Lab 9: Web Application Server and WTP

In this lab, you will develop a web application using Apache Tomcat Web Application server. You must be wondering what is the difference between a Web Server and a Web Application Server, right? Your **Milestone 1 and 2** are, in fact, asking you to implement a **Web Server**; your **Milestone 3** will ask you to implement a **Web Application Server** by extending the Web Server you developed in Milestone 1 and 2. So, here is the high-level difference between them (adopted from stack overflow):

Both terms are very generic, one containing the other one and vice versa in some cases:

- **Web server**: serves (in most cases) static content to the web using http protocol
- **Application server**: hosts and exposes business logic and processes

The main point is that the web server exposes everything through the http protocol, while the application server is not restricted to it. Let's look at some more differences:

- Web Server is designed to serve HTTP Content. App Server can also serve HTTP Content but is not limited to just HTTP. It can provide other protocol support such as RMI/RPC

- Web Server is mostly designed to serve static content, though most Web Servers have plugins to support scripting languages like Perl, PHP, ASP, JSP etc. through which these servers can generate dynamic HTTP content

- Most of the application servers have Web Server as integral part of them, that means App Server can do whatever Web Server is capable of. Additionally, App Server have components and features to support Application level services such as Connection Pooling, Object Pooling, Transaction Support, Messaging services, etc.

- As web servers are well suited for static content and app servers for dynamic content, most of the production environments have web server acting as reverse proxy to app server. That means while servicing a page request, static contents (such as images/Static HTML) are served by web server that interprets the request. Using filtering technique (mostly extension of requested resource) web server identifies dynamic content request and transparently forwards to app server

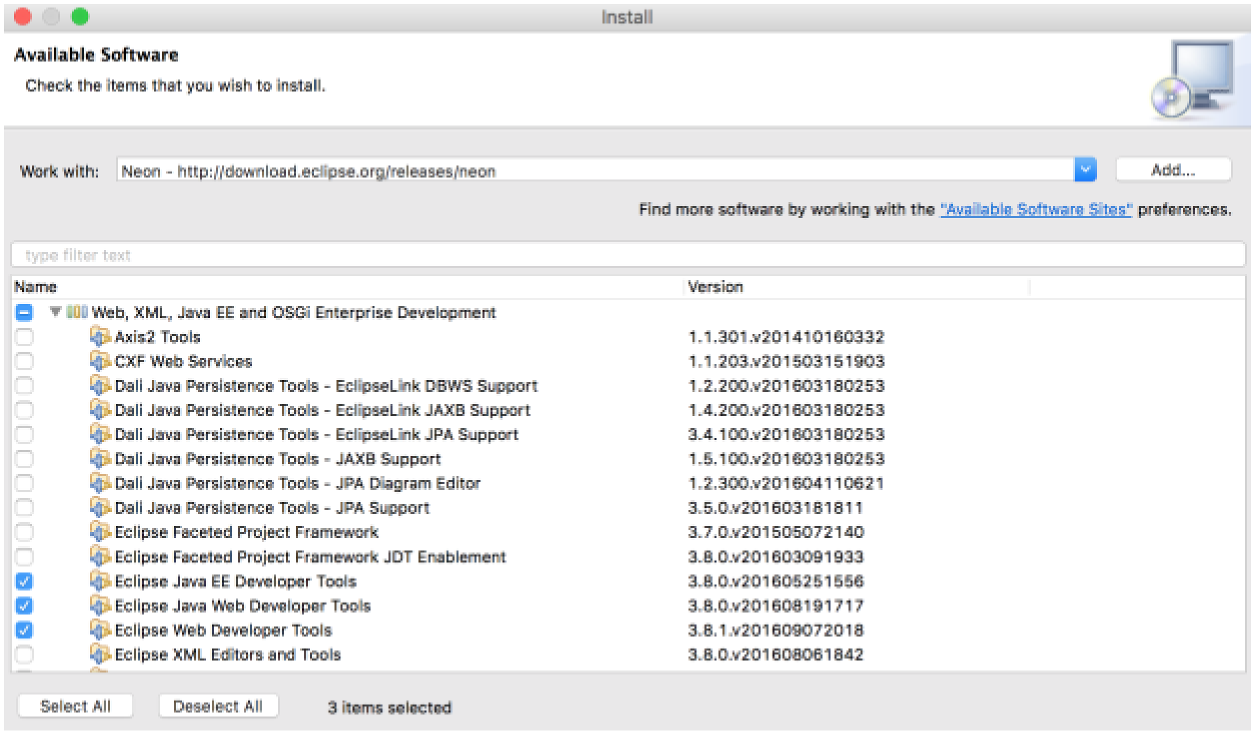
Now you know the difference between web server and web application server. To help you prepare for **Milestone 3,** let's learn how a web application server works by developing two simple web applications using it. In this lab, we will follow an **Eclipse Web Tool Platform (WTP)** tutorial. The tutorial is based on Eclipse Luna and Tomcat 7. We will, instead, use **Eclipse Neon and Tomcat 8.5** (latest) in this tutorial. Please find the necessary patches to the document below:

2. Tomcat Installation

Ignore the instruction for this section in the tutorial, follow the instruction here. Download the latest distribution of Tomcat (8.5.11). Extract it to a folder where you normally would extract other non-installed software.

3.1 Installation of WTP

For **Installation of WTP** section of the tutorial, choose **neon release update site** instead of Luna as shown below (note that JST packages are not shown here, but you should scroll down to select them):


Install

Available Software

Check the items that you wish to install.

Work with: Neon - http://download.eclipse.org/releases/neon

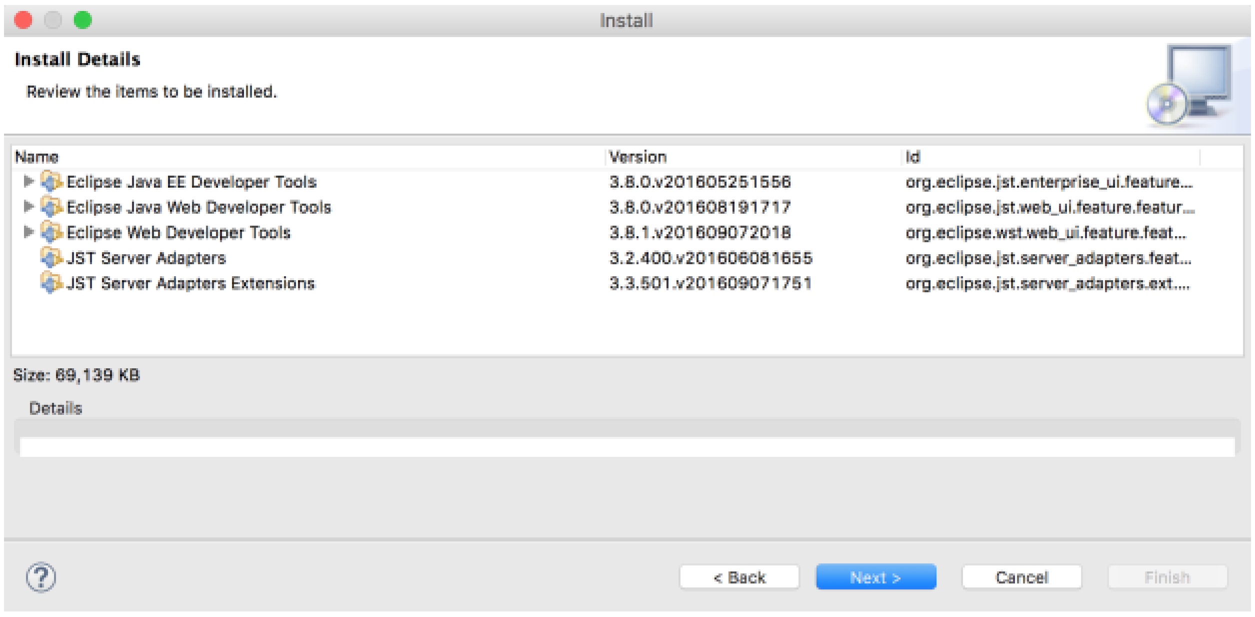
Find more software by working with the "Available Software Sites" preferences.

