Ex.No : 3	Implementation of SOAP and RESTful web services in Java
Date:	

Aim:

To implement the SOAP and RESTFUL Web services in Java for our Application's Signup Page along with its installation processes.

Web Services:

A web service is any piece of software that makes itself available over the internet and uses a standardized XML messaging system. XML is used to encode all communications to a web service. For example, a client invokes a web service by sending an XML message, then waits for a corresponding XML response.

SOAP Web Services:

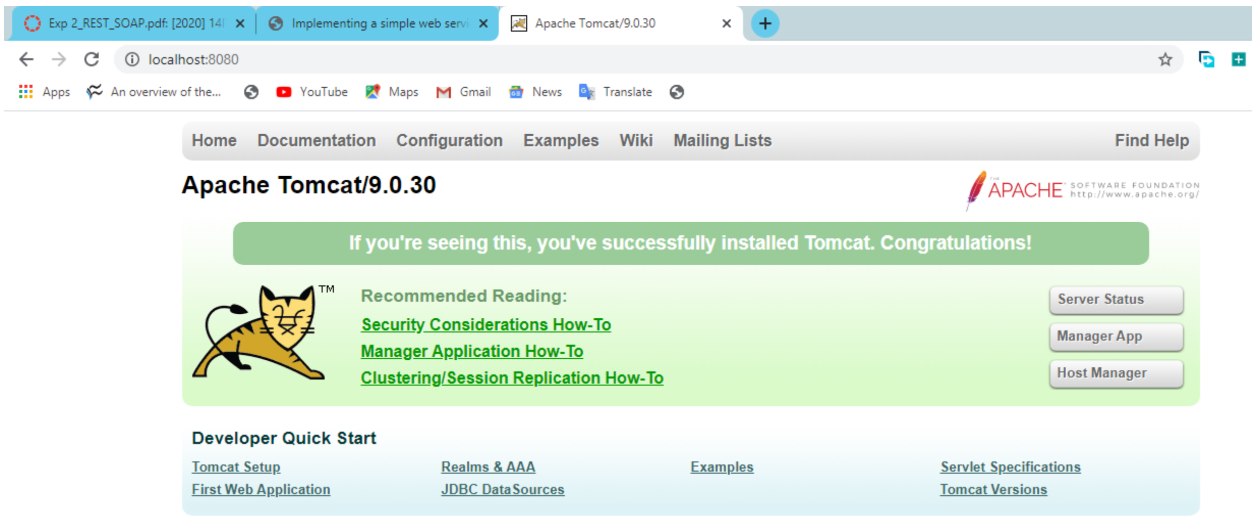
SOAP is an XML based industry standard protocol for designing and developing **web services.** Since it's XML based, it's platform and language independent. So our server can be based on **JAVA** and client can be on .

RESTFUL Web Services:

RESTful Web Services are basically **REST** Architecture based **Web Services**. In **REST** Architecture everything is a resource. **RESTful web services** are light weight, highly scalable and maintainable and are very commonly used to create APIs for **web-**based applications.

Web Service Installation:

- Check the Apache Tomcat/9.0.30 is properly installed or not by entering **localhost:8080** in Browser's URL Field.



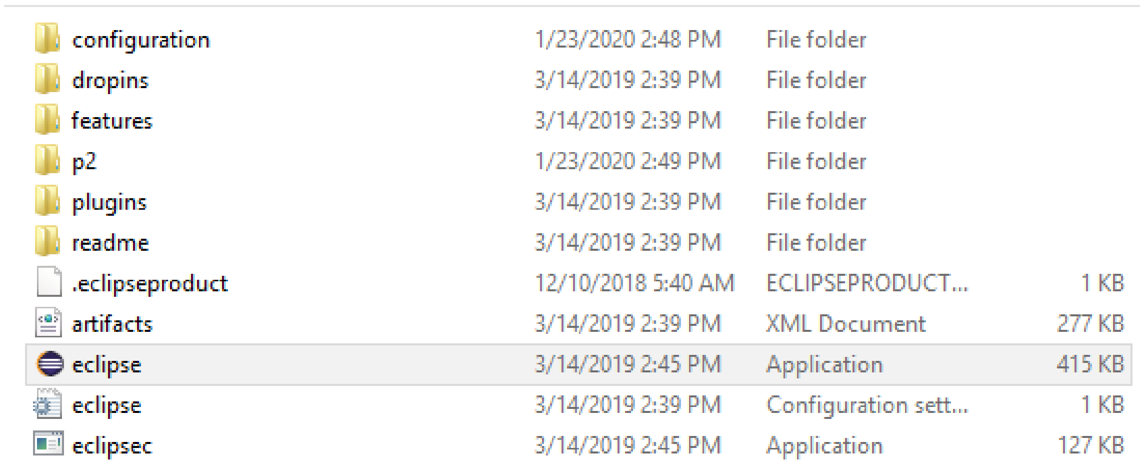
Apache Tomcat/9.0.30

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

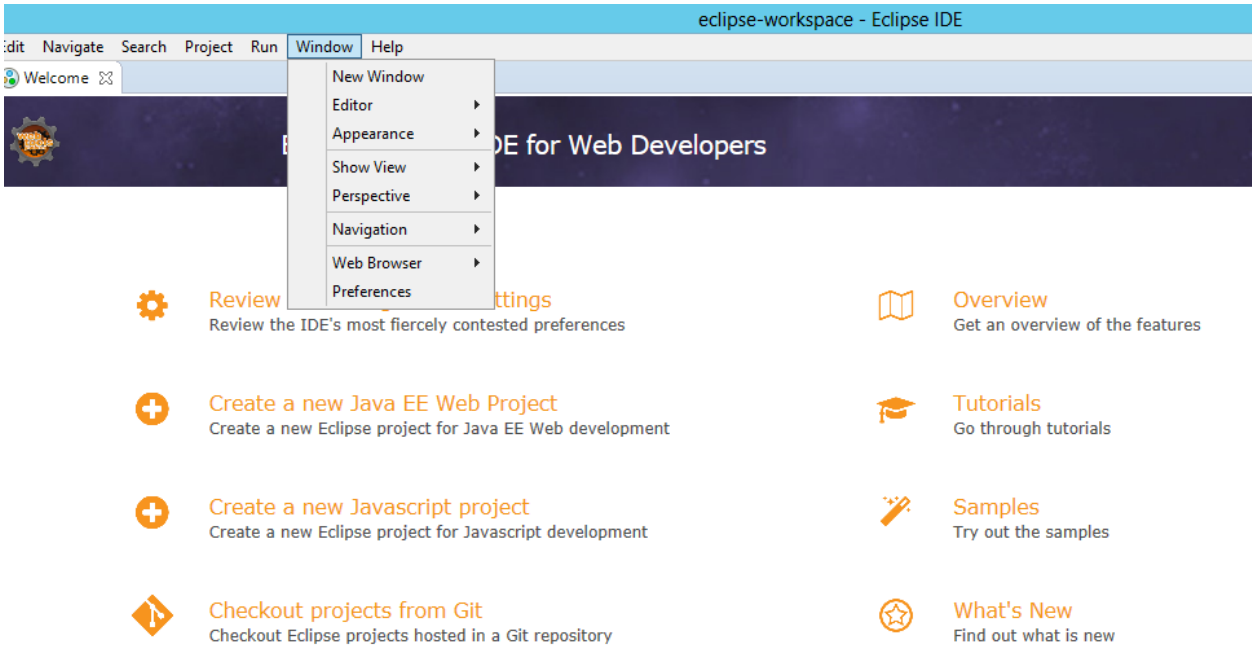
Developer Quick Start
Tomcat Setup
First Web Application
Realms & AAA
JDBC DataSources
Examples
Servlet Specifications
Tomcat Versions

- Opening the Eclipse IDE to run our Program



configuration	1/23/2020 2:48 PM	File folder	
dropins	3/14/2019 2:39 PM	File folder	
features	3/14/2019 2:39 PM	File folder	
p2	1/23/2020 2:49 PM	File folder	
plugins	3/14/2019 2:39 PM	File folder	
readme	3/14/2019 2:39 PM	File folder	
.eclipseproduct	12/10/2018 5:40 AM	ECLIPSEPRODUCT...	1 KB
artifacts	3/14/2019 2:39 PM	XML Document	277 KB
eclipse	3/14/2019 2:45 PM	Application	415 KB
eclipse	3/14/2019 2:39 PM	Configuration sett...	1 KB
eclipsec	3/14/2019 2:45 PM	Application	127 KB

➢ After Opening the Eclipse select the **Window** tab then choose the **Preference** which is listed when clicking the window tab in navigation bar.



Review ttings
Review the IDE's most fiercely contested preferences

Overview
Get an overview of the features

Create a new Java EE Web Project
Create a new Eclipse project for Java EE Web development

Tutorials
Go through tutorials

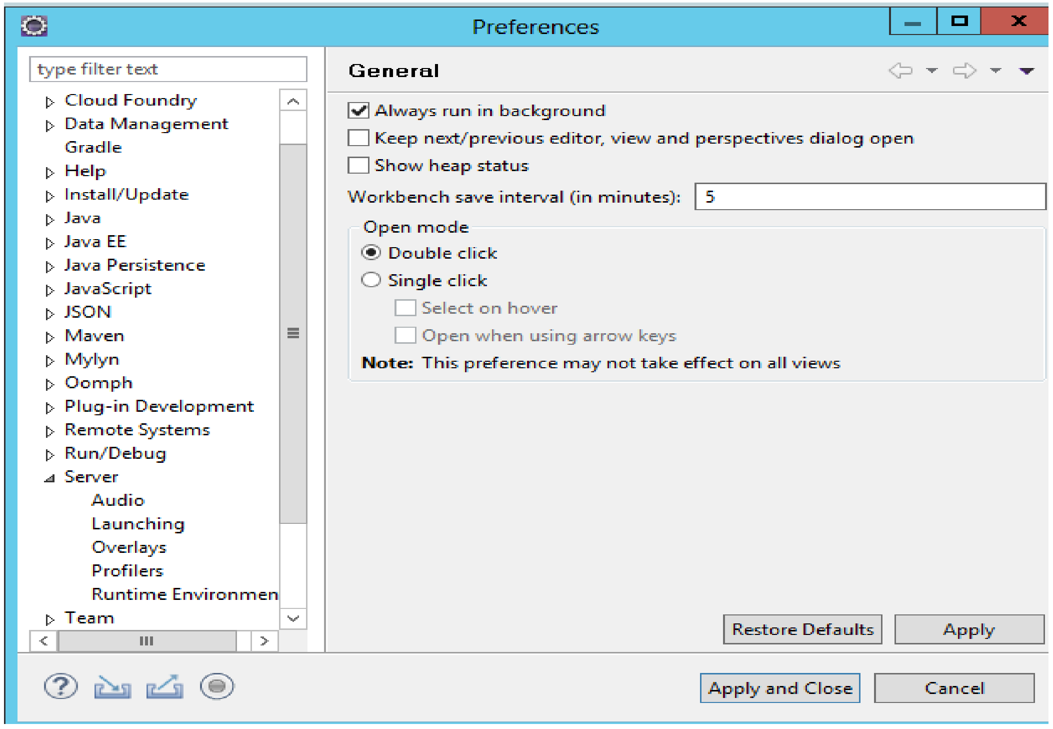
Create a new Javascript project
Create a new Eclipse project for Javascript development

Samples
Try out the samples

Checkout projects from Git
Checkout Eclipse projects hosted in a Git repository

What's New
Find out what is new

➢ Select **Run time environment** listed from **Server** tab



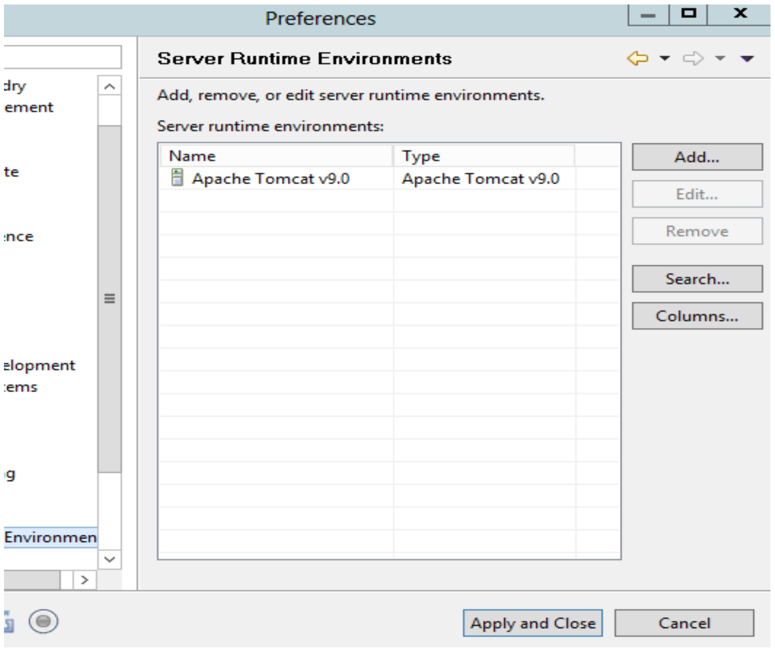
- [✓] Always run in background
- [ ] Keep next/previous editor, view and perspectives dialog open
- [ ] Show heap status

