Object Oriented Programming

LESSON 10

Servlet Technology

Outline

1. Introduction
2. Servlet API
3. Servlet Example

Overview

In this chapter, you are going to learn about

- Know Servlet
- Know how to create webpage using Servlet Technology
- Know how to use Servlet
- Know how to use Servlet
- Know how to implement website using Servlet

Learning content

1. Introduction
   - What is Servlet?
   - Flow of Servlet
   - Web terminology
2. Servlet API
   - Interfaces and classes in javax.servlet package
   - Interfaces and classes in javax.servlet.http package
   - Servlet interface
3. Servlet example
   - Apache Tomcat
   - Web module
   - Servlet with Eclipse

1. Introduction

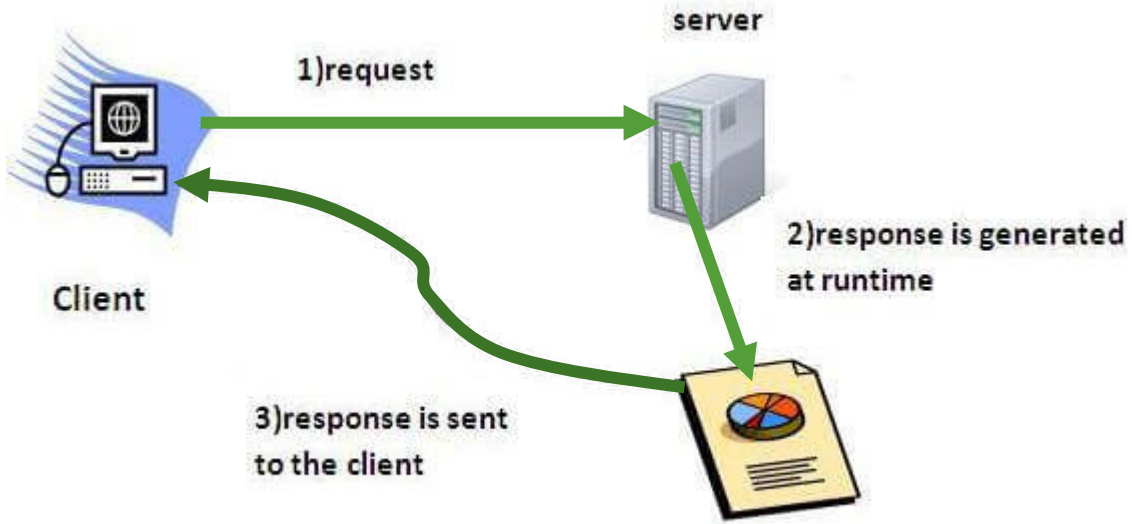
- **Servlet** technology is used to create a web application (resides at server side and generates a dynamic web page).
- **Servlet** technology is robust and scalable because of java language. Before Servlet, CGI (Common Gateway Interface) scripting language was common as a server-side programming language. However, there were many disadvantages to this technology. We have discussed these disadvantages below.
- There are many interfaces and classes in the Servlet API such as Servlet, GenericServlet, HttpServlet, ServletRequest, ServletResponse, etc.

1.1. What is Servlet?

- Servlet is a technology used to create a web application.
- Servlet is an API that provides many interfaces and classes including documentation.
- Servlet is an interface that must be implemented for creating any Servlet.
- Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests.
- Servlet is a web component that is deployed on the server to create a dynamic web page.

1.2. Flow of Servlet

http://example.com:8080/



1.3. Web terminology

Servlet Terminology	Description
Website: static vs dynamic	It is a collection of related web pages that may contain text, images, audio and video.
HTTP	It is the data communication protocol used to establish communication between client and server.
HTTP Requests	It is the request send by the computer to a web server that contains all sorts of potentially interesting information.
Get vs Post	It gives the difference between GET and POST request.
Container	It is used in java for dynamically generating the web pages on the server side.
Server: Web vs Application	It is used to manage the network resources and for running the program or software that provides services.
Content Type	It is HTTP header that provides the description about what are you sending to the browser.

2. Servlet API

- The **javax.servlet** package contains many interfaces and classes that are used by the servlet or web container. These are not specific to any protocol.
- The **javax.servlet.http** package contains interfaces and classes that are responsible for http requests only.

2.1. Interfaces and classes in javax.servlet package

Interfaces

- Servlet
- ServletRequest
- ServletResponse
- RequestDispatcher
- ServletConfig
- ServletContext
- SingleThreadModel
- Filter
- FilterConfig
- FilterChain
- ServletRequestListener
- ServletRequestAttributeListener
- ServletContextListener
- ServletContextAttributeListener

Classes

- GenericServlet
- ServletInputStream
- ServletOutputStream
- ServletRequestWrapper
- ServletResponseWrapper
- ServletRequestEvent
- ServletContextEvent
- ServletRequestAttributeEvent
- ServletContextAttributeEvent
- ServletException
- UnavailableException

2.2. Classes and Interfaces in javax.servlet.http package

Interfaces

- HttpServletRequest
- HttpServletResponse
- HttpSession
- HttpSessionListener
- HttpSessionAttributeListener
- HttpSessionBindingListener
- HttpSessionActivationListener
- HttpSessionContext (deprecated now)

Classes

- HttpServlet
- Cookie
- HttpServletRequestWrapper
- HttpServletResponseWrapper
- HttpSessionEvent
- HttpSessionBindingEvent
- HttpUtils (deprecated now)

2.3. Servlet Interface

- **Servlet interface provides** common behavior to all the servlets. Servlet interface defines methods that all servlets must implement.
- Servlet interface needs to be implemented for creating any servlet (either directly or indirectly). It provides 3 life cycle methods that are used to initialize the servlet, to service the requests, and to destroy the servlet and 2 non-life cycle methods.

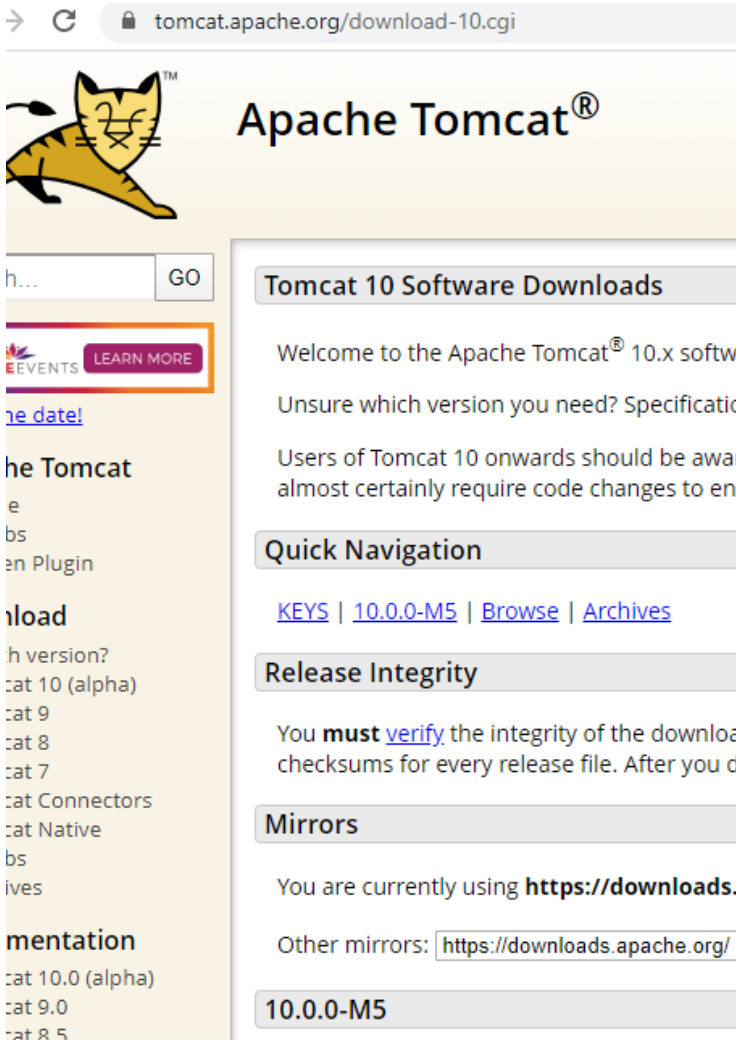
2.3. Servlet Interface

Method	Description
public void init(ServletConfig config)	initializes the servlet. It is the life cycle method of servlet and invoked by the web container only once.
public void service(ServletRequest request, **ServletResponse response)**	provides response for the incoming request. It is invoked at each request by the web container.
public void destroy()	is invoked only once and indicates that servlet is being destroyed.
public ServletConfig getServletConfig()	returns the object of ServletConfig.
public String getServletInfo()	returns information about servlet such as writer, copyright, version etc.

3. Servlet Example

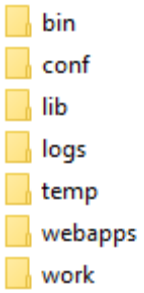
- Prerequisite
  - Apache Tomcat (Web server Java like Apache PHP)
  - OpenJDK or JDK 8.0+
  - Eclipse IDE for JEE developer

3.1. Apache Tomcat



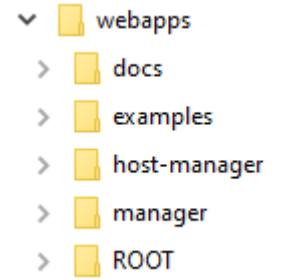
- Download Apache Tomcat 9.x at: https://tomcat.apache.org/download-90.cgi
- Download as **64-bit Windows zip**
- After download, extract it, and then a folder is extracted with name: apache-tomcat-9.0.35 or any version number greater than 8.

3.1. Apache Tomcat



- **/bin** - Startup, shutdown, and other scripts. The *.sh files (for Unix systems) are functional duplicates of the *.bat files (for Windows systems). Since the Win32 command-line lacks certain functionality, there are some additional files in here.
- **/conf** - Configuration files and related DTDs. The most important file in here is server.xml. It is the main configuration file for the container.
- **/logs** - Log files are here by default.
- **/webapps** - This is where your webapps go.

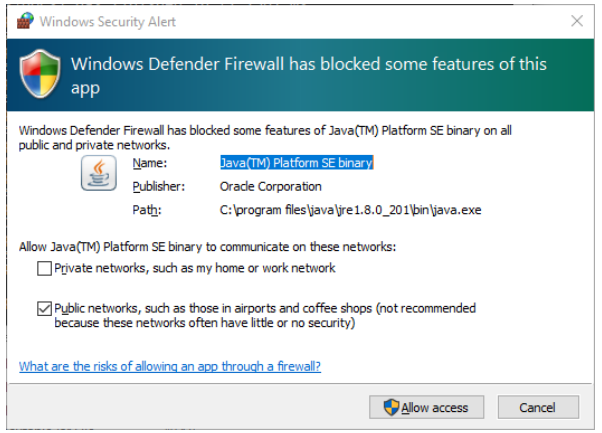
3.1. Apache Tomcat

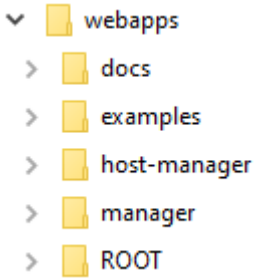


- **docs/** - user manual for Tomcat users
- **examples/** - examples web apps
- **host-manager/** - Web App enables you to create, delete, and otherwise manage virtual hosts within Tomcat
- **manager/** - a web app used to deploy a new web application, or undeploy an existing one, without having to shut down and restart the entire container
- **ROOT/** - root web app (ex: http://localhost:8080/)

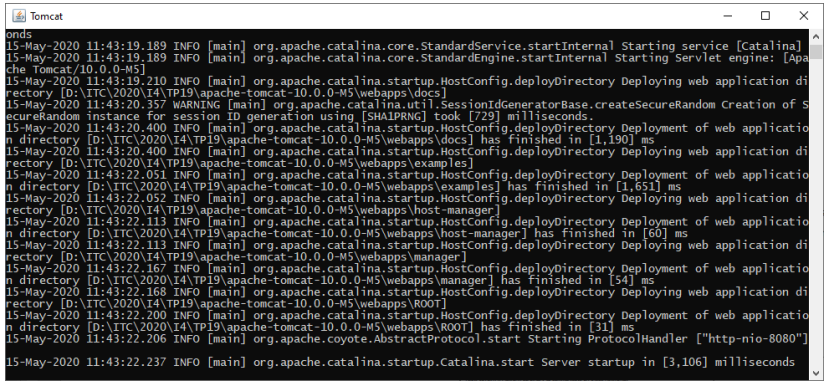
3.1. Apache Tomcat

- **Starting Server:**
  - Double click on bin/startup.bat
  - If prompt, click allow:



