-type>application/vnd.zul</mime-type>
    </mime-mapping>
    <mime-mapping>
        <extension>zirz</extension>
        <mime-type>application/vnd.zul</mime-type>
    </mime-mapping>
    <mime-mapping>
        <extension>zmm</extension>
        <mime-type>application/vnd.handheld-entertainment+xml</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 no welcome files 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  -->
  <!-- to 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.

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, 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 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 pages 7 and 8.)

  - 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.

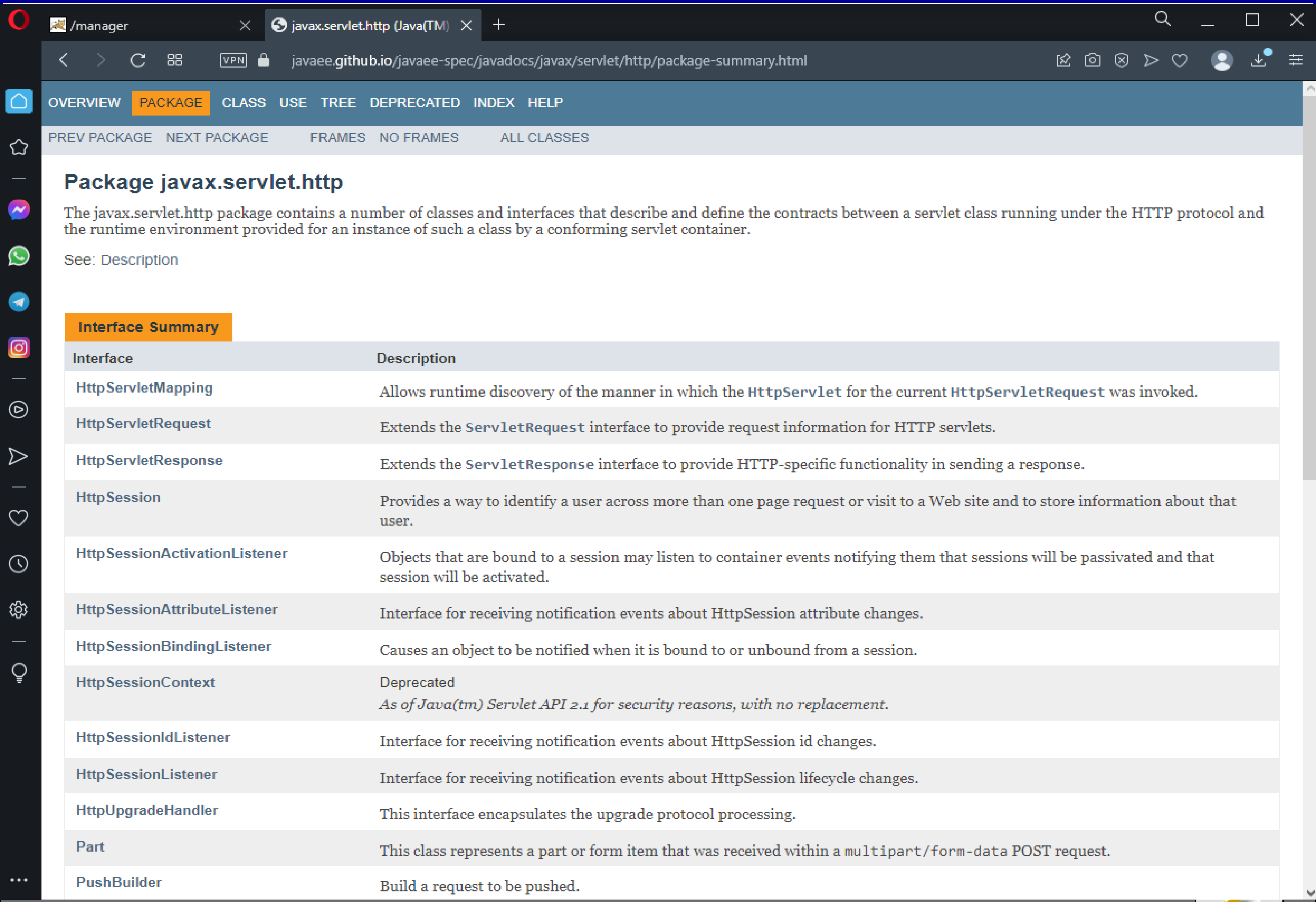
- Class `HttpServlet` defines methods `doGet` and `doPost` to respond to get and post requests from a client.

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: https://javaee.github.io/javaee-spec/javadocs/javax/servlet/http/package-summary.html, (see next page), however, a few of the more common ones are shown on page 59. (Note: you can also get to them from Tomcat, see page 57.)


/manager
javax.servlet.http (Java(TM)
javaee.github.io/javaee-spec/javadocs/javax/servlet/http/package-summary.html
OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP
PREV PACKAGE NEXT PACKAGE FRAMES NO FRAMES ALL CLASSES

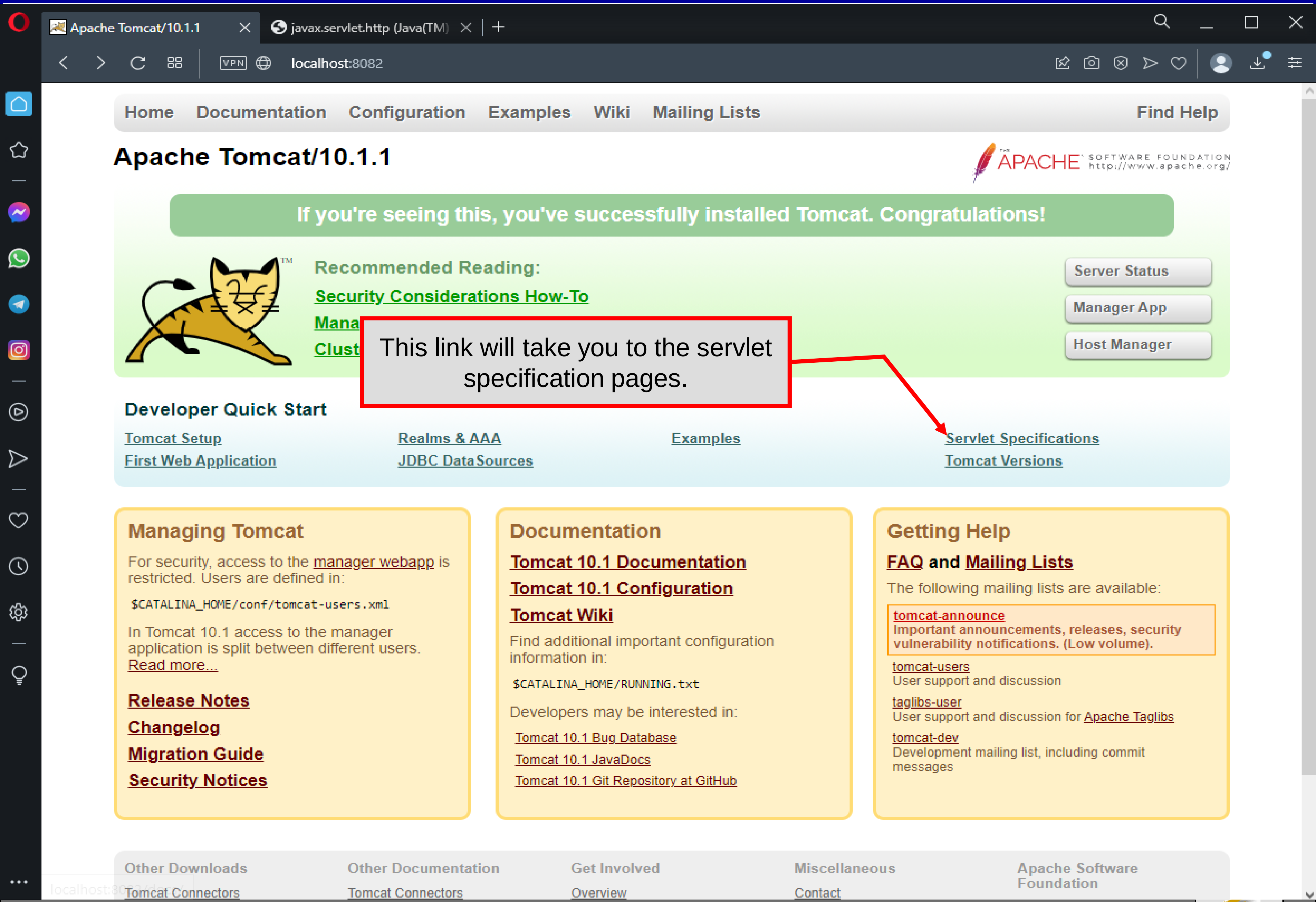
Package javax.servlet.http

The javax.servlet.http package contains a number of classes and interfaces that describe and define the contracts between a servlet class running under the HTTP protocol and the runtime environment provided for an instance of such a class by a conforming servlet container.