Workbench save interval (in minutes): 5

Open mode
- [✓ Double click]
- Single click
  - [ ] Select on hover
  - [ ] Open when using arrow keys

Note: This preference may not take effect on all views

➢ Add Apache Tomcat Server v9.0 in Run time Environment

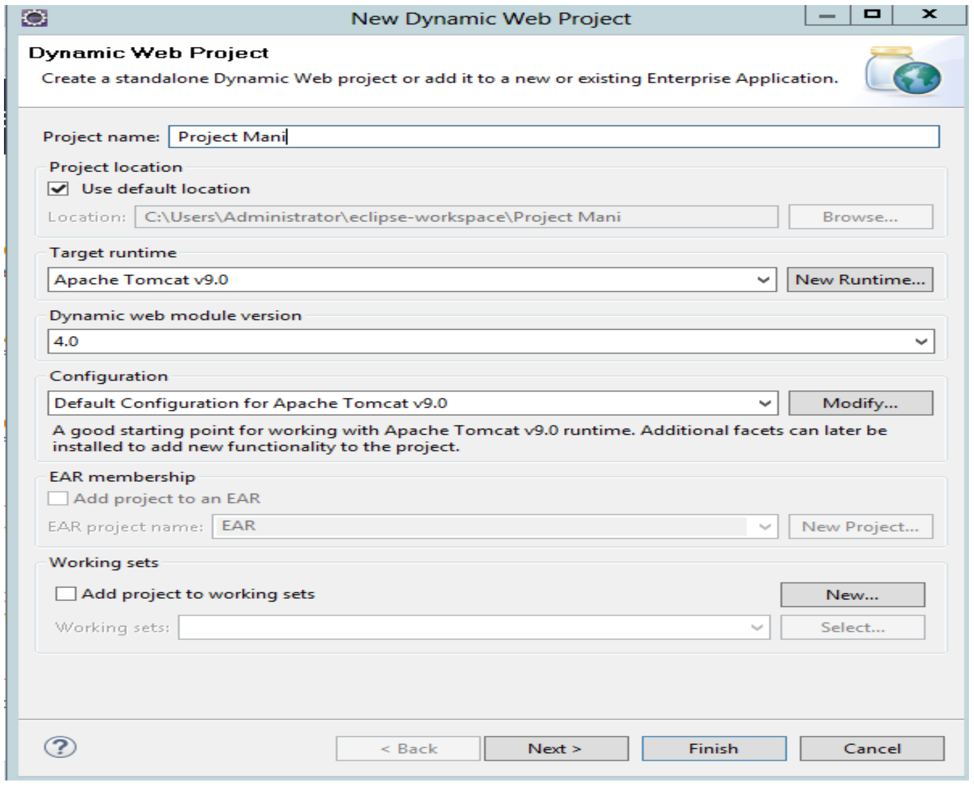


Add, remove, or edit server runtime environments.

Server runtime environments:

Name	Type
Apache Tomcat v9.0	Apache Tomcat v9.0

➢ Now Creating the **New Web Project** with the Target Run time Environment as Apache Tomcat Server



Dynamic Web Project

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name: Project Mani

Project location

[✓] Use default location

Location: C:\Users\Administrator\eclipse-workspace\Project Mani

Target runtime

Apache Tomcat v9.0

Dynamic web module version

4.0

Configuration

Default Configuration for Apache Tomcat v9.0

A good starting point for working with Apache Tomcat v9.0 runtime. Additional facets can later be installed to add new functionality to the project.

EAR membership

[ ] Add project to an EAR

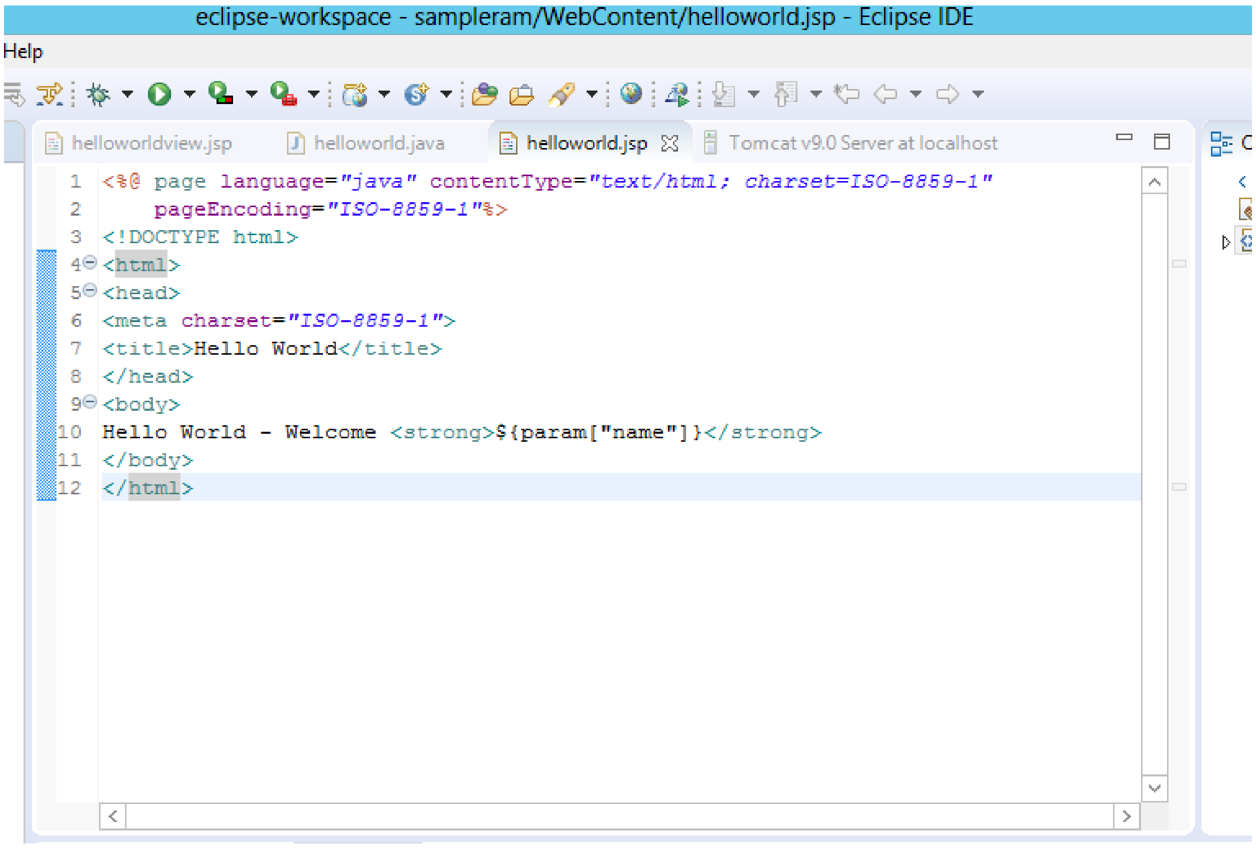
EAR project name: EAR

Working sets

[ ] Add project to working sets

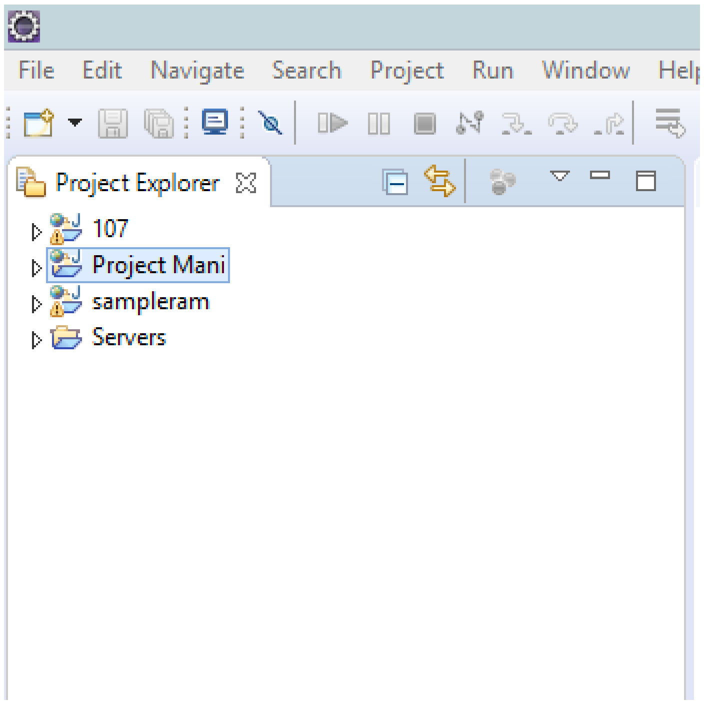
Working sets:

➢ Now, We can Create our **jsp file** which consists of some **HTML Tags**

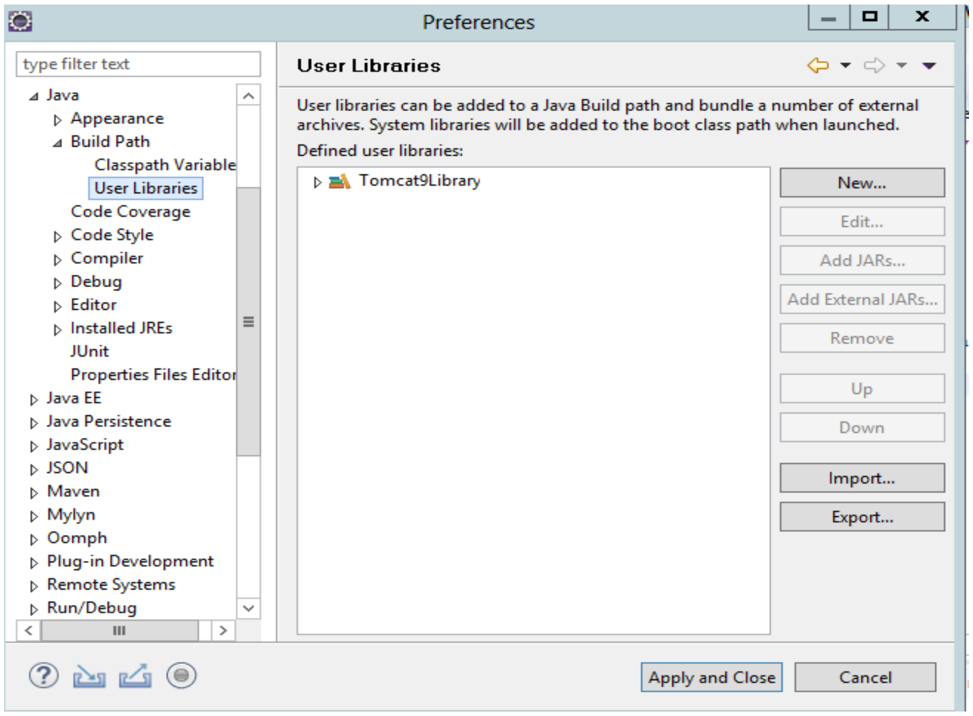


eclipse-workspace - sampleram/WebContent/helloworld.jsp - Eclipse IDE

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Hello World</title>
</head>
<body>
Hello World - Welcome <strong>${param["name"]}</strong>
</body>
</html>
```



- We need to add some Jar file in libraries folder of the created Web Project

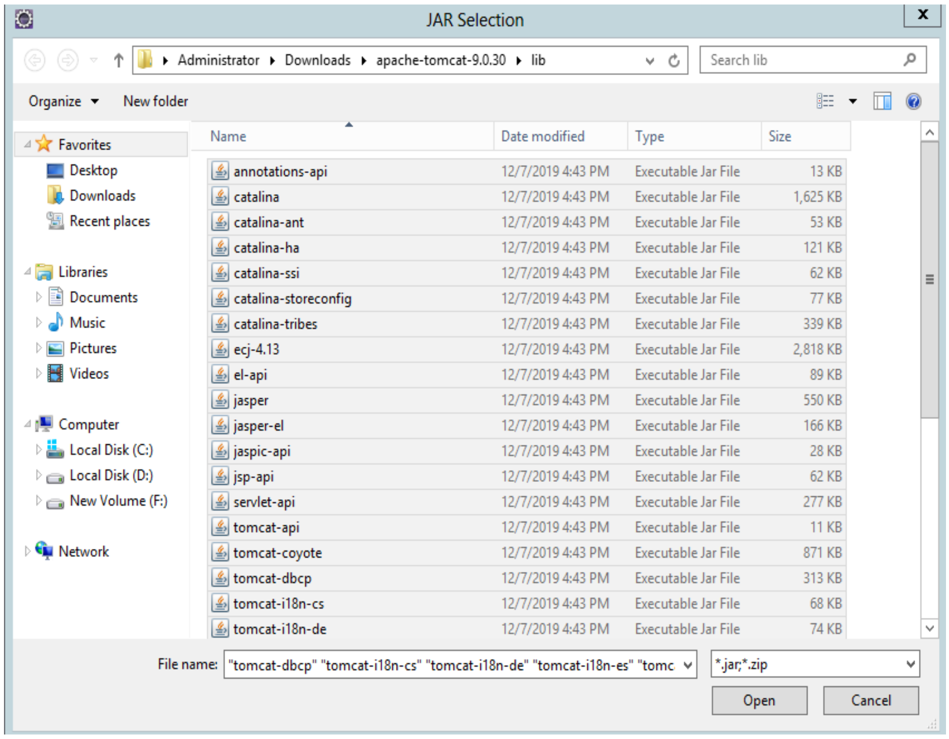


User libraries can be added to a Java Build path and bundle a number of external archives. System libraries will be added to the boot class path when launched.

Defined user libraries:

Tomcat9Library

- Select all jar files which are installed previously

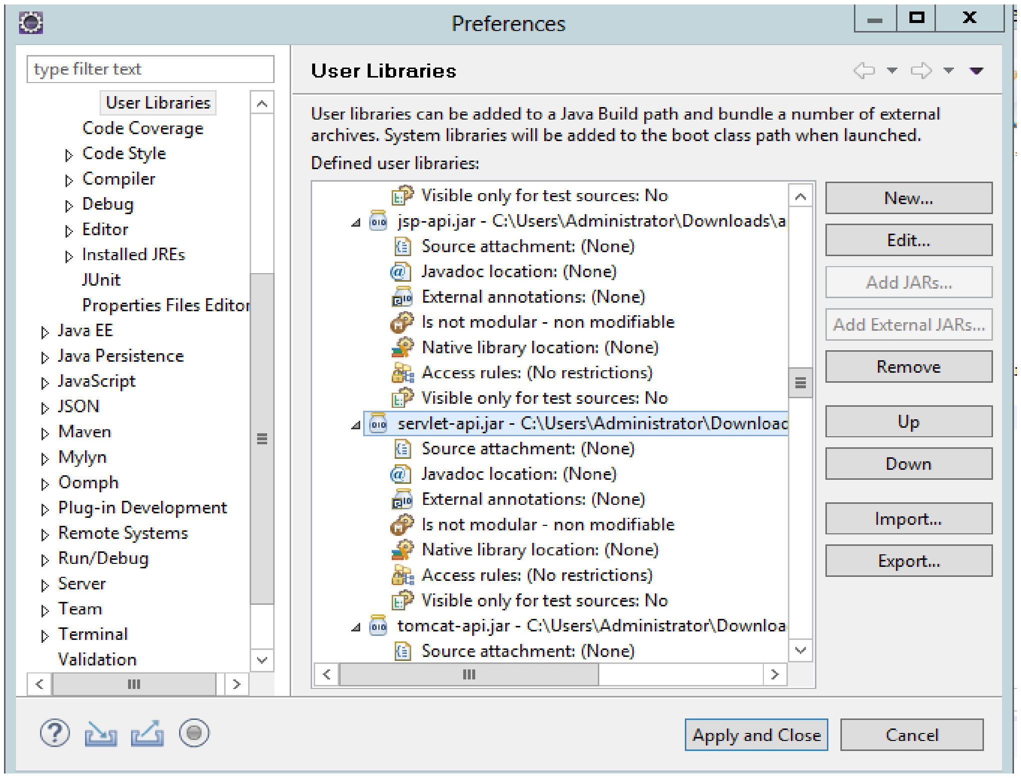


Name	Date modified	Type	Size
annotations-api	12/7/2019 4:43 PM	Executable Jar File	13 KB
catalina	12/7/2019 4:43 PM	Executable Jar File	1,625 KB
catalina-ant	12/7/2019 4:43 PM	Executable Jar File	53 KB
catalina-ha	12/7/2019 4:43 PM	Executable Jar File	121 KB
catalina-ssi	12/7/2019 4:43 PM	Executable Jar File	62 KB
catalina-storeconfig	12/7/2019 4:43 PM	Executable Jar File	77 KB
catalina-tribes	12/7/2019 4:43 PM	Executable Jar File	339 KB
ecj-4.13	12/7/2019 4:43 PM	Executable Jar File	2,818 KB
el-api	12/7/2019 4:43 PM	Executable Jar File	89 KB
jasper	12/7/2019 4:43 PM	Executable Jar File	550 KB
jasper-el	12/7/2019 4:43 PM	Executable Jar File	166 KB
jaspic-api	12/7/2019 4:43 PM	Executable Jar File	28 KB
jsp-api	12/7/2019 4:43 PM	Executable Jar File	62 KB
servlet-api	12/7/2019 4:43 PM	Executable Jar File	277 KB
tomcat-api	12/7/2019 4:43 PM	Executable Jar File	11 KB
tomcat-coyote	12/7/2019 4:43 PM	Executable Jar File	871 KB
tomcat-dbcp	12/7/2019 4:43 PM	Executable Jar File	313 KB
tomcat-i18n-cs	12/7/2019 4:43 PM	Executable Jar File	68 KB
tomcat-i18n-de	12/7/2019 4:43 PM	Executable Jar File	74 KB

File name: "tomcat-dbcp" "tomcat-i18n-cs" "tomcat-i18n-de" "tomcat-i18n-es" "tomc

.jar;.zip

- Check the important jar files like **servlet-apt.jar** file are copied perfectly or not.


Preferences

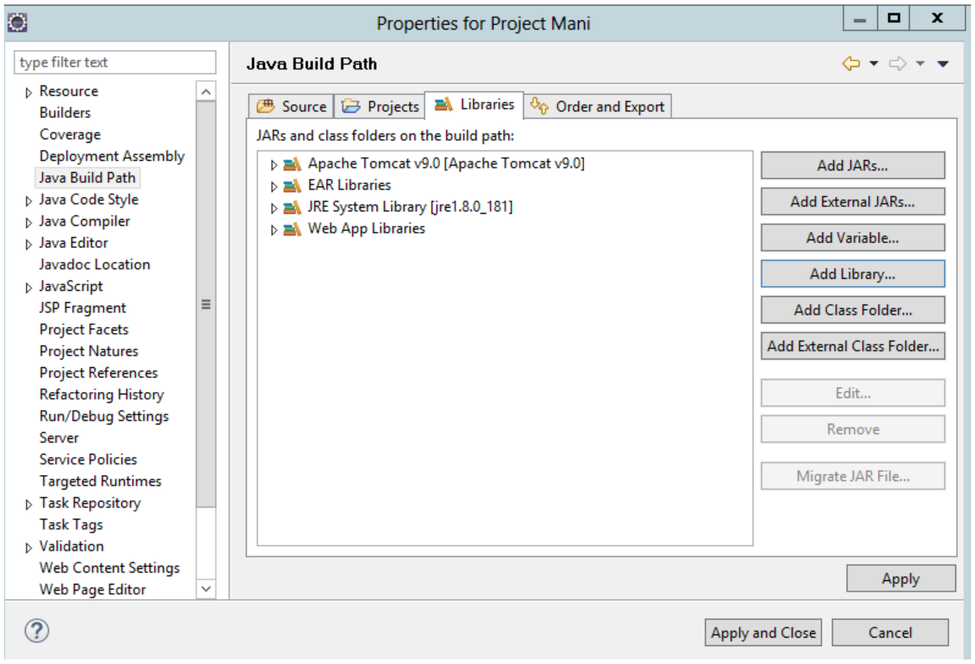
User Libraries

User libraries can be added to a Java Build path and bundle a number of external archives. System libraries will be added to the boot class path when launched.

Defined user libraries:

```
      Visible only for test sources: No
   jsp-api.jar - C:\Users\Administrator\Downloads\a
      Source attachment: (None)
      Javadoc location: (None)
      External annotations: (None)
      Is not modular - non modifiable
      Native library location: (None)
      Access rules: (No restrictions)
      Visible only for test sources: No
   servlet-api.jar - C:\Users\Administrator\Downloac
      Source attachment: (None)
      Javadoc location: (None)
      External annotations: (None)
      Is not modular - non modifiable
      Native library location: (None)
      Access rules: (No restrictions)
      Visible only for test sources: No
   tomcat-api.jar - C:\Users\Administrator\Downloa
      Source attachment: (None)
```

- Click the Java Build path tab to view all our imported jar files

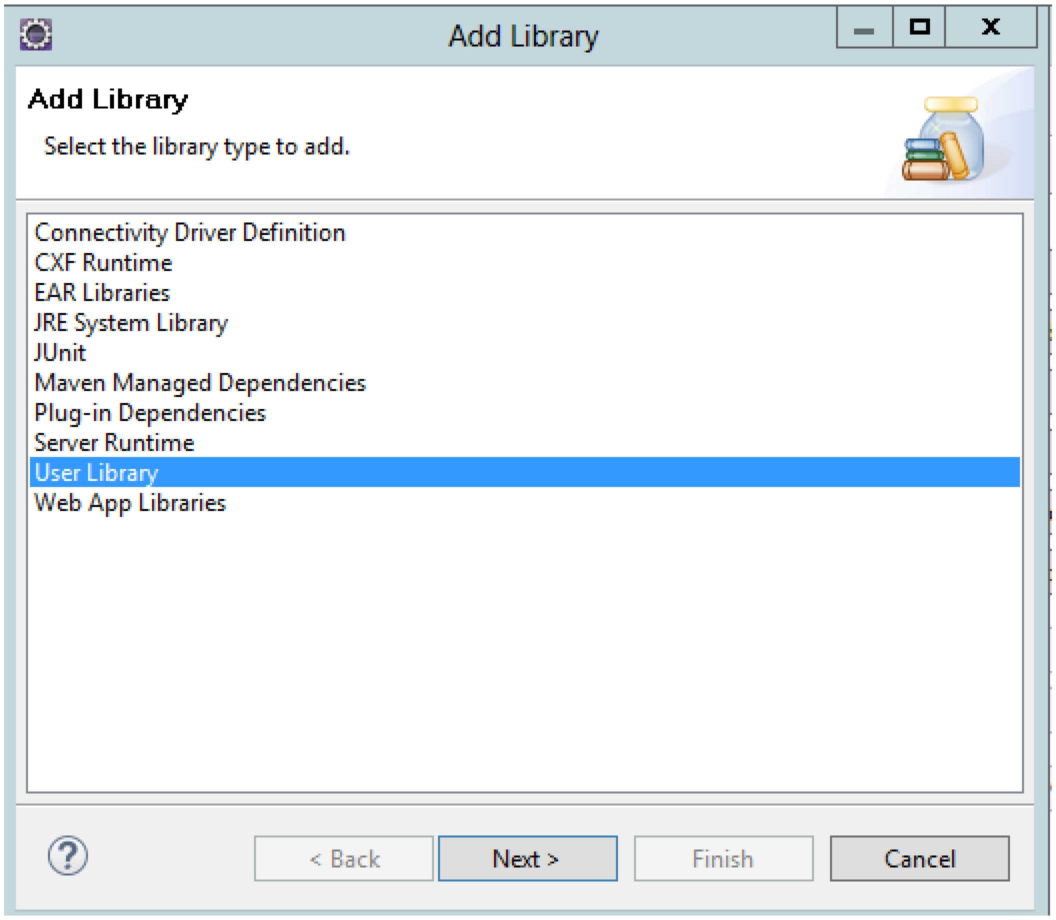

Properties for Project Mani

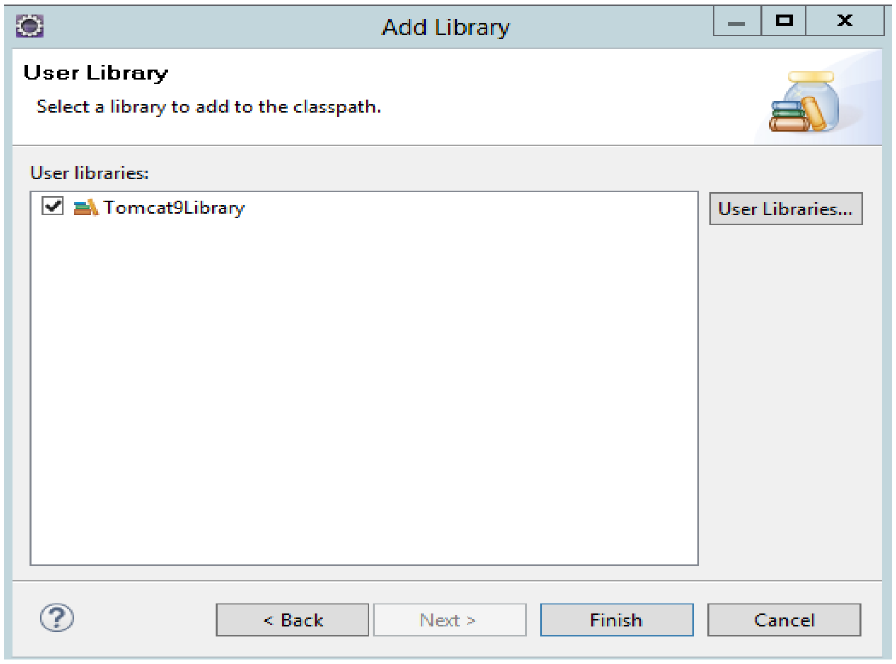
Java Build Path

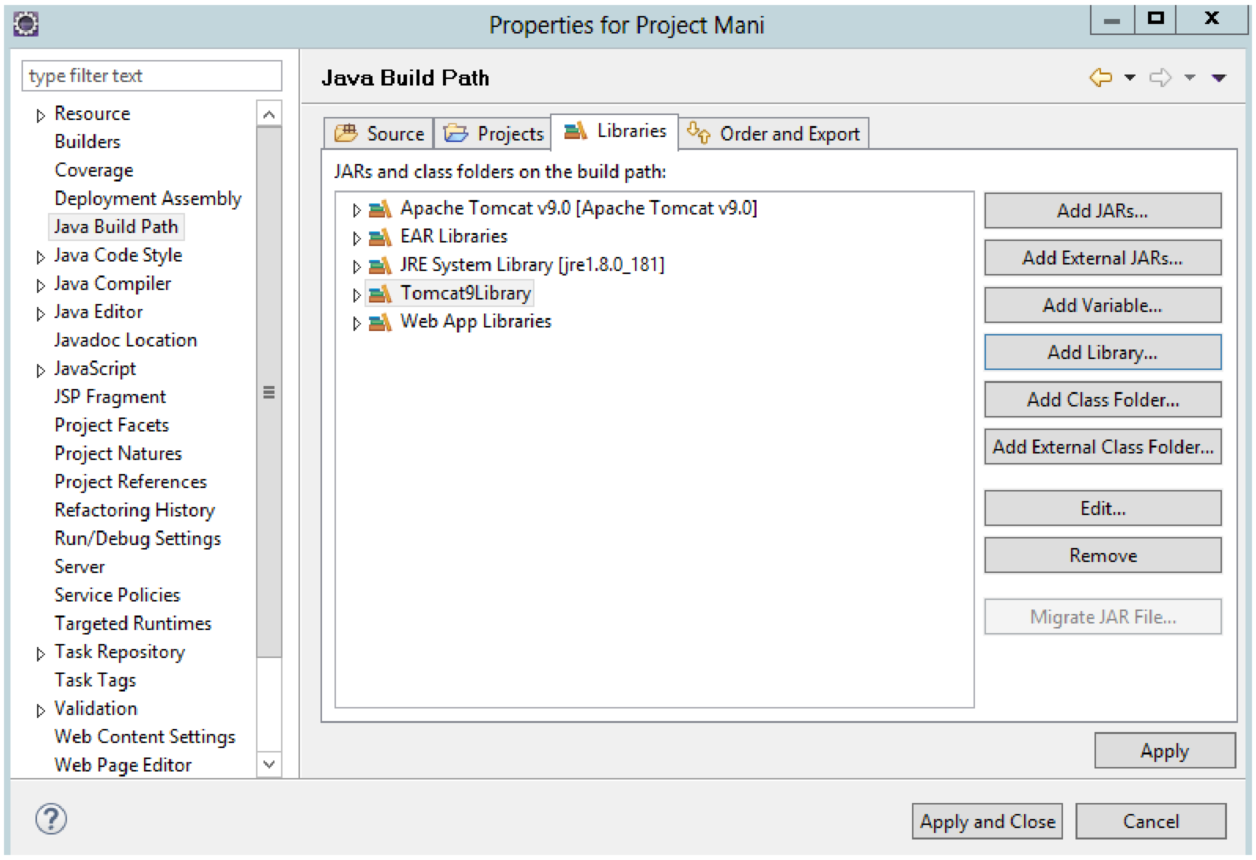
JARs and class folders on the build path:

```
Apache Tomcat v9.0 [Apache Tomcat v9.0]
EAR Libraries
JRE System Library [jre1.8.0_181]
Web App Libraries
```

➢ We can add our library file manually by Clicking Add library tab the source file is downloaded from any third party web site

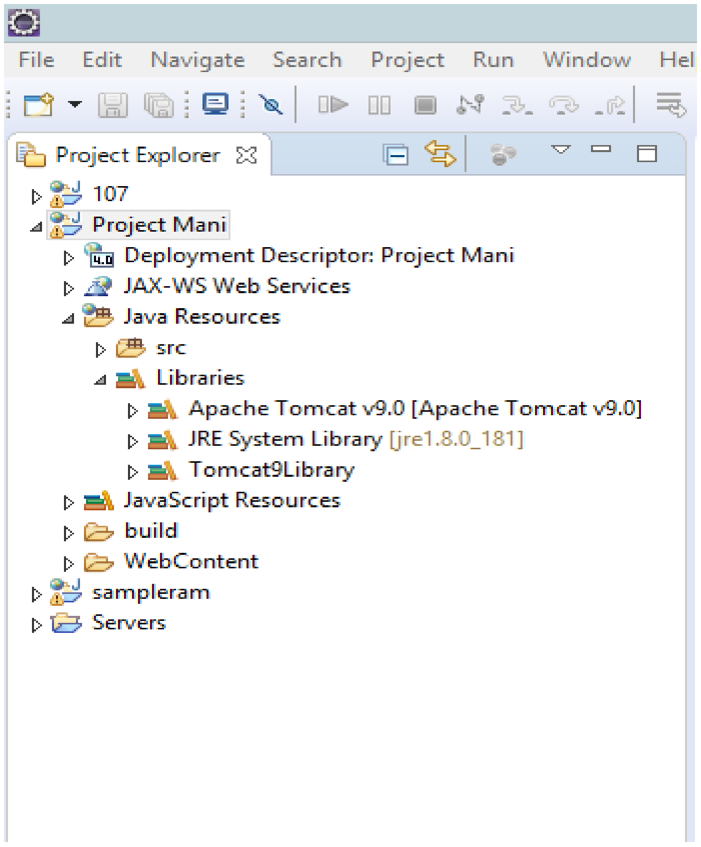






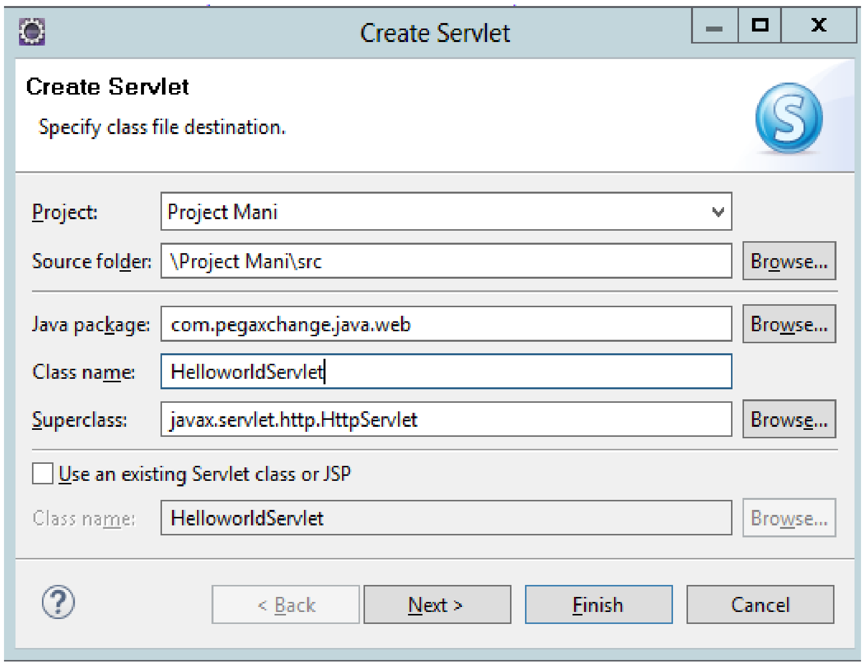
JARs and class folders on the build path:

- Apache Tomcat v9.0 [Apache Tomcat v9.0]
- EAR Libraries
- JRE System Library [jre1.8.0_181]
- Tomcat9Library
- Web App Libraries