type filter text

Name	Version
Web, XML, Java EE and OSGi Enterprise Development	
☐ Axis2 Tools	1.1.301.v201410160332
☐ CXF Web Services	1.1.203.v201503151903
☐ Dali Java Persistence Tools - EclipseLink DBWS Support	1.2.200.v201603180253
☐ Dali Java Persistence Tools - EclipseLink JAXB Support	1.4.200.v201603180253
☐ Dali Java Persistence Tools - EclipseLink JPA Support	3.4.100.v201603180253
☐ Dali Java Persistence Tools - JAXB Support	1.5.100.v201603180253
☐ Dali Java Persistence Tools - JPA Diagram Editor	1.2.300.v201604110621
☐ Dali Java Persistence Tools - JPA Support	3.5.0.v201603181811
☐ Eclipse Faceted Project Framework	3.7.0.v201505072140
☐ Eclipse Faceted Project Framework JDT Enablement	3.8.0.v201603091933
✓ Eclipse Java EE Developer Tools	3.8.0.v201605251556
✓ Eclipse Java Web Developer Tools	3.8.0.v201608191717
✓ Eclipse Web Developer Tools	3.8.1.v201609072018
☐ Eclipse XML Editors and Tools	3.8.0.v201608061842

3 items selected

When you press **Next** you should see the following packages:


Install

Install Details

Review the items to be installed.

Name	Version	Id
Eclipse Java EE Developer Tools	3.8.0.v201605251556	org.eclipse.jst.enterprise_ui.feature...
Eclipse Java Web Developer Tools	3.8.0.v201608191717	org.eclipse.jst.web_ui.feature.featur...
Eclipse Web Developer Tools	3.8.1.v201609072018	org.eclipse.wst.web_ui.feature.feat...
JST Server Adapters	3.2.400.v201606081655	org.eclipse.jst.server_adapters.feat...
JST Server Adapters Extensions	3.3.501.v201609071751	org.eclipse.jst.server_adapters.ext....

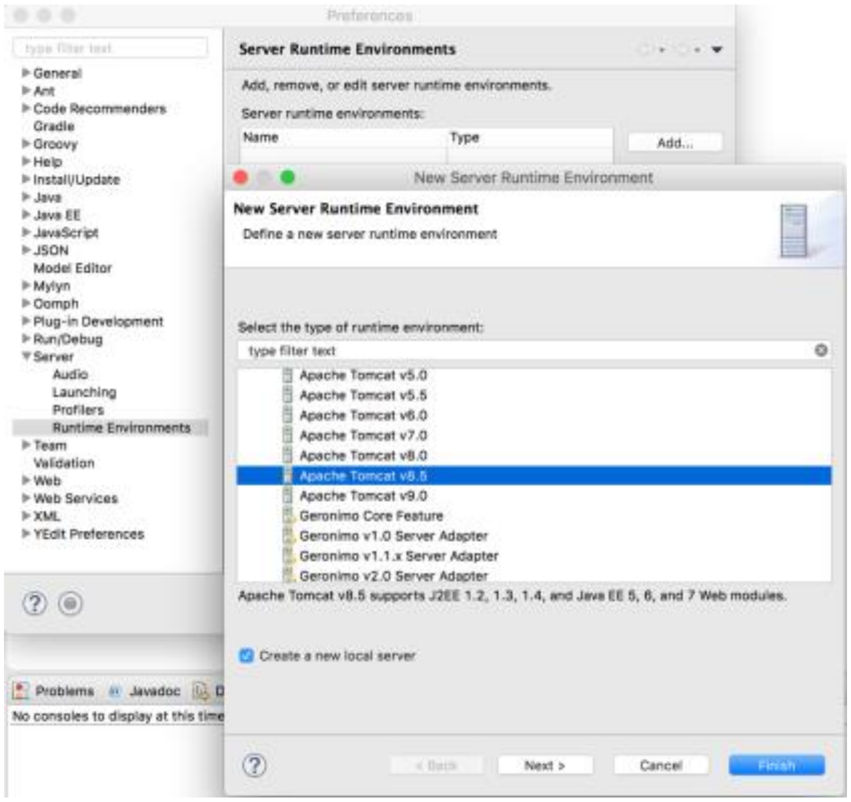
Size: 69,139 KB

Details

< Back | Next > | Cancel | Finish

4.1 Setting up runtime environments

Ignore the instruction for this section in the tutorial, follow the instruction here. Go to Windows -> Preferences -> Server Menu -> Runtime Environments -> **Add**. Select your version of Tomcat (8.5) and select the **Create a new local server** flag and click **Next**.