See: Description

Interface Summary

Interface	Description
HttpServletMapping	Allows runtime discovery of the manner in which the `HttpServlet` for the current `HttpServletRequest` was invoked.
HttpServletRequest	Extends the `ServletRequest` interface to provide request information for HTTP servlets.
HttpServletResponse	Extends the `ServletResponse` interface to provide HTTP-specific functionality in sending a response.
HttpSession	Provides a way to identify a user across more than one page request or visit to a Web site and to store information about that user.
HttpSessionActivationListener	Objects that are bound to a session may listen to container events notifying them that sessions will be passivated and that session will be activated.
HttpSessionAttributeListener	Interface for receiving notification events about HttpSession attribute changes.
HttpSessionBindingListener	Causes an object to be notified when it is bound to or unbound from a session.
HttpSessionContext	Deprecated *As of Java(tm) Servlet API 2.1 for security reasons, with no replacement.*
HttpSessionIdListener	Interface for receiving notification events about HttpSession id changes.
HttpSessionListener	Interface for receiving notification events about HttpSession lifecycle changes.
HttpUpgradeHandler	This interface encapsulates the upgrade protocol processing.
Part	This class represents a part or form item that was received within a `multipart/form-data` POST request.
PushBuilder	Build a request to be pushed.



Apache Tomcat/10.1.1 | javax.servlet.http (Java(TM)

localhost:8082

Home Documentation Configuration Examples Wiki Mailing Lists Find Help

Apache Tomcat/10.1.1

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

Recommended Reading:

Security Considerations How-To

Mana

Clust

This link will take you to the servlet specification pages.

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 10.1 access to the manager application is split between different users. Read more...

Release Notes

Changelog

Migration Guide

Security Notices

Documentation

Tomcat 10.1 Documentation

Tomcat 10.1 Configuration

Tomcat Wiki

Find additional important configuration information in:

```
$CATALINA_HOME/RUNNING.txt
```

Developers may be interested in:

Tomcat 10.1 Bug Database

Tomcat 10.1 JavaDocs

Tomcat 10.1 Git Repository at GitHub

Getting Help

FAQ and Mailing Lists

The following mailing lists are available:

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

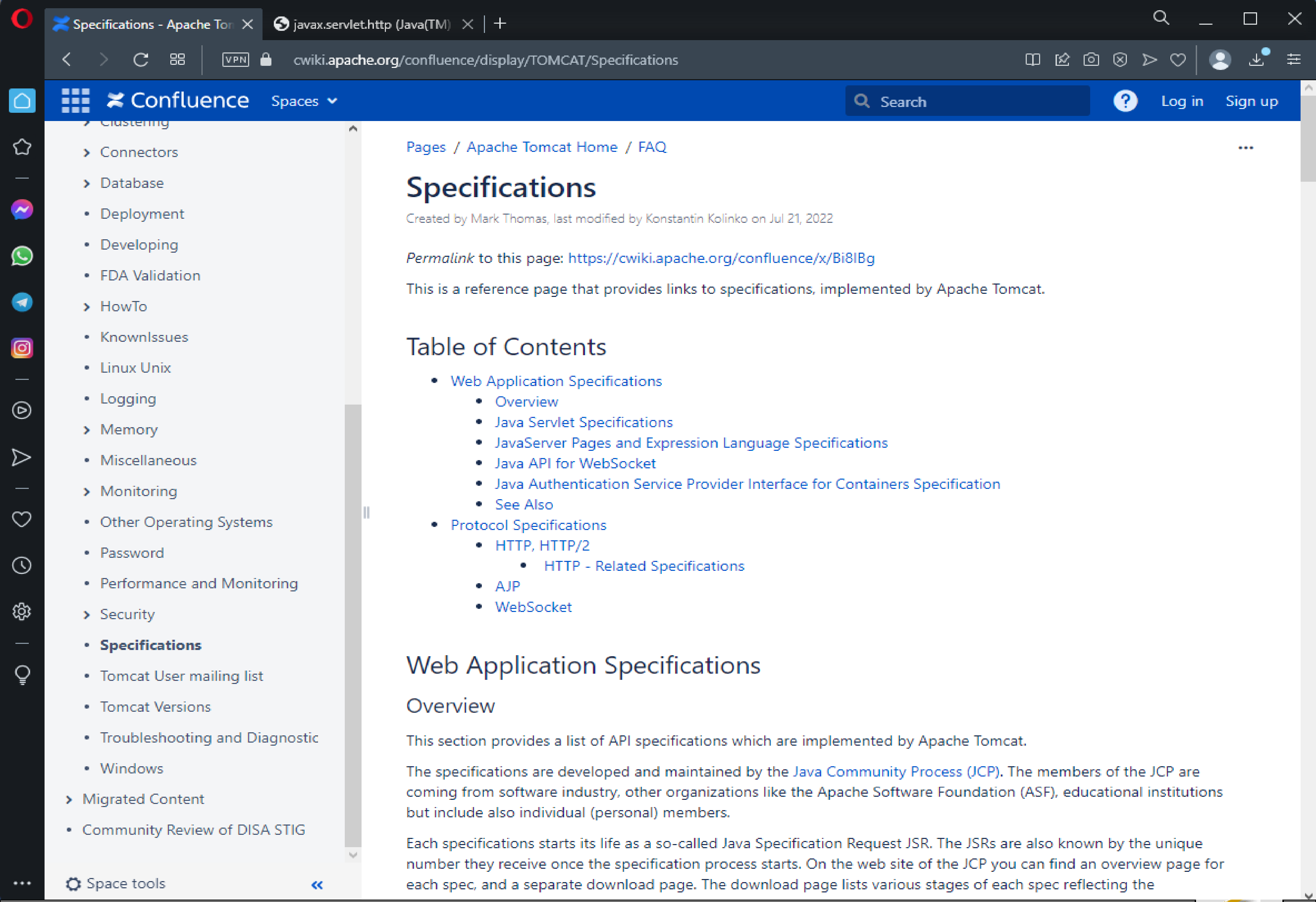
tomcat-users
User support and discussion

taglibs-user
User support and discussion for Apache Taglibs

tomcat-dev
Development mailing list, including commit messages

Other Downloads | Other Documentation | Get Involved | Miscellaneous | Apache Software Foundation

Tomcat Connectors | Tomcat Connectors | Overview | Contact



Pages / Apache Tomcat Home / FAQ

Specifications

Created by Mark Thomas, last modified by Konstantin Kolinko on Jul 21, 2022

Permalink to this page: https://cwiki.apache.org/confluence/x/Bi8IBg

This is a reference page that provides links to specifications, implemented by Apache Tomcat.

Table of Contents

Web Application Specifications

Overview

This section provides a list of API specifications which are implemented by Apache Tomcat.

The specifications are developed and maintained by the Java Community Process (JCP). The members of the JCP are coming from software industry, other organizations like the Apache Software Foundation (ASF), educational institutions but include also individual (personal) members.

Each specifications starts its life as a so-called Java Specification Request JSR. The JSRs are also known by the unique number they receive once the specification process starts. On the web site of the JCP you can find an overview page for each spec, and a separate download page. The download page lists various stages of each spec reflecting the