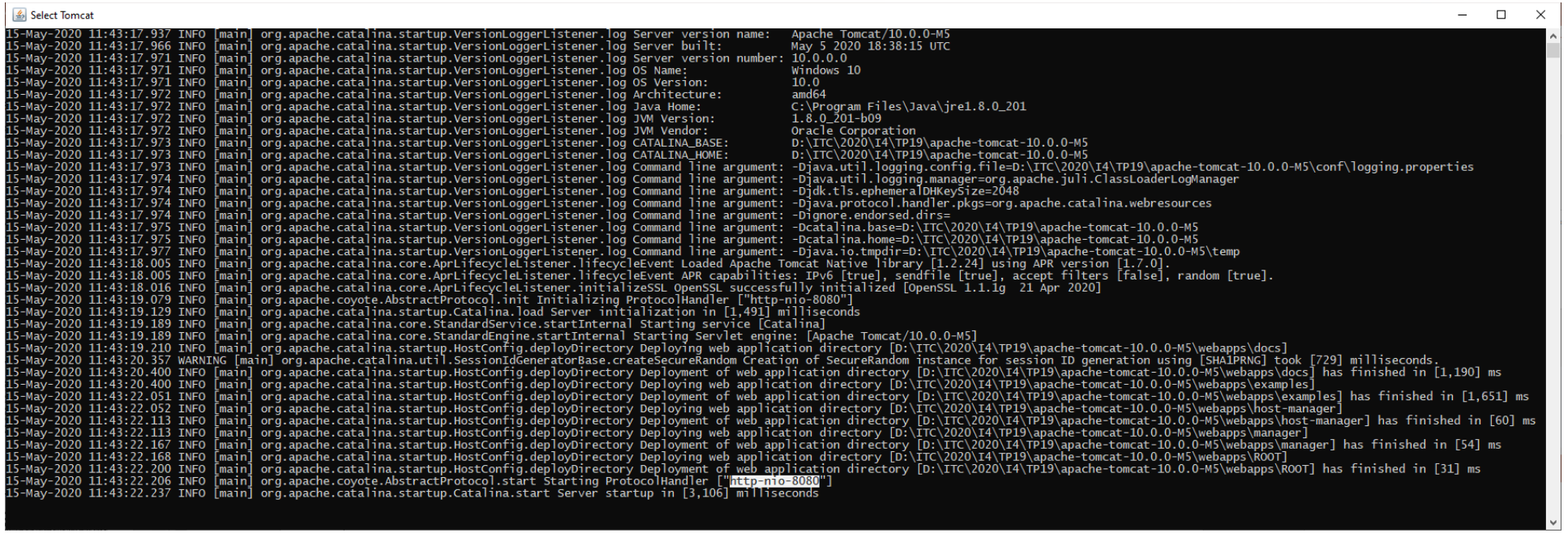


  - If successful startup, it will show interface like bellow:
  - DO NOT CLOSE THIS DOS WINDOWS UNLESS SHUTDOWN SERVER!



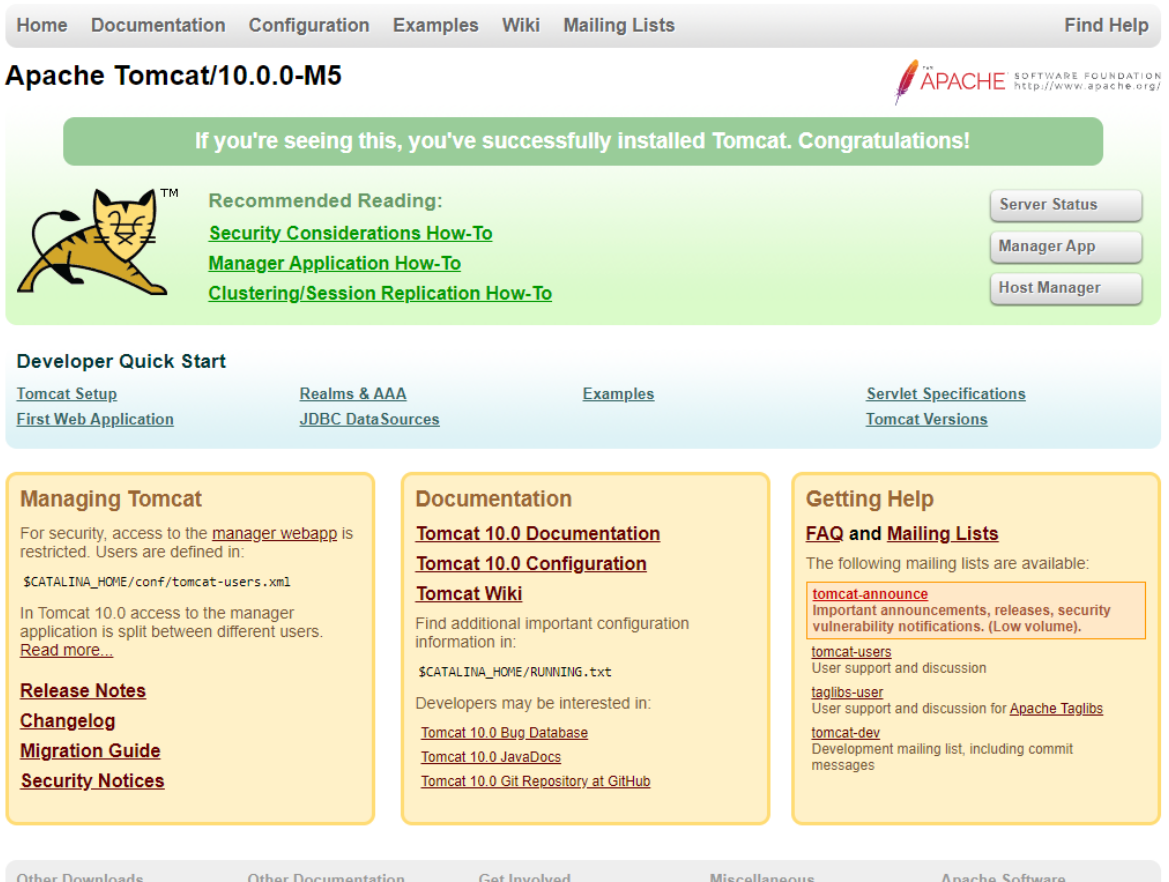
3.1. Apache Tomcat

- Access to server with address: http://localhost:8080/
- Or any port indicated in the DOS windows when start up successfully.



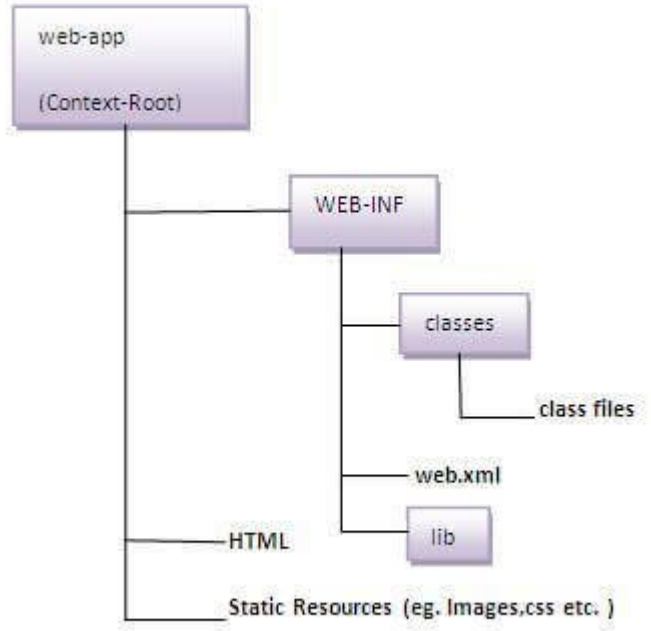
3.1. Apache Tomcat

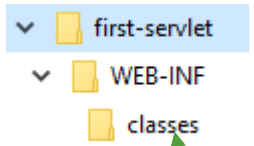
- If success start up, web page at address: http://localhost:8080/



3.2. Web module

- Folder structure of a Web App in Tomcat:





Your servlet classes here.

3.2. Web module

- First.java file created in folder first-servlet/WEB-INF/classes, then Enter content:

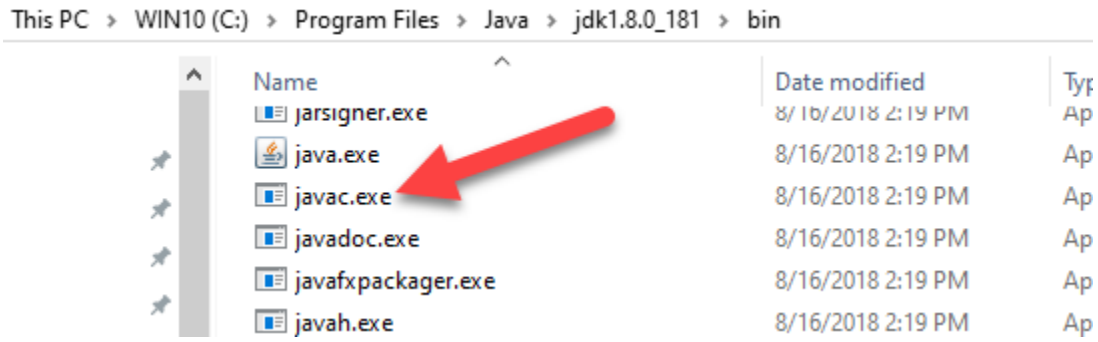
```
import java.io.*;

import javax.servlet.*;

public class First implements Servlet {
  ServletConfig config = null;

  public void init(ServletConfig config) {
    this.config = config;
    System.out.println("servlet is initialized");
  }

  public void service(ServletRequest req, ServletResponse res)
                    throws IOException, ServletException {

    res.setContentType("text/html");

    PrintWriter out = res.getWriter();
    out.print("<html><body>");
    out.print("<b>hello simple servlet</b>");
    out.print("</body></html>");
  }

  public void destroy() {
    System.out.println("servlet is destroyed");
  }

  public ServletConfig getServletConfig() {
    return config;
  }

  public String getServletInfo() {
    return "copyright 2007-1010";
  }
}
```

3.2. Web module

- Save the file First.java and then compile:

```
javac -cp ../../../../lib/servlet-api.jar First.java
```

- The javac.exe is located in <JDK_DIR>/bin (example: C:\Program Files\Java\jdk1.8.0_181\bin\javac.exe)



3.2. Web module

- If Javac.exe is not found, it will message as bellow:

```
Microsoft Windows [Version 10.0.18362.836]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\DICE>cd D:\ITC\2020\I4\TP19\apache-tomcat-9.0.35\webapps\first-jsp\WEB-INF\classes

C:\Users\DICE>D:

D:\ITC\2020\I4\TP19\apache-tomcat-9.0.35\webapps\first-jsp\WEB-INF\classes>javac First.java
'javac' is not recognized as an internal or external command,
operable program or batch file.

D:\ITC\2020\I4\TP19\apache-tomcat-9.0.35\webapps\first-jsp\WEB-INF\classes>
```

- In this case you need to put full path

```
D:\ITC\2020\I4\TP19\apache-tomcat-9.0.35\webapps\first-jsp\WEB-
INF\classes>"C:\Program Files\Java\jdk1.8.0_181\bin\javac.exe" -cp
../../../../lib/servlet-api.jar First.java

D:\ITC\2020\I4\TP19\apache-tomcat-9.0.35\webapps\first-jsp\WEB-INF\classes>
```

3.2. Web module

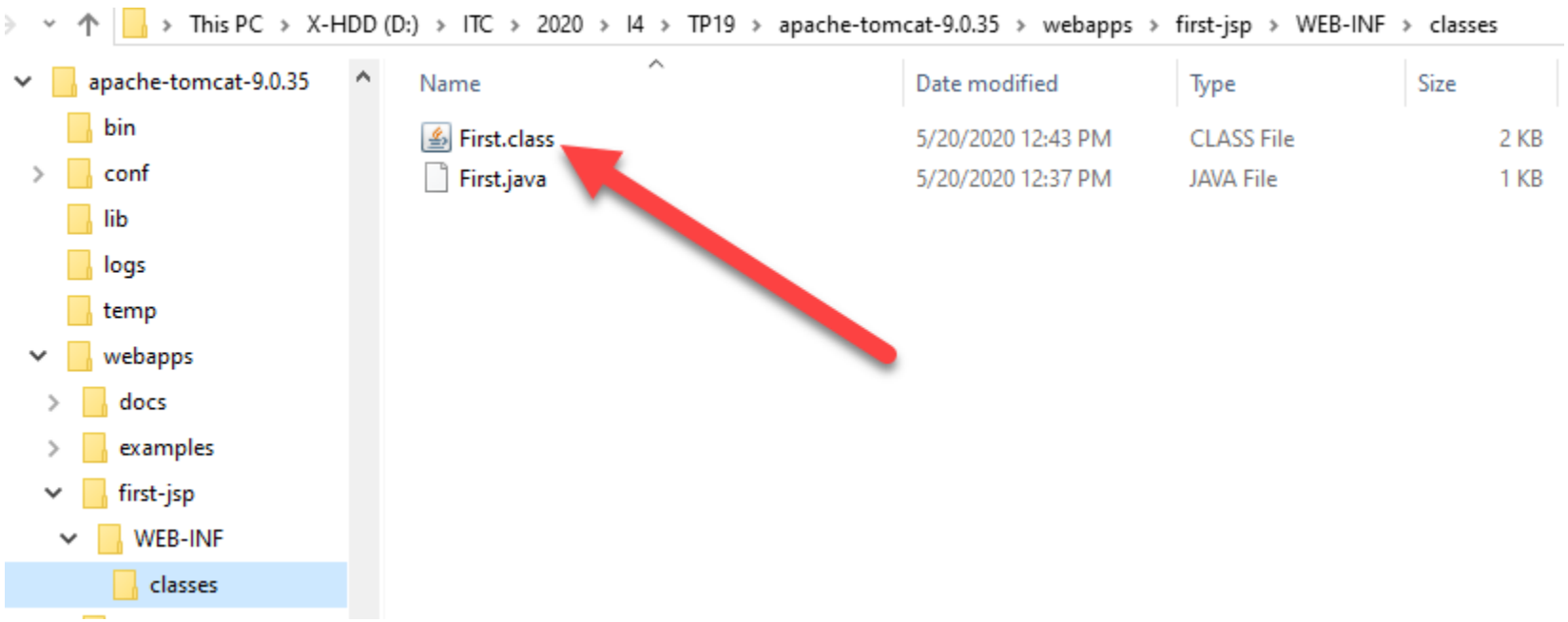
- Analysis

```
D:\ITC\2020\I4\TP19\apache-tomcat-9.0.35\webapps\first-jsp\WEB-
INF\classes>"C:\Program Files\Java\jdk1.8.0_181\bin\javac.exe" -cp
../../../../lib/servlet-api.jar First.java

D:\ITC\2020\I4\TP19\apache-tomcat-9.0.35\webapps\first-jsp\WEB-INF\classes>
```

