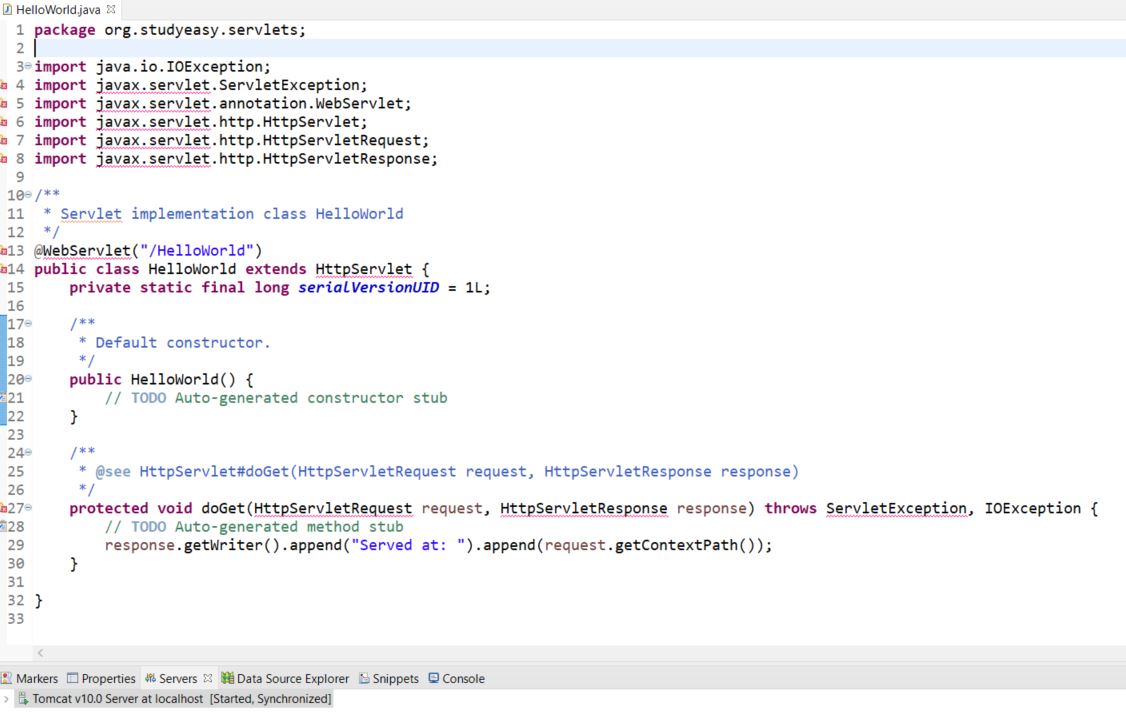
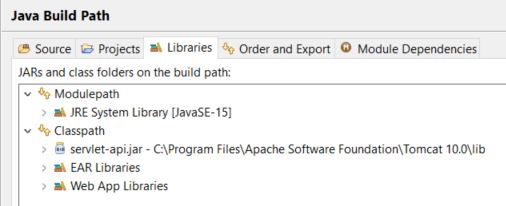
FAQ list

**Q1.** As of March 2021, I have installed Eclipse 2020-12 with JDK15 and Tomcat 10. I am unable to get rid of the initial errors even that I have added the servlet-api.jar and tomcat is running.

****

**Can't run Hello Servlets project.**

****

****

**Ans:** Firstly, change all the import statements from javax to jakarta (because Tomcat 10 uses Jakarta based requests)

Then add the following code in your Hello.java file

1. import jakarta.servlet.ServletException;
2. import jakarta.servlet.annotation.WebServlet;
3. import jakarta.servlet.http.HttpServlet;
4. import jakarta.servlet.http.HttpServletRequest;
5. import jakarta.servlet.http.HttpServletResponse;
7. /\*\*
8. \* Servlet implementation class Servlets
9. \*/
10. @WebServlet("/Hello")
11. /\*This is the name of my servlet file (check in your case it might be of different name) \*/
12. public class Hello extends HttpServlet {
13. private static final long serialVersionUID = 1L;
15. /\*\*
16. \* Default constructor.
17. \*/
18. public Hello() {
20. }
22. /\*\*
23. \* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
24. \*/
25. protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
26. response.getWriter().println("<h1>Hello World!</h1>");
27. }
29. }

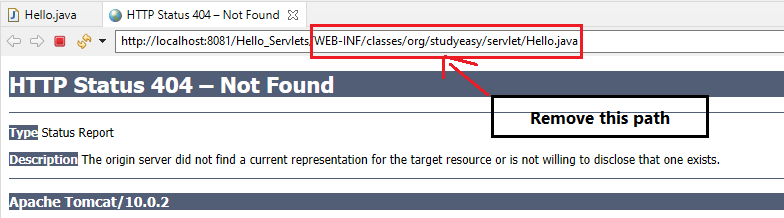
After adding the above code restart Eclipse IDE.

Executing a project in Tomcat 10 is slightly different as compared to previous Tomcat versions.

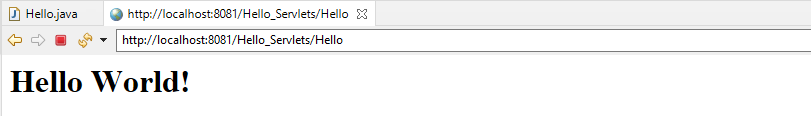
**Steps:**

1. First select the filename and right-click to run on the server

2. Then you will get an error 404(the reason for it to happen is that tomcat doesn't know the path of your file, you need to explicitly(manually) provide the path).

3. Below is the error page image same as yours

4. Now provide the path manually, follow the below image

[](javascript:void(0))

5. After you mentioned the path, press Enter

**Q.2. failed to create a maven project** could not resolve the archetype web app from any of the configured repositories.

