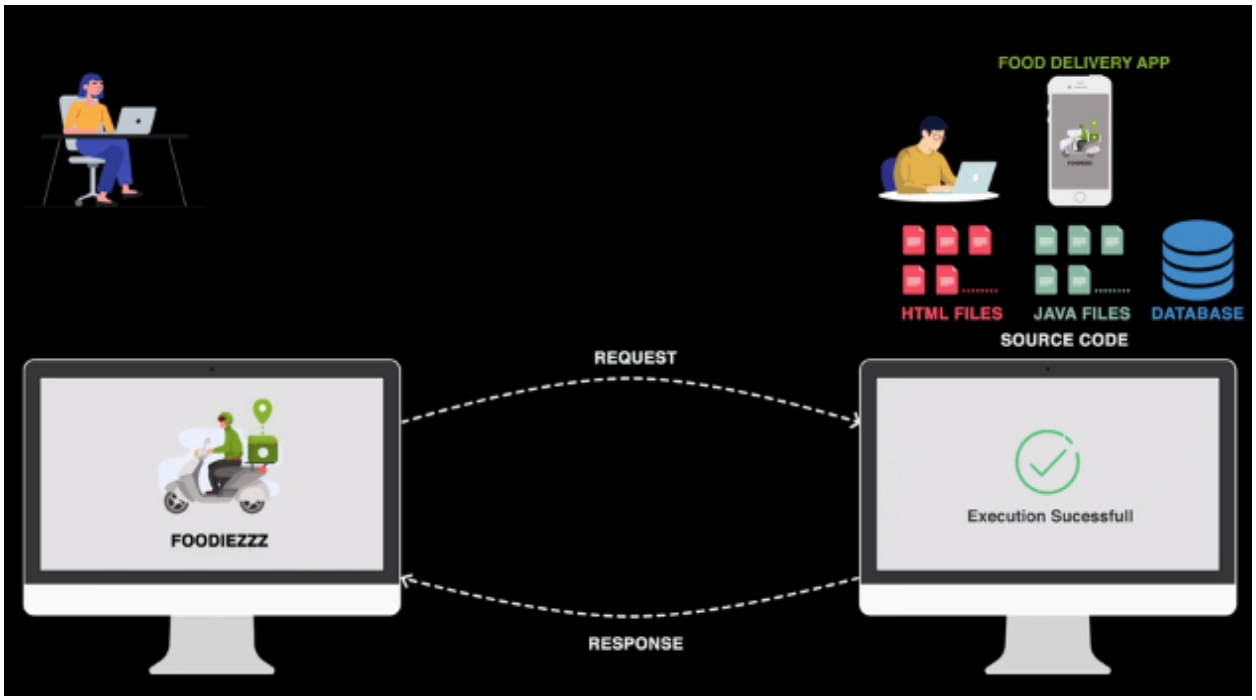
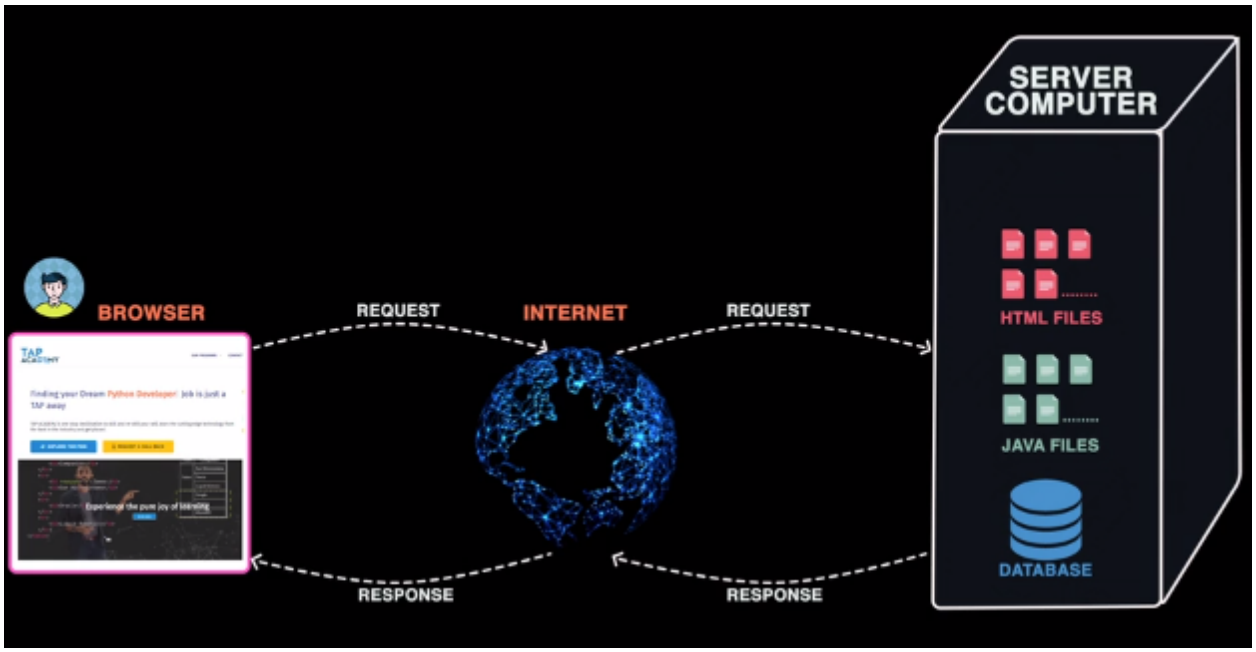
DAY 1

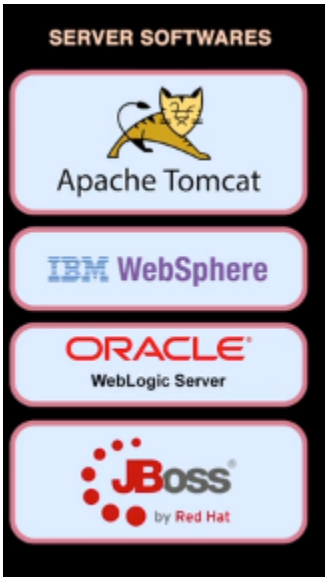
In the previous session we had seen the need of web applications, now let us see the difference between Stand Alone Applications and Web Applications through an example-
Let us assume that a developer has developed a web application called “Foodiezzz” and all the front end(HTML files), back end(Java files) and database are present in his computer where he can compile and execute the program and start using the application.
But if you want to use the application in your system, then you have to download and install the entire source code(front end, back end and database) in your computer, compile and execute it. Now this application would be a Stand Alone Application and not a Web Application. But the biggest disadvantage would be that the entire application should be present in the computer, if you want to access the application but in real life such applications would be a huge application with millions of lines of code and it would be impractical to download such huge application so the companies which develop the applications should take this burden and now the customers (client) should be able to send a request and now the based on the type of request, automatically compilation should happen in the developer’s computer and whatever is being requested for should be executed and the output should be displayed in the client’s computer. And such an application is called Web Application.