  - -cp: means to use Library (*.jar) (CLASSPATH)
  - `-cp ../../../../lib/servlet-api.jar`
    Means that we need Servlet-API from Tomcat/lib/servlet-api.jar

3.2. Web module

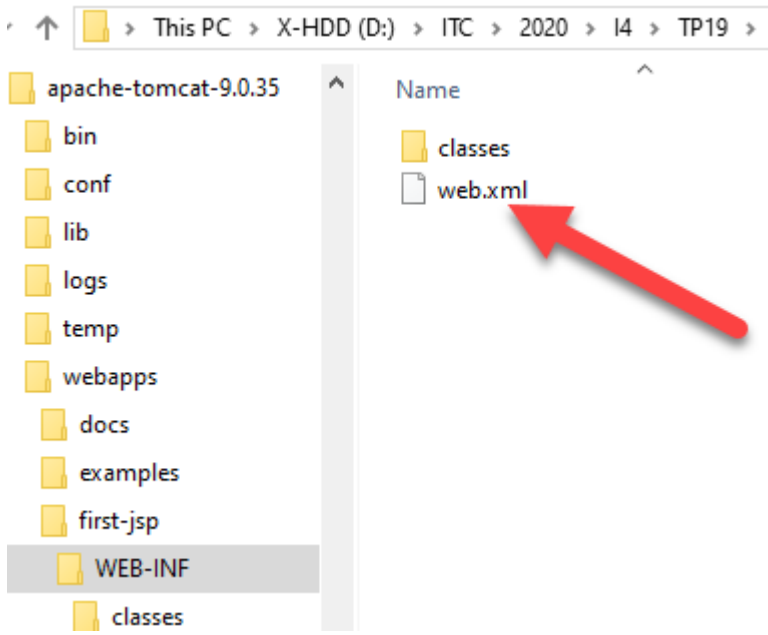
- If compile successful, it will generate a class file (*.class) in WEB-INF/classes:



3.2. Web module

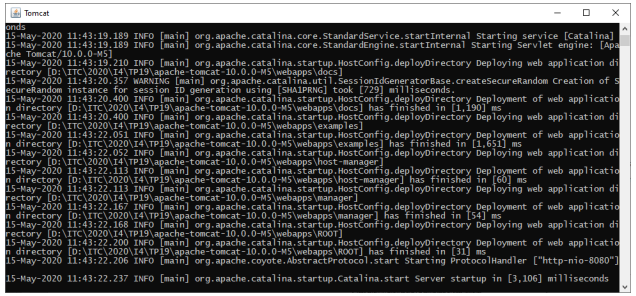
- Next, we create web.xml file in /WEB-INF/ with content:

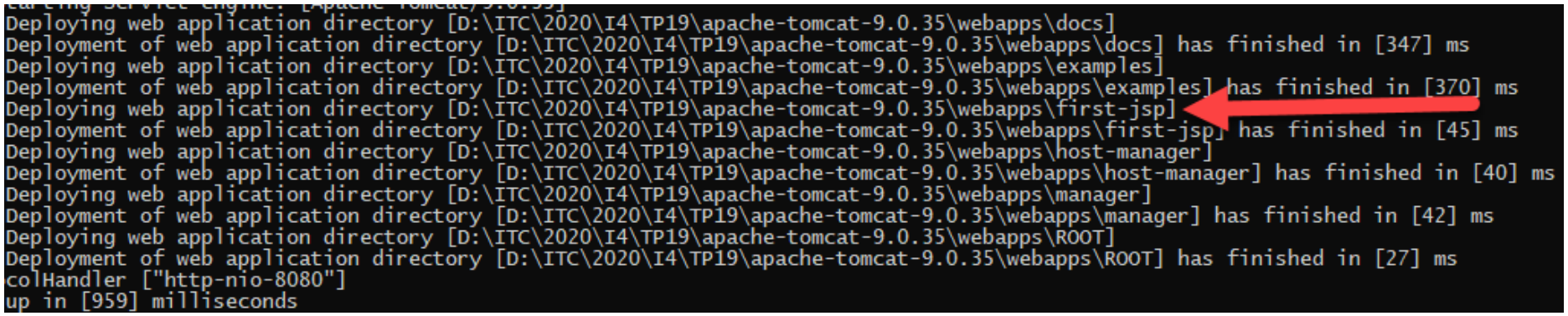
```
<web-app>

  <servlet>
    <servlet-name>my-first-servlet</servlet-name>
    <servlet-class>First</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>my-first-servlet</servlet-name>
    <url-pattern>/welcome</url-pattern>
  </servlet-mapping>

</web-app>
```



3.2. Web module

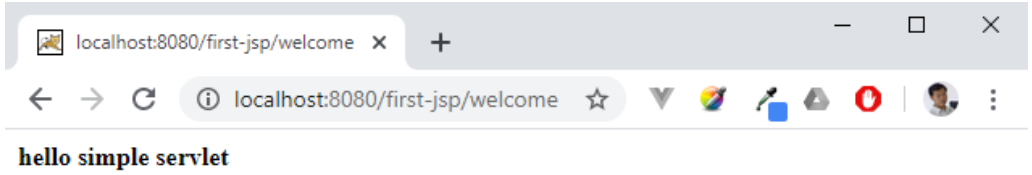
- Now, it is all setup!!
- You can start (or restart the server) to apply the changes
  - To do that, you can close the server console, and then run startup.bat again
  - After restarted the server, you will notice the different:

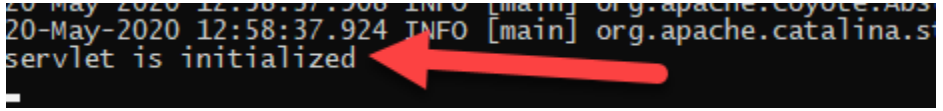




3.2. Web module

- To test whether it works, open your browser, and enter the address below:
  http://localhost:8080/first-jsp/welcome
- If everything's fine, you will see in browser below:



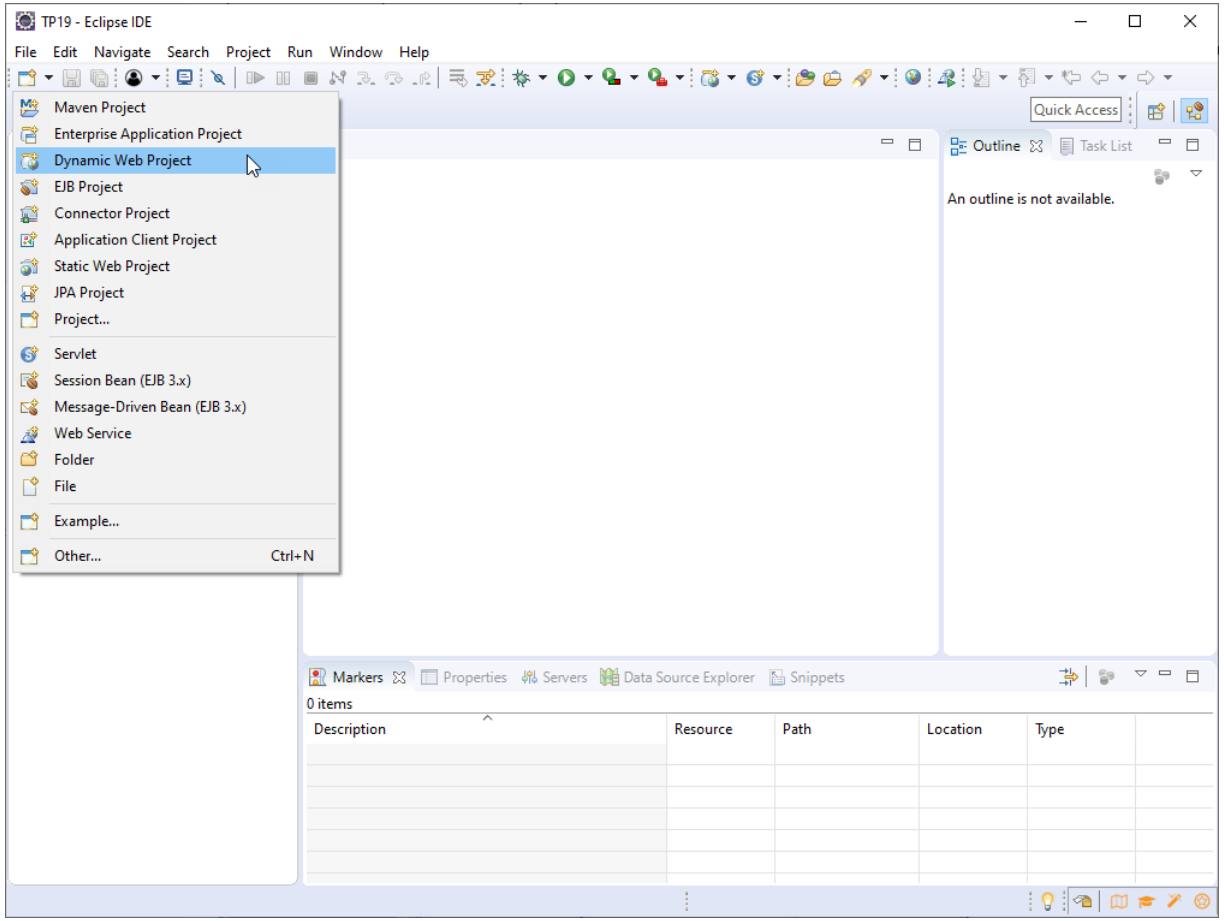


3.3. Servlet with Eclipse

- Eclipse is an open-source ide for developing JavaSE and JavaEE (J2EE) applications. You can download the eclipse ide from the eclipse website http://www.eclipse.org/downloads/.
- You need to download the eclipse ide for **JavaEE developers**.
- Steps:
  1. Create a Dynamic web project
  2. Create a servlet
  3. Run the servlet