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

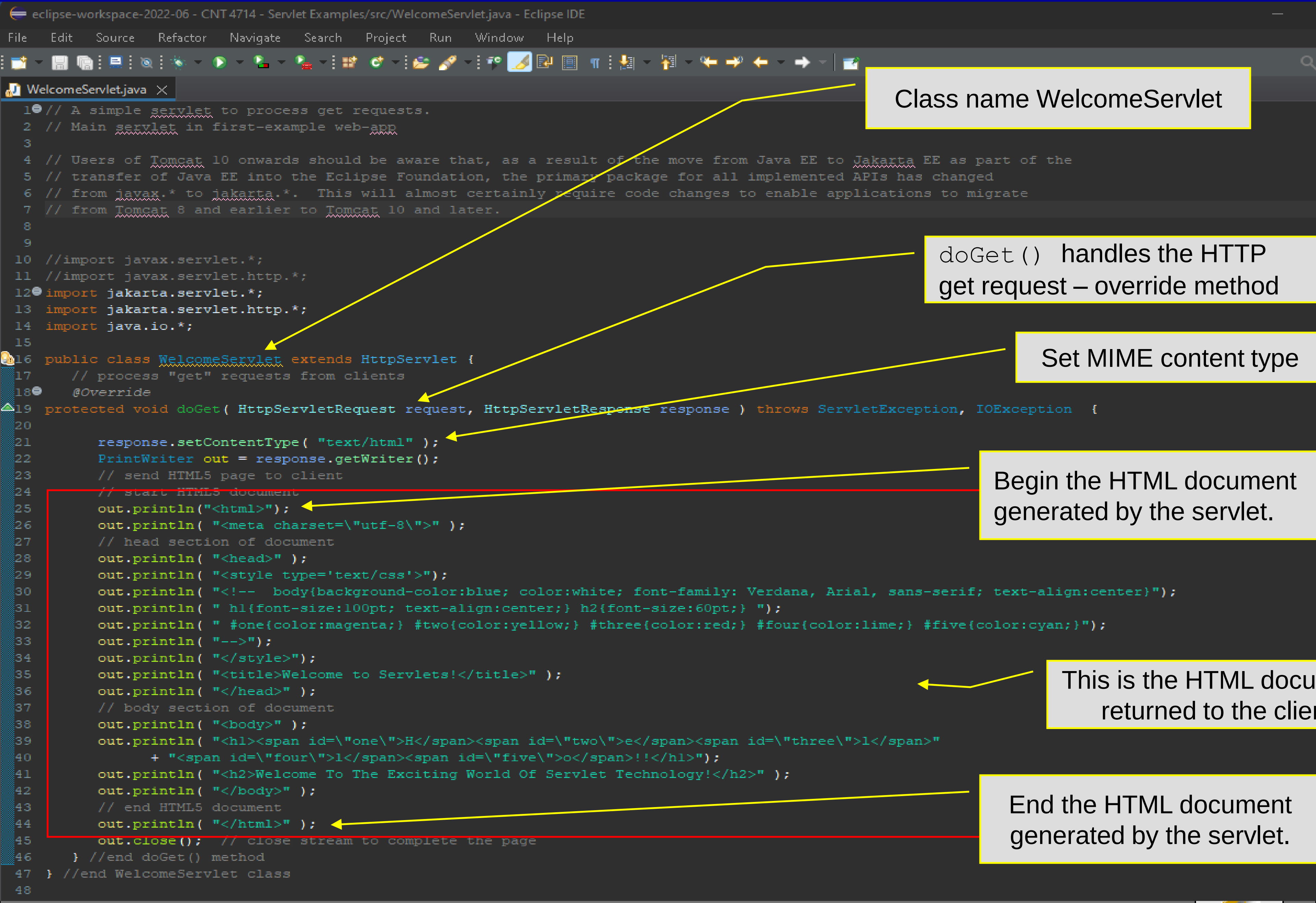
- Every invocation of `doGet` or `doPost` for an `HttpServlet` receives an object that implements interface `HttpServletResponse`.
- The servlet container creates an `HttpServletResponse` object and passes it to the servlet's `service` method, which in turn, passes it to `doGet` or `doPost`.
- This object provides methods that enable the servlet to formulate the response to the client.
- The full list of HttpServletResponse methods is available at: https://javaee.github.io/javaee-spec/javadocs/javax/servlet/http/package-summary.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.


eclipse-workspace-2022-06 - CNT 4714 - Servlet Examples/src/WelcomeServlet.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
WelcomeServlet.java

```java
// A simple servlet to process get requests.
// Main servlet in first-example web-app

// Users of Tomcat 10 onwards should be aware that, as a result of the move from Java EE to Jakarta EE as part of the
// transfer of Java EE into the Eclipse Foundation, the primary package for all implemented APIs has changed
// from javax.* to jakarta.*.  This will almost certainly require code changes to enable applications to migrate
// from Tomcat 8 and earlier to Tomcat 10 and later.


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

public class WelcomeServlet extends HttpServlet {
  // process "get" requests from clients
  @Override
protected void doGet( HttpServletRequest request, HttpServletResponse response ) throws ServletException, IOException  {

    response.setContentType( "text/html" );
    PrintWriter out = response.getWriter();
    // send HTML5 page to client
    // start HTML5 document
    out.println("<html>");
    out.println( "<meta charset=\"utf-8\">" );
    // head section of document
    out.println( "<head>" );
    out.println( "<style type='text/css'>");
    out.println( "<!--  body{background-color:blue; color:white; font-family: Verdana, Arial, sans-serif; text-align:center}");
    out.println( " h1{font-size:100pt; text-align:center;} h2{font-size:60pt;} ");
    out.println( " #one{color:magenta;} #two{color:yellow;} #three{color:red;} #four{color:lime;} #five{color:cyan;}");
    out.println( "-->");
    out.println( "</style>");
    out.println( "<title>Welcome to Servlets!</title>" );
    out.println( "</head>" );
    // body section of document
    out.println( "<body>" );
    out.println( "<h1><span id=\"one\">H</span><span id=\"two\">e</span><span id=\"three\">l</span>"
          + "<span id=\"four\">l</span><span id=\"five\">o</span>!!</h1>");
    out.println( "<h2>Welcome To The Exciting World Of Servlet Technology!</h2>" );
    out.println( "</body>" );
    // end HTML5 document
    out.println( "</html>" );
    out.close();  // close stream to complete the page
  } //end doGet() method
} //end WelcomeServlet class
```

Class name WelcomeServlet

`doGet()` handles the HTTP get request – override method

Set MIME content type

Begin the HTML document generated by the servlet.

This is the HTML docu
returned to the clie

End the HTML document generated by the servlet.

Handling HTTP `get` Requests (cont.)

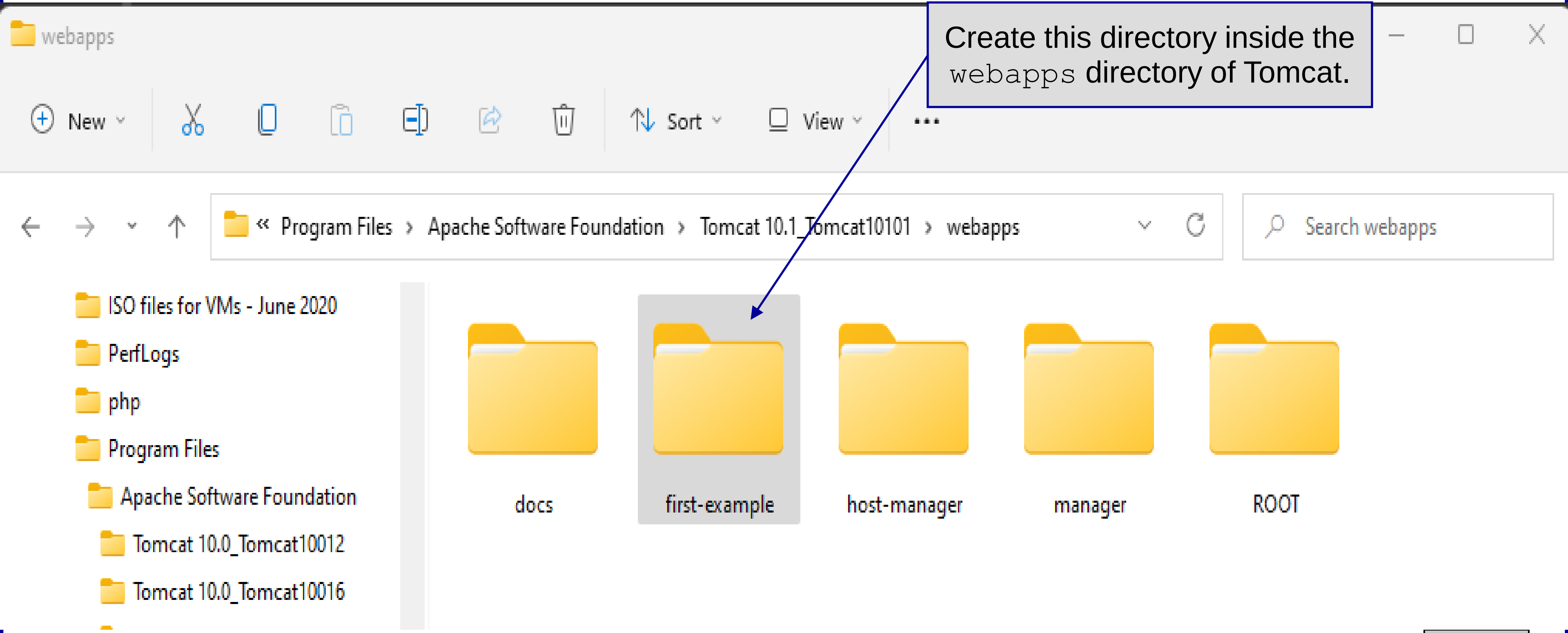
- The servlet creates an HTML document containing the text “Hello! Welcome to the Exciting World of Servlet Technology!”
- This text is the response to the client and is sent through the `PrintWriter` object obtained from the `HttpServletRepsonse` object.
- The response object’s `setContentType` method 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.)