Web applications should be stored in a special type of computer known as Server Computer. User will have a browser installed in their computer and with the help of internet can send the request to the server computer and now the server computer should recognise the type of request and send the appropriate file as response



Any computer can be used as a server computer and for that we have to install a software called Server software and there are many options available in the market-



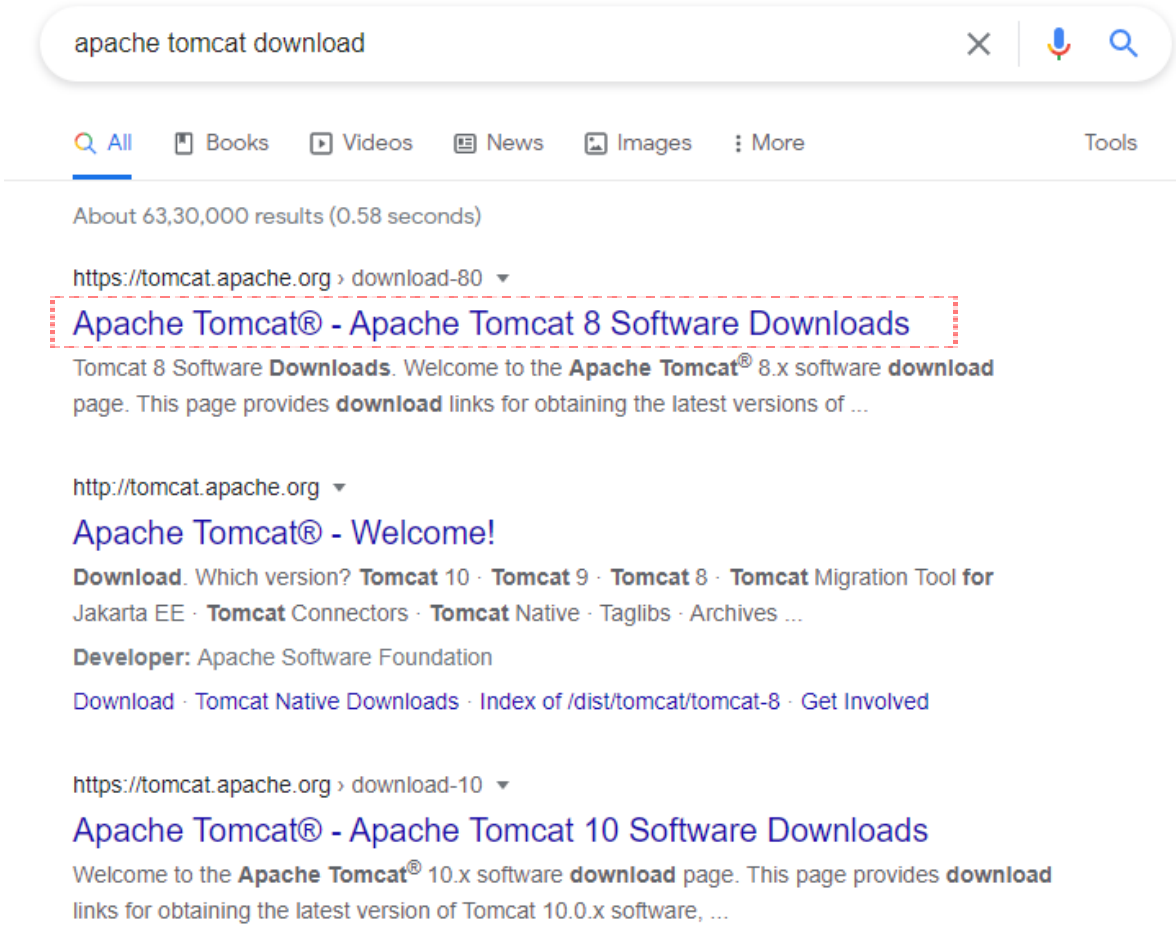
And we will be using Apache Tomcat server for two main reasons-

- Open Source
- Free of cost

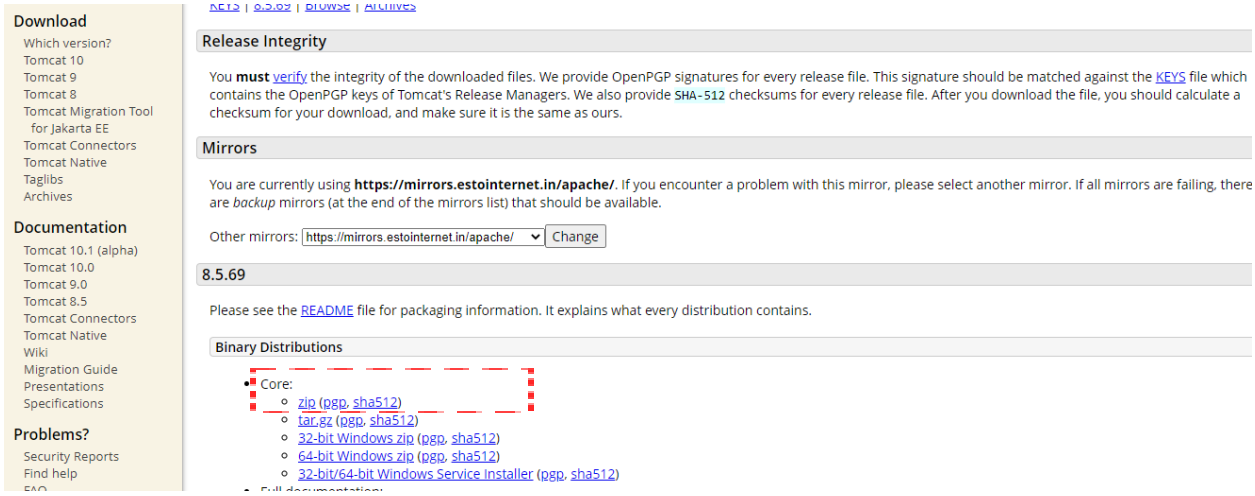
The most important part of a server software is called Web Container and A web container is **the component of a web server that interacts with Java servlets**.Inside this container we have to place all the files which must accept a request and send back the response.