**Answer:** It seems like Maven tries to find the jar for the relative webapp and when it cannot find it, it throws this error. As per the documentation available Maven should install all the required files/jars automatically, but for some reason, it can't in several cases.

Download “maven-archetype-webapp-1.0.jar" from below link,

Link: <https://mvnrepository.com/artifact/org.apache.maven.archetypes/maven-archetype-webapp>

and placed it at the following location in “.m2 folder”,

**.m2>\.m2\repository\org\apache\maven\archetypes\maven.archetype.webapp**

Alternatively, you can add the maven archetype catalog to eclipse. Steps are provided below:

1. Open Window tab
2. Preferences - > Open Maven
3. Archetypes Click 'Add Remote Catalog’
4. And add the following:

Catalog File: <http://repo1.maven.org/maven2/archetype-catalog.xml>

Description: maven catalog

**Q.3**.An unknown version of tomcat was specified error

**Answer:** As per the question, it seems like it is because of the path issue.

Reason: Unknown version of tomcat was specified

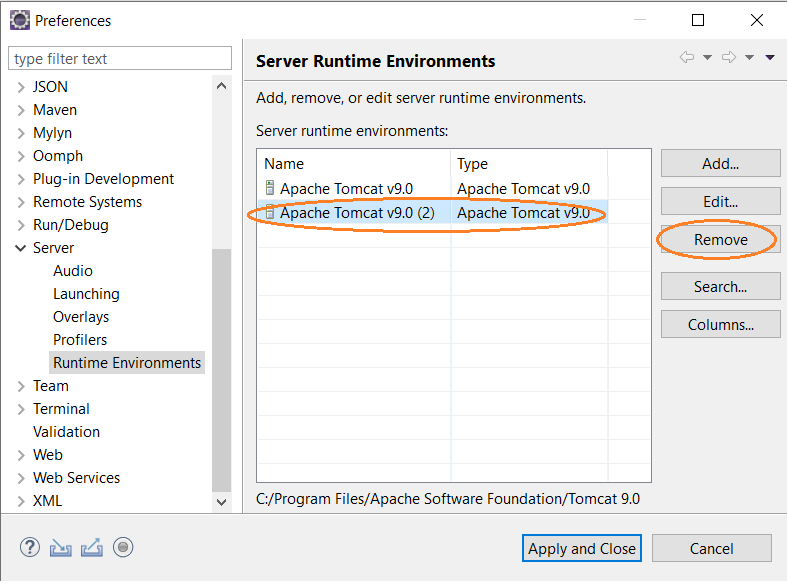
I’m assuming that you are using the latest Tomcat version v9.0. So, delete the previous Tomcat server installed, from the server tab, also delete the **Servers** folder in Project Explorer.

**Note:** Tomcat Installation directory field is meant to be pointed at your Tomcat base directory.

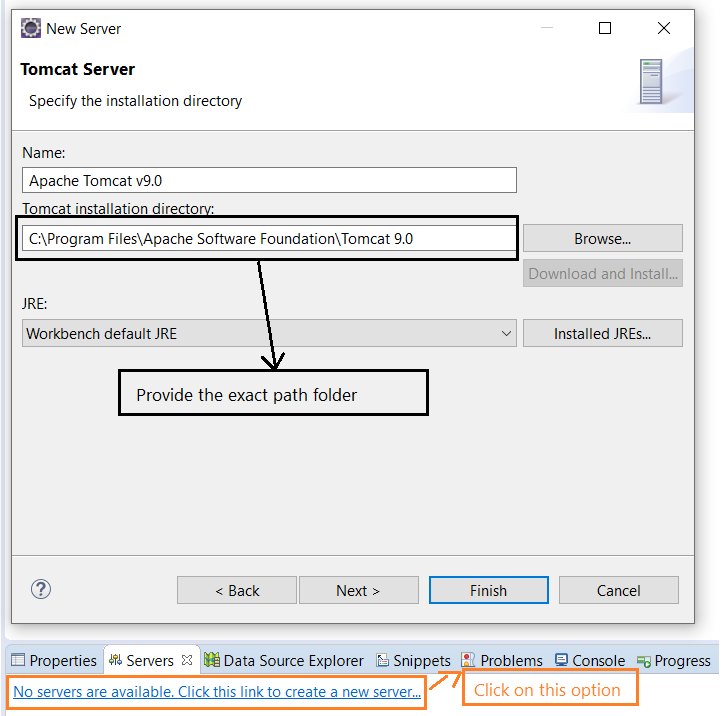
Steps are provided for your convenience.

**Steps:**

1. Remove the previously installed server from the **Runtime Environments** tab.

****

1. Click on Apply and Close option.
2. Add New Server from the **Server** tab.

****

1. Click on Finish.

Below is a YouTube link that you can take reference.

Link: <https://youtu.be/iheKhnk7fGM>

**Q.4.** Dependency error for mapping results.

**Answer:** If you are using any of the versions of Alpha jar, there might be a chance of dependency issues.

Like**, “hibernate-core-Alpha6-release”** jar file from the maven repository.

Switch to “**hibernate-core-5.4.20.Final”** and add this jar file externally.

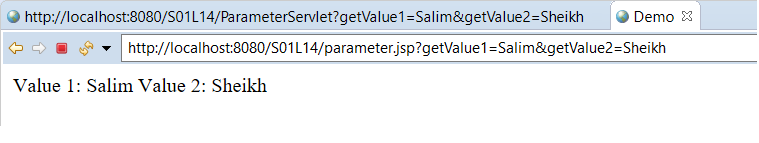
Download link is provided below,

Link: <https://mvnrepository.com/artifact/org.hibernate/hibernate-core/5.4.20.Final>

**Q.5.** Input params are not working in eclipse

**Answer:** This is because whenever you run a servlet or JSP program in this example; you have to add “**get parameters**” to get information from the servlet and JSP file.

