

Lab 2: Apache Tomcat Installation & Servlet Programming

Objectives:

- To install and administer the Apache Tomcat Servlet Container and to learn servlet programming.

Submission:

- Checkpoints that need to be shown to the course teacher.

Introduction:

In this lab, you will install and administer the most widely-used servlet container in the world: Apache Tomcat. In addition, you will learn the very basic of servlet programming.

Servlet (Server-Applet) technology is used to create a web application using Java. A web application resides at the server side and generates a dynamic web page based on the received request. The generated web page is then sent back as a response to be displayed in the browser. This is illustrated in the figure¹ (taken from <https://www.javatpoint.com/servlet-tutorial>) below.

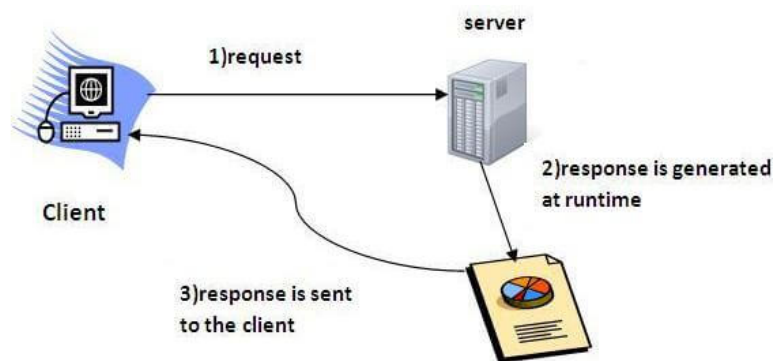


Figure 1: Basic servlet operation

Before Servlet, CGI (Common Gateway Interface) scripting language was common as a server-side programming language. A CGI enables the web server to call an external program and pass HTTP request information to the external program to process the request. For each request, it starts a new process. The following figure illustrated this concept (taken from <https://www.javatpoint.com/servlet-tutorial>).

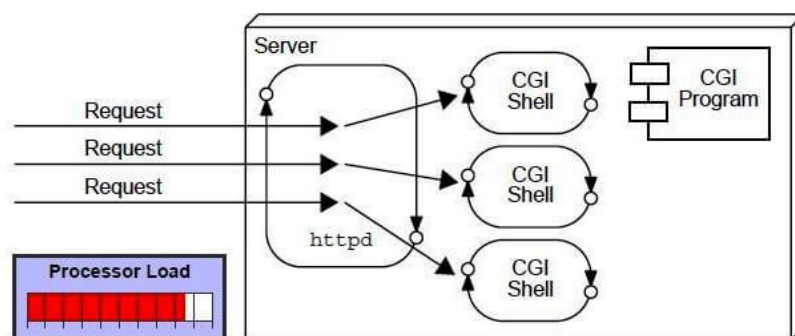


Figure 2: CGI operation

The main disadvantage of CGI is that it used to rely on platform dependant programming languages such as C, C++ and Perl. Servlet in a way is an evolution of CGI with the integration of platform-independent Java programming language.

Because of its usage of Java, Servlet offers several advantages such as:

- It creates a thread for each request, not a process.
- It is portable because of its usage of Java.
- It is robust because a JVM is leveraged to handle request which takes care of memory leak, garbage collection and so on.

A servlet container is a servlet-enabled web server. Among the servlet containers, Apache Tomcat (<http://tomcat.apache.org/>) is the most widely-used one. In this lab, you will learn how to install and administer an Apache Tomcat servlet container and how to do servlet programming. There will be several checkpoints to carry out different tasks. **Complete the checkpoints and show it to your teacher.**

Task-1: Install Apache Tomcat (6 marks)

The first thing that you will need to check if you have Java installed in your computer. To do this, use the following command:

- `$ java -version`

If Java is installed you will see an output like the following:

```
java version "1.8.0_144"  
Java(TM) SE Runtime Environment (build 1.8.0_144-b01)  
Java HotSpot(TM) 64-Bit Server VM (build 25.144-b01, mixed mode)
```

Otherwise, you will need to install Java using the following commands:

- `$ sudo apt-get update`
- `$ sudo apt-get install default-jdk`

We will need to know the location of the Java installation directory. We can get this using the following command:

- `$ sudo update-java-alternatives -l`

The output of this will show a path which points to the Java installation directory. Copy this path in a text document. You will need it later.