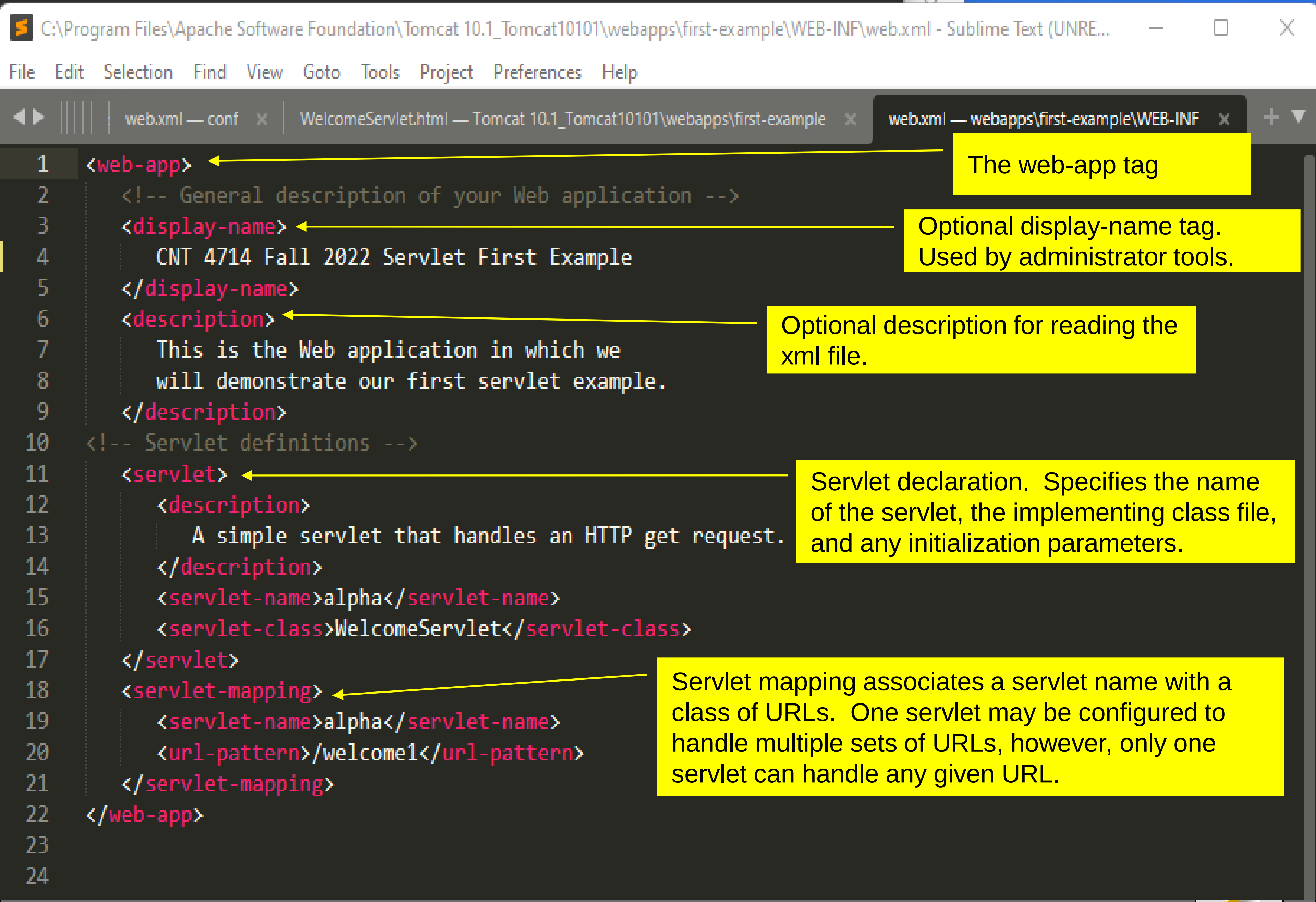
- The most basic web application in Tomcat will require the creation of a directory inside the `webapps` directory to hold the web application. For this first example, we'll create a subdirectory called `first-example`.



Create this directory inside the `webapps` directory of Tomcat.

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.


C:\Program Files\Apache Software Foundation\Tomcat 10.1_Tomcat10101\webapps\first-example\WEB-INF\web.xml - Sublime Text (UNRE...
File Edit Selection Find View Goto Tools Project Preferences Help
web.xml — conf
WelcomeServlet.html — Tomcat 10.1_Tomcat10101\webapps\first-example
web.xml — webapps\first-example\WEB-INF

```xml
<web-app>
   <!-- General description of your Web application -->
   <display-name>
      CNT 4714 Fall 2022 Servlet First Example
   </display-name>
   <description>
      This is the Web application in which we
      will demonstrate our first servlet example.
   </description>
<!-- Servlet definitions -->
   <servlet>
      <description>
         A simple servlet that handles an HTTP get request.
      </description>
      <servlet-name>alpha</servlet-name>
      <servlet-class>WelcomeServlet</servlet-class>
   </servlet>
   <servlet-mapping>
      <servlet-name>alpha</servlet-name>
      <url-pattern>/welcome1</url-pattern>
   </servlet-mapping>
</web-app>
```

The web-app tag

Optional display-name tag. Used by administrator tools.

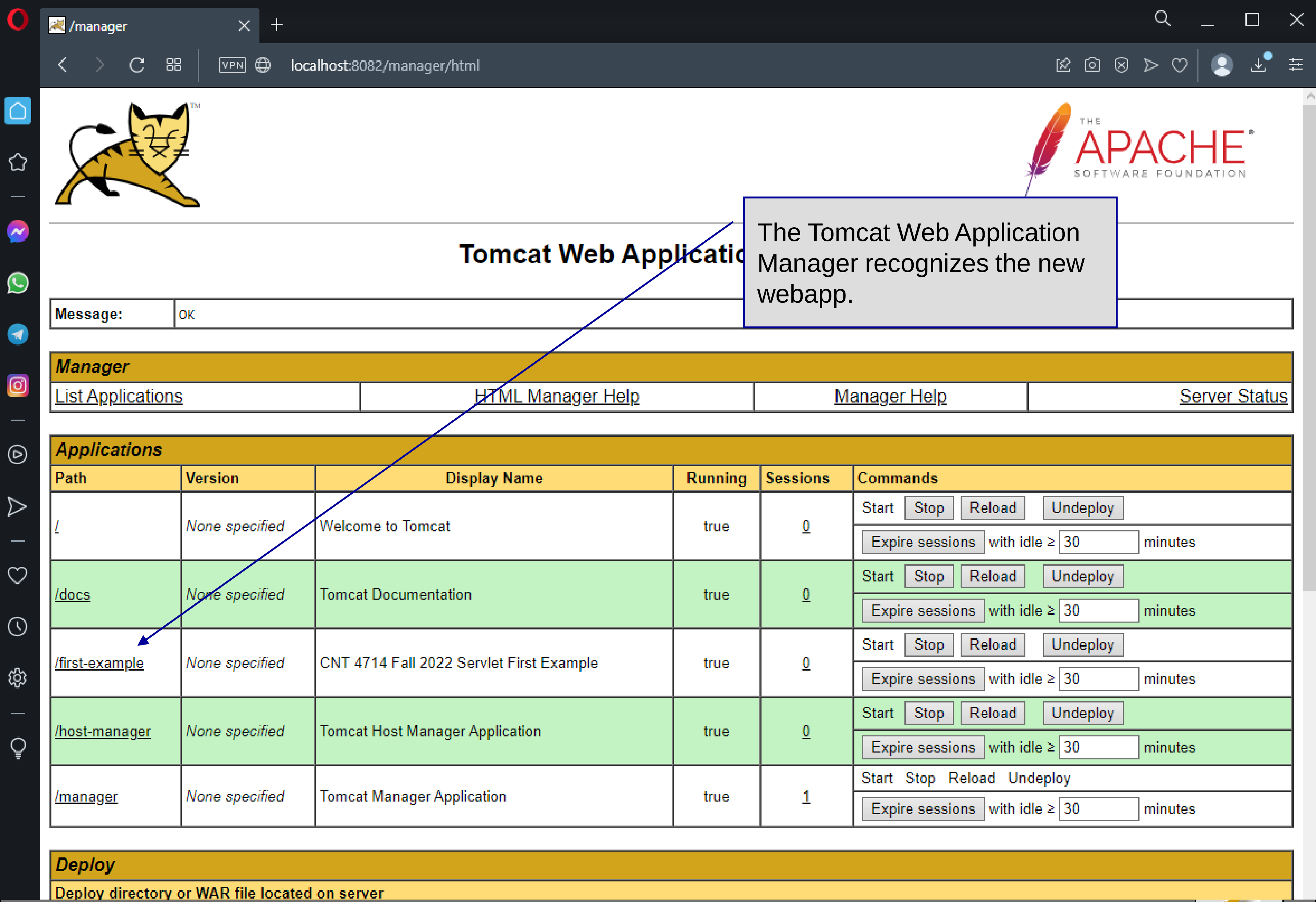
Optional description for reading the xml file.

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.

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:8082/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).