```
▷ 107
◢ Project Mani
    ▷ Deployment Descriptor: Project Mani
    ▷ JAX-WS Web Services
    ◢ Java Resources
        ▷ src
        ◢ Libraries
            ▷ Apache Tomcat v9.0 [Apache Tomcat v9.0]
            ▷ JRE System Library [jre1.8.0_181]
            ▷ Tomcat9Library
    ▷ JavaScript Resources
    ▷ build
    ▷ WebContent
▷ sampleram
▷ Servers
```

➤ Import our java package as a **com.pegaxchange.java.web** and class name as HelloworldServlet



Create Servlet

Specify class file destination.

Project: Project Mani

Source folder: \Project Mani\src

Java package: com.pegaxchange.java.web

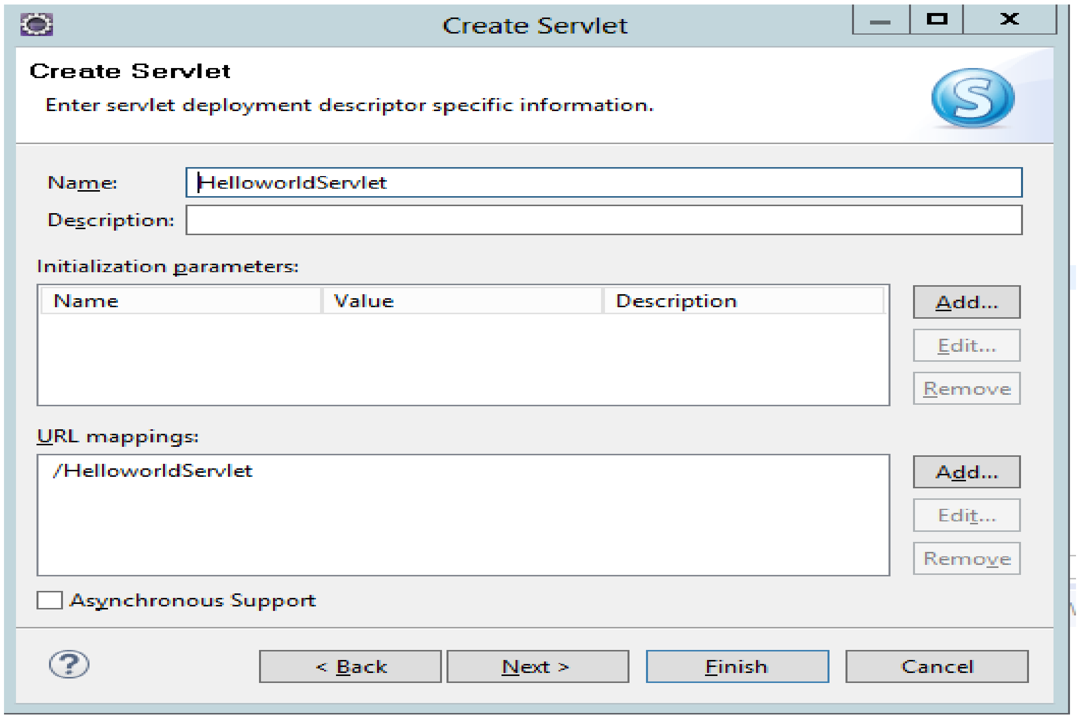
Class name: HelloworldServlet

Superclass: javax.servlet.http.HttpServlet

☐ Use an existing Servlet class or JSP

Class name: HelloworldServlet

➤ Servlet Creation Process



Create Servlet

Enter servlet deployment descriptor specific information.

Name: HelloworldServlet

Description:

Initialization parameters:

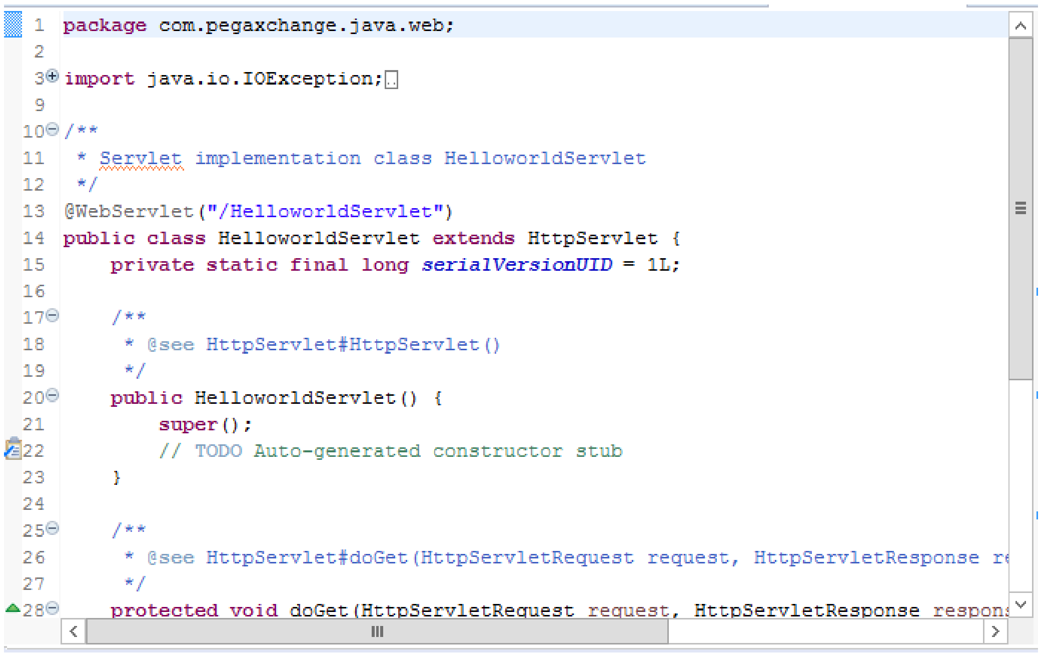
Name	Value	Description

URL mappings:

/HelloworldServlet

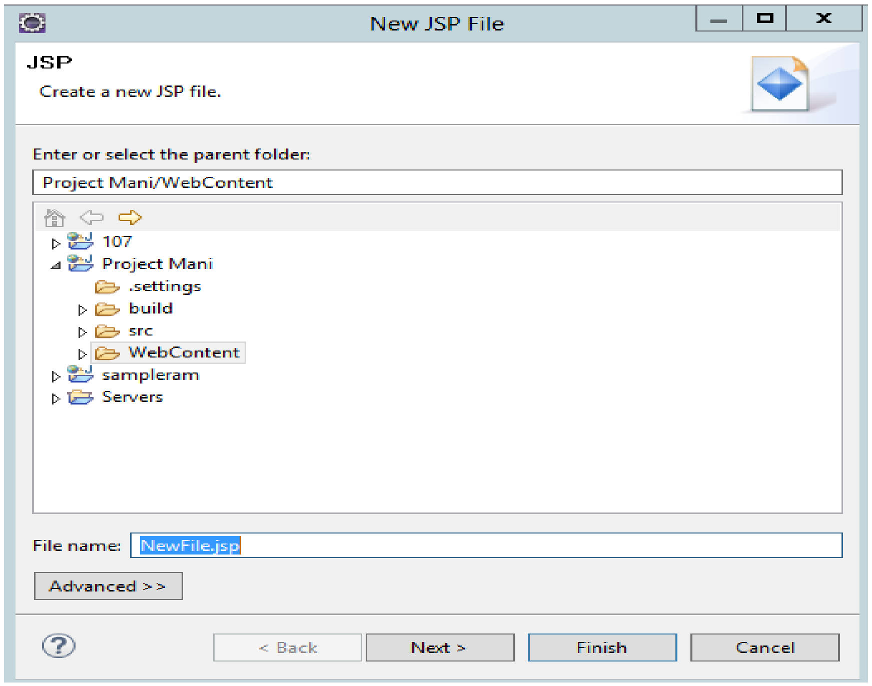
☐ Asynchronous Support

➢ Java Code implementation of the Web Project



```
package com.pegaxchange.java.web;

import java.io.IOException;

/**
 * Servlet implementation class HelloworldServlet
 */
@WebServlet("/HelloworldServlet")
public class HelloworldServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public HelloworldServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse re
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse respons
```

➢ Creating the new JSP file by **RIGHT** Clicking the project name. Purpose of JSP file is to made a link between the Java File and the HTML file which is used to implement the Client Side of the Web Page



New JSP File

JSP

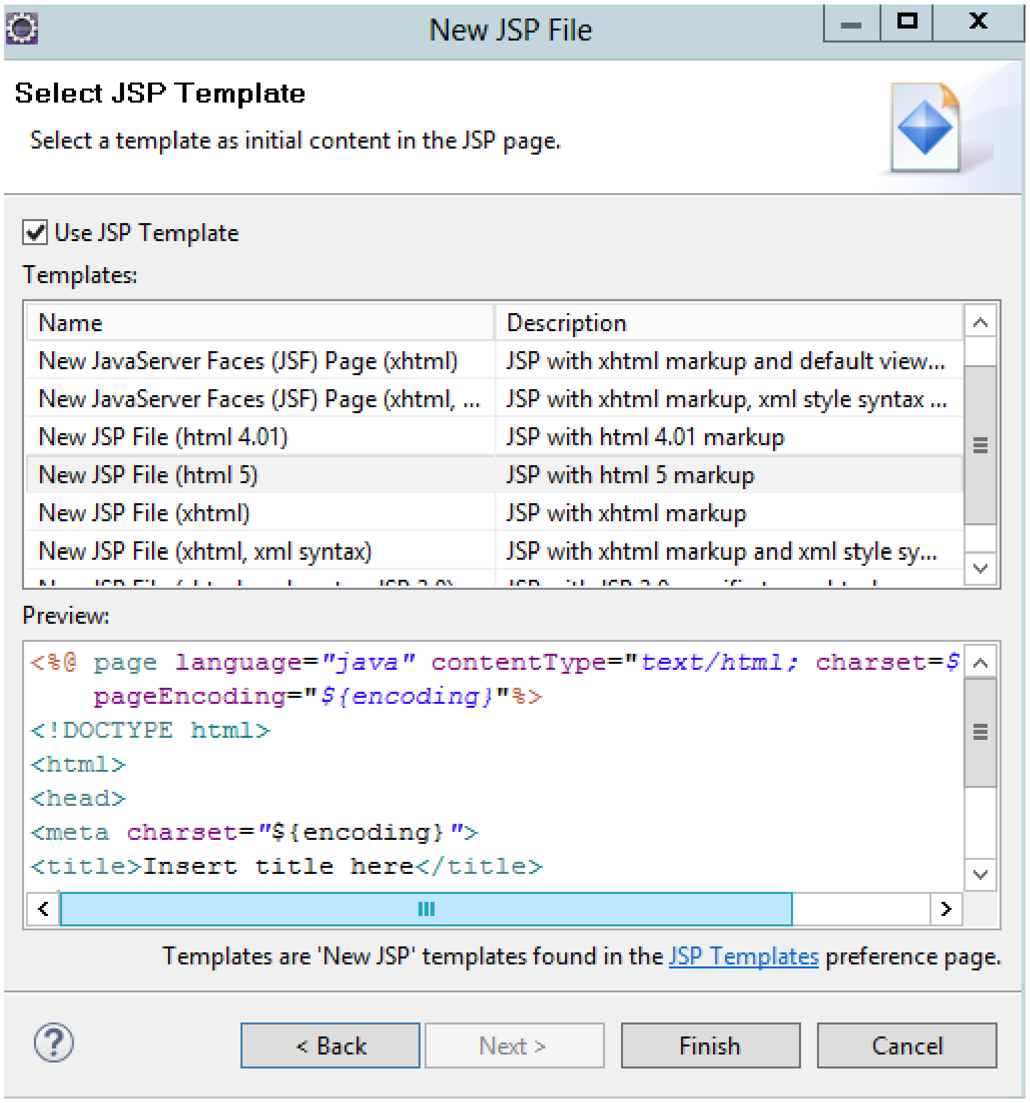
Create a new JSP file.

Enter or select the parent folder:

Project Mani/WebContent

File name: NewFile.jsp

- Also we need to Choose our JSP Template generally we can select JSP File(html 5) because it supports all of the HTML tags.



Select JSP Template

Select a template as initial content in the JSP page.

[✓] Use JSP Template

Templates:

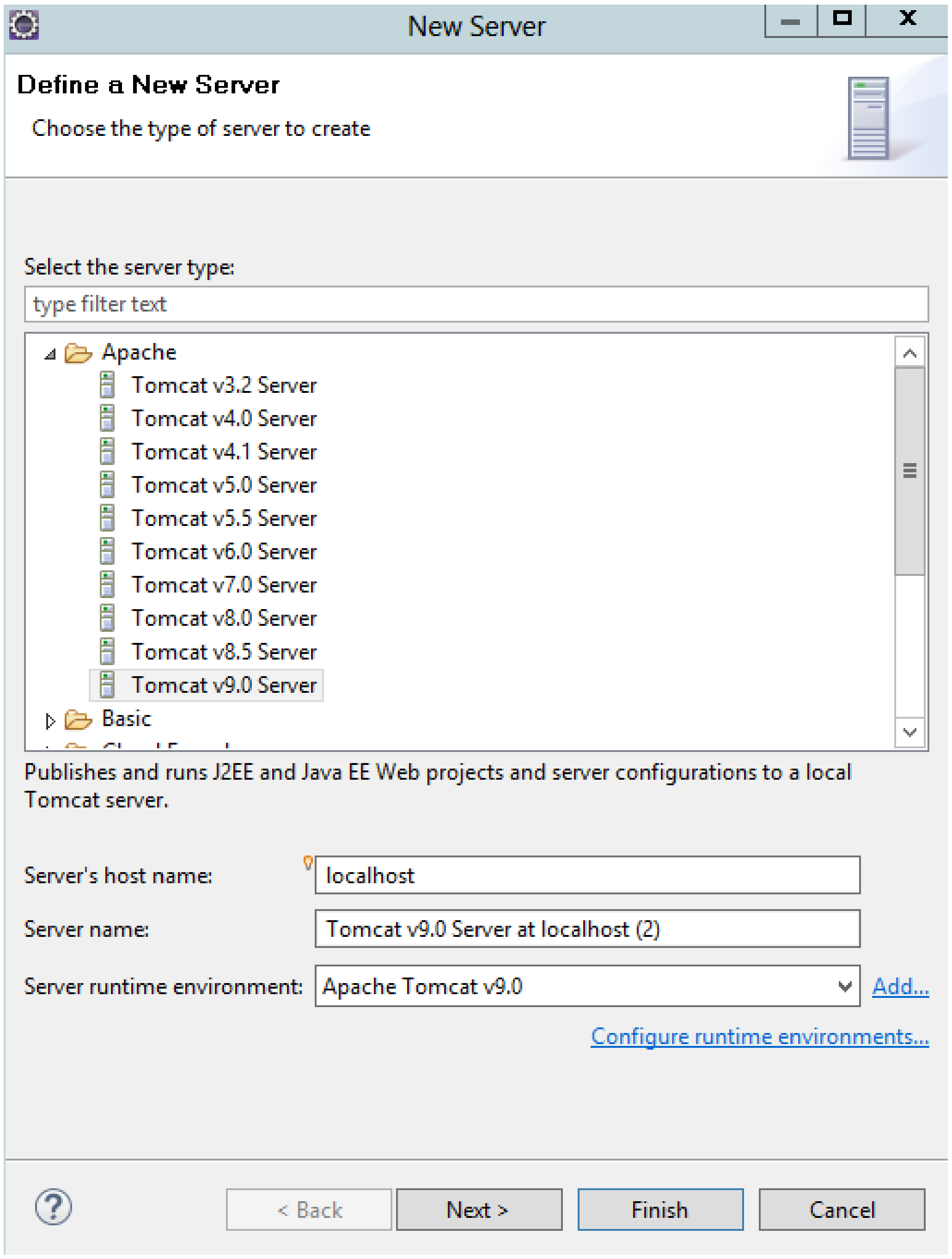
Name	Description
New JavaServer Faces (JSF) Page (xhtml)	JSP with xhtml markup and default view...
New JavaServer Faces (JSF) Page (xhtml, ...	JSP with xhtml markup, xml style syntax ...
New JSP File (html 4.01)	JSP with html 4.01 markup
New JSP File (html 5)	JSP with html 5 markup
New JSP File (xhtml)	JSP with xhtml markup
New JSP File (xhtml, xml syntax)	JSP with xhtml markup and xml style sy...

Preview:

```
<%@ page language="java" contentType="text/html; charset=$
    pageEncoding="${encoding}"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="${encoding}">
<title>Insert title here</title>
```

Templates are 'New JSP' templates found in the JSP Templates preference page.

- After selecting the Template we need to choose out Server like Apache Tomcat v3.2 Server and running environment that is host type.



Define a New Server

Choose the type of server to create

Select the server type:

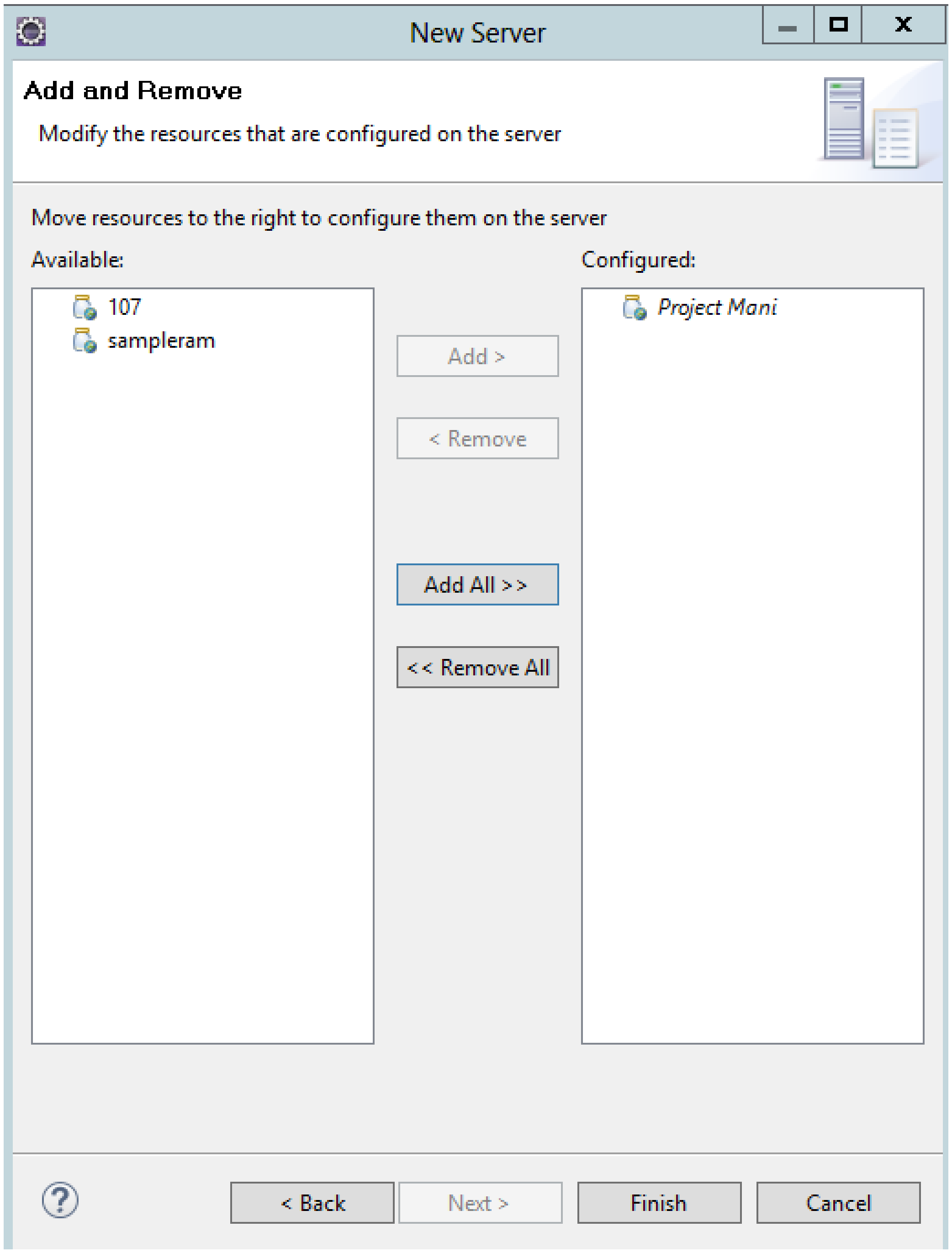
type filter text

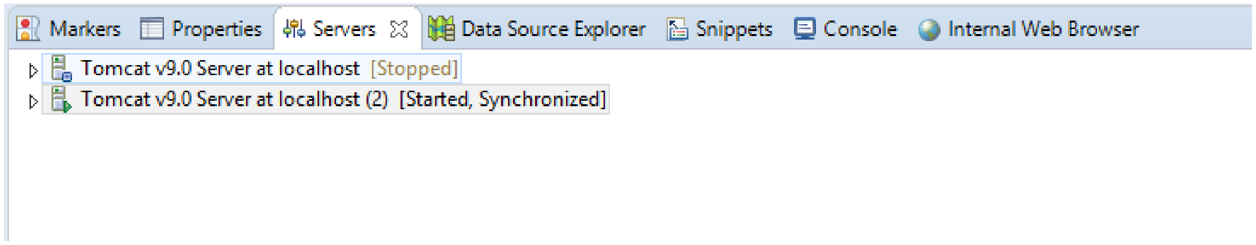
- Apache
  - Tomcat v3.2 Server
  - Tomcat v4.0 Server
  - Tomcat v4.1 Server
  - Tomcat v5.0 Server
  - Tomcat v5.5 Server
  - Tomcat v6.0 Server
  - Tomcat v7.0 Server
  - Tomcat v8.0 Server
  - Tomcat v8.5 Server
  - Tomcat v9.0 Server
- Basic

Publishes and runs J2EE and Java EE Web projects and server configurations to a local Tomcat server.

Server's host name:	localhost
Server name:	Tomcat v9.0 Server at localhost (2)
Server runtime environment:	Apache Tomcat v9.0

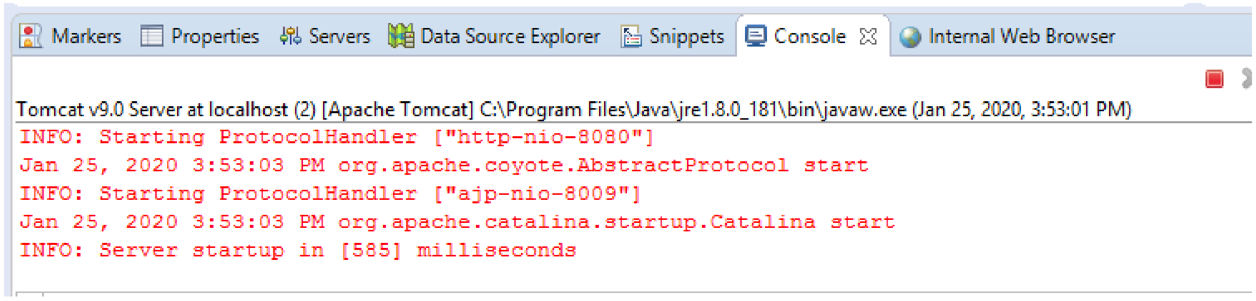
- We can add our Server from external field



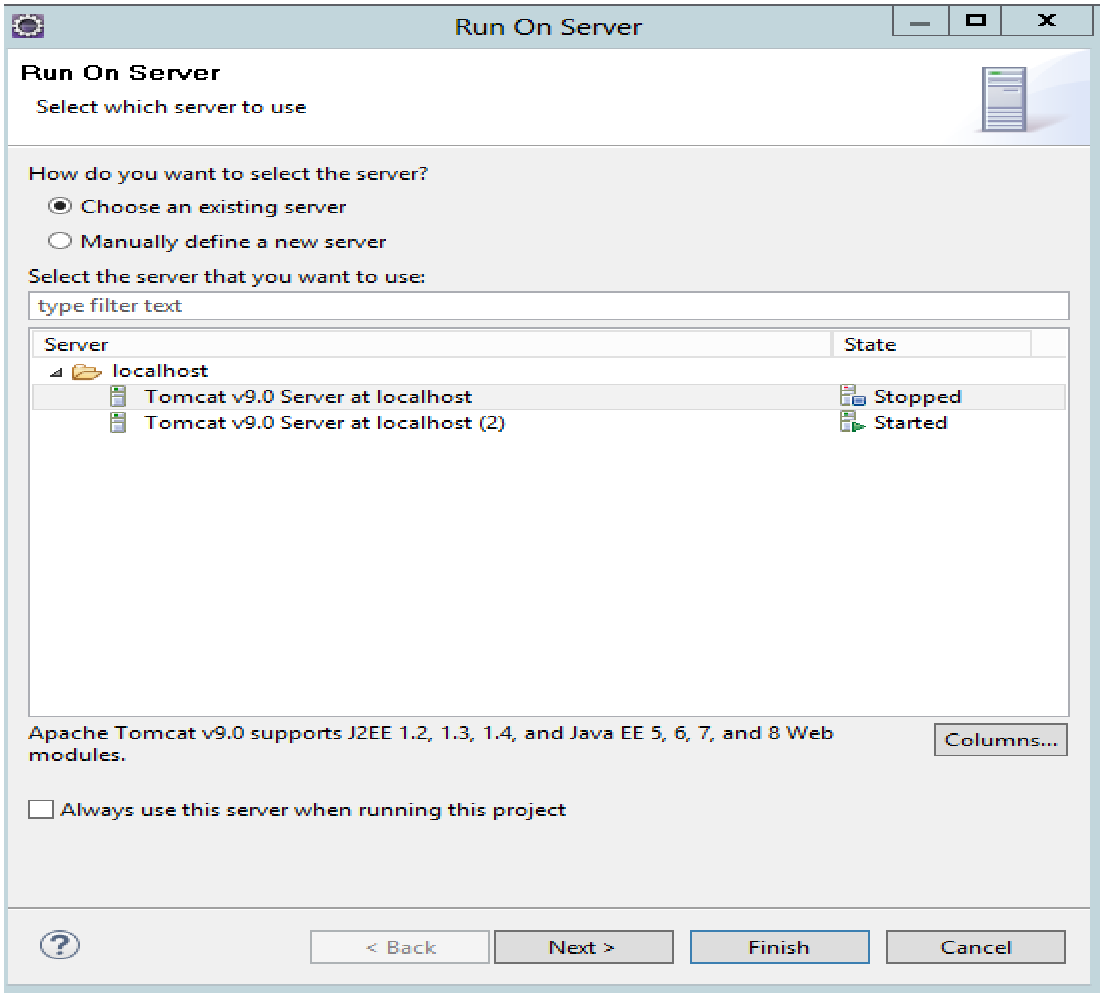


Tomcat v9.0 Server at localhost [Stopped]
Tomcat v9.0 Server at localhost (2) [Started, Synchronized]

➢ Execution of the JSP file and observe the output which is like a HTML page.



```
Tomcat v9.0 Server at localhost (2) [Apache Tomcat] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (Jan 25, 2020, 3:53:01 PM)
INFO: Starting ProtocolHandler ["http-nio-8080"]
Jan 25, 2020 3:53:03 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["ajp-nio-8009"]
Jan 25, 2020 3:53:03 PM org.apache.catalina.startup.Catalina start
INFO: Server startup in [585] milliseconds
```



Run On Server

Select which server to use

How do you want to select the server?

[● Choose an existing server]

○ Manually define a new server

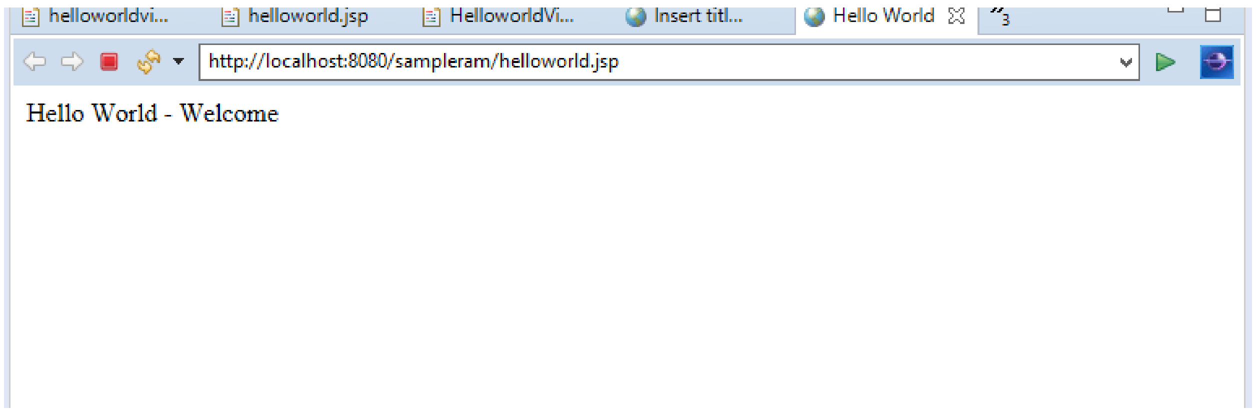
Select the server that you want to use:

type filter text

Server	State
localhost	
Tomcat v9.0 Server at localhost	Stopped
Tomcat v9.0 Server at localhost (2)	Started

Apache Tomcat v9.0 supports J2EE 1.2, 1.3, 1.4, and Java EE 5, 6, 7, and 8 Web modules.

☐ Always use this server when running this project



http://localhost:8080/sampleram/helloworld.jsp

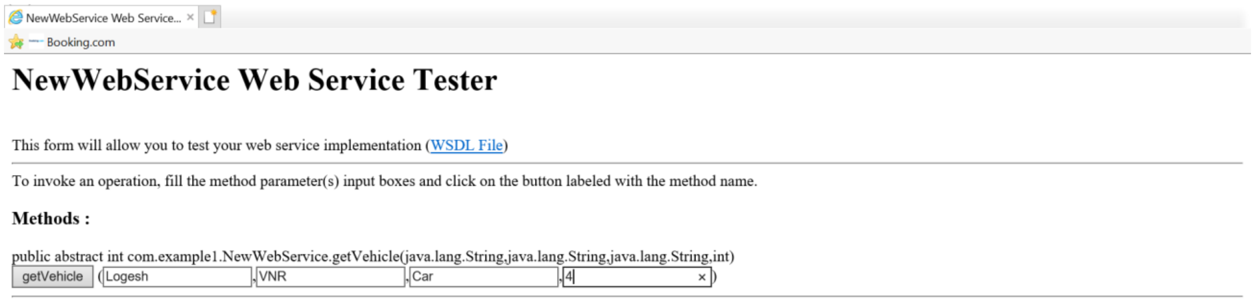
Hello World - Welcome

<u>Web Service Programming:</u>

SOAP:

```
package com.example1;

import javax.jws.WebService;

import javax.jws.WebMethod;

import javax.jws.WebParam;

@WebService(serviceName = "NewWebService")

public class NewWebService {

  @WebMethod(operationName = "Get_Vehicle")

  public int Get_Vehicle(@WebParam(name = "Customer_Name") String Customer_Name,
@WebParam(name = "City") String City, @WebParam(name = "Type_of_Vehicle") String
Type_of_Vehicle, @WebParam(name = "Duration") int Duration) {

    int res;

    switch(Type_of_Vehicle)

    {

      case "Cycle":
        {
          res = 10;
          break;
        }
        case "Bike":
        {
          res = 100;
          break;
        }
        case "Car":
        {
          res = 1000;
          break;
        }
        default:
        {
          res = 0;
          break;
        }
      }
      return res*Duration;
    }}
```

Output:


NewWebService Web Service... Booking.com

NewWebService Web Service Tester

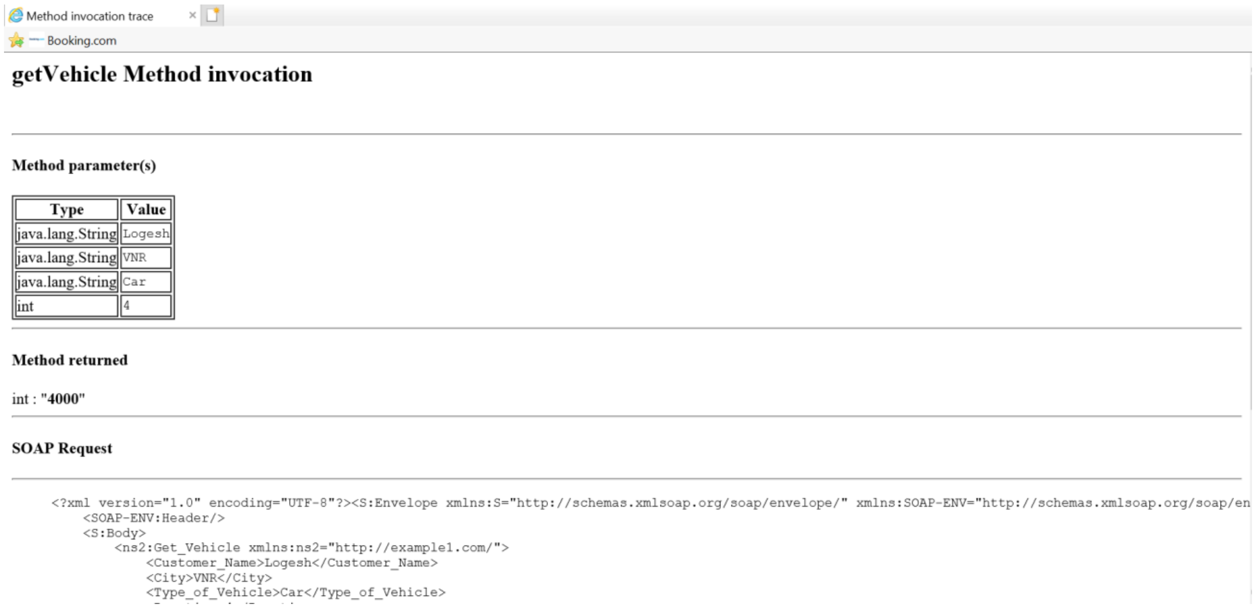
This form will allow you to test your web service implementation (WSDL File)

To invoke an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.

Methods :

public abstract int com.example1.NewWebService.getVehicle(java.lang.String,java.lang.String,java.lang.String,int)

getVehicle (Logesh, VNR, Car, 4)


Method invocation trace Booking.com

getVehicle Method invocation

Method parameter(s)

Type	Value
java.lang.String	Logesh
java.lang.String	VNR
java.lang.String	Car
int	4

Method returned

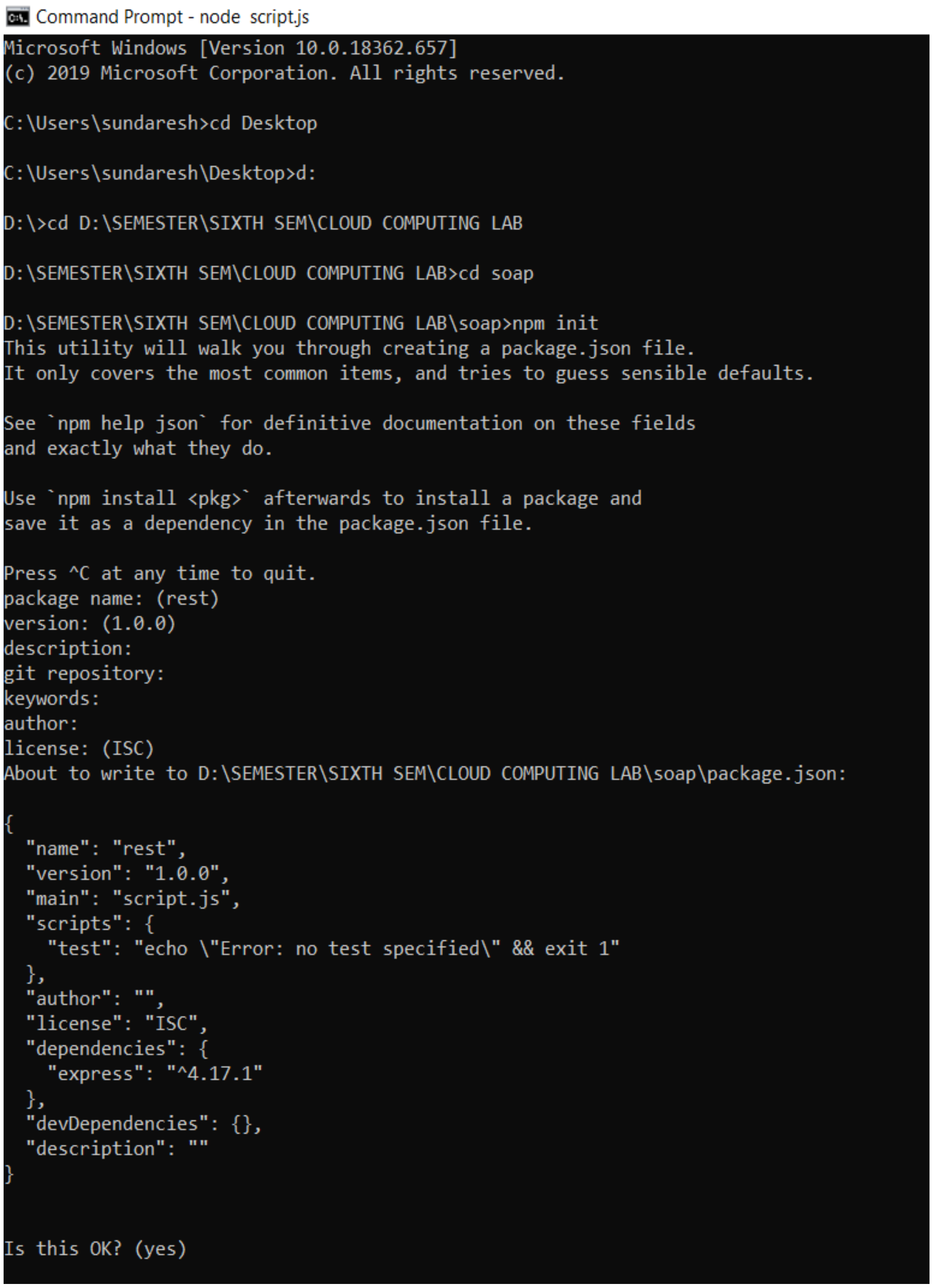
int : "4000"

SOAP Request

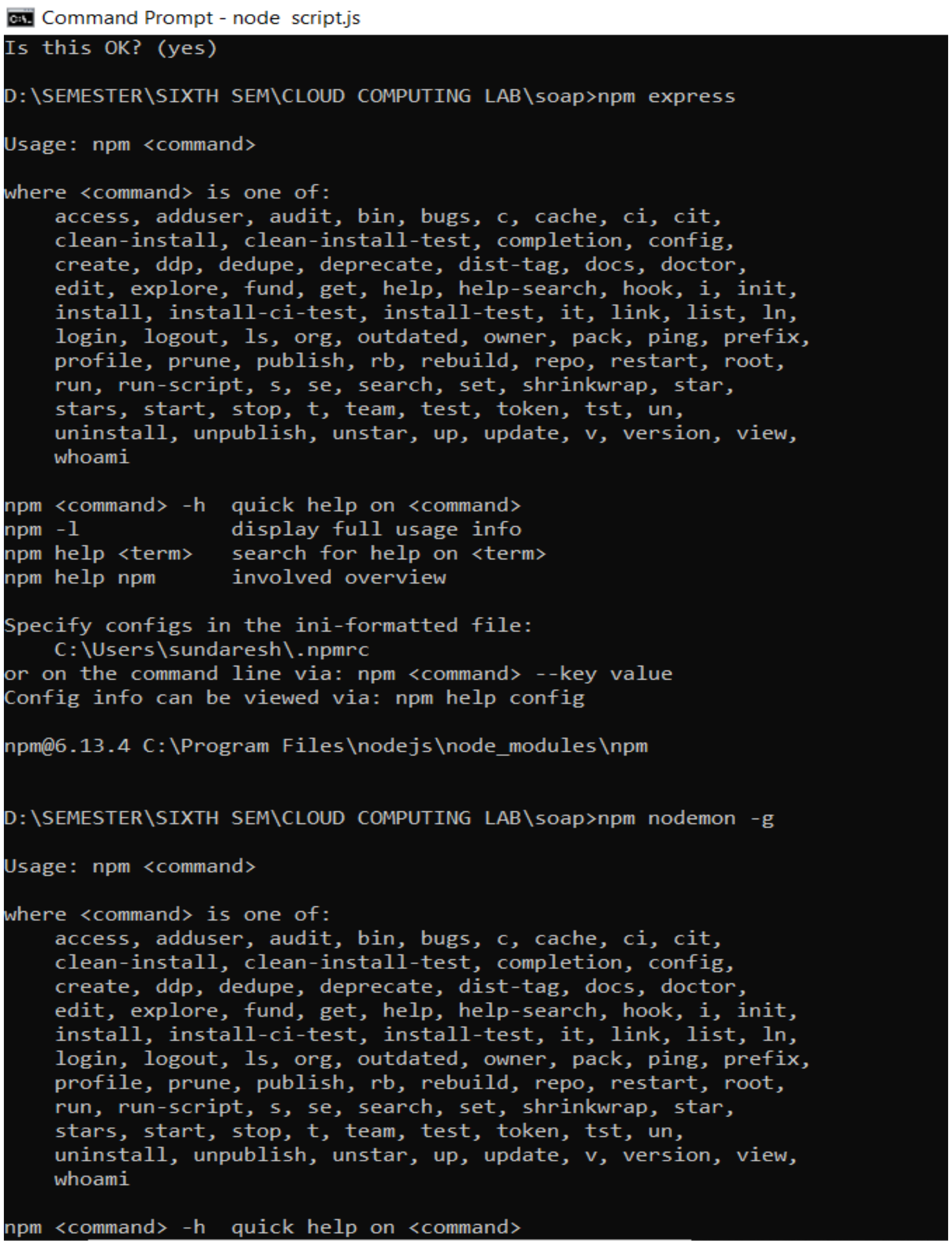
```
     <?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/en
         <SOAP-ENV:Header/>
         <S:Body>
             <ns2:Get_Vehicle xmlns:ns2="http://example1.com/">
                 <Customer_Name>Logesh</Customer_Name>
                 <City>VNR</City>
                 <Type_of_Vehicle>Car</Type_of_Vehicle>
```

Node.js:

We need to download node js package to execute the **npm** commands.By using **npm** only we get two important packages such as **express** and **nodemon** to execute our javascript code.By executing java script program we can get a host with Port Number.

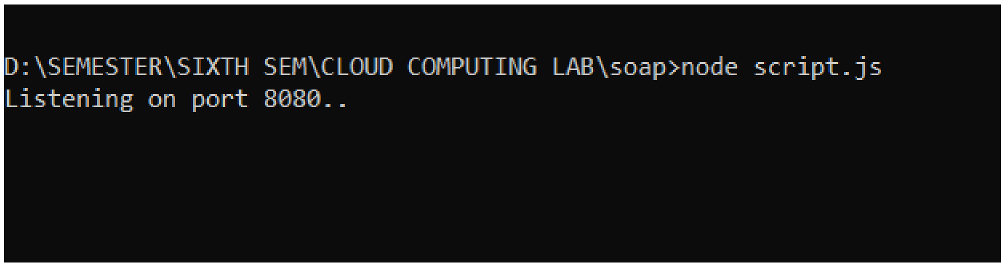