Preferences

Server Runtime Environments

Add, remove, or edit server runtime environments.

Server runtime environments:

Name | Type | Add...

New Server Runtime Environment

New Server Runtime Environment

Define a new server runtime environment

Select the type of runtime environment:

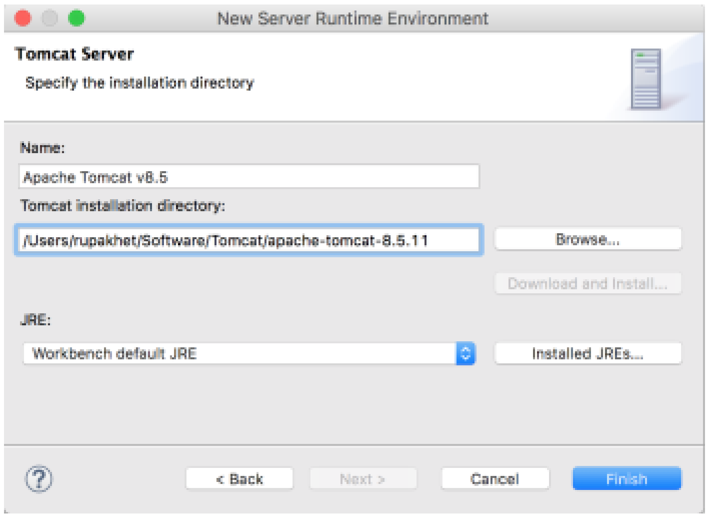
- Apache Tomcat v5.0
- Apache Tomcat v5.5
- Apache Tomcat v6.0
- Apache Tomcat v7.0
- Apache Tomcat v8.0
- Apache Tomcat v8.5
- Apache Tomcat v9.0
- Geronimo Core Feature
- Geronimo v1.0 Server Adapter
- Geronimo v1.1.x Server Adapter
- Geronimo v2.0 Server Adapter

Apache Tomcat v8.5 supports J2EE 1.2, 1.3, 1.4, and Java EE 5, 6, and 7 Web modules.

[✓] Create a new local server

No consoles to display at this time

Browse to the installation directory of the Tomcat as shown below and press **Finish**.


New Server Runtime Environment

Tomcat Server

Specify the installation directory

Name: Apache Tomcat v8.5

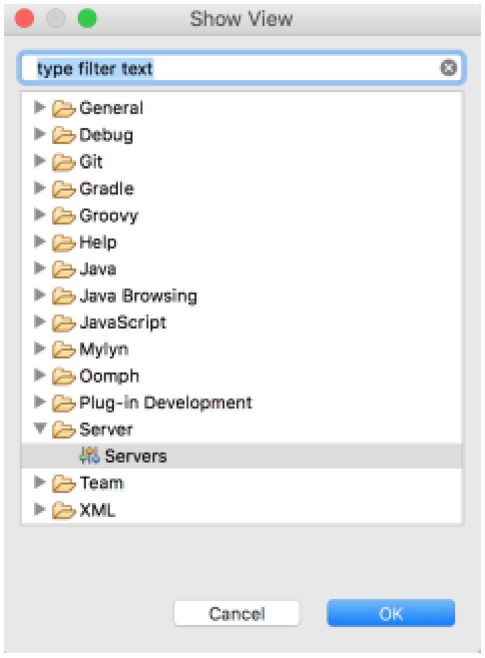
Tomcat installation directory: /Users/rupakhet/Software/Tomcat/apache-tomcat-8.5.11

JRE: Workbench default JRE

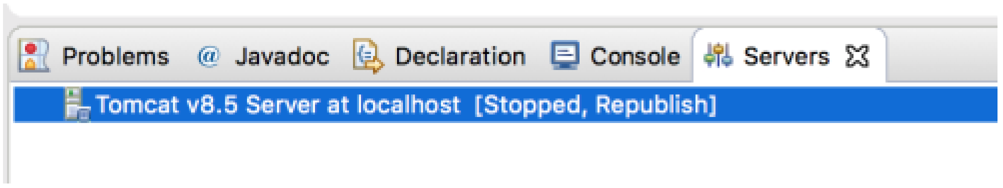
You are now ready to use Tomcat. Go to **4.2 Starting a Web Server**.