/manager

localhost:8082/manager/html

Tomcat Web Applicatio

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 Fall 2022 Servlet First Example	true	0	Start Stop Reload Undeploy / Expire sessions with idle ≥ 30 minutes
/host-manager	None specified	Tomcat Host Manager Application	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

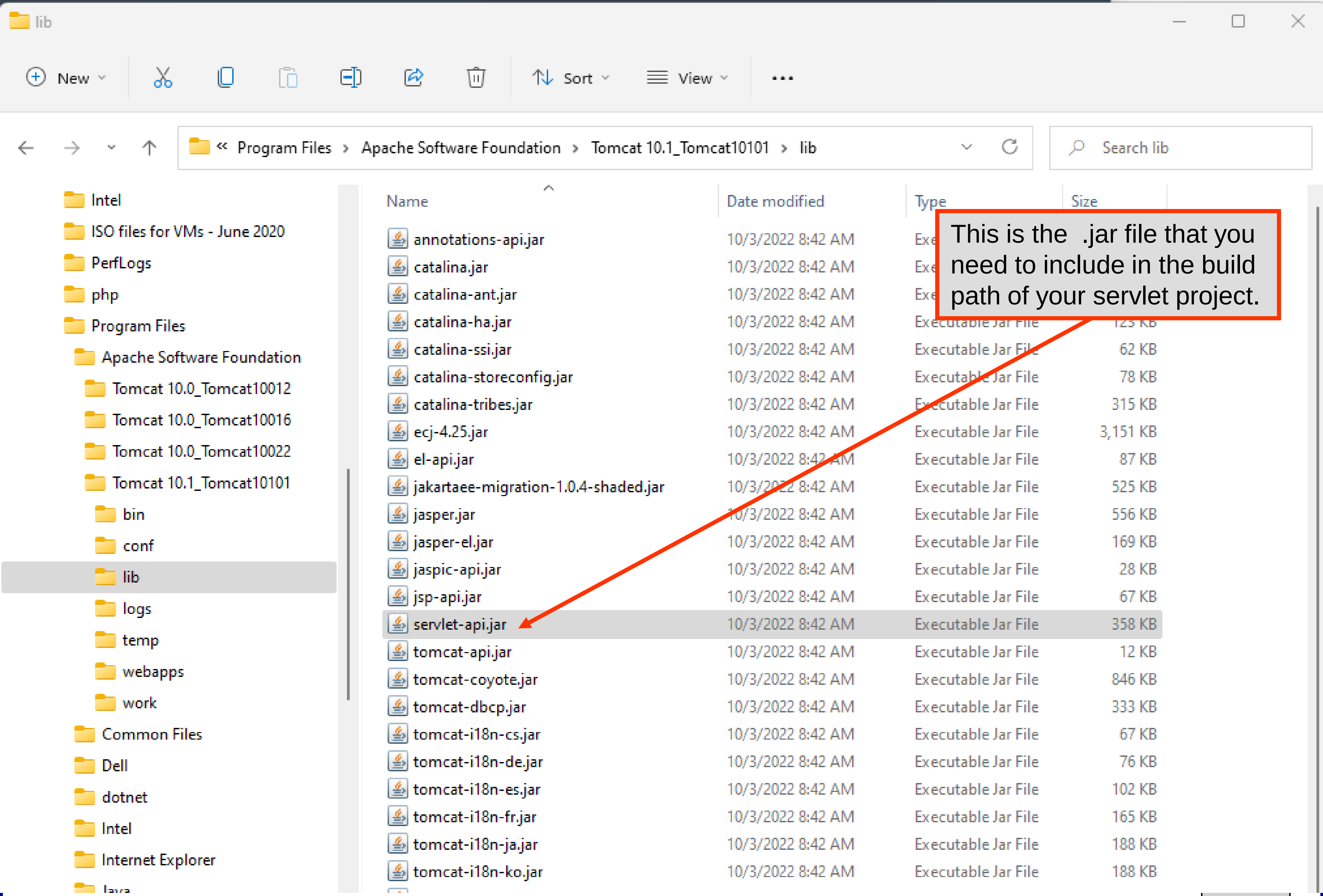
Deploy directory or WAR file located on server

The Tomcat Web Application Manager recognizes the new webapp.

Tomcat/Java Configuration - The Servlet API

IMPORTANT ! !

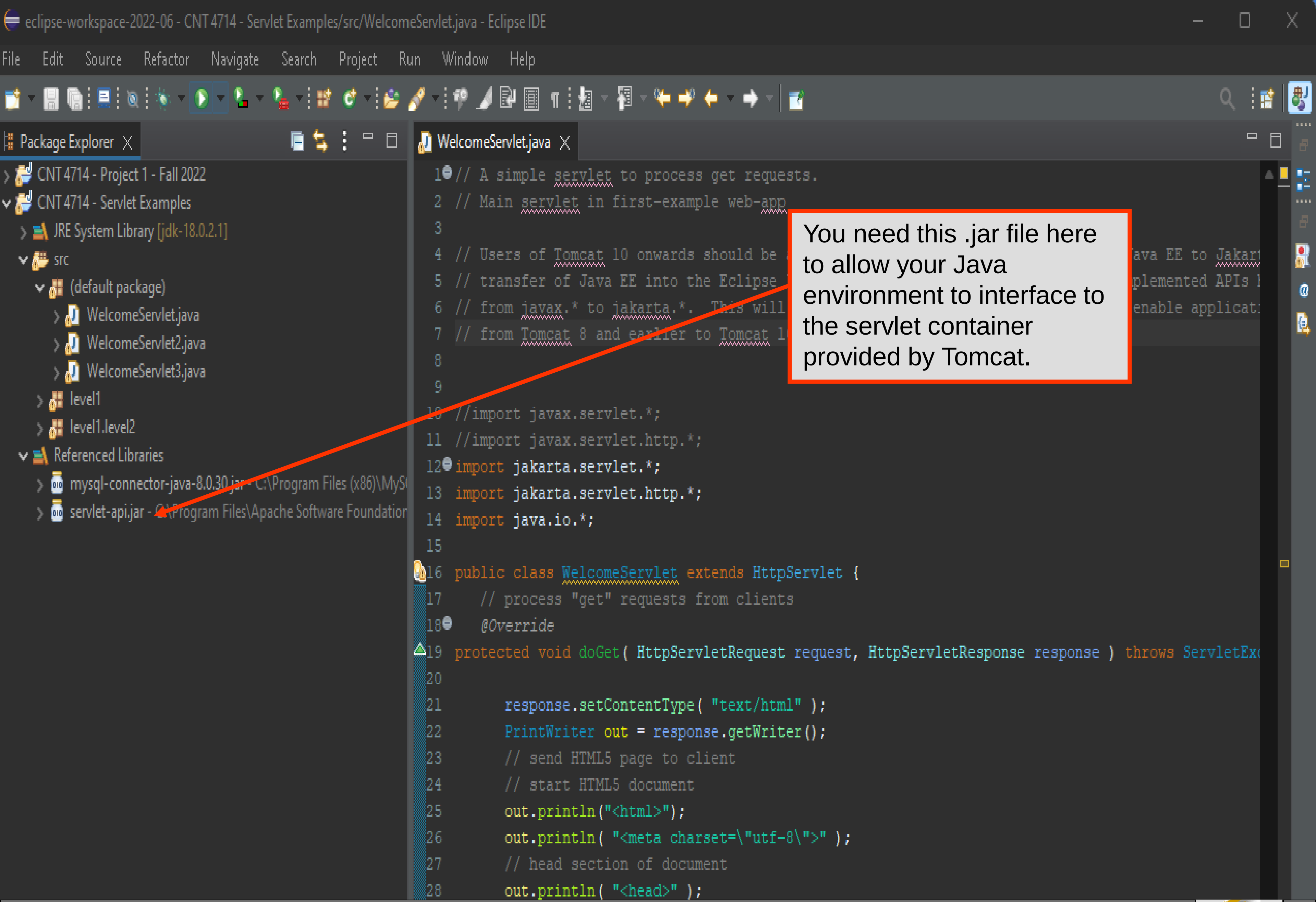
- Your Tomcat installation includes the `servlet-api.jar` file. This file can be found in the `lib` folder in Tomcat.
- This file must be included in the build path of the project where you are creating and compiling your servlet.java files.
- Add this archive to your Java project in the same fashion that you added the MySQL connector file for the JDBC examples.
- See the screen shots on the next two pages.


lib
New
Sort
View
Program Files › Apache Software Foundation › Tomcat 10.1_Tomcat10101 › lib
Search lib

Intel
ISO files for VMs - June 2020
PerfLogs
php
Program Files
Apache Software Foundation
Tomcat 10.0_Tomcat10012
Tomcat 10.0_Tomcat10016
Tomcat 10.0_Tomcat10022
Tomcat 10.1_Tomcat10101
bin
conf
lib
logs
temp
webapps
work
Common Files
Dell
dotnet
Intel
Internet Explorer
Java