Let us see how to install Apache Tomcat server -

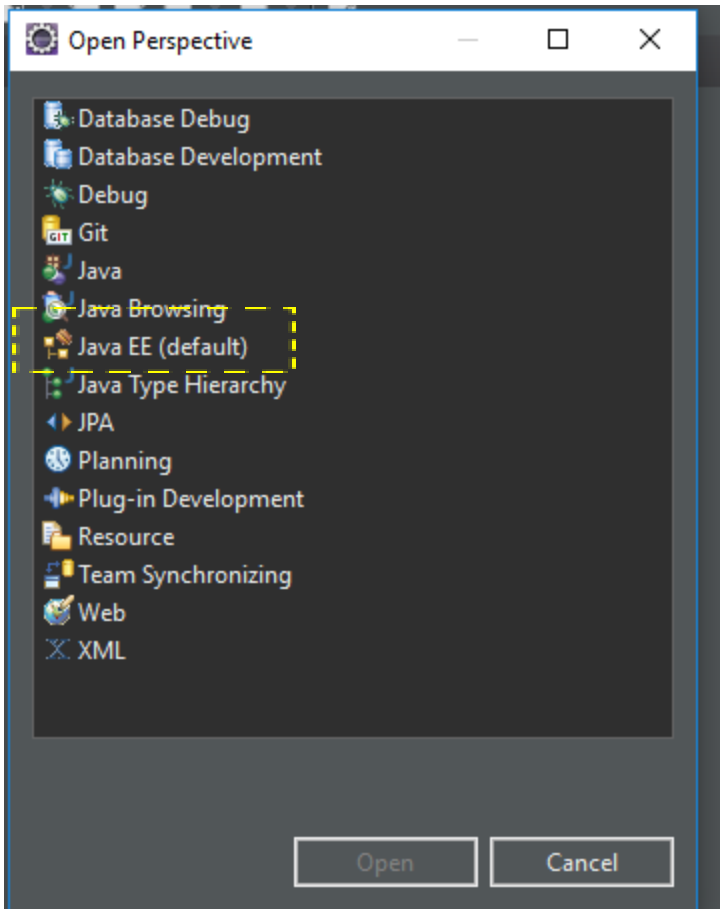
1. Go to google and search “Apache Tomcat download” and click on the 1st link



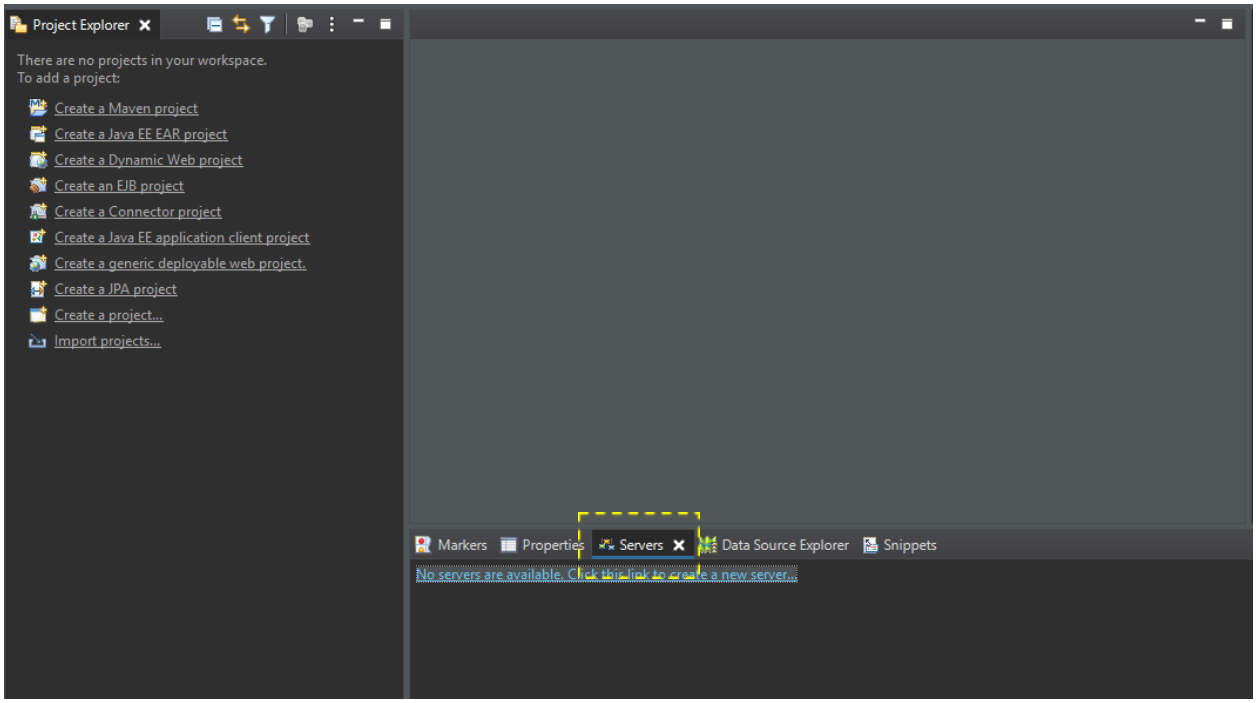
2. We will be using Tomcat version 8. Click on Tomcat 8 and then click on zip file to download