I have reprogrammed this code and provided a screenshot for your reference,



**Note:** The reason is if you don’t enter the “**get parameters**” on the URL, you won’t be able to fetch information from both servlet and JSP files.

Otherwise, the output will show you null values.

**Q.6.** Jersey version 3.0.0-M6 and tomcat-9 shows HTTP 404 not found error

**Answer:** The reason is with dependency and the version.

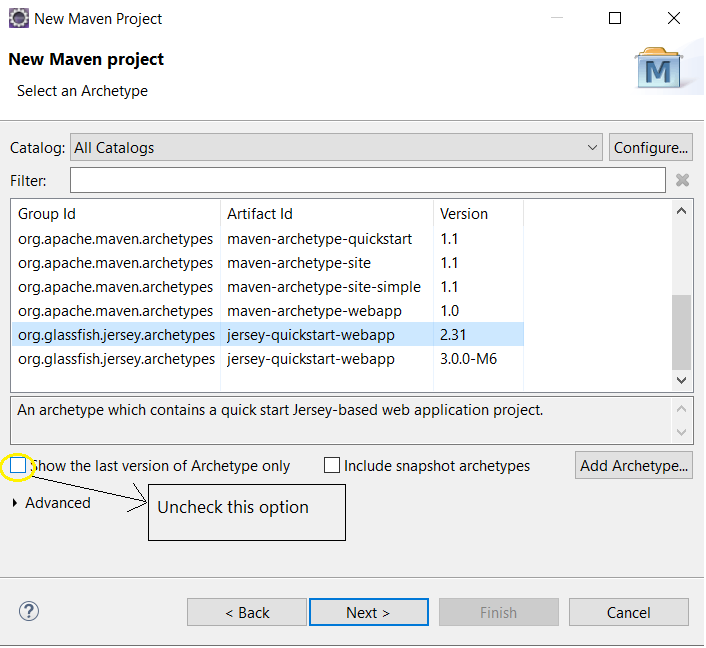
I was using the same Jersey version “3.0.0-M6” but the project doesn’t work.

As **Tomcat 9** uses “javax.servlet.Filter”, whereas for **Jersey 3.x**, which is part of Jakarta EE 9 and depends on **Jakarta** based API packages “jakarta.servlet.Filter” is available in **Tomcat 10**.

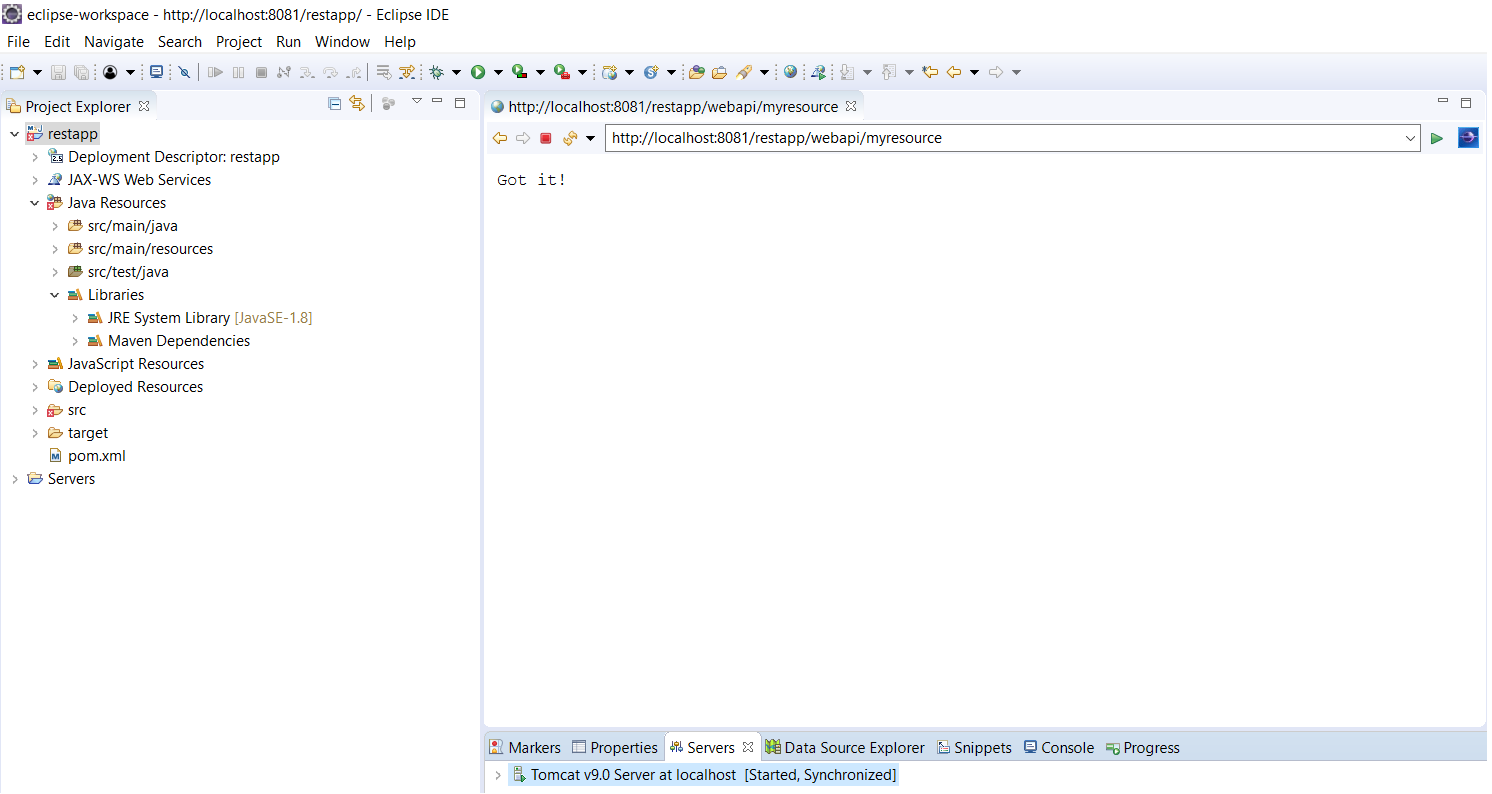
If you are using Tomcat 9, use **Jersey 2.x** and if you want to use Jersey 3.x with Servlets, you need to use a servlet container that implements Servlet 5 API, e.g. Tomcat 10, Jetty 11, Glassfish 6.