Name	Date modified	Type	Size
annotations-api.jar	10/3/2022 8:42 AM	Exe	
catalina.jar	10/3/2022 8:42 AM	Exe	
catalina-ant.jar	10/3/2022 8:42 AM	Exe	
catalina-ha.jar	10/3/2022 8:42 AM	Executable Jar File	123 KB
catalina-ssi.jar	10/3/2022 8:42 AM	Executable Jar File	62 KB
catalina-storeconfig.jar	10/3/2022 8:42 AM	Executable Jar File	78 KB
catalina-tribes.jar	10/3/2022 8:42 AM	Executable Jar File	315 KB
ecj-4.25.jar	10/3/2022 8:42 AM	Executable Jar File	3,151 KB
el-api.jar	10/3/2022 8:42 AM	Executable Jar File	87 KB
jakartaee-migration-1.0.4-shaded.jar	10/3/2022 8:42 AM	Executable Jar File	525 KB
jasper.jar	10/3/2022 8:42 AM	Executable Jar File	556 KB
jasper-el.jar	10/3/2022 8:42 AM	Executable Jar File	169 KB
jaspic-api.jar	10/3/2022 8:42 AM	Executable Jar File	28 KB
jsp-api.jar	10/3/2022 8:42 AM	Executable Jar File	67 KB
servlet-api.jar	10/3/2022 8:42 AM	Executable Jar File	358 KB
tomcat-api.jar	10/3/2022 8:42 AM	Executable Jar File	12 KB
tomcat-coyote.jar	10/3/2022 8:42 AM	Executable Jar File	846 KB
tomcat-dbcp.jar	10/3/2022 8:42 AM	Executable Jar File	333 KB
tomcat-i18n-cs.jar	10/3/2022 8:42 AM	Executable Jar File	67 KB
tomcat-i18n-de.jar	10/3/2022 8:42 AM	Executable Jar File	76 KB
tomcat-i18n-es.jar	10/3/2022 8:42 AM	Executable Jar File	102 KB
tomcat-i18n-fr.jar	10/3/2022 8:42 AM	Executable Jar File	165 KB
tomcat-i18n-ja.jar	10/3/2022 8:42 AM	Executable Jar File	188 KB
tomcat-i18n-ko.jar	10/3/2022 8:42 AM	Executable Jar File	188 KB

This is the .jar file that you need to include in the build path of your servlet project.

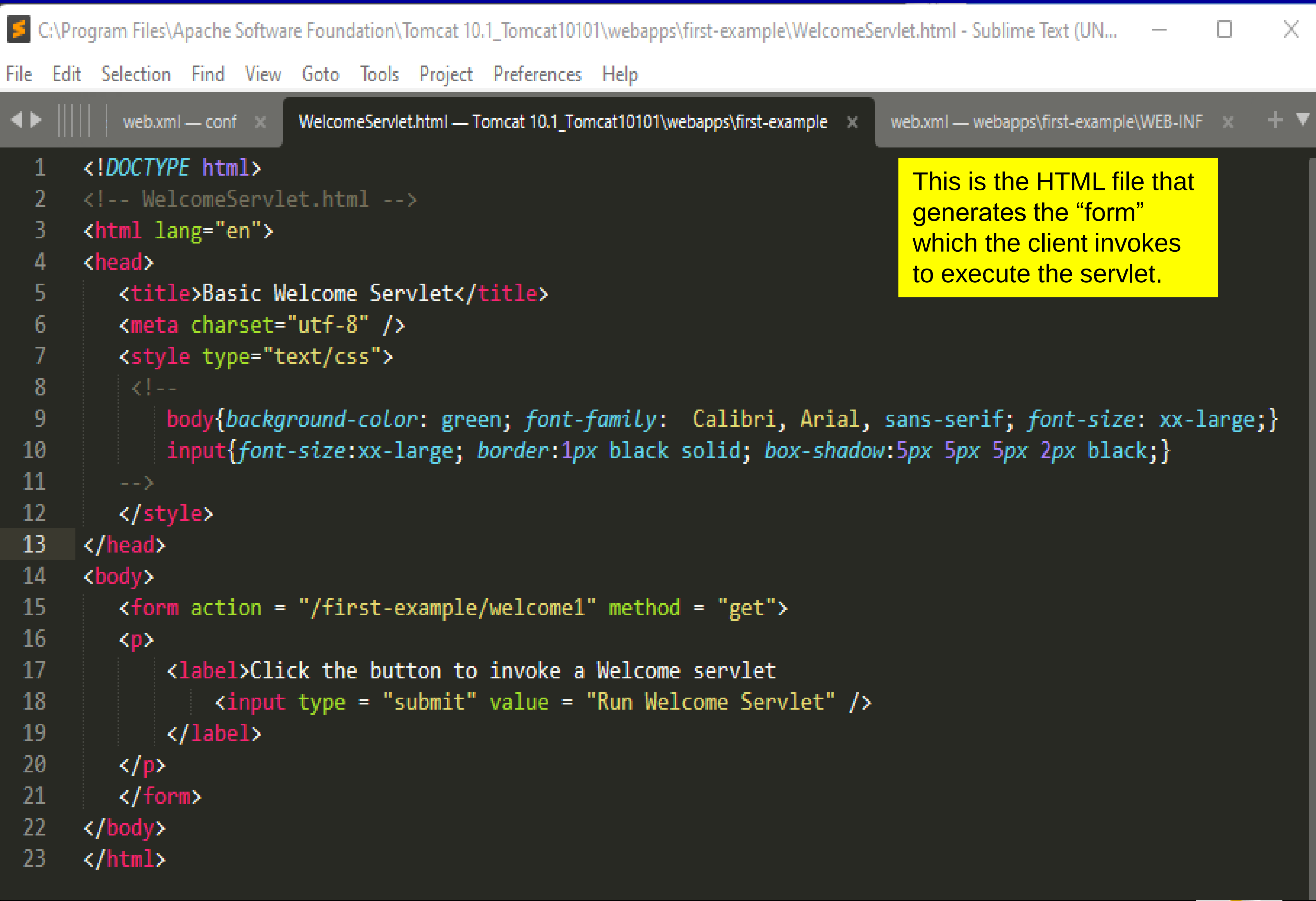


eclipse-workspace-2022-06 - CNT 4714 - Servlet Examples/src/WelcomeServlet.java - Eclipse IDE

```
CNT 4714 - Project 1 - Fall 2022
CNT 4714 - Servlet Examples
  JRE System Library [jdk-18.0.2.1]
  src
    (default package)
      WelcomeServlet.java
      WelcomeServlet2.java
      WelcomeServlet3.java
    level1
    level1.level2
  Referenced Libraries
    mysql-connector-java-8.0.30.jar - C:\Program Files (x86)\MyS
    servlet-api.jar - C:\Program Files\Apache Software Foundatio
```