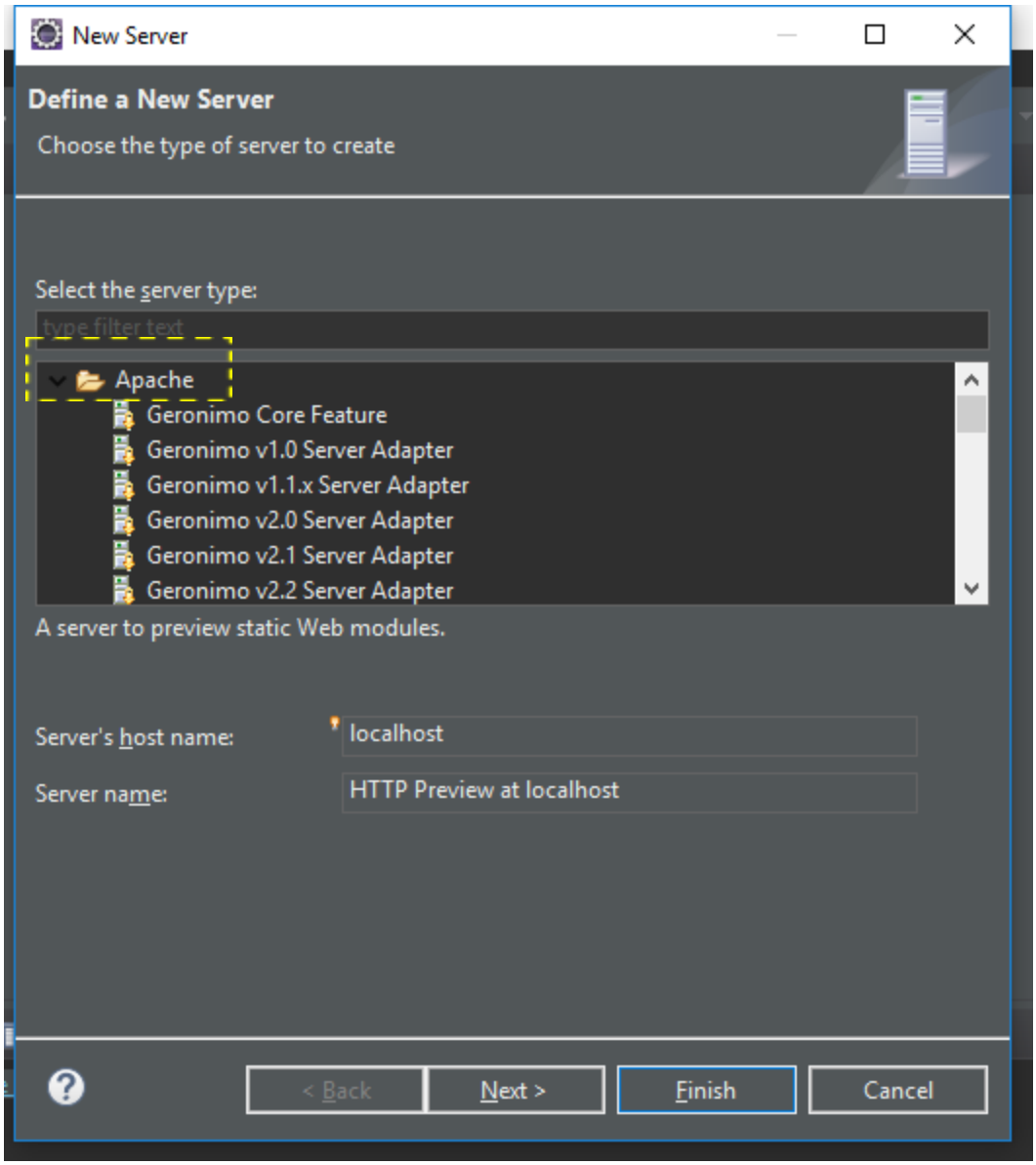
3. Extract the zip file
4. Open Eclipse and click on open perspective. Select Java EE and click on “Open”.



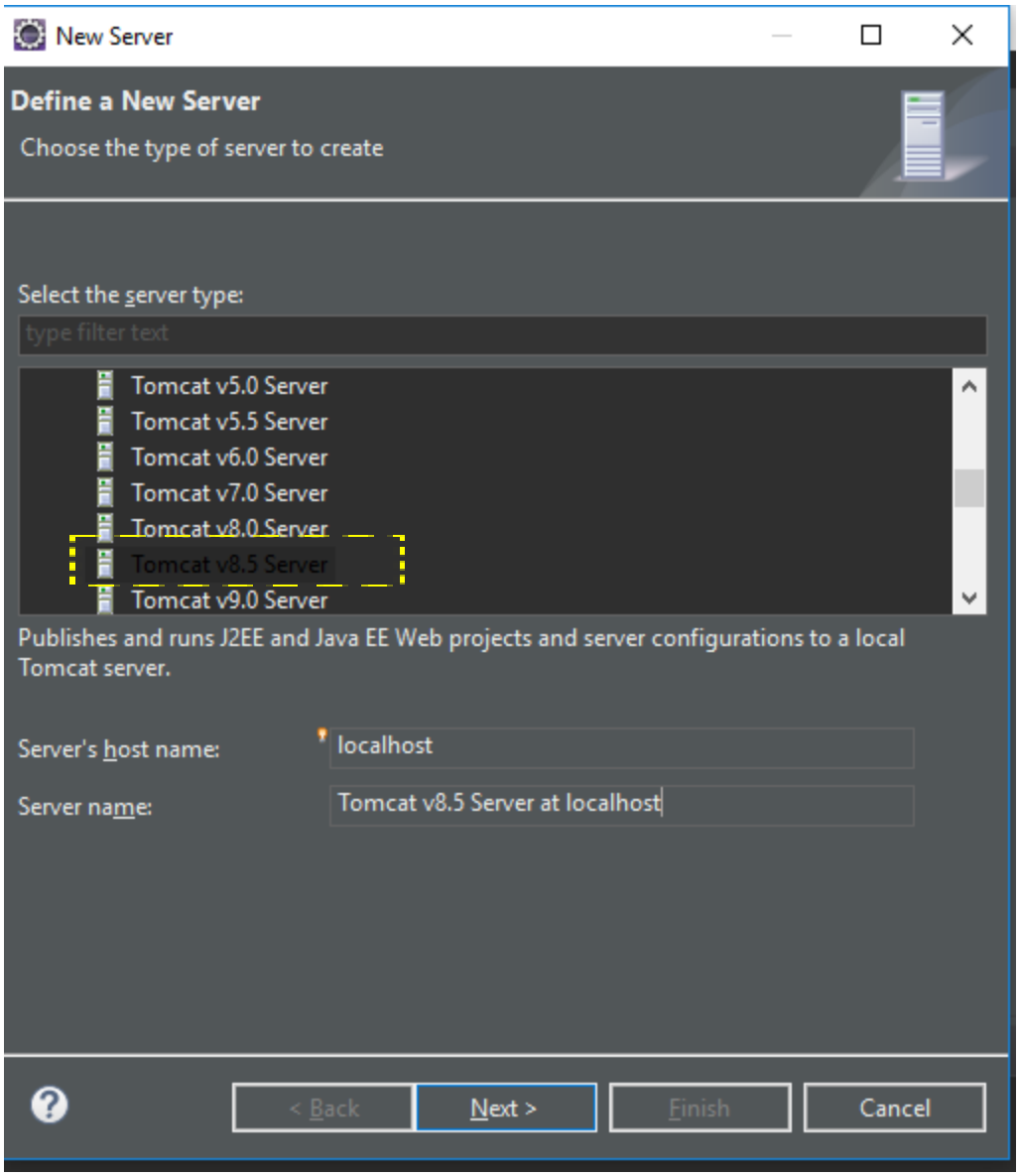
5. Click on Server tab below



6. Click on “No servers are available. Click this link to create a new server...” link
7. Now select “Apache” in the pop up