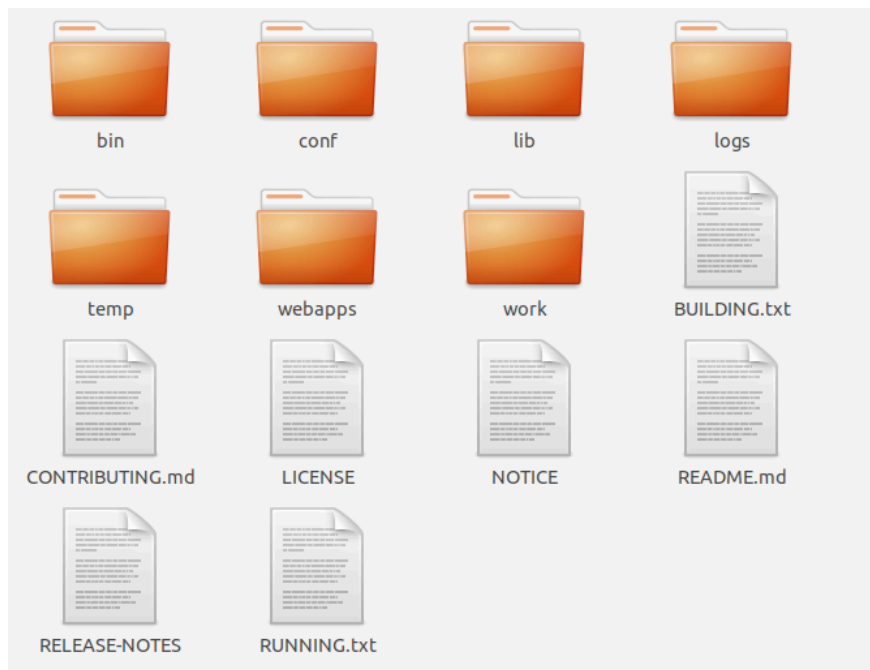
Next, while in your home directory, download the Tomcat container using the following command:

- `$ wget https://www-us.apache.org/dist/tomcat/tomcat-9/v9.0.16/bin/apache-tomcat-9.0.16.tar.gz`

The latest Tomcat (Tomcat 9) will be downloaded in your home directory. You need to untar it using the following command:

- `$ tar xzf apache-tomcat-9.0.16.tar.gz`

This will create a directory *apache-tomcat-9.0.16* into your home directory. Check if it has the following directory structure. If so, then your Tomcat download has been successful.



Take a few moments to explore the tomcat root directory to look at its different directory and files.

In the next step, we need to configure several environment variables. Issue the following commands one after one to set up the variables.

- `$ echo "export CATALINA_HOME=~/.apache-tomcat-9.0.16" >> ~/.bashrc`
- `$ echo "export JAVA_HOME=/path_to_java_saved_before" >> ~/.bashrc`
- `$ source ~/.bashrc`

Test if these variables have been set up properly using the following commands:

- `$ echo CATALINA_HOME`
- `$ echo $JAVA_HOME`

Finally we need to create user accounts to secure and access admin/manager pages. Edit `conf/tomcat-users.xml` file within the tomcat directory in your editor and paste the following contents inside `<tomcat-users>` `</tomcat-users>` tags.

```
<!-- user manager can access only manager section -->
<role rolename="manager-gui" />
<user username="manager" password="_add_password" roles="manager-gui" />

<!-- user admin can access manager and admin section both -->
<role rolename="admin-gui" />
<user username="admin" password="_add_password_" roles="manager-gui,admin-gui" />
```

This configurations above are creating two users, manager and admin, to allow different types of accesses for different resources. For example, the manager is allowed to only manager section GUI while the admin is allowed to access both the manager GUI and the admin GUI.

Next, cd into the tomcat root directory (~/apache-tomcat-9.0.16) and issue the following commands:

- `$ chmod +x bin/startup.sh`
- `$./bin/startup.sh`

If you see the following output, then the tomcat has been successfully started.


```
Using CATALINA_BASE:   /home/ripul/apache-tomcat-9.0.16
Using CATALINA_HOME:   /home/ripul/apache-tomcat-9.0.16
Using CATALINA_TMPDIR: /home/ripul/apache-tomcat-9.0.16/temp
Using JRE_HOME:        /usr/lib/jvm/java-11-openjdk-amd64
Using CLASSPATH:       /home/ripul/apache-tomcat-9.0.16/bin/bootstrap.jar:/home/ripul/apache-tomcat-9.0.16/bin/tomcat-juli.jar
Tomcat started.
```

Test the installation by visiting localhost:8080 page in your preferred browser. If you see the following page, you have successfully completed Task-1. Show it to your teacher to tick off checkpoint 1.