So, I have used another Jersey version “2.31” and this works fine for this project.

**Note:** Whenever you add a new Archetype, remember to uncheck the option shown in the below image for showing all the “quickstart Jersey-based web application” projects.



Here is an output of this project.



As per the question, It is recommend using **Jersey 2.x** versions as these are stable versions and you can implement this project using this version Archetype.

**Q.7.Exception:**  java.lang.ClassNotFoundException :org.springframework.web.servlet.DispatcherServlet.class

**Answer:** You need to add the "Maven Dependency" in the Deployment Assembly

**Steps:**

1. Right-click on your project and choose properties.
2. Click on Deployment Assembly.
3. Click on Add
4. Click on "Java Build Path Entries"
5. select Maven Dependencies"
6. Click on Finish.

Rebuild and deploy again.

Most probably the related jar files are not loaded and deployed on tomcat startup. These files are in your classpath and hence you are not getting any error in Eclipse IDE during development time, it happens only during runtime.

Steps are provided in the below link, which might help you to resolve this problem.

Credits: <https://crunchify.com/how-to-fix-java-lang-classnotfoundexception-org-springframework-web-servlet-dispatcherservlet-exception-spring-mvc-tomcat-and-404-error/>

**Q.8.** Error: Could not find or load main class com.App

**Answer:** According to the error message ("Could not find or load main class"), there are two categories of problems:

1. Main class could not be **found**
2. Main class could not be **loaded**

The main class could not be **found** when there is a **typo or wrong syntax in the fully qualified class name** or it **does not exist in the provided classpath**.

The main class could not be **loaded** when **the class cannot be initiated**, typically the main class extends another class and that class does not exist in the provided classpath.

**Note:** It is found that the basic syntax errors and cases are the root causes of such problems.

Clean your entire Build Path and everything you ever added to it manually. This includes additional sources, Projects, Libraries.

**Steps:**

1. Project -> Clean
2. Make sure Project -> Build automatically is active
3. Project -> Properties -> Java Build Path -> Libraries: Remove any external libs you have ever added.

**Don't remove standard libraries like the JRE System Library.**

1. Try to run your main class now.
2. Try adding your external libs/jars one after each other.

The reason behind this: The compiler had issues linking the libraries to the project. It failed and produced a wrong error message.

If this is still not working, have a look at the **built-in ErrorLog of Eclipse**:

* Window -> Show View -> General -> Error Log

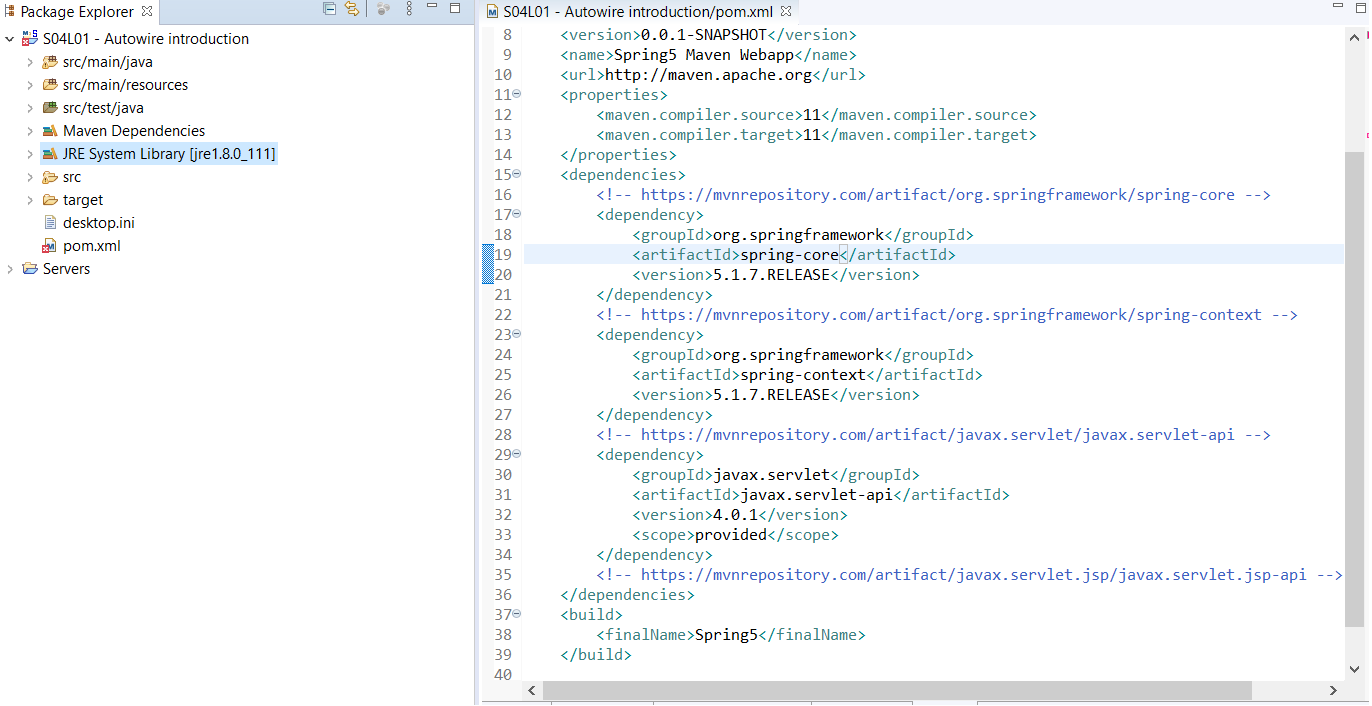
To see, where the code is throwing such error.