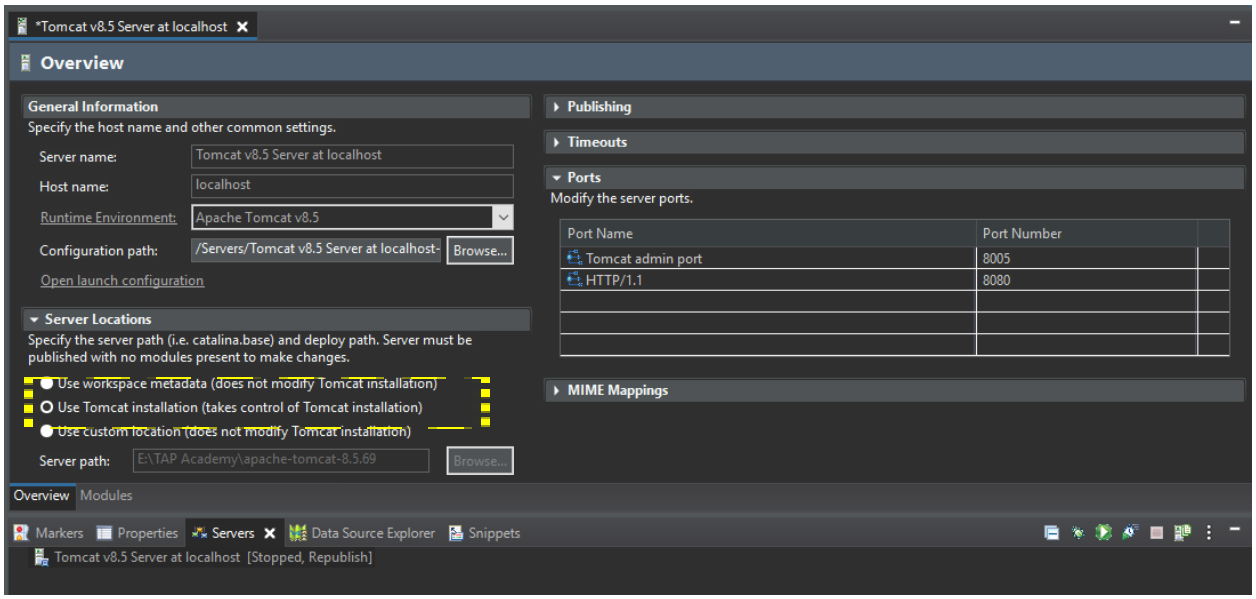


8. Scroll down and click on “Tomcat v8.5 Server”

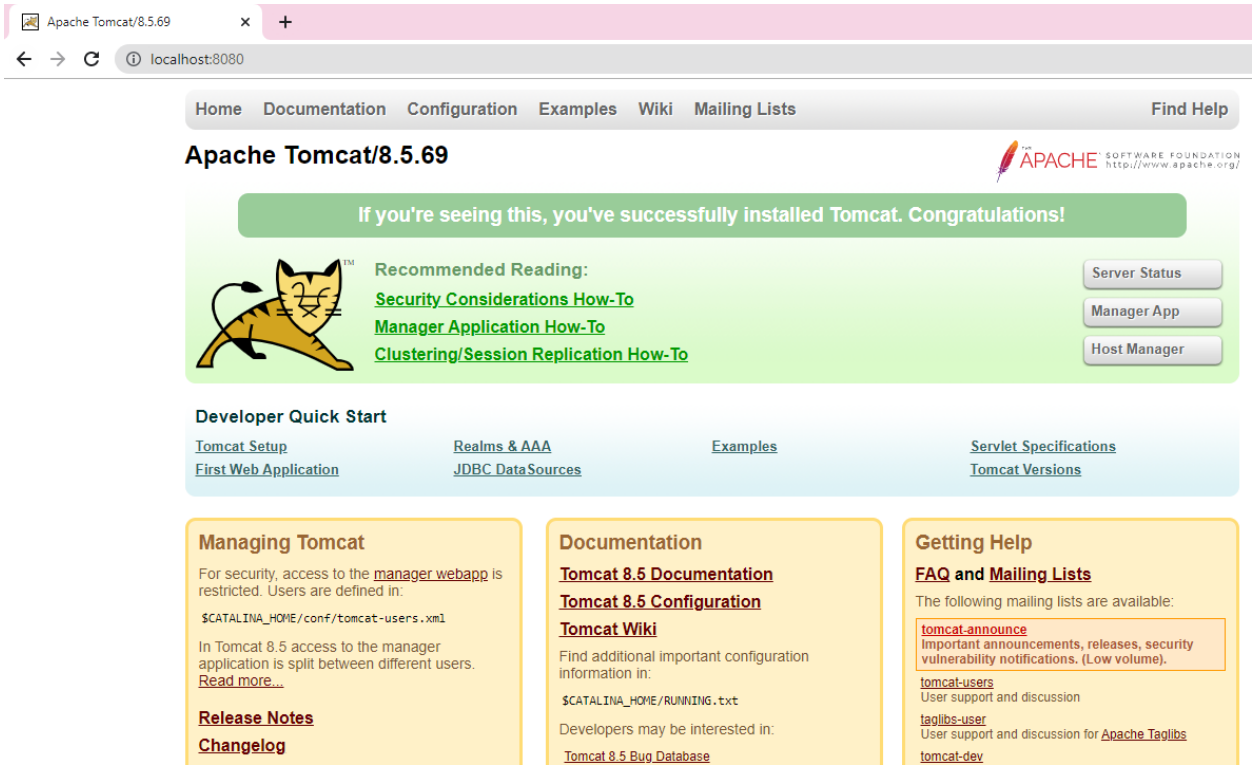


9. Click on Next and select the folder in which you have downloaded the tomcat server.
10. Click on Finish.

How to check whether the server was installed successfully or not?
Any process executing on the RAM will have a Process ID or Port Number and the Port Number of Tomcat Server is by default **8080.**
Double click on the server in eclipse and select the 2nd option in the server locations tab and save the file.