Apache Tomcat/9.0.16



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](#)

[Server Status](#)

[Manager App](#)

[Host Manager](#)

Developer Quick Start

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

Managing Tomcat

For security, access to the [manager webapp](#) is restricted. Users are defined in:

```
$CATALINA_HOME/conf/tomcat-users.xml
```

In Tomcat 9.0 access to the manager application is split between different users. [Read more...](#)

[Release Notes](#)

Documentation

[Tomcat 9.0 Documentation](#)

[Tomcat 9.0 Configuration](#)

[Tomcat Wiki](#)

Find additional important configuration information in:

```
$CATALINA_HOME/RUNNING.txt
```

Developers may be interested in:

Getting Help

FAQ and Mailing Lists

The following mailing lists are available:

[tomcat-announce](#)
Important announcements, releases, security vulnerability notifications. (Low volume).

[tomcat-users](#)
User support and discussion

[tomcat-dev](#)
Developer discussions

You can stop the tomcat server using the following command:

- `$./bin/shutdown.sh`

Task-2: Servlet programming with Eclipse (6 marks)

In this task, we will use Eclipse IDE for servlet programming. At first, we need to download and install Eclipse IDE in Ubuntu.

Download Eclipse IDE for Enterprise Java from the following location:

- <https://www.eclipse.org/downloads/packages/>

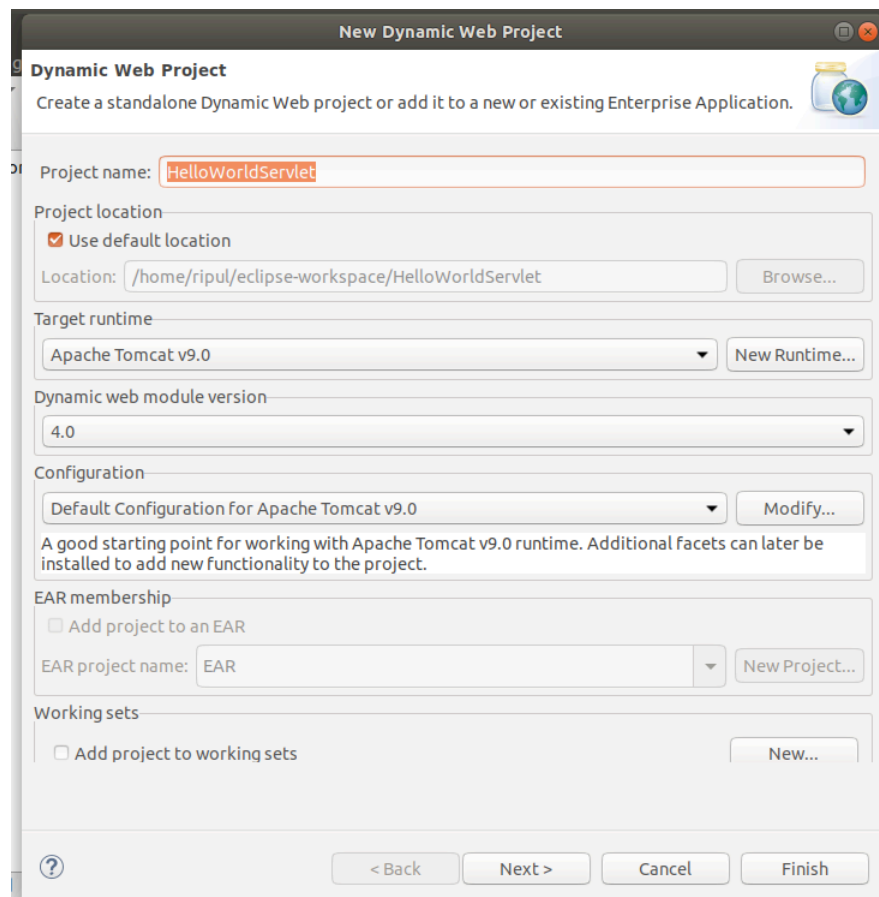
Once downloaded, untar it and install it by executing the installed called: eclipse-inst.