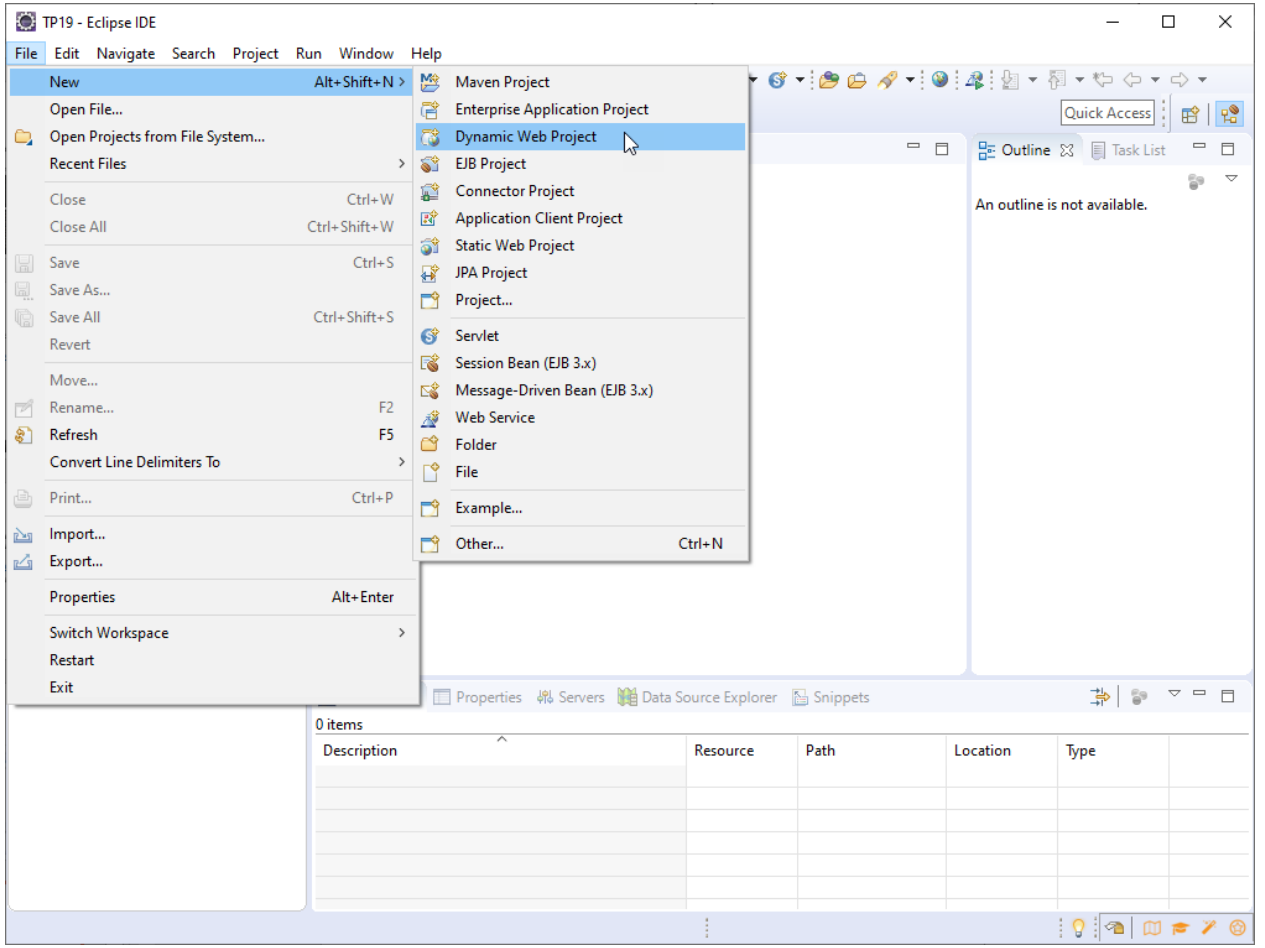
3.3. Servlet with Eclipse

1. Create a Dynamic web project (Method 1)



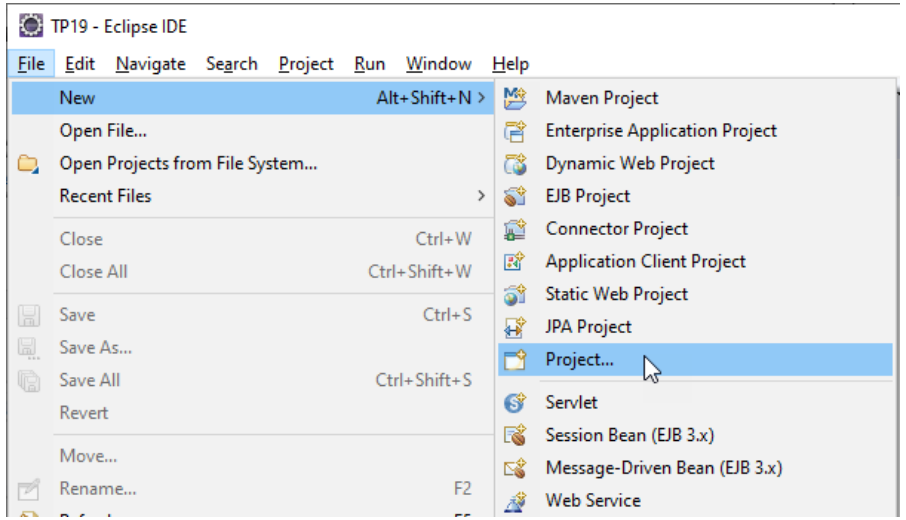
3.3. Servlet with Eclipse

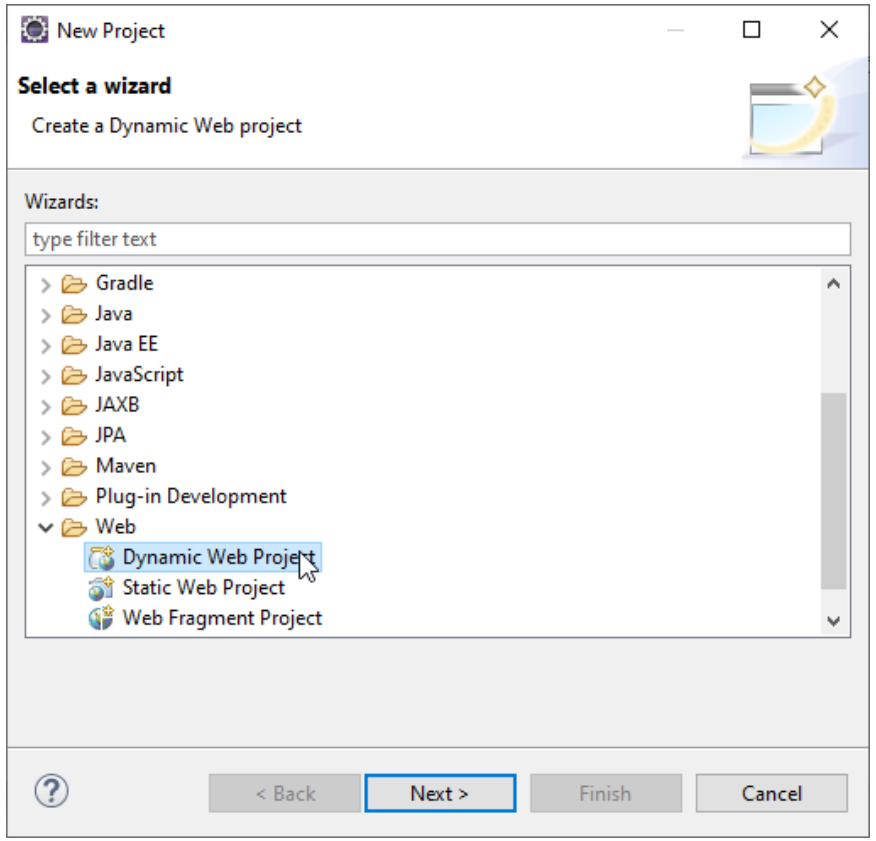
1. Create a Dynamic web project (Method 2)



3.3. Servlet with Eclipse

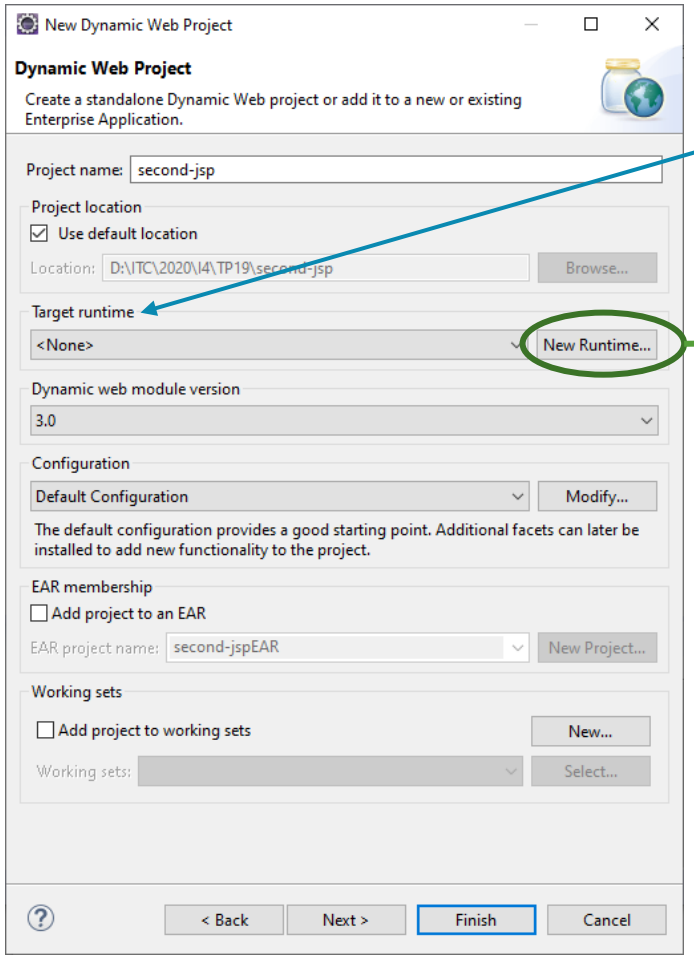
1. Create a Dynamic web project (Method 3)



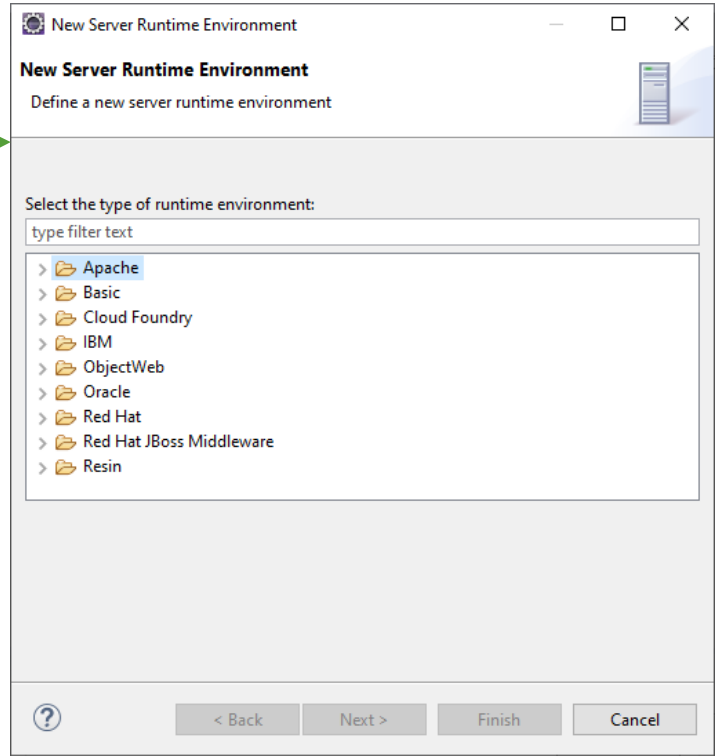


3.3. Servlet with Eclipse

1. Create a Dynamic web project (Step 2)

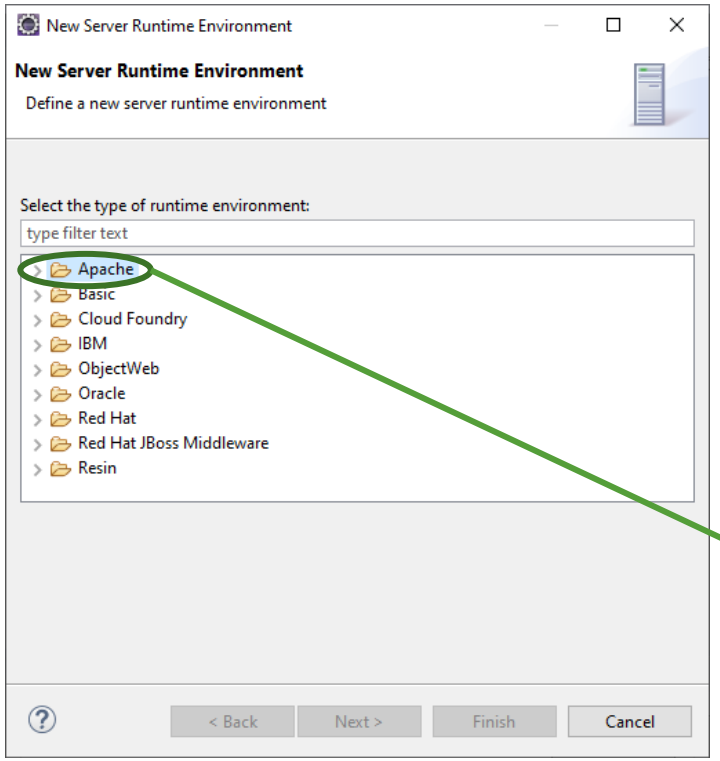


Target runtime: is server type you want to deploy to such as Apache Tomcat, Jetty, GlassFish, ...