4.2 Starting a Web Server

Ignore the instruction for this section in the tutorial, follow the instruction here. Go to Windows -> Show View -> Others… -> Server -> Servers.

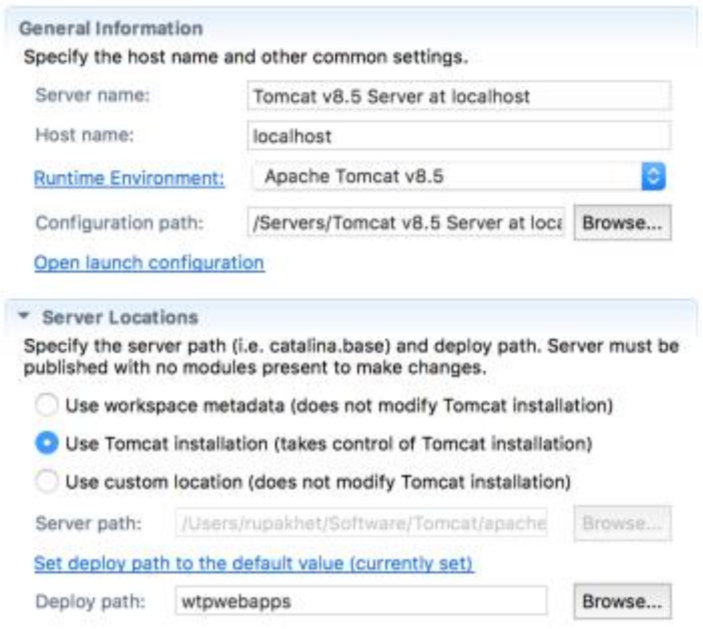


You should see the Tomcat Server in the **Servers view** as follows. Double click on the server to edit its configuration.



Tomcat v8.5 Server at localhost [Stopped, Republish]

Under **Server Locations**, instead of using *Use workspace metadata*, select **Use Tomcat installation** as shown below.



General Information

Specify the host name and other common settings.

Server name: Tomcat v8.5 Server at localhost

Host name: localhost

Runtime Environment: Apache Tomcat v8.5

Configuration path: /Servers/Tomcat v8.5 Server at loca

Open launch configuration

Server Locations

Specify the server path (i.e. catalina.base) and deploy path. Server must be published with no modules present to make changes.

- ○ Use workspace metadata (does not modify Tomcat installation)
- ● Use Tomcat installation (takes control of Tomcat installation)
- ○ Use custom location (does not modify Tomcat installation)

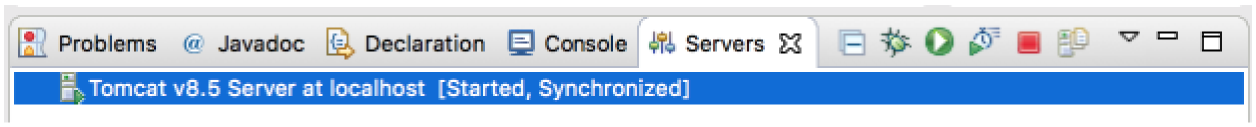
Server path: /Users/rupakhet/Software/Tomcat/apache

Set deploy path to the default value (currently set)

Deploy path: wtpwebapps

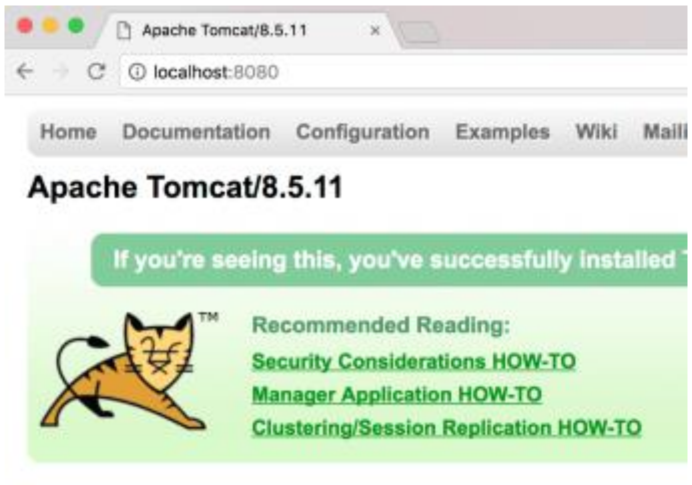
Make sure to save the configuration before moving on to the next step.