After installing, open the Eclipse IDE. The first thing you will need to do is to add a Tomcat runtime environment within the Eclipse. This can be done using the following instructions:

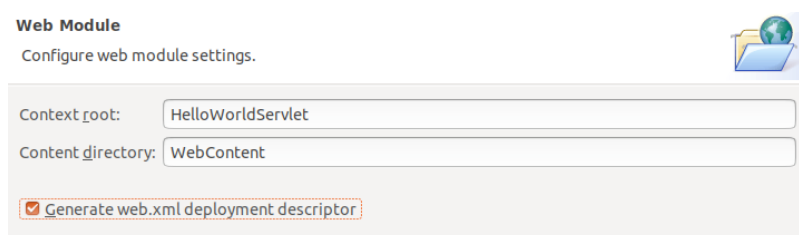
- Select Preferences from under Windows menu.
- Within the Preferences window, select Server and then select Runtime Environments.
- Click the Add button, select Apache Tomcat v9.0 and then click Next.
- In the next window, click the Browse button and select the directory of your Apache Tomcat root directory.
- Once selected, click Finish.
- Finally, click Apply and Close button to save this setting and close the Preferences window.

Now, Eclipse IDE is ready for Servlet programming. For this, you need to follow the steps below:

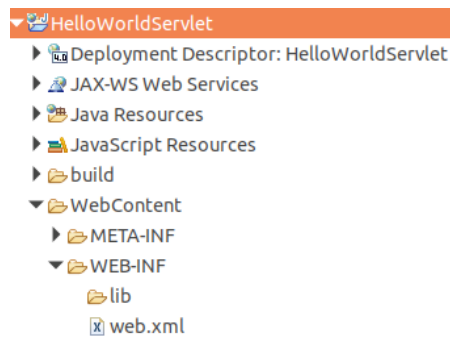
- Create a new servlet project by clicking the Dynamic Web Project from File->New
- Give it a name **HelloWorldServlet**. However, you can choose any name you want. Other options should be similar to the options as the following Figure.



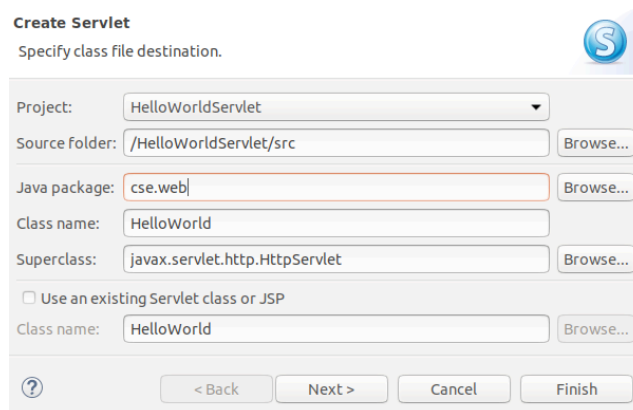
- Click, Next -> Next and then in the following window, tick the **Generate web.xml deployment descriptor** as in the following figure.



- Next, click Finish and check if the project has the hierarchy similar to the Figure below.



- Explore this hierarchy to get the idea about the contents of each directory.
- Now, this is just a project template which does not have the required code. Next, we will populate this template with code so that it can function as a web server.
- While selecting the project, right click and select Servlet from New.
- In the Create Servlet window, fill in the Java Package and Class Name as in the following figure:



- Click, Next->Next->Finish. In each window examine the objects present.
- When Finish is clicked, it will create a Java Class called HelloWorld.java as in the following figure.

```
@WebServlet("/HelloWorld")
public class HelloWorld extends HttpServlet {
    private static final long serialVersionUID = 1L;

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

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException {
        // TODO Auto-generated method stub
        response.getWriter().append("Served at: ").append(request.getContextPath());
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException {
        // TODO Auto-generated method stub
        doGet(request, response);
    }
}
```