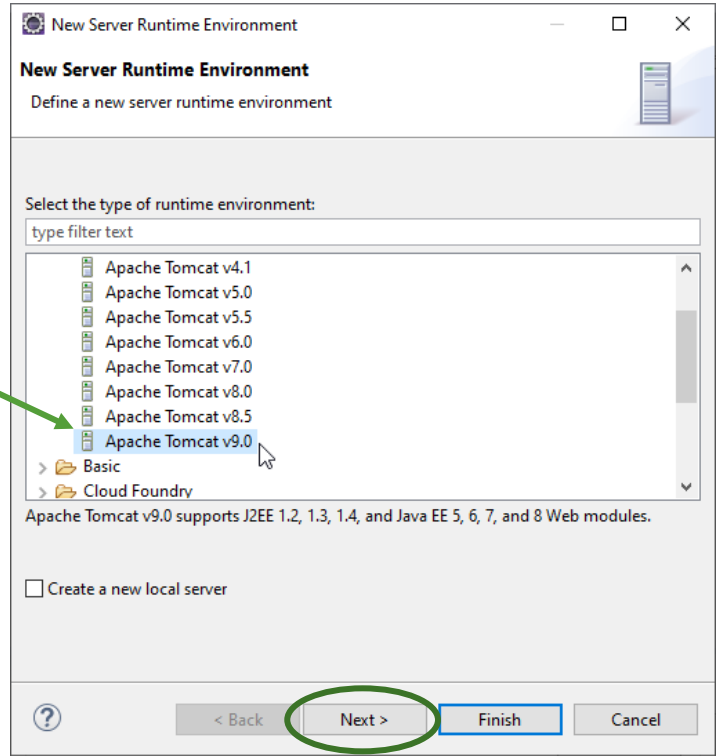


3.3. Servlet with Eclipse

1. Create a Dynamic web project (Step 2)

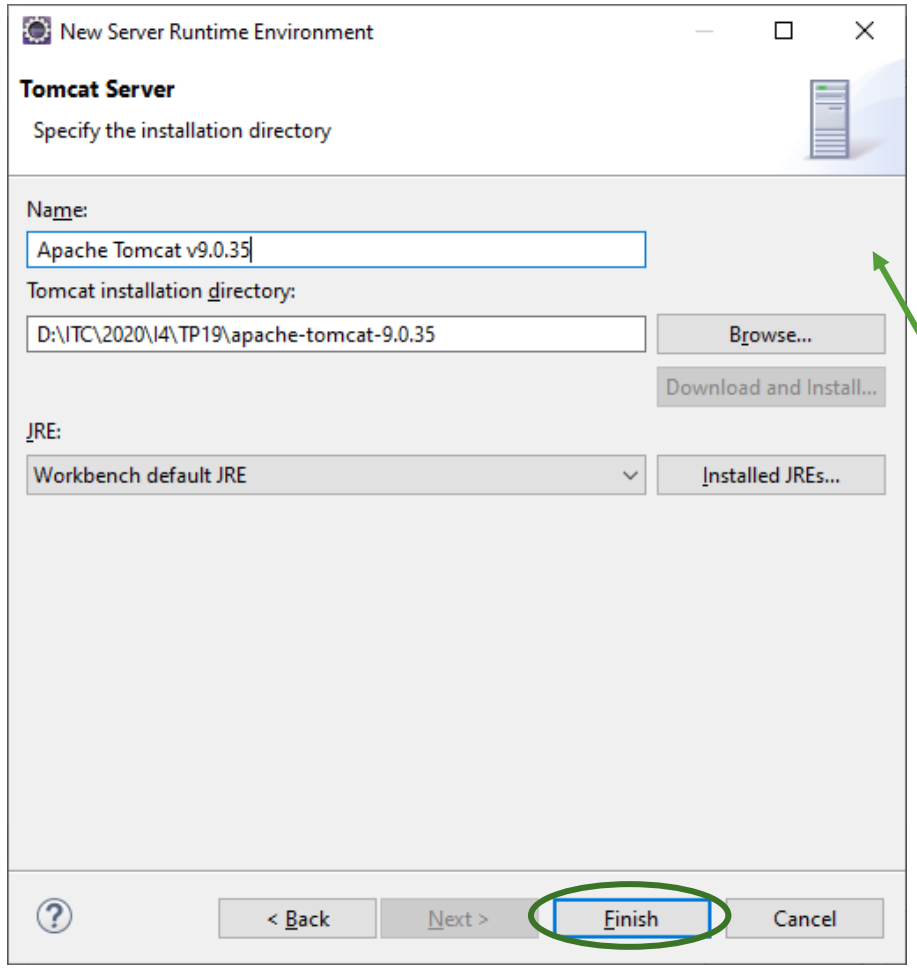


Select Target runtime as **Apache Tomcat v9.0** (because we downloaded last time)

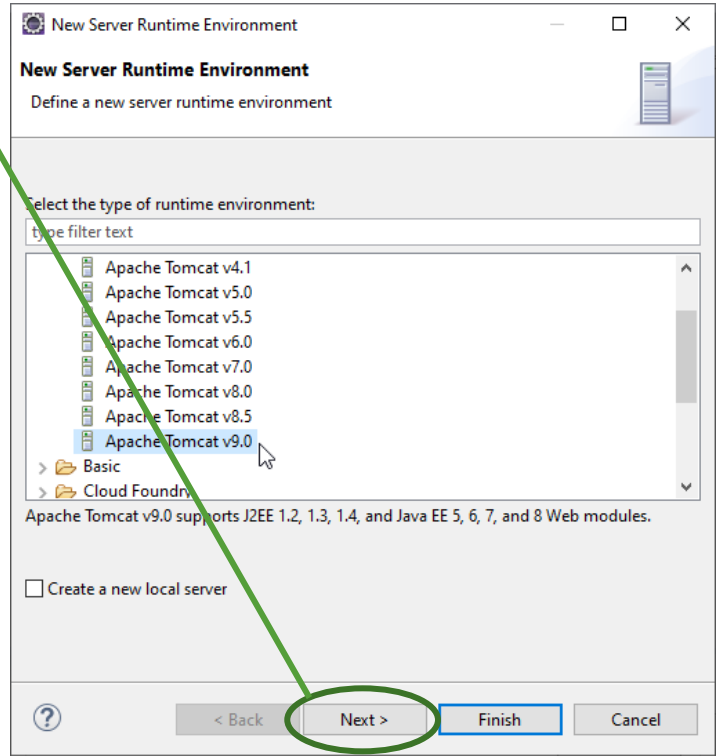


3.3. Servlet with Eclipse

1. Create a Dynamic web project (Step 2)

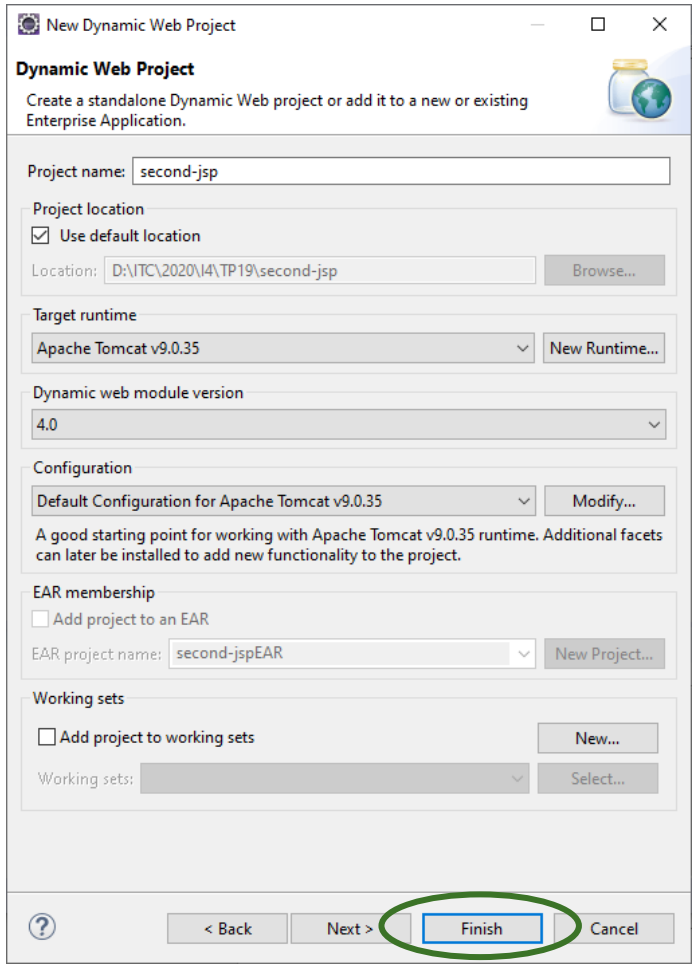


Select Target runtime as **Apache Tomcat v9.0** (because we downloaded last time)