```
Microsoft Windows [Version 10.0.18362.657]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\sundaresh>cd Desktop

C:\Users\sundaresh\Desktop>d:

D:\>cd D:\SEMESTER\SIXTH SEM\CLOUD COMPUTING LAB

D:\SEMESTER\SIXTH SEM\CLOUD COMPUTING LAB>cd soap

D:\SEMESTER\SIXTH SEM\CLOUD COMPUTING LAB\soap>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (rest)
version: (1.0.0)
description:
git repository:
keywords:
author:
license: (ISC)
About to write to D:\SEMESTER\SIXTH SEM\CLOUD COMPUTING LAB\soap\package.json:

{
  "name": "rest",
  "version": "1.0.0",
  "main": "script.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.17.1"
  },
  "devDependencies": {},
  "description": ""
}


Is this OK? (yes)
```



Command Prompt - node script.js

```
Is this OK? (yes)

D:\SEMESTER\SIXTH SEM\CLOUD COMPUTING LAB\soap>npm express

Usage: npm <command>

where <command> is one of:
    access, adduser, audit, bin, bugs, c, cache, ci, cit,
    clean-install, clean-install-test, completion, config,
    create, ddp, dedupe, deprecate, dist-tag, docs, doctor,
    edit, explore, fund, get, help, help-search, hook, i, init,
    install, install-ci-test, install-test, it, link, list, ln,
    login, logout, ls, org, outdated, owner, pack, ping, prefix,
    profile, prune, publish, rb, rebuild, repo, restart, root,
    run, run-script, s, se, search, set, shrinkwrap, star,
    stars, start, stop, t, team, test, token, tst, un,
    uninstall, unpublish, unstar, up, update, v, version, view,
    whoami

npm <command> -h  quick help on <command>
npm -l            display full usage info
npm help <term>   search for help on <term>
npm help npm      involved overview

Specify configs in the ini-formatted file:
    C:\Users\sundaresh\.npmrc
or on the command line via: npm <command> --key value
Config info can be viewed via: npm help config

npm@6.13.4 C:\Program Files\nodejs\node_modules\npm


D:\SEMESTER\SIXTH SEM\CLOUD COMPUTING LAB\soap>npm nodemon -g

Usage: npm <command>

where <command> is one of:
    access, adduser, audit, bin, bugs, c, cache, ci, cit,
    clean-install, clean-install-test, completion, config,
    create, ddp, dedupe, deprecate, dist-tag, docs, doctor,
    edit, explore, fund, get, help, help-search, hook, i, init,
    install, install-ci-test, install-test, it, link, list, ln,
    login, logout, ls, org, outdated, owner, pack, ping, prefix,
    profile, prune, publish, rb, rebuild, repo, restart, root,
    run, run-script, s, se, search, set, shrinkwrap, star,
    stars, start, stop, t, team, test, token, tst, un,
    uninstall, unpublish, unstar, up, update, v, version, view,
    whoami

npm <command> -h  quick help on <command>
```



```
D:\SEMESTER\SIXTH SEM\CLOUD COMPUTING LAB\soap>node script.js
Listening on port 8080..
```

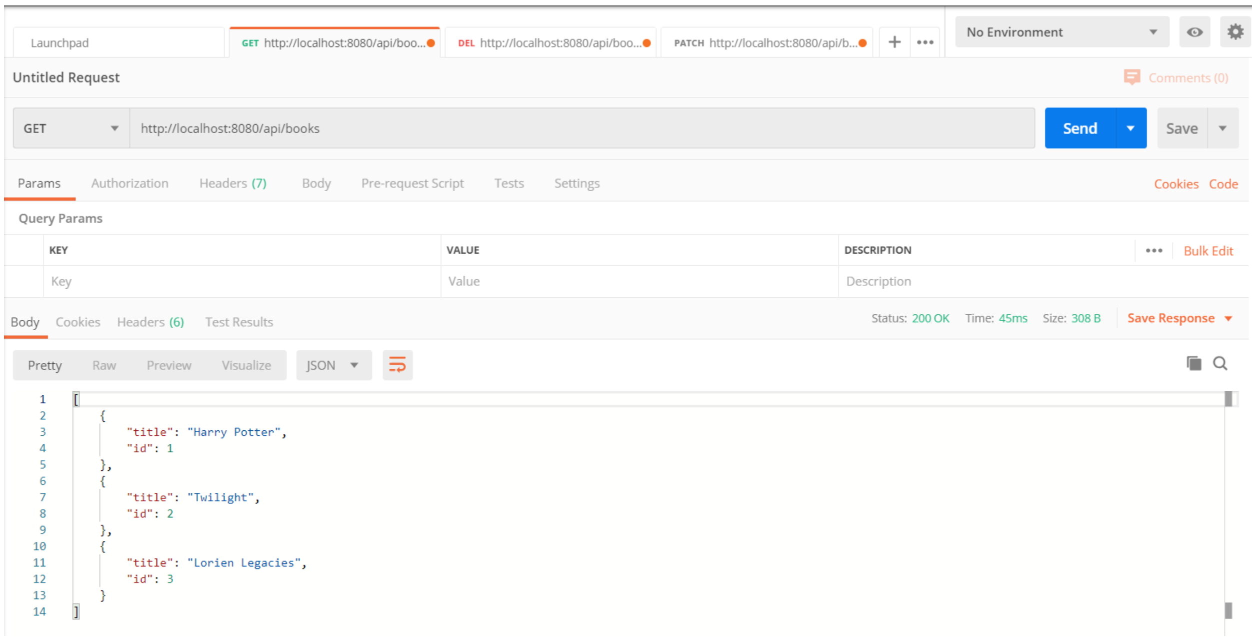
Server.js : (Java Script File)

```
const express = require('express');

const app = express();

app.use(express.json());

const books = [

{title: 'Harry Potter', id: 1},

{title: 'Twilight', id: 2},

{title: 'Lorien Legacies', id: 3}

]

 //READ Request Handlers

app.get('/', (req, res) => {

res.send('Welcome to Edurekas REST API with Node.js Tutorial!!');

});


app.get('/api/books', (req,res)=> {

res.send(books);

});


app.get('/api/books/:id', (req, res) => {

const book = books.find(c => c.id === parseInt(req.params.id));
if (!book) res.status(404).send('<h2 style="font-family: Malgun Gothic; color:
darkred;">Ooops... Cant find what you are looking for!</h2>');

res.send(book);

});


//CREATE Request Handler


app.post('/api/books', (req, res)=> {

const { error } = validateBook(req.body);

if (error){

res.status(400).send(error.details[0].message)

return;

}

const book = {

id: books.length + 1,

title: req.body.title

};

books.push(book);

res.send(book);

});
```

```
//UPDATE Request Handler

app.put('/api/books/:id', (req, res) => {

const book = books.find(c=> c.id === parseInt(req.params.id));

if (!book) res.status(404).send('<h2 style="font-family: Malgun Gothic; color:
darkred;">Not Found!! </h2>');

const { error } = validateBook(req.body);

if (error){

res.status(400).send(error.details[0].message);

return;

}

book.title = req.body.title;

res.send(book);

});


//DELETE Request Handler



app.delete('/api/books/:id', (req, res) => {

 const book = books.find( c=> c.id === parseInt(req.params.id));

if(!book) res.status(404).send('<h2 style="font-family: Malgun Gothic; color: darkred;">
Not Found!! </h2>');

 const index = books.indexOf(book);

books.splice(index,1);
```

```
 res.send(book);

});

 function validateBook(book) {

const schema = {

title: Joi.string().min(3).required()

};

return Joi.validate(book, schema);

}


//PORT ENVIRONMENT VARIABLE


const port = process.env.PORT || 8080;

app.listen(port, () => console.log(`Listening on port ${port}..`));
```

By using port number only we can made our host in the Post man tool.In which we can perform INSERT users and DELETE users operations.Observe the results below:

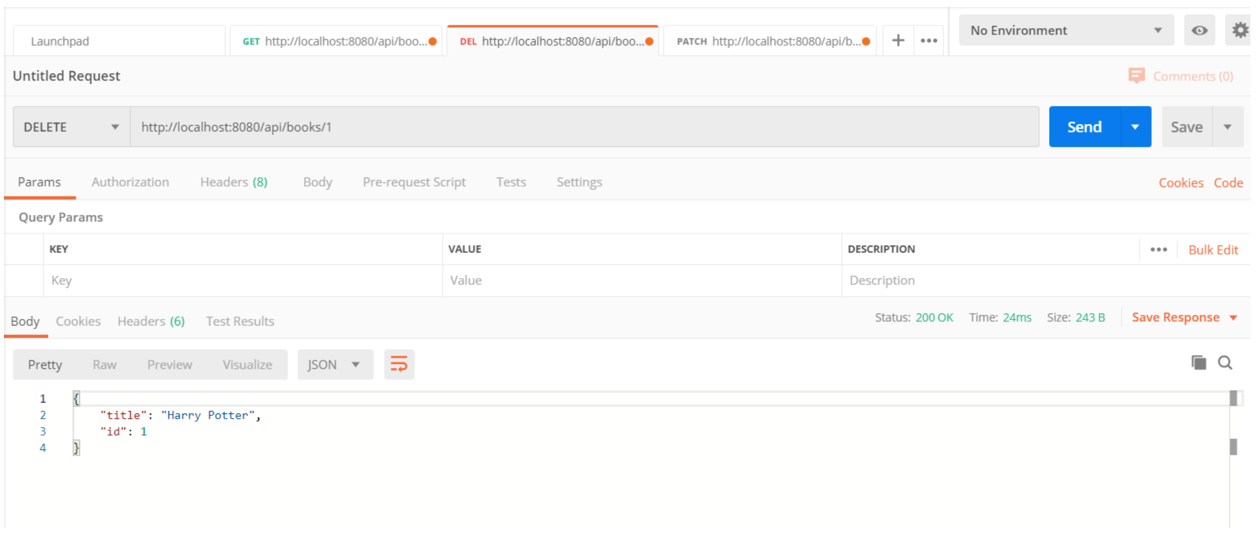


Untitled Request

GET http://localhost:8080/api/books

Status: 200 OK Time: 45ms Size: 308 B

```
[
    {
        "title": "Harry Potter",
        "id": 1
    },
    {
        "title": "Twilight",
        "id": 2
    },
    {
        "title": "Lorien Legacies",
        "id": 3
    }
]
```



Untitled Request

DELETE http://localhost:8080/api/books/1

Status: 200 OK Time: 24ms Size: 243 B

```
{
    "title": "Harry Potter",
    "id": 1
}
```

<u>Result:</u>

Thus, the installation or configuration of SOAP and RESTFUL webservices and Signup page of our application using SOAP and RESTFUL webServices are implemented.