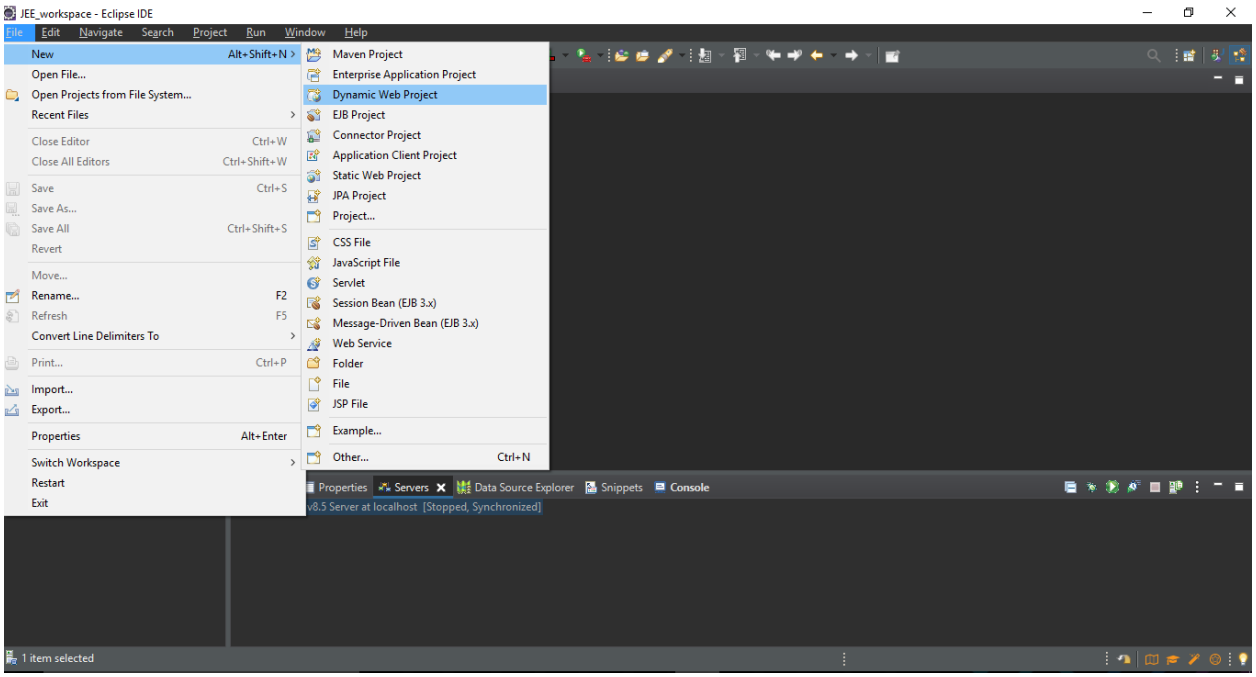


Now right click on the server and select ‘start’.
Open the browser and type “http://localhost:8080” in the search bar and press enter. If the page below is displayed, then Tomcat is installed successfully.

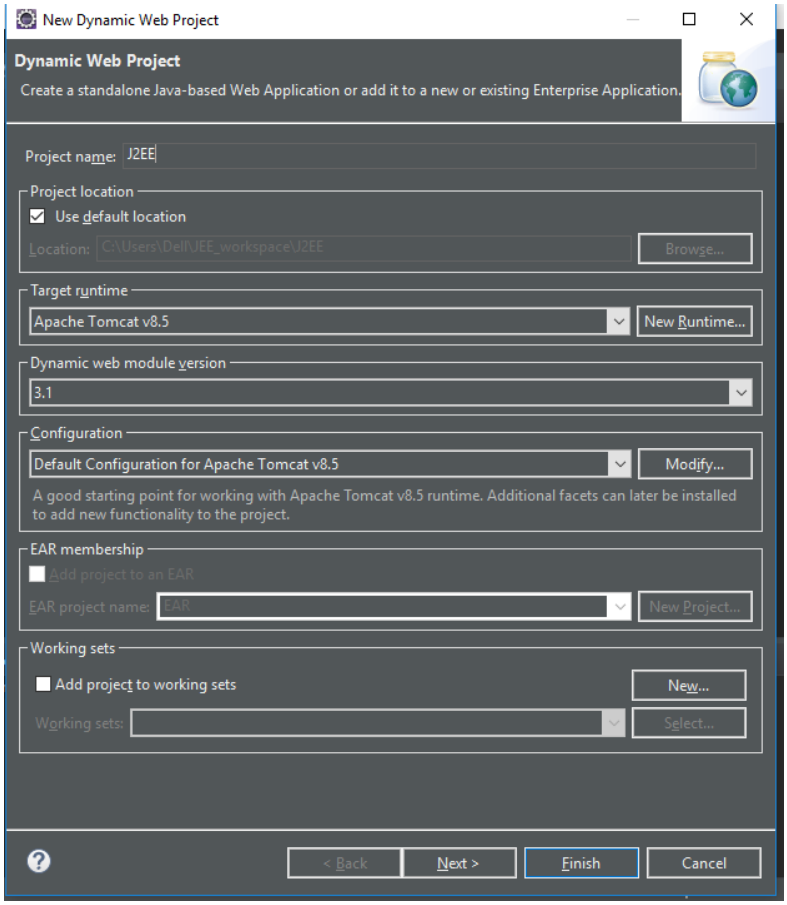


Now let us create a web application from scratch. We will be creating Dynamic Web project in Eclipse
Steps to create Dynamic Web Project:

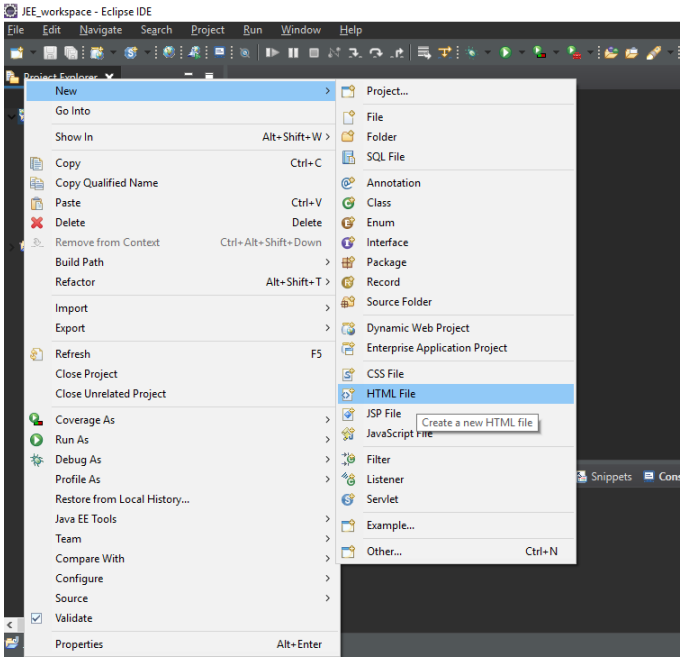
1. Launch Eclipse and click on file and select “Dynamic Web Project”.