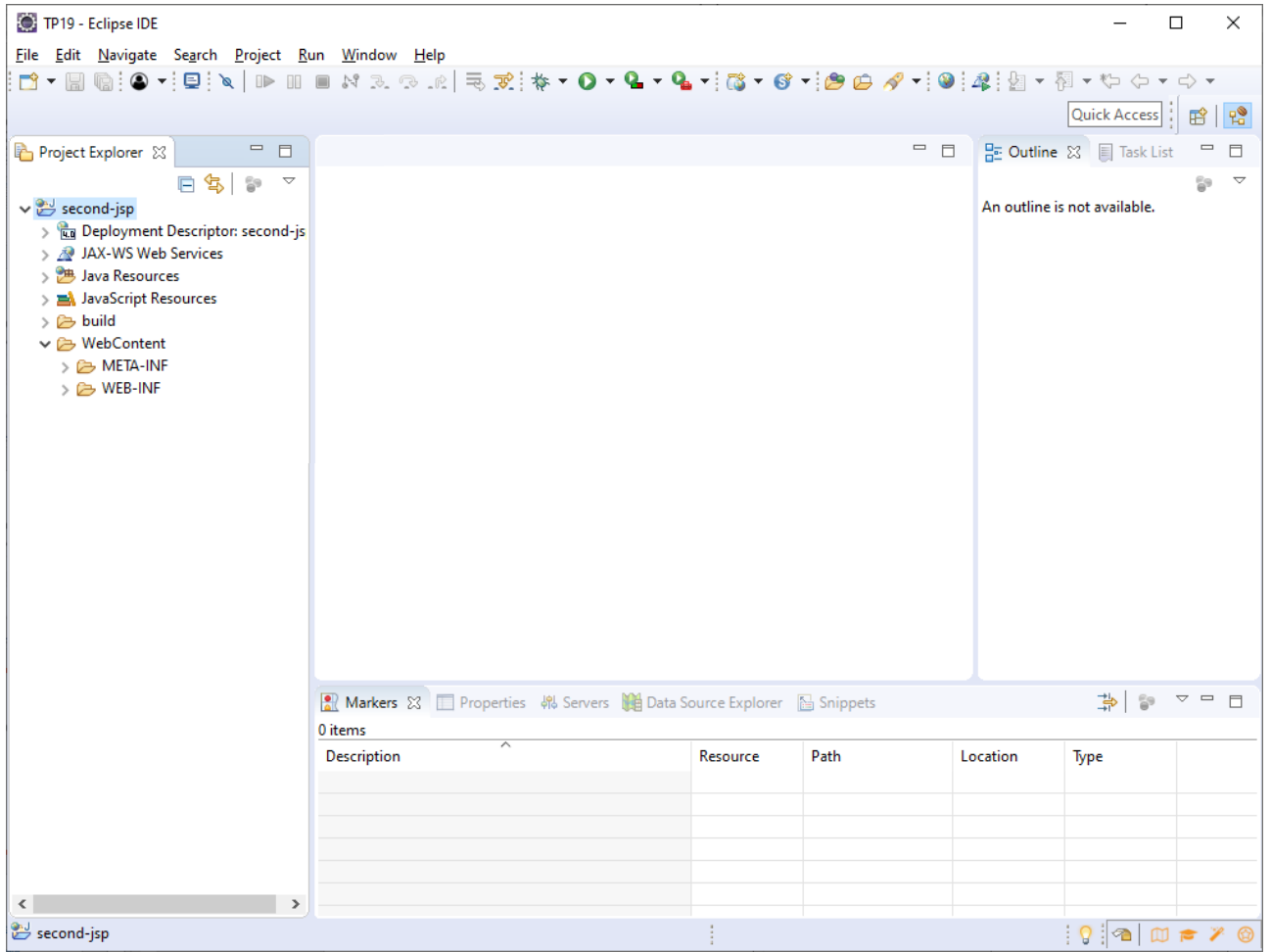
3.3. Servlet with Eclipse

1. Create a Dynamic web project (Step 2)



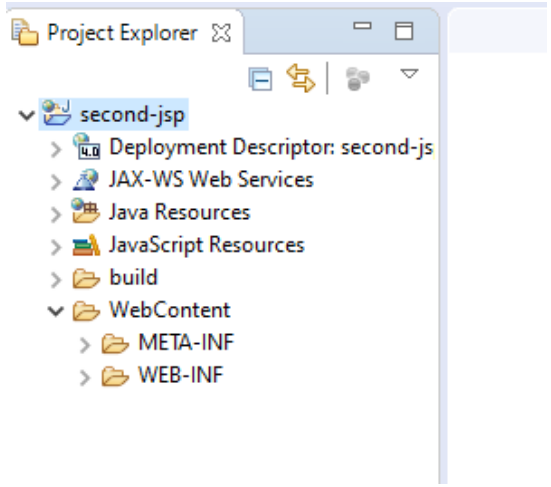
3.3. Servlet with Eclipse

1. Create a Dynamic web project



3.3. Servlet with Eclipse

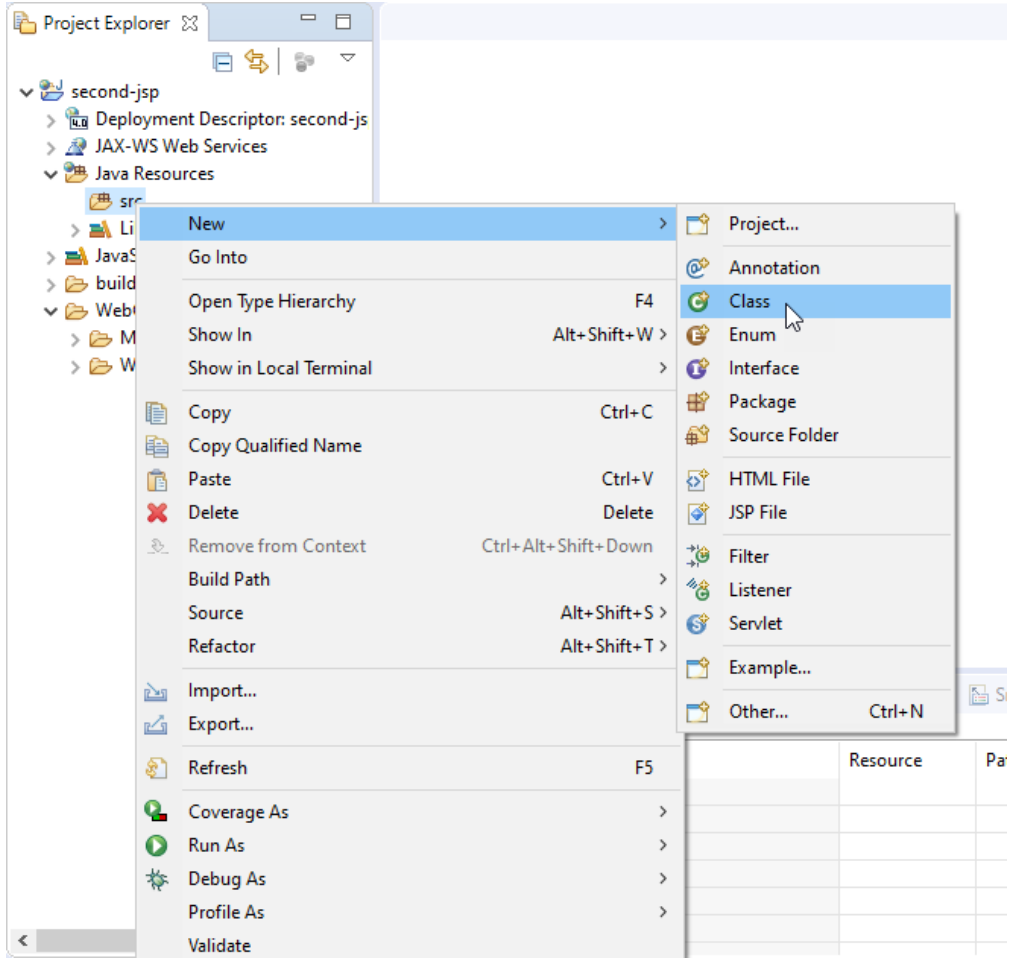
1. Create a Dynamic web project

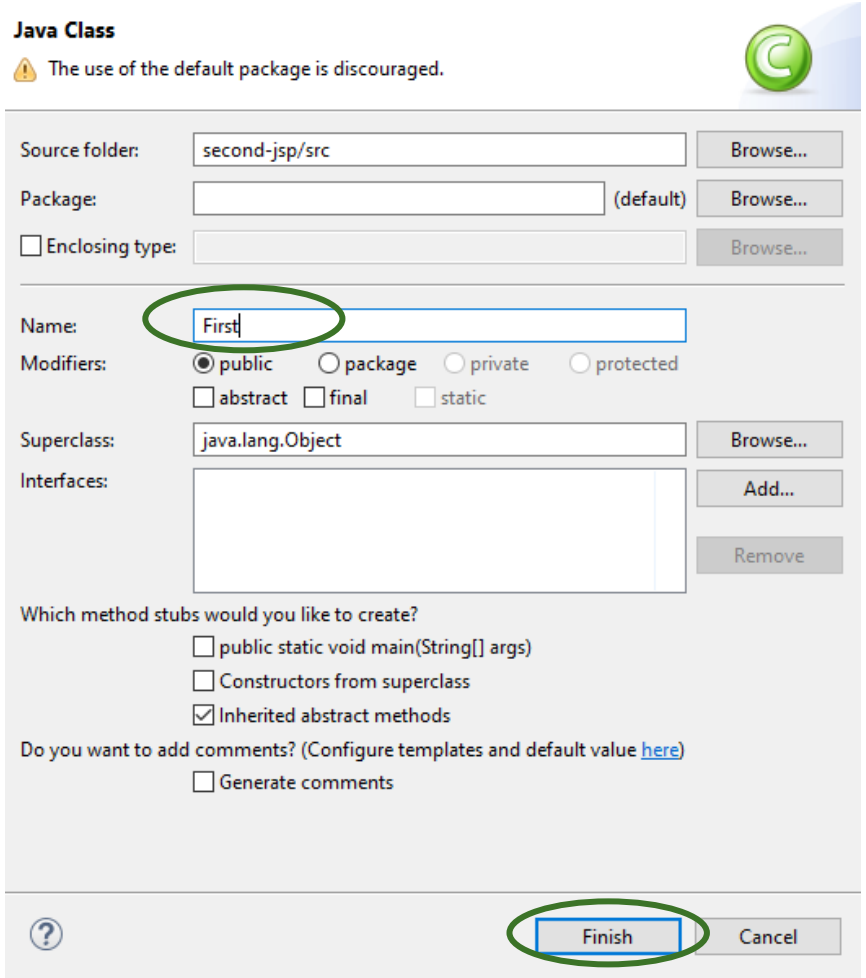


Deployment Descriptor: is web.xml visual editor.
JAX-WS Web Services: is for Web Services configurations
Java Resources: it is where you put Java source codes, packages and libraries
JavaScript Resources: it is where you put your JavaScript codes
build: is web module built (*.war,…)
WebContent: is web root folder ant it is used to put HTML,CSS,JS,Configs,*.classes files.

3.3. Servlet with Eclipse

2. Create a Servlet - “First.java”





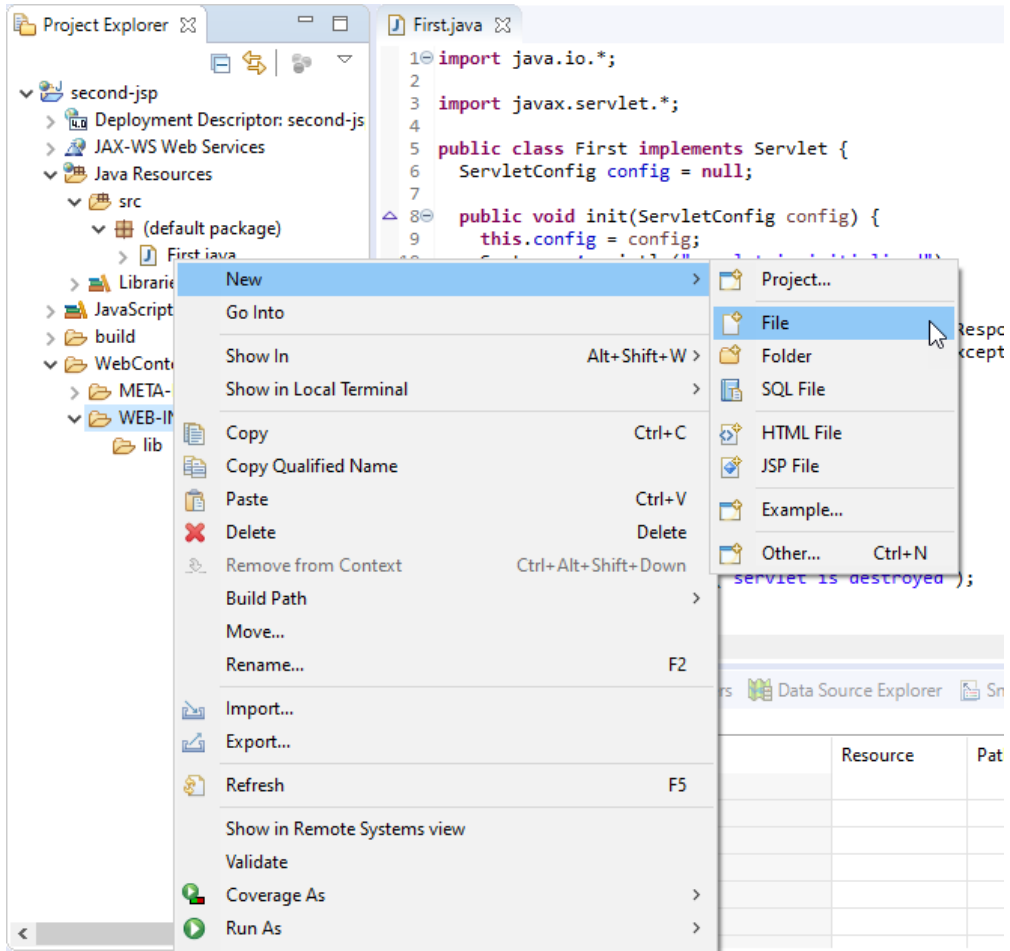
3.3. Servlet with Eclipse

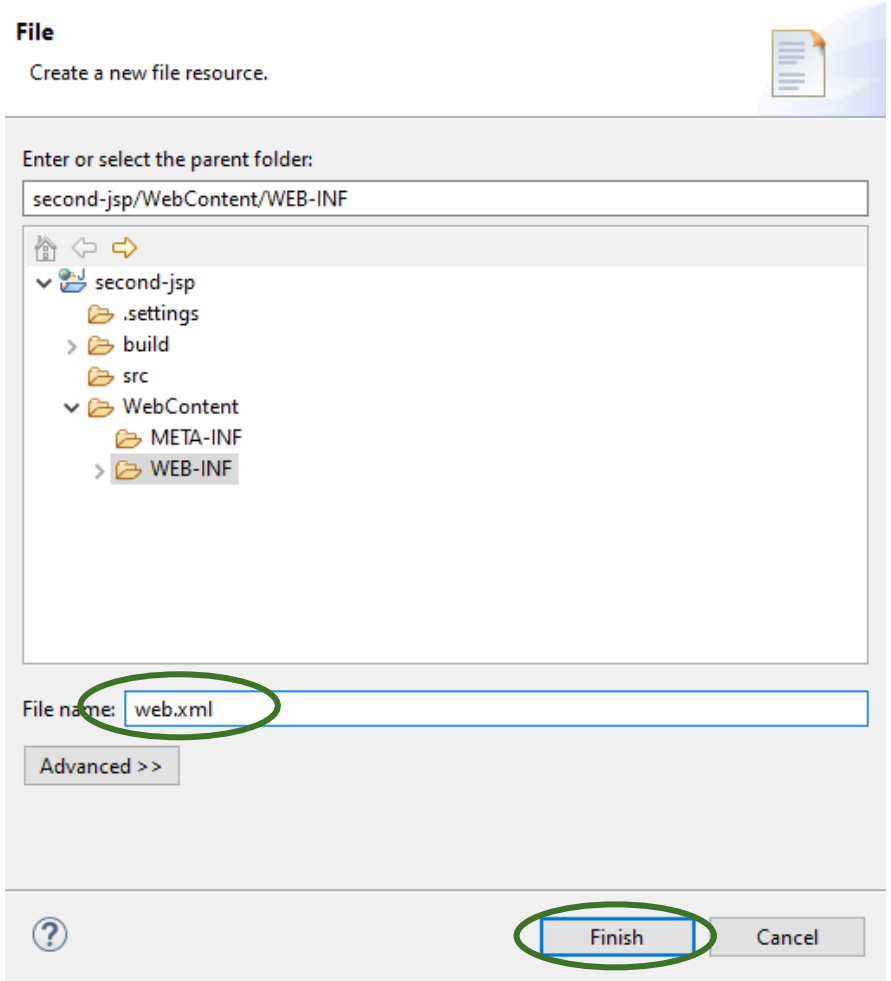
2. Create a Servlet - “First.java”

```
import java.io.*;

import javax.servlet.*;

public class First implements Servlet {
  ServletConfig config = null;

  public void init(ServletConfig config) {
    this.config = config;
    System.out.println("servlet is initialized");
  }

  public void service(ServletRequest req, ServletResponse res)
                    throws IOException, ServletException {

    res.setContentType("text/html");

    PrintWriter out = res.getWriter();
    out.print("<html><body>");
    out.print("<b>hello simple servlet</b>");
    out.print("</body></html>");
  }

  public void destroy() {
    System.out.println("servlet is destroyed");
  }

  public ServletConfig getServletConfig() {
    return config;
  }

  public String getServletInfo() {
    return "copyright 2007-1010";
  }
}
```

3.3. Servlet with Eclipse

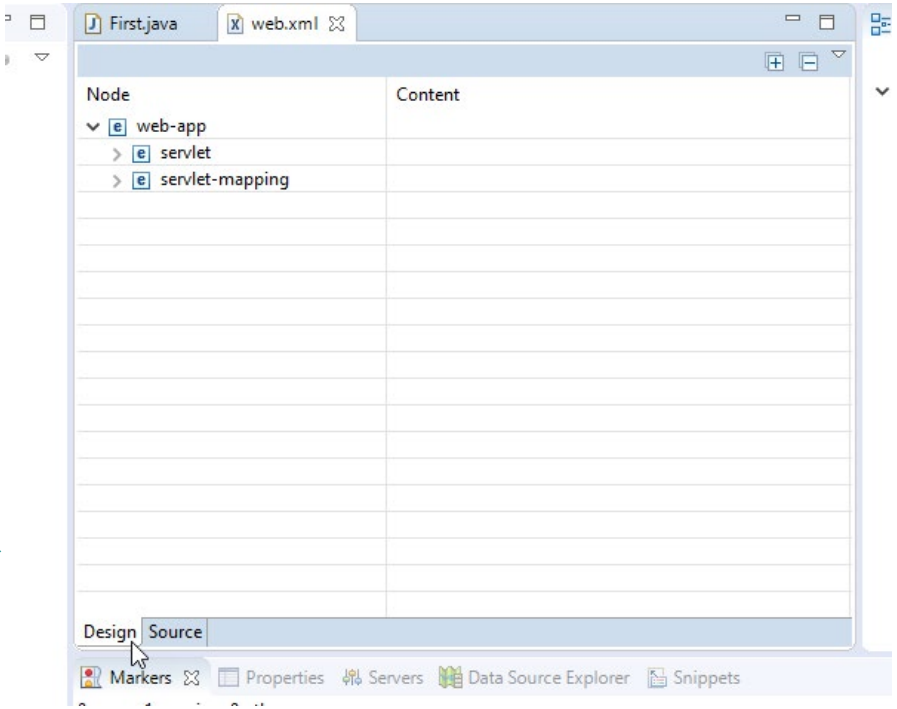
2. Create a Servlet - “web.xml”