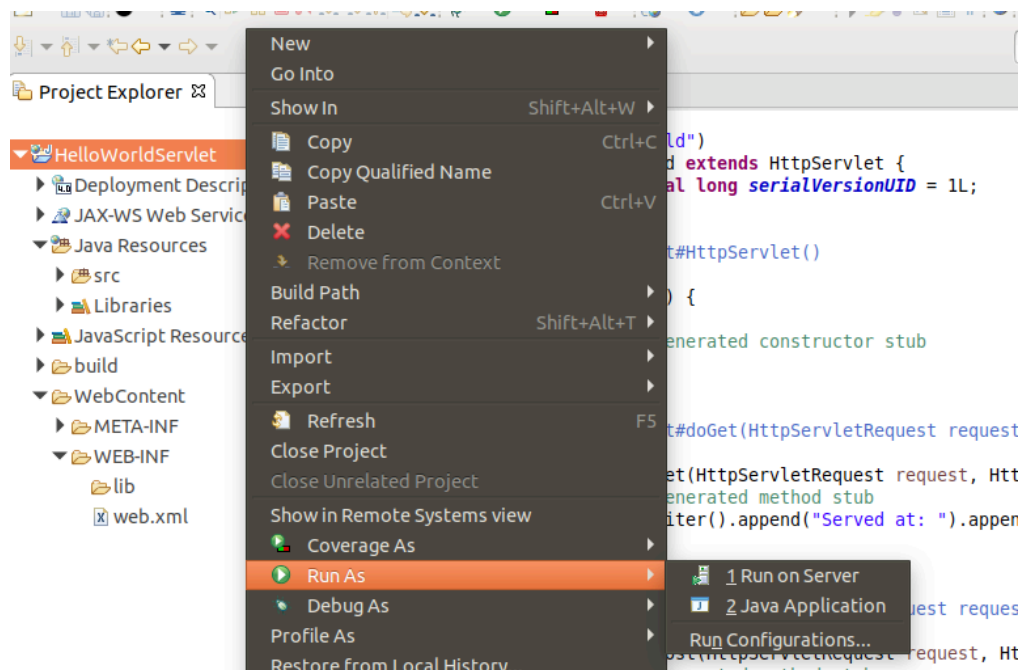
- In this, look at the `@WebServlet` annotation which indicates that `"/HelloWorld"` is the servlet url that we need to specify in the browser url to access this servlet.
- Also, take a look at the `doGet` and `doPost` methods. These methods will be used when you will submit a HTTP GET or POST request respectively. At this point, there is no content in either method. Also, examine the location where this class is created.
- Now, update the class as the following figure.

```

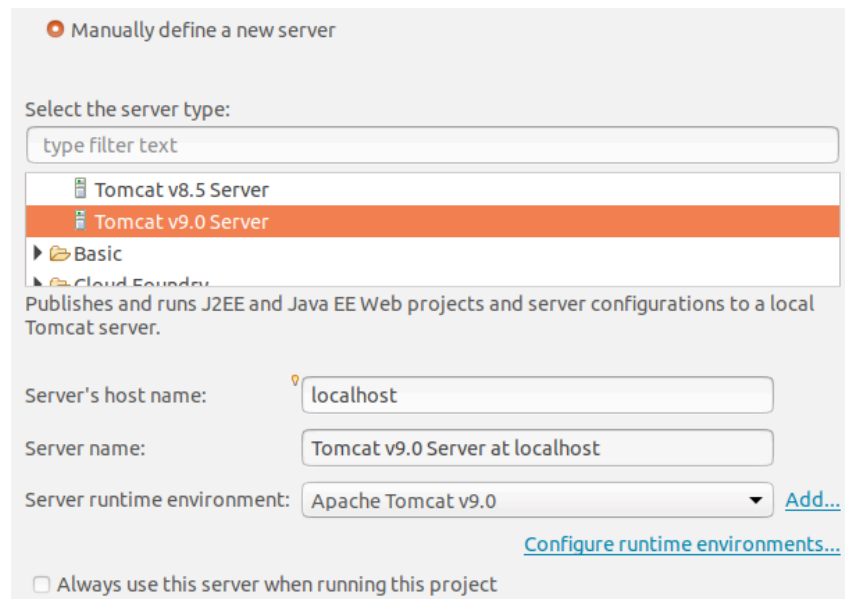
4  import java.io.PrintWriter;
5
6  import javax.servlet.ServletException;
7  import javax.servlet.annotation.WebServlet;
8  import javax.servlet.http.HttpServlet;
9  import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11
12 /**
13  * Servlet implementation class HelloWorld
14  */
15 @WebServlet("/HelloWorld")
16 public class HelloWorld extends HttpServlet {
17     private static final long serialVersionUID = 1L;
18
19     /**
20      * @see HttpServlet#HttpServlet()
21      */
22     public HelloWorld() {
23         super();
24         // TODO Auto-generated constructor stub
25     }
26
27     /**
28      * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
29      */
30     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException {
31         response.setContentType("text/html");
32         PrintWriter printWriter = response.getWriter();
33         printWriter.println("<h1>Hello World!</h1>");
34     }

```

- Then, right click on the project 'HelloWorldServlet' and select from context menu 'Run As' --> 'Run on Server' like the figure below.