Start the server by pressing the **green run button** in the Servers view (shown below). To stop the server, you can press **the square red button**, but don't do that yet.



Tomcat v8.5 Server at localhost [Started, Synchronized]

Open a browser and go to the following url: http://localhost:8080/, you should see the landing page as shown below. Now your Tomcat web application server is ready for use.



Apache Tomcat/8.5.11

If you're seeing this, you've successfully installed

Recommended Reading:
Security Considerations HOW-TO
Manager Application HOW-TO
Clustering/Session Replication HOW-TO

Complete the rest of the tutorial. After you are done, **take two snapshots** of the web app in your browser showing the counter increments and add them to a pdf file.

Going Further

Explore all available methods in HttpServlet. In Milestone 2 of the project, you are asked to implement a basic support for HEAD, POST, PUT, DELETE for static contents. Let's understand how a web application server would do these things using the HttpServlet API. Create another dynamic web app project that implements the following features.

The best way to implement these features is to think about a servlet as the Façade to a file. Let's call the file: **file.txt**. (Also see the **Other Considerations** section for details on where to place this file on the server).

F1 – Basic Support for the GET Request

Your GET request should return the contents of **file.txt**. If the file does not exist, it should return 404 status code with right error message. When the Servlet initializes, you should create **file.txt** with a sample content for testing.

F2 – Basic Support for the HEAD Request

Your HEAD request will provide necessary info about **file.txt** without returning the content.

F3 – Basic Support for the POST Request

Your POST request will **create** a new **file.txt** if one does not exist or **append** the contents in the body of the request to **file.txt** if it already exists.

F4 – Basic Support for the PUT Request

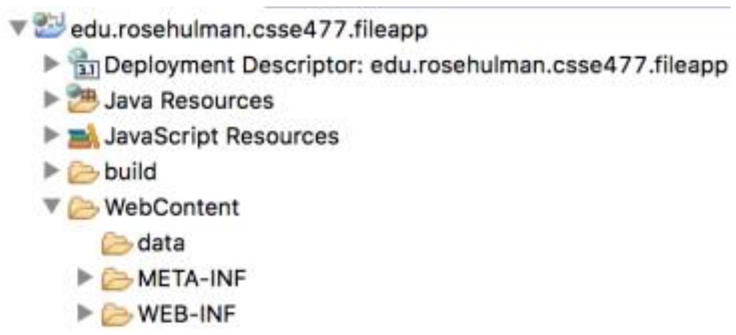
The PUT request must **create file.txt** if it does not exist or **overwrite** it with the content in the body (payload) of the request if it already exists.

F5 – Basic Support for the DELETE Request

Your DELETE request should delete **file.txt** if it already exists.

Other Considerations

Your web application should host a root directory to support the file operations. If you create a directory named **data** under the Project/WebContent directory in Eclipse as shown below, then the absolute path to the deployed **data** directory of the web app in the server can be acquired by calling the Java code that follows the picture. You can call this code in any methods [doGet() / doPost() / doPut() / doDelete()] of the servlet.


edu.rosehulman.csse477.fileapp
Deployment Descriptor: edu.rosehulman.csse477.fileapp
Java Resources
JavaScript Resources
build
WebContent
data
META-INF
WEB-INF

```
String dataDirPath = this.getServletContext().getRealPath("/data");
```

Important Note: When you save your changes in Eclipse project, it deploys an empty data directory to the server, so you will lose the previously written data.

Please handle exceptions and errors by properly returning the appropriate HTTP status code to the client.

Use Postman or Advanced REST Client to test these features. You may also use Google HTTP Java client or similar tools if you want to do these programmatically. In any case, take snapshots of request and reply that clearly demonstrate that these features are working as expected. Please note that you may have to make a GET request after each POST, PUT, and DELETE requests to show that they worked.

Deliverables

1. Bundle the two projects in one zip file and turn it in on Moodle.
2. **Separately** turn in the **pdf** file with the snapshots.