3.3. Servlet with Eclipse

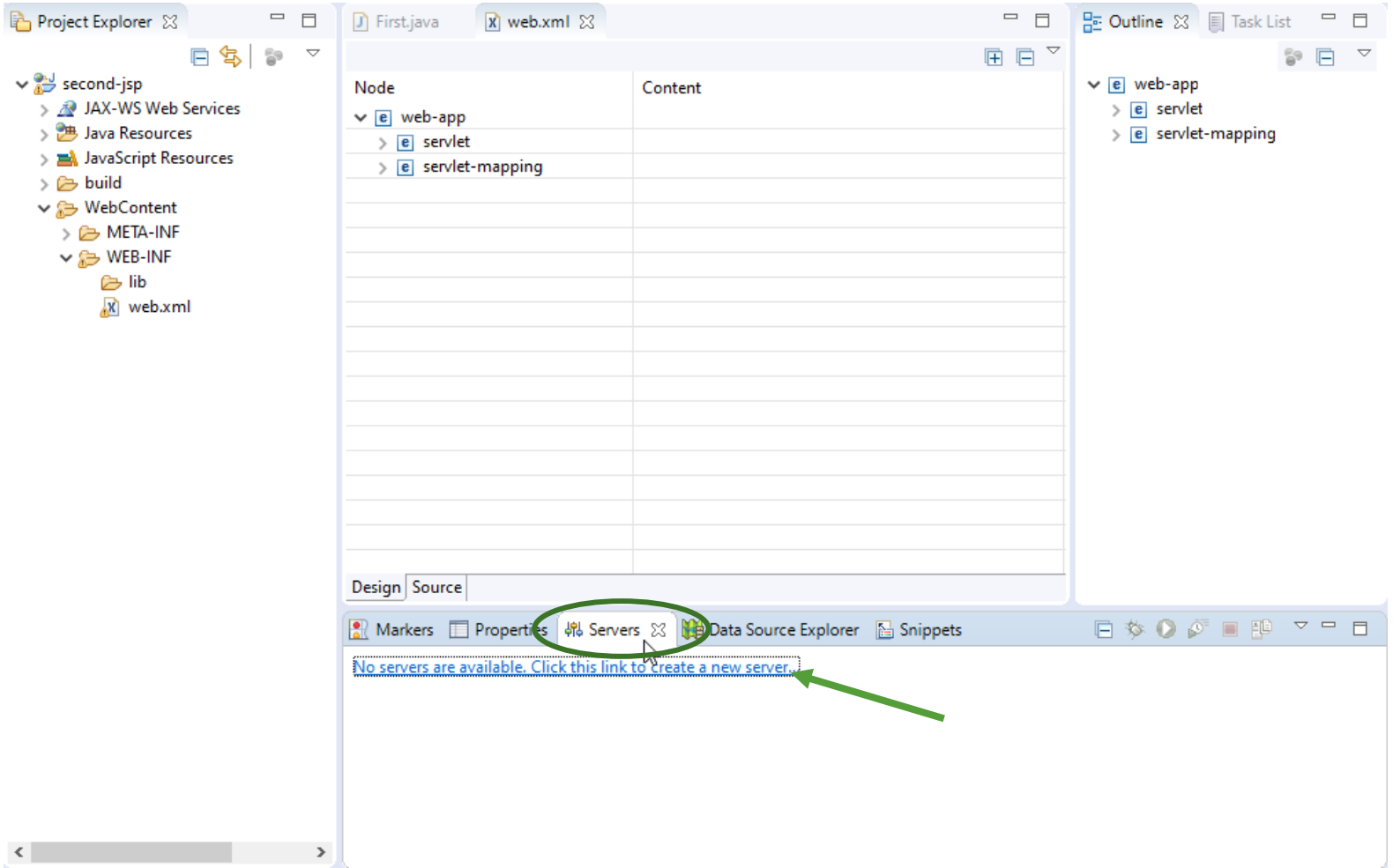
2. Create a Servlet - “web.xml”




```
<web-app>

  <servlet>
    <servlet-name>my-first-servlet</servlet-name>
    <servlet-class>First</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>my-first-servlet</servlet-name>
    <url-pattern>/welcome</url-pattern>
  </servlet-mapping>

</web-app>
```