- In the next window, select the options as per the following figure.



- Click Next and then Finish.
- Eclipse might create browser window with some URLs and show HTTP 404. This is because the URL is not correct.
- In your browser, browse this location:
 - <http://localhost:8080/HelloWorldServlet/HelloWorld>
- If you can see, Hello World in the browser window, then you have created your first servlet. Congratulations!
- Examine the Servlet code to understand what is going on. Try to change the contents that you would like see and save. Refresh the browser window and see what happens.
- You can Start/Stop the Apache Tomcat Server by right-clicking in the Servers tab within Show View (the lower part of with IDE).
- Stop the Tomcat Server and try to refresh the browser to examine what is going on.
- Show this to your teacher and explain your observations to tick off the checkpoint.

Task-3: Handling Form Data using Servlet (5 marks)

In this task, we will create a servlet which can handle HTML form data by following the steps below.

- Create another Dynamic Web Project following the steps described before.
- Create a servlet called MyServlet under the package of cse.web
- Update its doPost method with the following contents:

```
response.setContentType("text/html;charset=UTF-8");
PrintWriter out = response.getWriter();
try {

    String user=request.getParameter("user");
    out.println("<h2> Welcome "+user+"</h2>");
} finally {
    out.close();
}
```


- Create a new HTML file called `index.html` under the **WebContent** directory of the project and add the following contents under its *body* tag.

```
<form method="post" action="check">  
  Name <input type="text" name="user" >  
  <input type="submit" value="submit">  
</form>
```

- Open the `web.xml` file and add the following contents after the *welcome-file-list* tag.

```
<servlet>  
  <servlet-name>check</servlet-name>  
  <servlet-class>cse.web.MyServlet</servlet-class>  
</servlet>  
<servlet-mapping>  
  <servlet-name>check</servlet-name>  
  <url-pattern>/check</url-pattern>  
</servlet-mapping>
```

- Now, run this servlet following the methods described previously. Since you already have created the Tomcat instance you don't need to create any more again.
- Explore the webpage (<http://localhost:8080/FormData/>) in your browser and submit Form data.
- Examine the servlet code to understand what is happening.
- Show this to your teacher and explain your observations to tick off this checkpoint.

Task-4: Creating dynamic web pages using JSP (3 marks)

In this task, you will learn how to utilise JSP (JavaServer Pages) within a servlet. It is a technology with similar capabilities as PHP and ASP, however, it utilises Java in the back-end. Therefore, you can take advantage of a wide-range of libraries available for Java to enhance the performance of your Tomcat server.

Like before you will need to follow the instructions given below:

- Create a new Dynamic Web Project called *JSPServletExample* following the steps described before.
- Create a new servlet called *HelloJSP* as before and edit its `doGet` method with the following contents. Also resolve any error by importing the required package.

```
RequestDispatcher view=request.getRequestDispatcher("data.jsp");  
view.forward(request,response);
```

- Create a jsp page, called *index.jsp*, under the `WebContent` directory with the following contents:

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>

<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>JSP Servlet Example</title>
</head>
<body>

    <div align="center" style="margin-top: 50px;">

        <form action="JSPServlet">
            Please enter your Username: <input type="text" name="username" size="20px"> <br>
            Please enter your Password: <input type="text" name="password" size="20px"> <br><br>
            <input type="submit" value="submit">
        </form>

    </div>

</body>
</html>

```

- Create another jsp page, called *data.jsp*, under the WebContent directory with the following contents:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
+http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title> Crunchify.com JSP Servlet Example </title>
</head>
<body>
<div align='center'>
    Username: <%= request.getParameter("username") %> <br> Password: <%=
request.getParameter("password") %>
</div>
</body>
</html>

```

- Open the web.xml file and add the following contents after the *welcome-file-list* tag.

```

<servlet>
    <servlet-name>Hello</servlet-name>
    <servlet-class>cse.web.HelloJSP</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>Hello</servlet-name>
    <url-pattern>/JSPServlet</url-pattern>
</servlet-mapping>

```

- Now, run this servlet following the methods described previously.
- Explore the webpage (<http://localhost:8080/JSPServletExample/>) in your browser and submit Form data.
- Examine the servlet code and the index.jsp And data.jsp to understand what is happening.
- Show this to your teacher and explain your observations to tick off this checkpoint.