2. Set project name to “J2EE” and click on next



3. Click on Next
4. Click on Next and Click on Finish.
5. Right click on the project and click on new and select HTML file



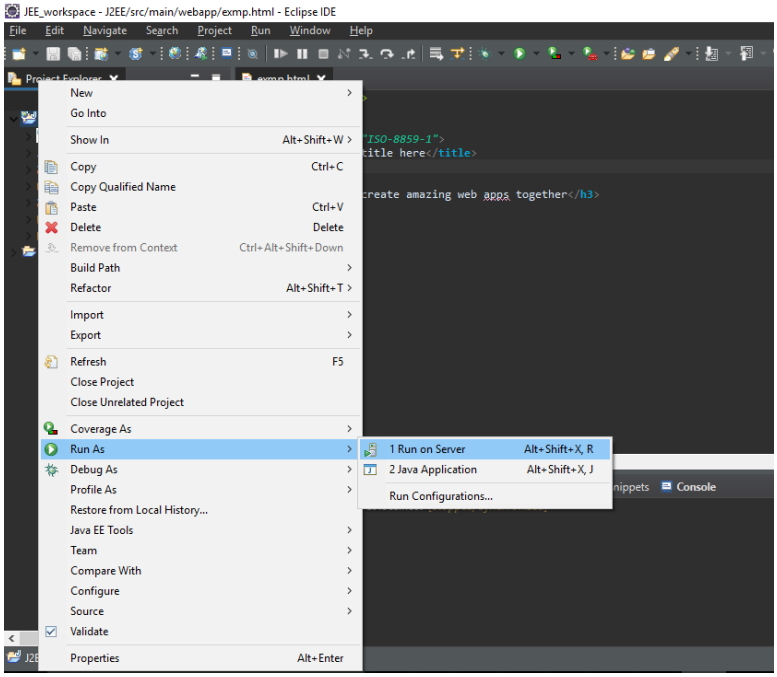
6. Set the name as ‘exmp.html’ and click on Finish.

   The HTML file is be present in WEB-INF folder and this is the project structure-

   In our server computer we have installed Tomcat and every server has a web container and Catalina is the name of the Tomcat server’s Web Container and the port number of Catalina is 8080.
   The dynamic web project which we have created is “J2EE” and this should be loaded into the web container(Catalina) and only then the project is accessible to the client and technically adding this project to Tomcat is called **Deployment**.

   How to deploy our project?

   - Right click on the project and click on “Run as” and click on “Run on Server”



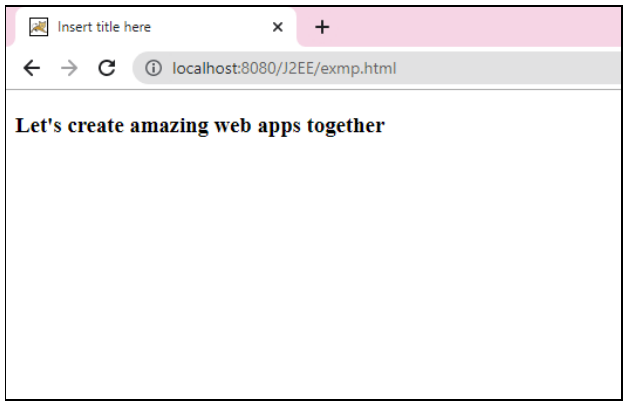
- Now click on Next, now into our Tomcat server our project J2EE has been added.
- Click on finish

So let us see how exactly this works-
In our server computer, Tomcat's web container i.e.,Catalina is installed and inside the web container, we have added our project J2EE. J2EE project contains an HTML file, exmp.html.
Now, if anyone wants to access exmp.html then, all they need is a computer in which a browser should be installed, with an active internet connection. Since our client and server are present in the same computer, internet connection is not mandatory.
If the client has to find the server computer with the help of URL(Uniform Resource Locator) and we are going to use a HTTP protocol, where HTTP stands for HyperText Transfer Protocol and protocol is nothing but the procedure we have to follow one must follow in order to find the server computer.
In the URL bar of the browser we have to enter the following -

http://192.167.1.2:8080/J2EE/exmp.html

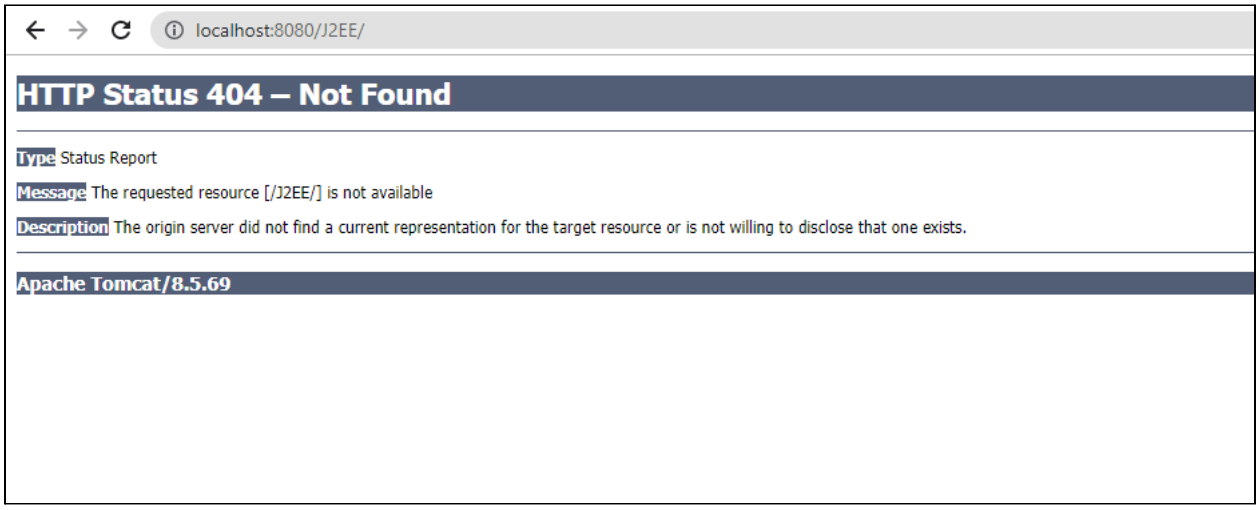
Where -
http: Protocol
192.167.1.2: IP address
8080: Port number
exmp.html: Resource(HTML file)

And when we press enter, we get the following output-



As you can see, we have used “localhost” instead of the IP address, which is nothing but the name given to the IP address by DNS(Domain Name System) DNS will convert the name to IP address.