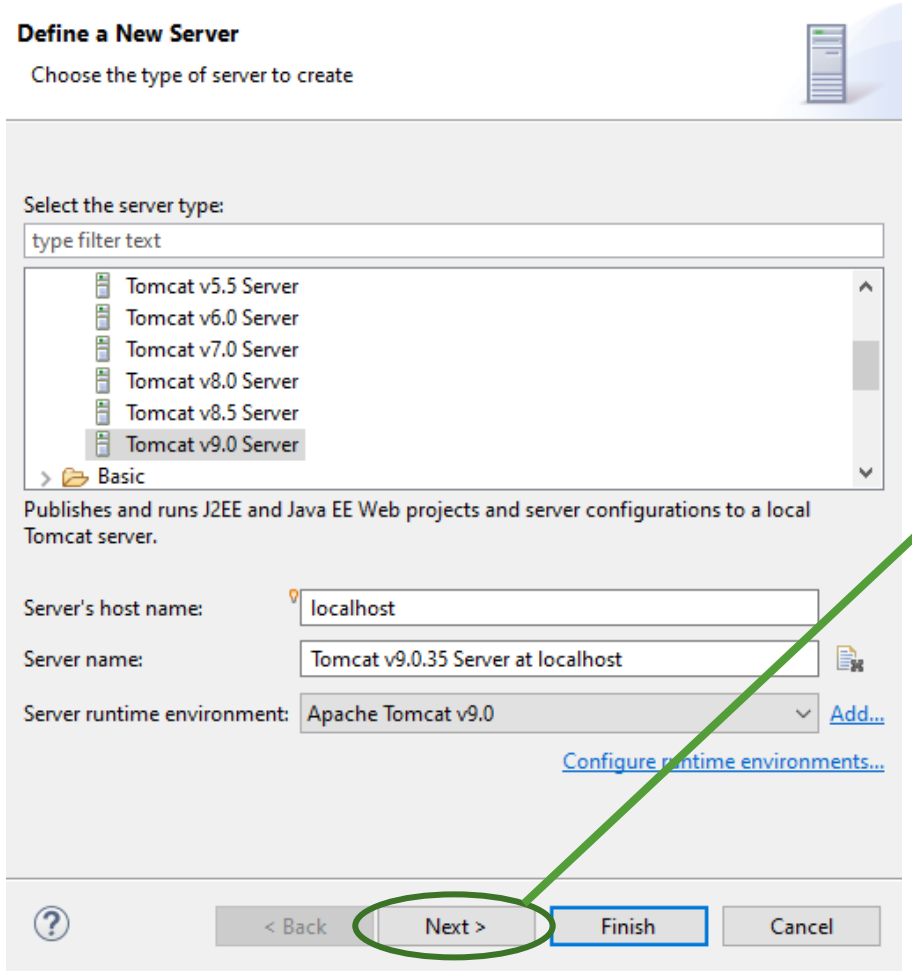
3.3. Servlet with Eclipse

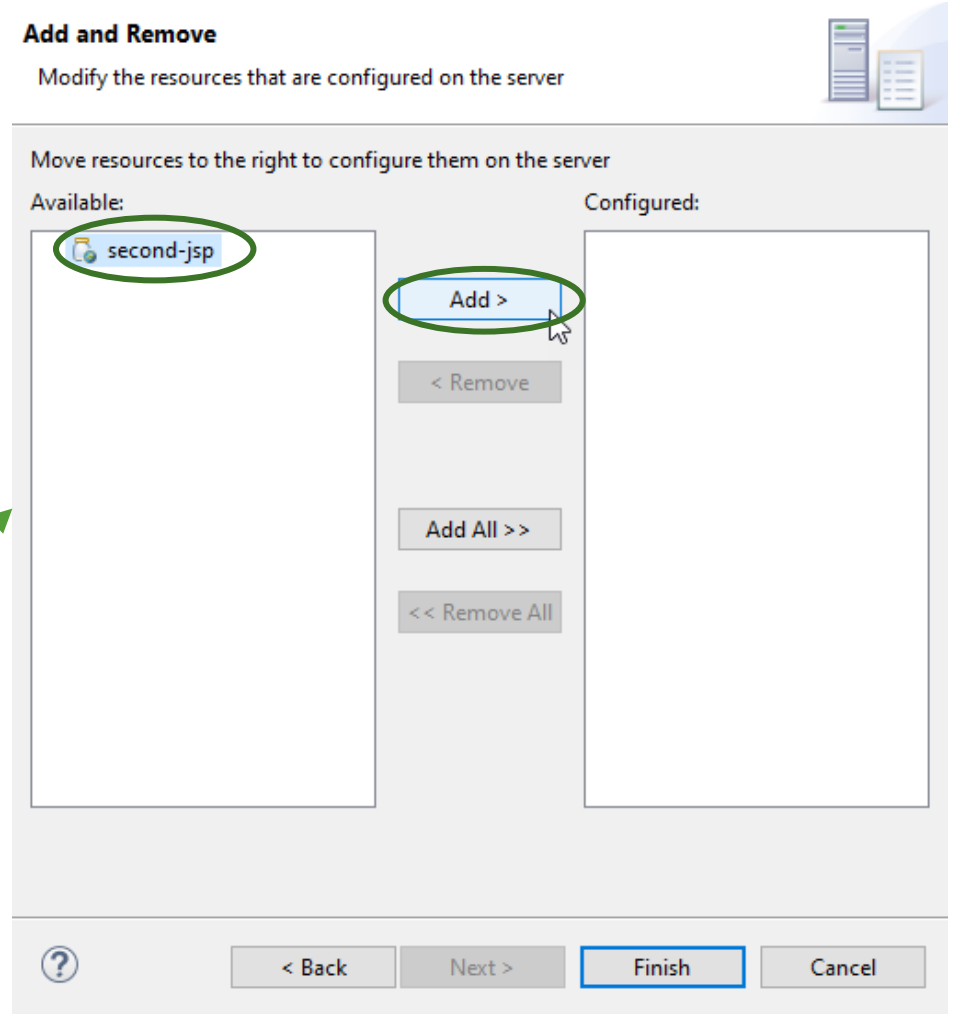
3. Run Servlet



3.3. Servlet with Eclipse

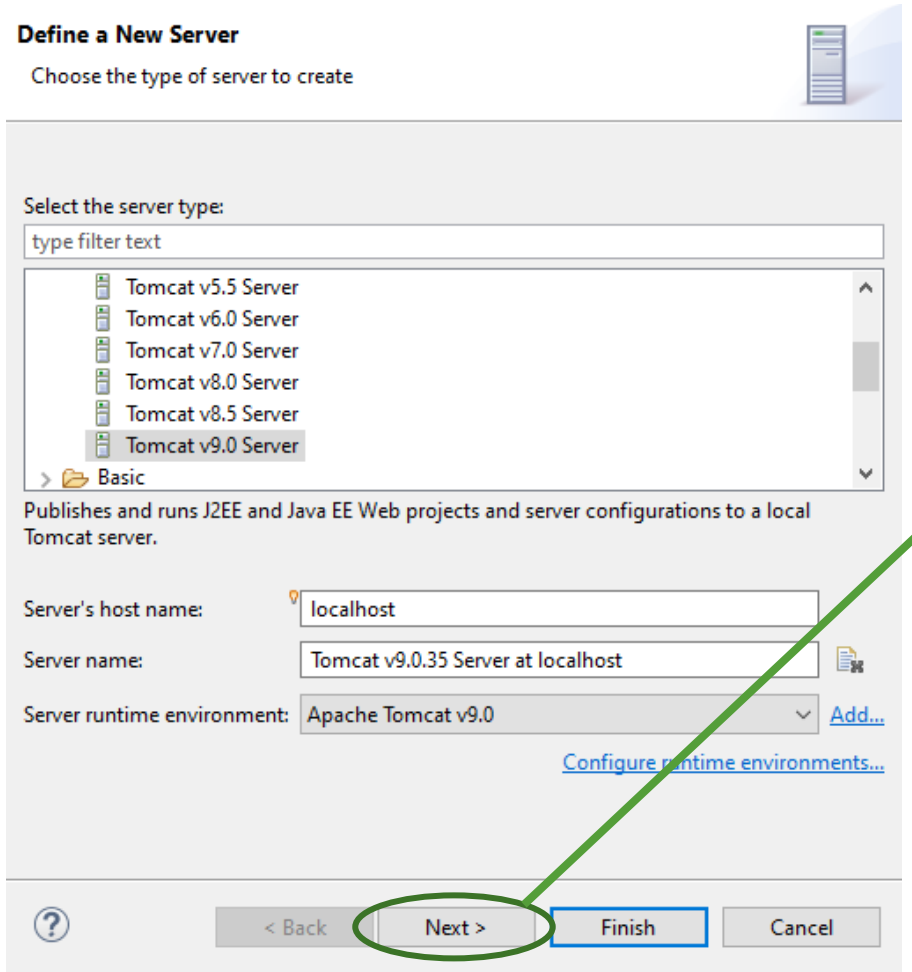
3. Run Servlet – Add Server

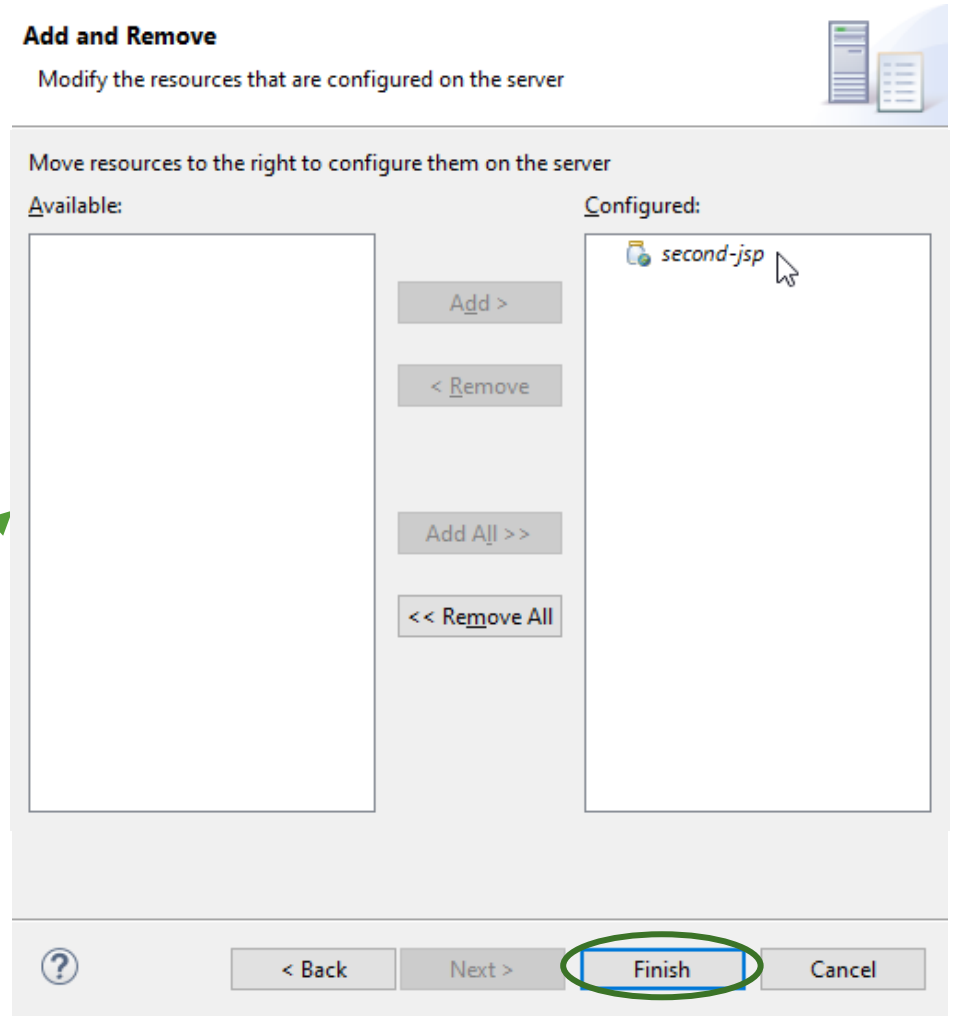




3.3. Servlet with Eclipse

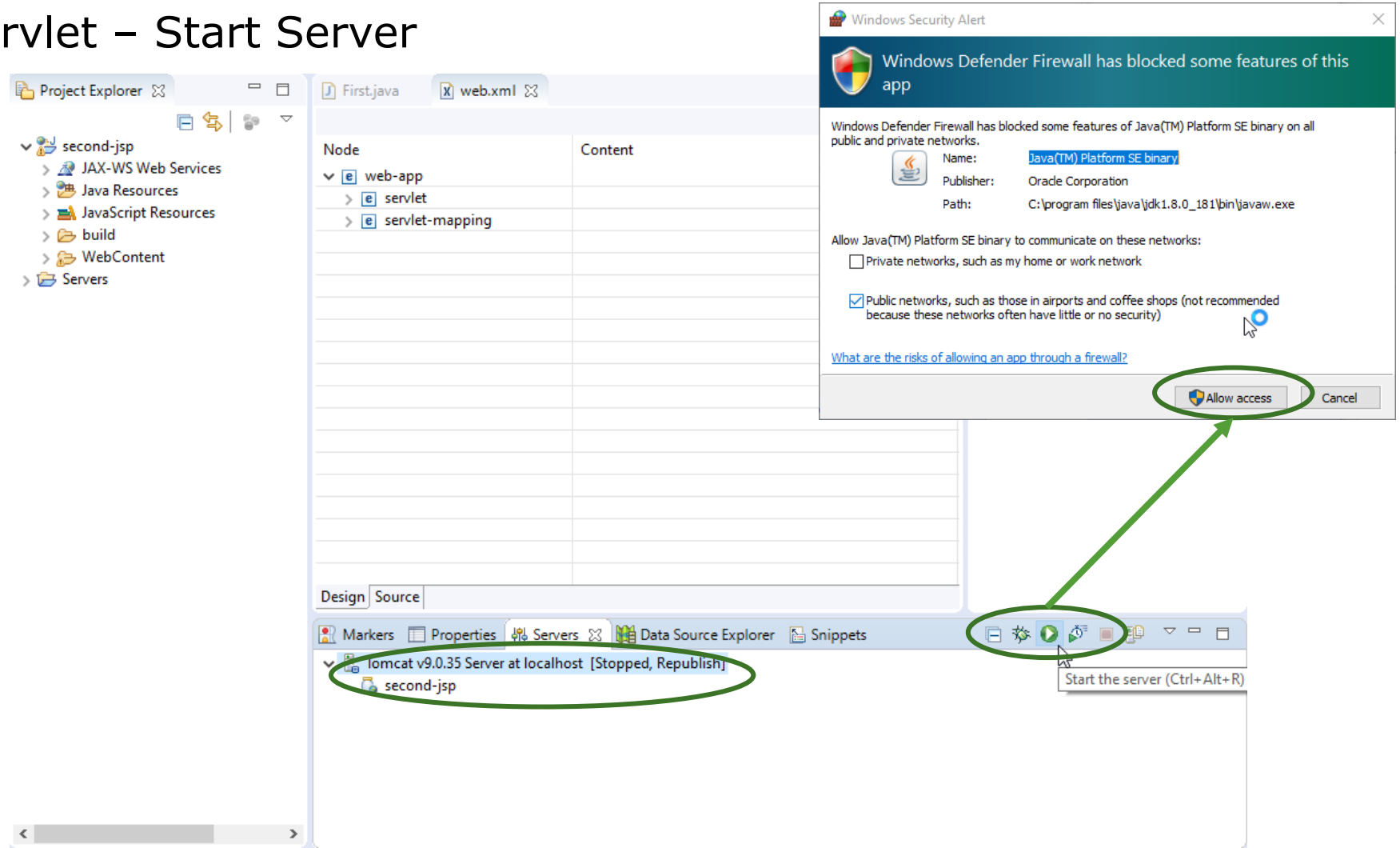
3. Run Servlet – Add Server





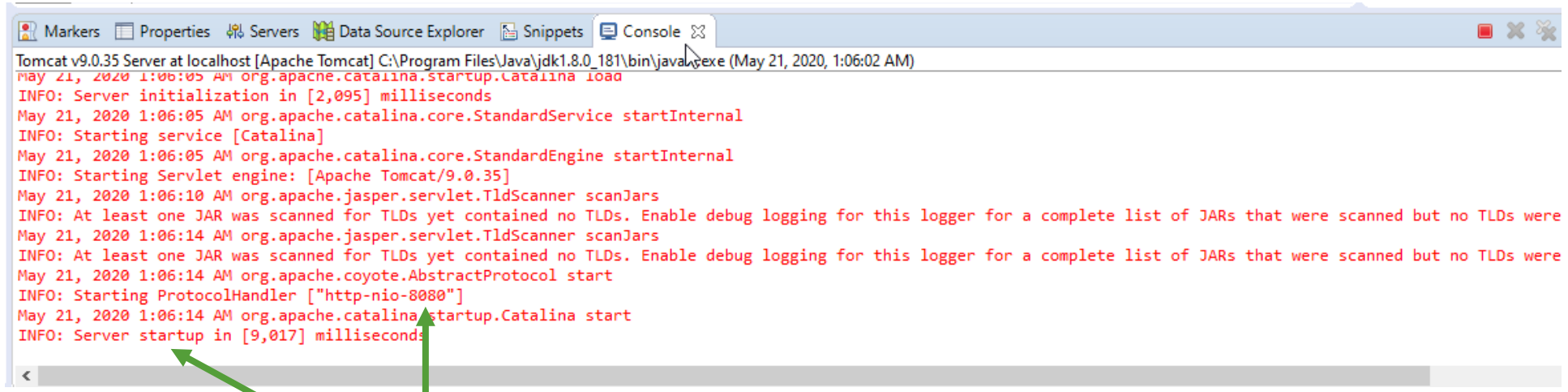
3.3. Servlet with Eclipse

3. Run Servlet – Start Server



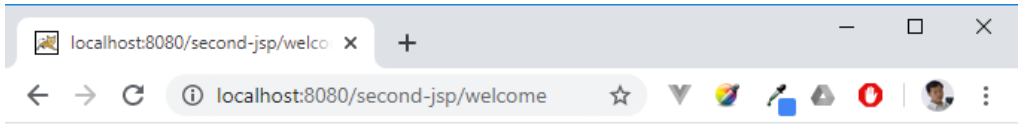
3.3. Servlet with Eclipse

3. Run Servlet – Check Server Status



Server is correctly started up at port 8080.

Now open browser and navigate to http://localhost:8080/second-jsp/w



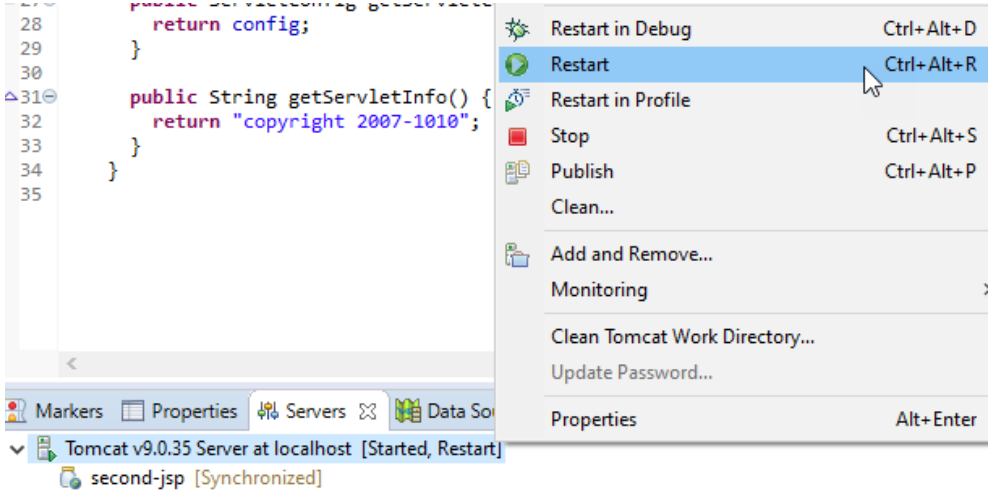
3.3. Servlet with Eclipse

3. Run Servlet – Updating codes

Now try to change some codes in First.java and then save it. You will notice the console says:

```
May 21, 2020 1:18:44 AM org.apache.catalina.core.StandardContext reload
INFO: Reloading Context with name [/second-jsp] has started
servlet is destroyed
May 21, 2020 1:18:49 AM org.apache.jasper.servlet.TldScanner scanJars
INFO: At least one JAR was scanned for TLDs yet contained no TLDs. Enable debug logg
May 21, 2020 1:18:49 AM org.apache.catalina.core.StandardContext reload
INFO: Reloading Context with name [/second-jsp] is completed
```

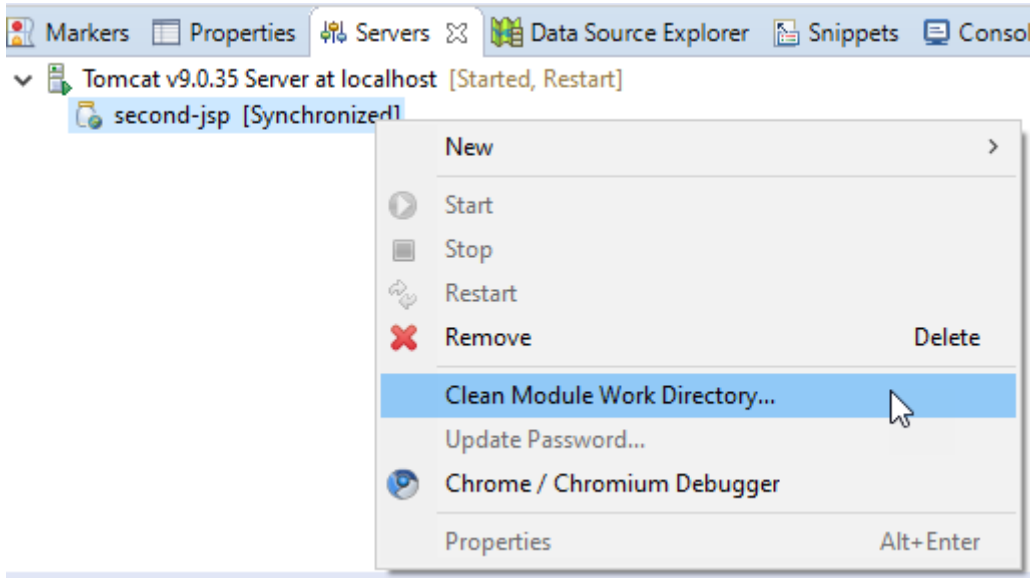
But sometimes it w
reload, so we neec
restart the server.



3.3. Servlet with Eclipse

3. Run Servlet – Updating codes

Sometimes when we rename class or restructure classes, we need clean cache:



References

- https://www.javatpoint.com/steps-to-create-a-servlet-using-tomcat-server
- https://www.javatpoint.com/jsp-tutorial
- https://www.javatpoint.com/servlet-tutorial
- https://www.javatpoint.com/Servlet-interface