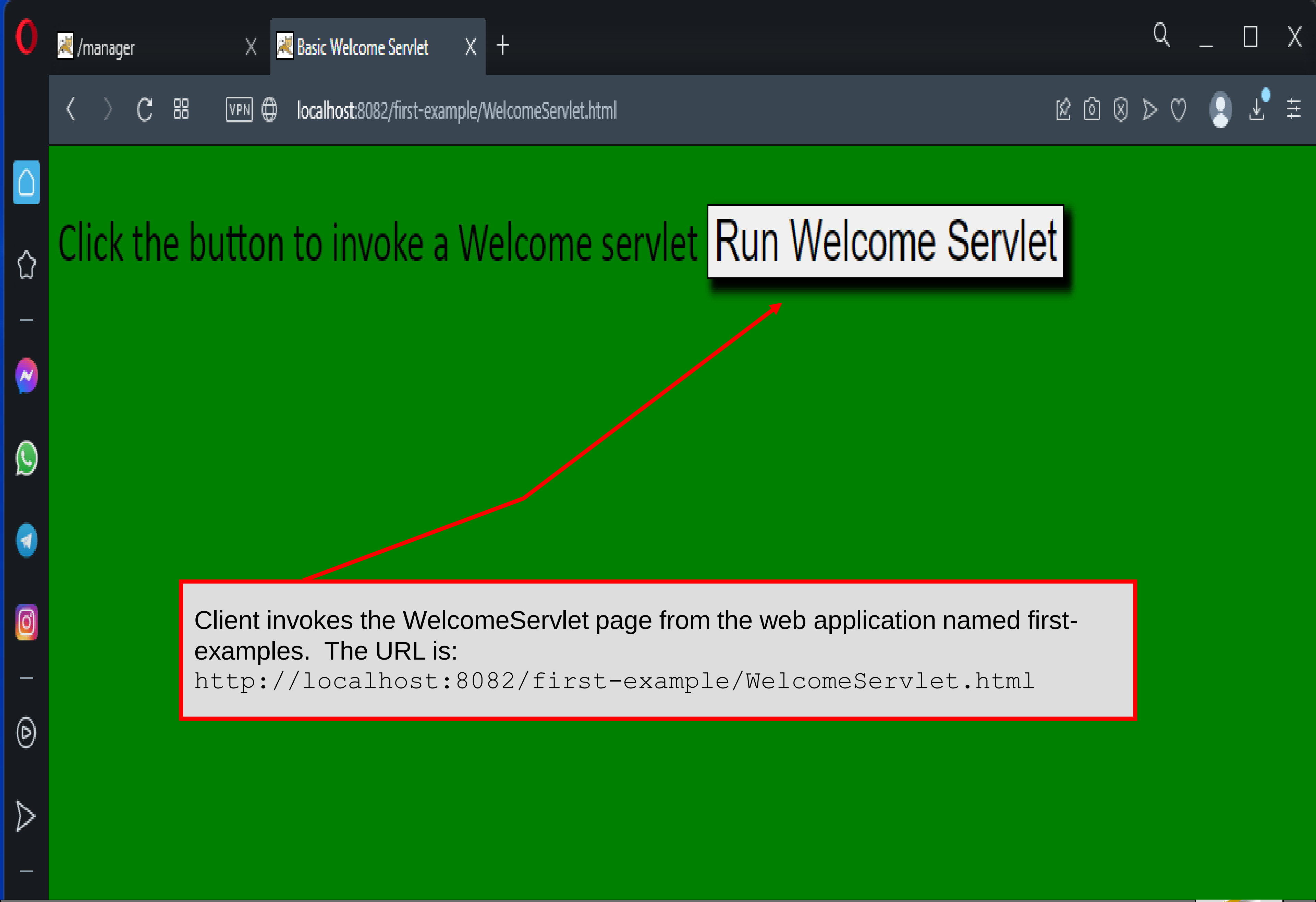
```java
// A simple servlet to process get requests.
// Main servlet in first-example web-app

// Users of Tomcat 10 onwards should be                                  ava EE to Jakar
// transfer of Java EE into the Eclipse                                  plemented APIs
// from javax.* to jakarta.*.  This will                                 enable applicat
// from Tomcat 8 and earlier to Tomcat 1


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

public class WelcomeServlet extends HttpServlet {
   // process "get" requests from clients
   @Override
protected void doGet( HttpServletRequest request, HttpServletResponse response ) throws ServletEx

      response.setContentType( "text/html" );
      PrintWriter out = response.getWriter();
      // send HTML5 page to client
      // start HTML5 document
      out.println("<html>");
      out.println( "<meta charset=\"utf-8\">" );
      // head section of document
      out.println( "<head>" );
```

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

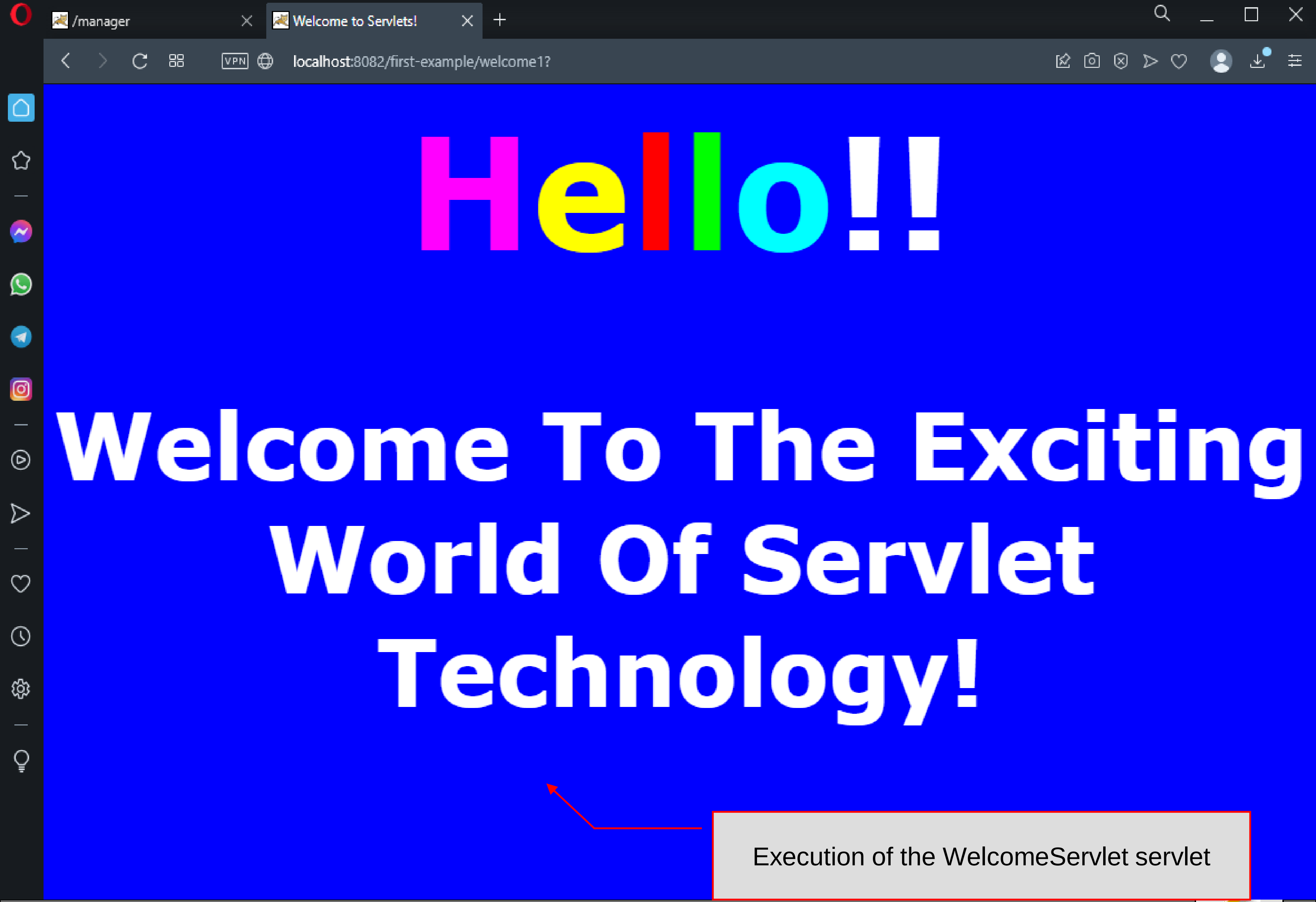

C:\Program Files\Apache Software Foundation\Tomcat 10.1_Tomcat10101\webapps\first-example\WelcomeServlet.html - Sublime Text (UN...
File Edit Selection Find View Goto Tools Project Preferences Help
web.xml — conf
WelcomeServlet.html — Tomcat 10.1_Tomcat10101\webapps\first-example
web.xml — webapps\first-example\WEB-INF

```html
<!DOCTYPE html>
<!-- WelcomeServlet.html -->
<html lang="en">
<head>
   <title>Basic Welcome Servlet</title>
   <meta charset="utf-8" />
   <style type="text/css">
    <!--
      body{background-color: green; font-family:  Calibri, Arial, sans-serif; font-size: xx-large;}
      input{font-size:xx-large; border:1px black solid; box-shadow:5px 5px 5px 2px black;}
   -->
   </style>
</head>
<body>
   <form action = "/first-example/welcome1" method = "get">
   <p>
      <label>Click the button to invoke a Welcome servlet
         <input type = "submit" value = "Run Welcome Servlet" />
      </label>
   </p>
   </form>
</body>
</html>
```

This is the HTML file that generates the “form” which the client invokes to execute the servlet.


/manager
Basic Welcome Servlet
localhost:8082/first-example/WelcomeServlet.html
Click the button to invoke a Welcome servlet
Run Welcome Servlet

Client invokes the WelcomeServlet page from the web application named first-examples. The URL is: `http://localhost:8082/first-example/WelcomeServlet.html`



Hello!!

Welcome To The Exciting World Of Servlet Technology!