**Q.9.** While updating the project, can’t update the JRE System Library to a specified JavaSE version.

**Answer:** The reason is that you might have not downloaded and installed **JDK 11 version on your system.**

Here is the download link: [https://www.oracle.com/java/technologies/javase-jdk11-downloads.html#license-lightbox](https://www.oracle.com/java/technologies/javase-jdk11-downloads.html%23license-lightbox)

Below are the steps you need to follow to add/update the default **JRE System Library** to [JavaSE-11] after updating the maven project:

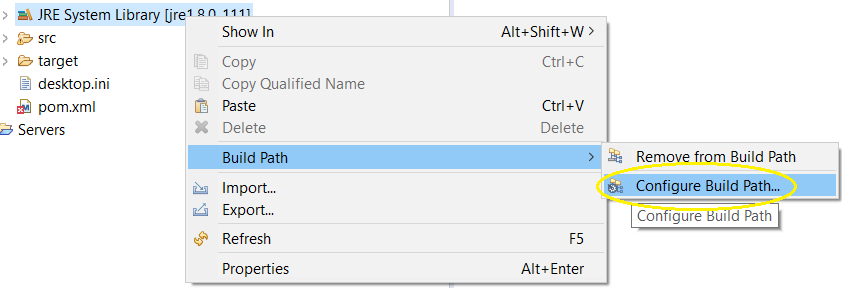


The above image specifies the **JDK 8** version and its equivalent **JRE 1.8** version is installed in the system.

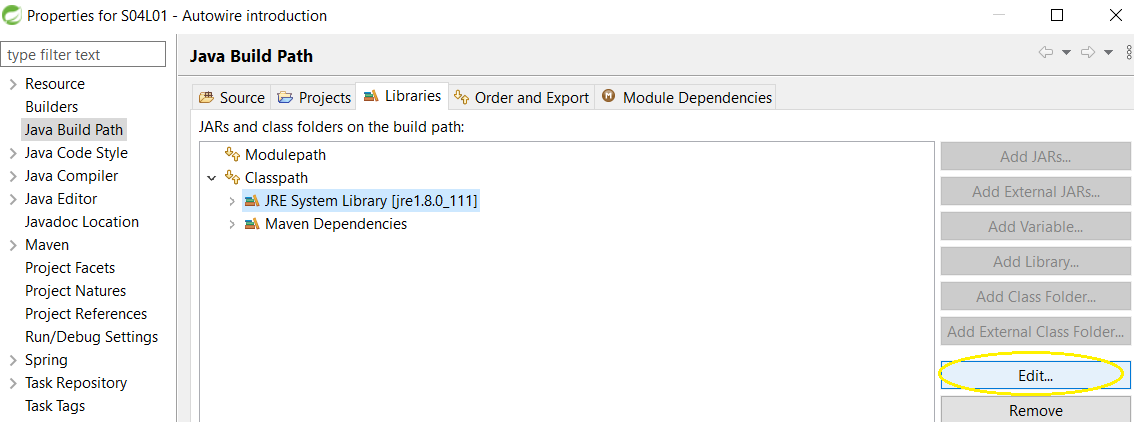
Now to change that to JDK 11 you have to install the JDK 11 on your system.

**Steps:**

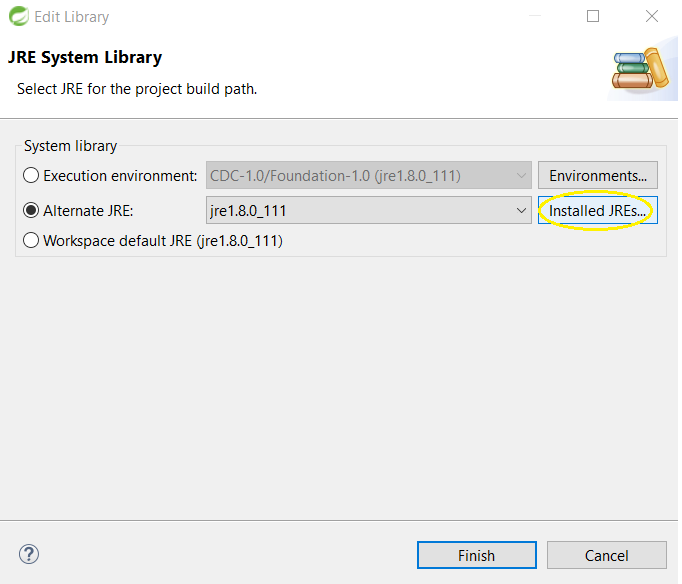
1. Right-click on the **JRE System Library** folder or the **project name** folder.
2. Choose the build path option ->Click on Configure build path