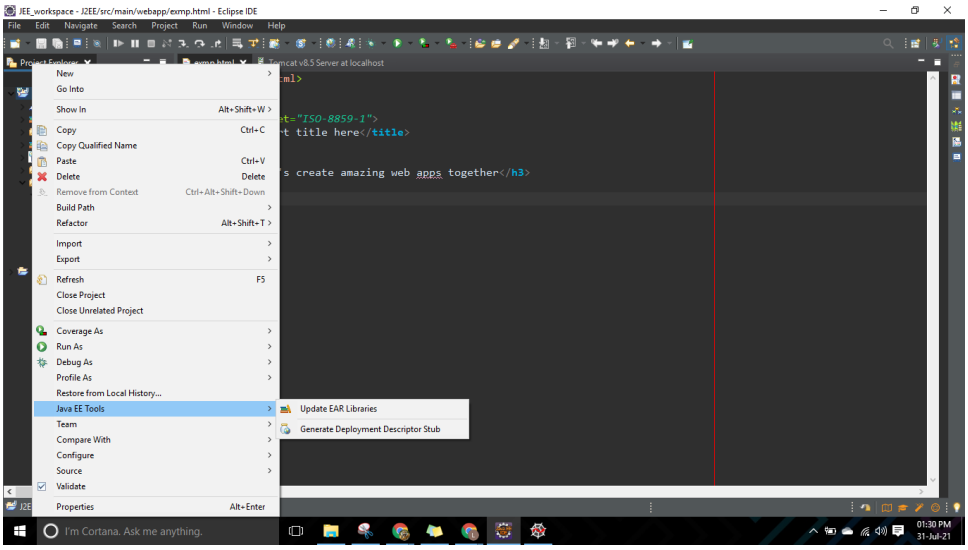
Example 2:
Whenever we enter a URL in our browser, then by default the Home page of that particular web page will be displayed. But in our program if we just enter “http://localhost:8080/J2EE”, we get the following error-



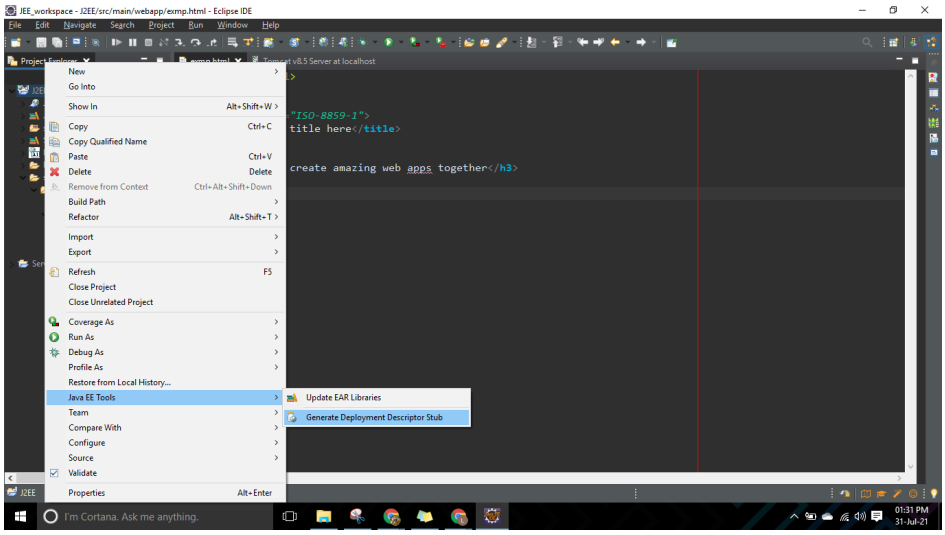
We don’t want such error messages to be displayed, instead if the user does not request “**exmp.html**” then by default another html file should be displayed.

How to achieve that?
We are going to use a xml file called ‘web.xml’ which is called **Deployment Descriptor** .
XML files are files which contain **tags,** and these tags are used for variety for purposes such as in eclipse they are used for deployment purpose, i.e., to find resources.
Steps to create Deployment Descriptor(web.xml):

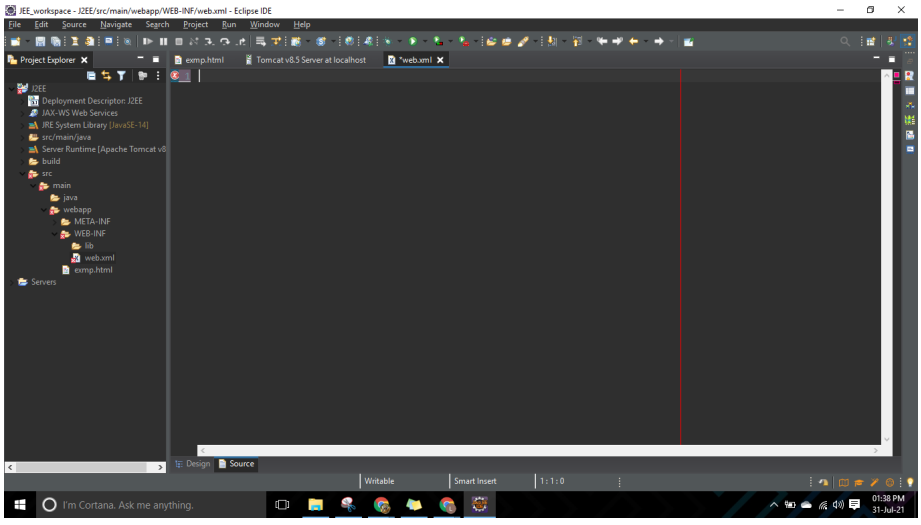
1. Right click on the project and select Java JEE tools



2. Click on “Generate Deployment Descriptor Stub”



3. “web.xml” file will be created in **WEB-INF** folder, under **lib** folder



4. Create a html file and name it as **'index.html'.** This is the file which should be automatically sent, if user does not explicitly request for **exmp.html**

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Home Page</title>
</head>
<body>
	<h3>A Web Application consists of 3 layers:</h3>
	<ul>
		<li>Front-End</li>
		<li>Back-End</li>
		<li>Database</li>
	</ul>
</body>
</html>
```

5. Now type in the following code inside web.xml

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

Now, whenever user does not request explicitly for a file, then automatically the request is sent to **web.xml,** in which we are redirecting to **index.html**