Execution of the WelcomeServlet servlet

An HTML Document

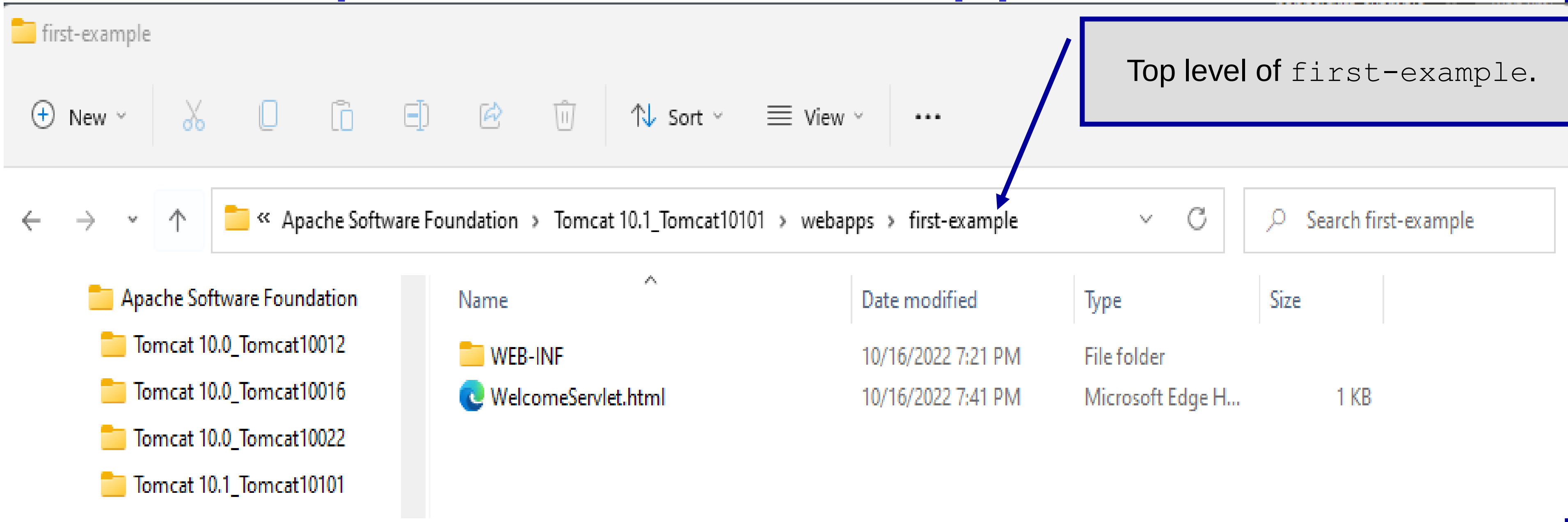
- The HTML document shown on page 74 provides a `form` that invokes the servlet defined on page 63.
- The form's `action` attribute (/first-example/welcome1) specifies the URL path that invokes the servlet.
- The form's method attribute indicates that the browser sends a get request to the server, which 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` create the `WelcomeServlet.html` file as shown on page 74.
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-examples should look like the picture at the top of the next page.

Set-Up For First Web Application (cont.)


first-example
New
Sort
View
« Apache Software Foundation › Tomcat 10.1_Tomcat10101 › webapps › first-example
Search first-example

Top level of `first-example`.

Apache Software Foundation
- Tomcat 10.0_Tomcat10012
- Tomcat 10.0_Tomcat10016
- Tomcat 10.0_Tomcat10022
- Tomcat 10.1_Tomcat10101

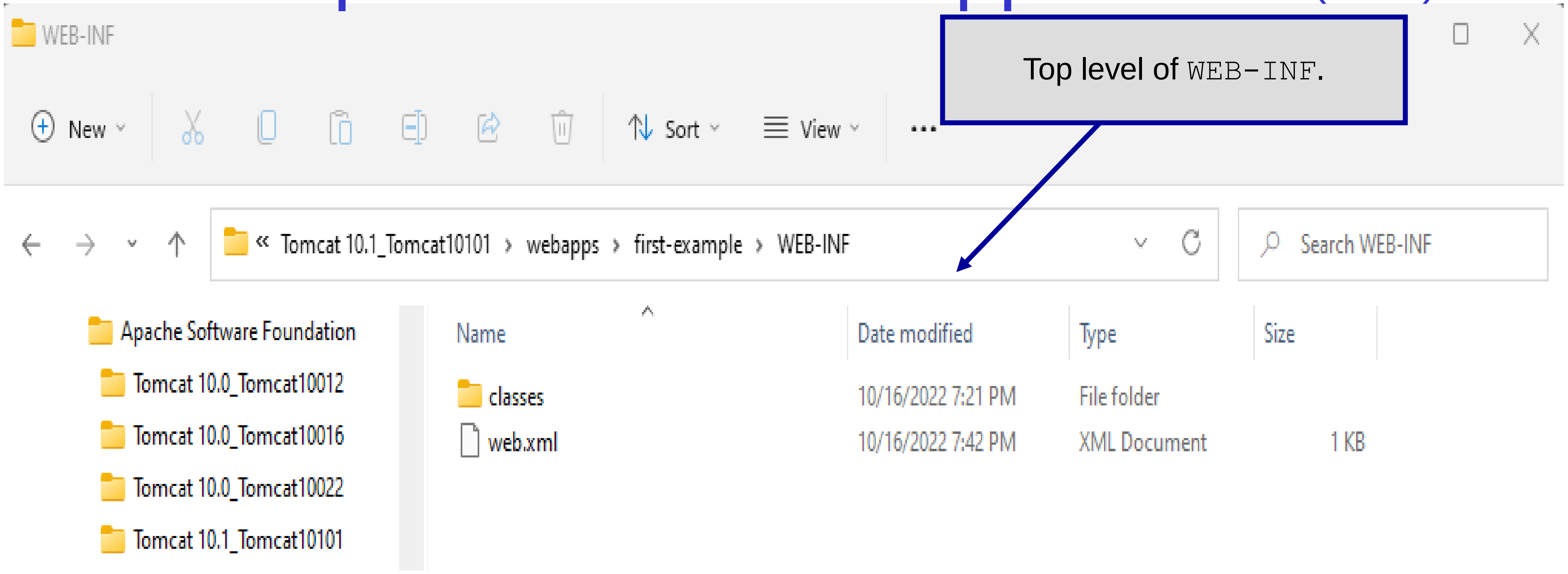
Name	Date modified	Type	Size
WEB-INF	10/16/2022 7:21 PM	File folder	
WelcomeServlet.html	10/16/2022 7:41 PM	Microsoft Edge H...	1 KB

5. Create the `web.xml` configuration file as shown on page 68.

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.)

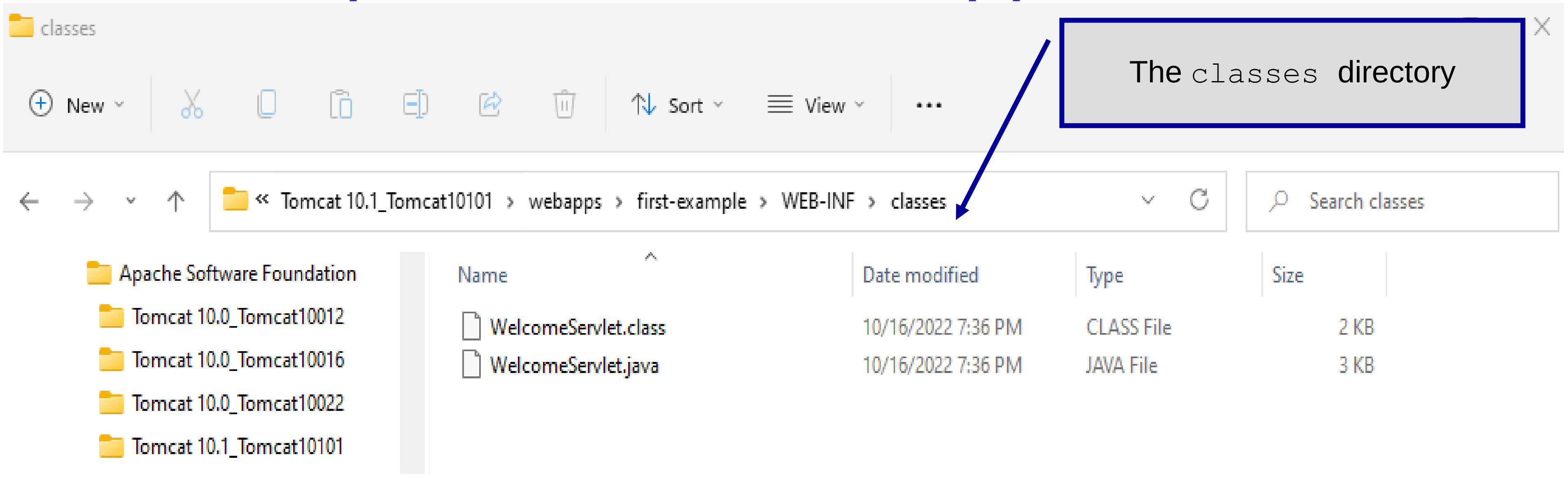


Top level of `WEB-INF`.

Name	Date modified	Type	Size
classes	10/16/2022 7:21 PM	File folder	
web.xml	10/16/2022 7:42 PM	XML Document	1 KB

8. Create the `WelcomeServlet.java` file as shown on page 63. When this file is compiled, place the `WelcomeServlet.class` file into 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.)



Name	Date modified	Type	Size
WelcomeServlet.class	10/16/2022 7:36 PM	CLASS File	2 KB
WelcomeServlet.java	10/16/2022 7:36 PM	JAVA File	3 KB

The `classes` directory

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:8082/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 75-76.