1. Click on Edit option

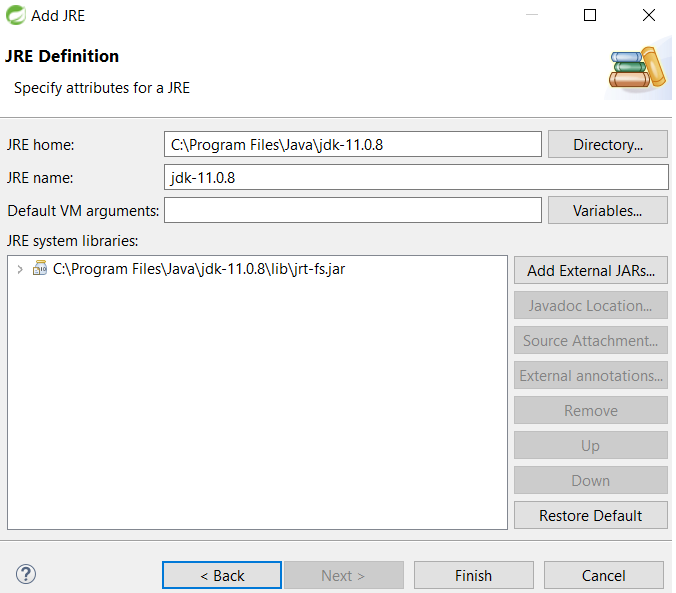


1. Choose the **Alternate JRE** option and click on **Installed JREs..** button

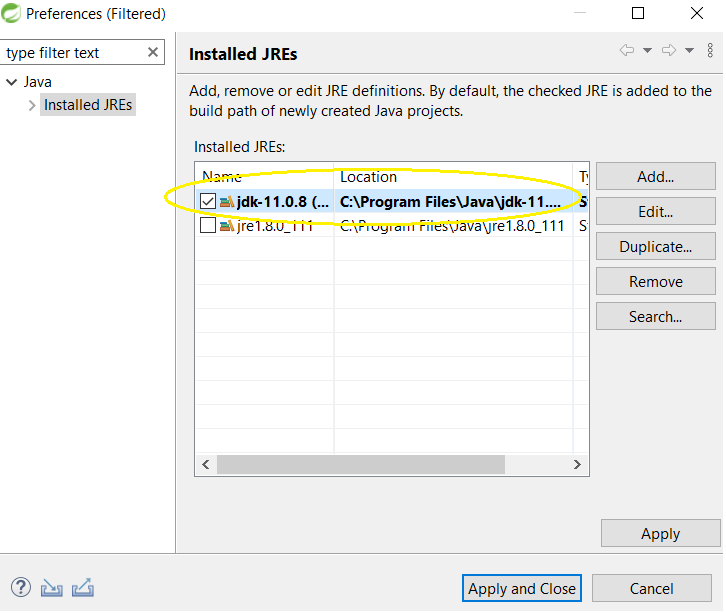


1. Now click on **Add** button and give the proper path of **JDK 11** installed on your system.

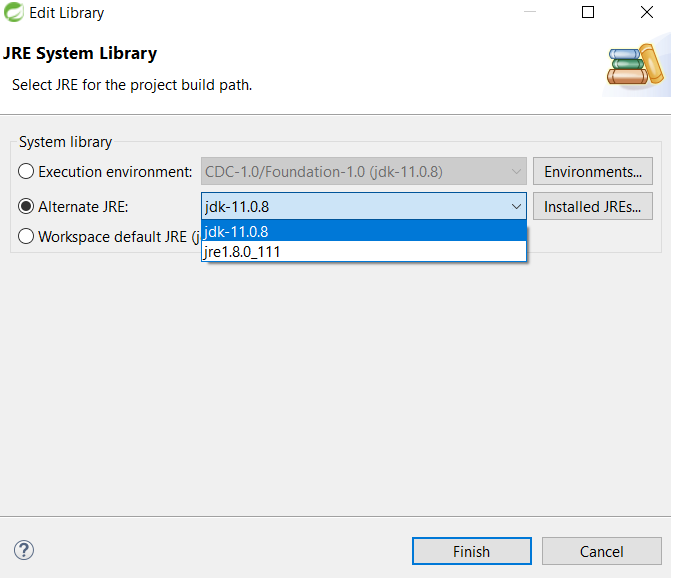
For example: **C:\Program Files\Java\jdk-11.0.8**



1. Now, make the JDK 11 as default using the checkbox option.

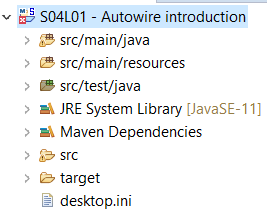


1. click on **Apply** -> **Apply and Close**
2. Now **Alternate JRE** option will show “two JRE versions”, choose the JDK 11 option.



1. Use the **Workspace default JRE** option and click on finish.
2. Now update the project into the maven project.

Below is the image after updating the maven project.



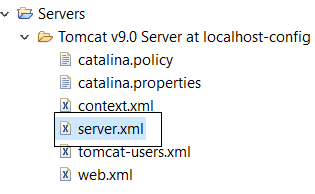
**Q.10.** 'Starting Tomcat v9.0 Server at localhost' has encountered a problem

**Answer:** Firstly, try to check your console and find the actual cause of error there.

Below is an example provided, that tells how does the error looks like (The error might be different in your case, this is just an example for your reference)

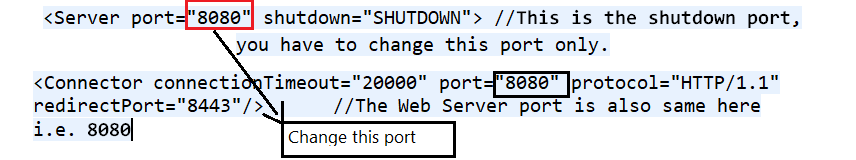
**Example:** Failed to create server shutdown socket on **address[localhost]** and **port[8080]** (base port [8080]) and offset [0]) java.net.BindException : Address already in use : JVM Bind.

Now, there is a reason why we get this type of exception is that, the “**server.xml**” file which can be found in the **Servers** folder in your project explorer. Below is the image provided by the **Servers** folder.



Now, in this file i.e. “**server.xml**” the actual issue arising here is that there is a conflict between your **shutdown** port and **HTTP** port are accessing the same ports.

Below is the image, where I have mentioned the ports which are similar and using the same address ports.



Change the shutdown port, i.e. <Server port=”8080”> to like: <Server port=”8008”> or any port number of your wish, other than **8080**

Now Start your tomcat server once again.

**Note:** Whenever your server does not start for any of the following reasons, try to see your console first and find out what is the error, why I’m getting this error, and then Google for the related solutions accordingly.

**Q.11.** Autocomplete not coming in Eclipse IDE, also the Project Facets dialog is not showing

**Answer:** If you would like to use autocomplete all the time without having to worry about hitting **Ctrl + Spacebar** or your keyboard shortcut, you can make the following adjustment in the Eclipse preferences to trigger autocomplete simply by typing several different characters:

1. Eclipse > Preferences > Java > Editor > Content Assist
2. Auto Activation > Auto activation triggers for Java
3. Enter all the characters you want to trigger autocomplete, such as the following:

abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ.\_

Now any time that you type any of these characters, Eclipse will trigger autocomplete suggestions based on the context.

**"further configuration required" is also not coming in the Project Facets dialog**

You can just add technology facets to your project, instead of configuring it to a different type of project.

**Steps:**

Right click on project -> **Properties** -> **Project Facet** -> Convert to faceted form (if needed) -> JPA.

The link that says "Further configuration required(available)" you may want to configure a JPA implementation. To use EclipseLink: You should see the **JPA Facet** dialog.

Select EclipseLink from Platform -> Select User library from Type. Click "Download Library (disk icon)" on the right. You would be able to do, the rest of the steps.

Here is an alternate solution from Stack Overflow, that might be helpful if the above steps don’t work for you,

Link: <https://stackoverflow.com/questions/13818155/eclipse-warning-unknown-faceted-project>

**Q.12.** Project build error: Non-resolvable parent POM

**Answer:** The issue might be with your project which is not able to complete maven build.

**Steps:**

1. Right Click Application and RunAs maven install.
2. If you get any error while reaching the repos online try giving the proxies in **settings.xml** under your **.m2 directory**. Check the below link for setting proxies for maven build.

Link: <https://maven.apache.org/guides/mini/guide-proxies.html>

1. Once done, try to Update Project by Right Click on **Project**, **Maven**->**Update Maven Project**, and select codebase and do check the **Force Update of Snapshot/Release** checkbox.

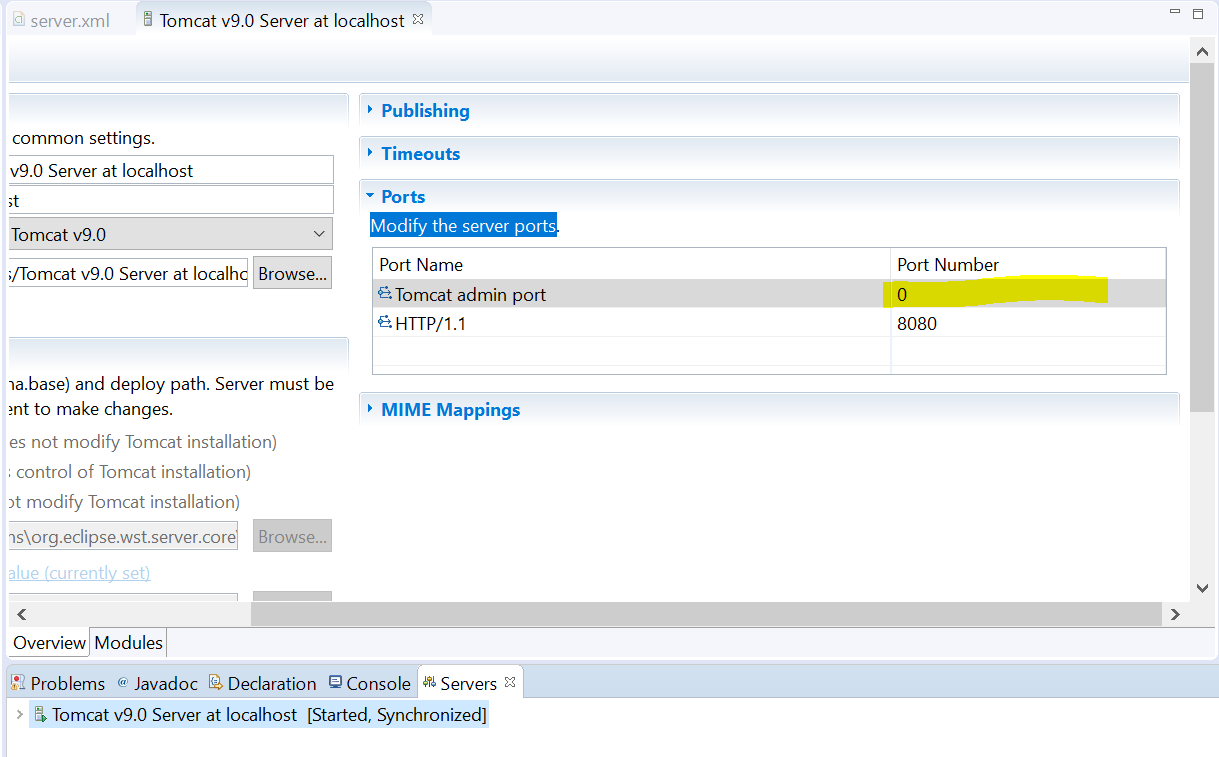
This will update your maven build and might remove your errors with pom.xml

**Q.13.** The server cannot be started because one or more of the ports is invalid.

**Answer:** You need to modify the server ports "Tomcat Admin Port" default value from hyphen (-) to zero (0)

**Steps:**

1. Double-click on the Tomcat server.
2. Change "Tomcat Admin Port" default value from hyphen (-) to zero(0)
3. Save the changes.
4. Restart Eclipse



**Note:** If the above steps do not work in your case, change the **Tomcat admin port** to some other port value. For ex: 8081, 8082, and so on.

**Q.14.** java.lang.ClassCastException: java.math.BigInteger cannot be cast to java.lang.Long occur on connecting to MySQL

**Answer:** Your error clearly says casting is not possible, because of a java.math.BigInteger class instance is not an instance of java.lang.Long class.

Now the question arises who is doing casting at what level, when we ask the JDBC driver to make a connection, it is doing a lot of work behind the scene before it gives us back the proper working object of connection.

The problem seems with your version of MySQL in combination with your version of mysql-connector.jar Try a newer version of MySQL Connector/J, the link is provided below.

(see <https://dev.mysql.com/downloads/connector/j/> for the latest version),

for example: upgrade to **5.1.47** or **8.0.12(**search for latest versions available now**)** if you are using an older version.

**Q.15.** Java EE tools option, and how to get it?

### Answer: Firstly, a minimal setup of JDK/JRE is needed to be installed on your system.

### Now you need to choose the “Eclipse IDE for Enterprise Java Developers” option for Java EE users and choose the appropriate version of OS you are working on.

### 5.png

### In Section 2(Lecture 6) “Download and installation process of Eclipse”, you can learn how to install Eclipse IDE for Java EE.

### Here is a download link: <https://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/2020-06/R/eclipse-jee-2020-06-R-win32-x86_64.zip>

**Q.16.** JDK, Eclipse, and Tomcat (which version is recommended to download)

**Answer:** Software versions used in our course and Recommended versions to download

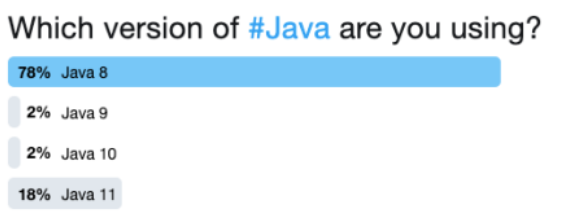
* **JAVA(JDK)**

|  |  |  |
| --- | --- | --- |
| **Product name** | **Version** | **Release date** |
| **Java SE 8(LTS)** | **1.8** | **March 2014** |
| **Java SE 9** | **9** | **September 2017** |
| **Java SE 10** | **10** | **March 2018** |
| **Java SE 11(LTS)** | **11** | **September 2018** |
| **Java SE 12** | **12** | **March 2019** |
| **Java SE 13** | **13** | **September 2019** |
| **Java SE 14** | **14** | **March 2020** |
| **Java SE 15** | **15** | **September 2020** |
| **Java SE 16** | **15** | **March 2021** |

**Note:** Java 14 is the latest stable release and shows promising new features. Below is the link provided you can check what are the additional features in the following link,

Credits: <https://www.stackchief.com/blog/Java%2014%20%7C%20New%20Features>

But remember, the latest version i.e. **JDK 16** will not offer long term support (LTS). For these reasons, LTS support editions are only recommendations for production releases including **Java SE 8** and **Java SE 11**.



When developers upgrade from Java 8, they generally upgrade to Java 11.

This explains the pattern of adoption generally; people will not be on Java 9 or 10. If they were using it at any stage, they should have moved on to at least Java 11.

Java SE 11 will be the next major version with long term support (LTS).

* **Eclipse(for JAVA EE)**

|  |  |  |
| --- | --- | --- |
| **Product name** | **Platform version** | **Release date** |
| **Eclipse Mars** | **4.5** | **24 June 2015** |
| **Eclipse Neon** | **4.6** | **22 June 2016** |
| **Eclipse Oxygen** | **4.7** | **28 June 2017 (Introduced JUnit, Java 9, 10 support)** |
| **Eclipse Photon** | **4.8** | **27 June 2018 (Dropped support for Windows 32-bit)** |
| **Eclipse 2018-09** | **4.9** | **19 September 2018** |
| **Eclipse 2018-12** | **4.10** | **19 December 2018 (Added Support for JAVA 11)** |
| **Eclipse 2019-03** | **4.11** | **20 March 2019** |
| **Eclipse 2019-06** | **4.12** | **19 June 2019** |
| **Eclipse 2019-09** | **4.13** | **18 September 2019** |
| **Eclipse 2019-12** | **4.14** | **18 December 2019** |
| **Eclipse 2020-03** | **4.15** | **18 March 2020** |
| **Eclipse 2020-06** | **4.16** | **17 June 2020** |

Try the latest version of **Eclipse** which always has a fixed update of bugs and issues. All of the releases in the last few years are fairly stable.

* **Apache Tomcat versions**

|  |  |  |
| --- | --- | --- |
| **Tomcat Series** | **Description** | **Declared stable** |
| **Tomcat 7.0** | **It supports Servlet 3.0, JSP 2.2, EL 2.2, and WebSocket specifications** | **14 January 2011** |
| **Tomcat 8.0(superseded)** | **It supports Servlet 3.1, JSP 2.3, and EL 3.0 specifications** | **25 June 2014** |
| **Tomcat 8.5** | **Supports several newer features like HTTP/2, but remains a Servlet 3 container.** | **13 June 2016** |
| **Tomcat 9.0** | **Supports Servlet 4.0 spec and requires JDK 8 or greater.** | **1 January 2018** |
| **Tomcat 10** | **Servlet 5.0**, **JSP 3.0**, **EL 4.0**, **WebSocket 2.0 and** **Authentication 2.0** **specifications** | **Unknown** |

**Note:** Users of Tomcat 8.0 should be aware that **Tomcat 8.0** has now reached the end of life.

Below is a link provided for details about **Tomcat 8.0**

Link: <http://tomcat.apache.org/tomcat-80-eol.html>

Users of **Tomcat 8.0.x** should upgrade to **Tomcat 8.5.x** or later.

**Apache Tomcat 10.x** is the current focus of development (the versions required by the **Jakarta EE 9** platform)