**JSP & Servlets**

1. **Introduction**

JSP (JavaServer Pages) and Servlets are both technologies used in Java web development for creating dynamic web applications. They are part of the Java EE (Enterprise Edition) platform, now evolved into Jakarta EE.

**Servlets**: Servlets are Java programs that run on a web server. They handle requests from clients, typically web browsers, and generate responses dynamically. Servlets are Java classes that extend the capabilities of servers, responding to various types of requests. They are the foundation of Java web development and are responsible for handling the low-level communication details.

**JSP (JavaServer Pages)**: JSP is a technology that simplifies the process of creating dynamic web content in Java. It allows developers to embed Java code into HTML pages, making it easier to create web pages that display dynamic data. JSP pages are translated into servlets by the web container at runtime. So, while Servlets focus on generating dynamic content in Java code, JSP allows developers to mix HTML and Java code seamlessly.

Here's a brief comparison:

**Servlets**:

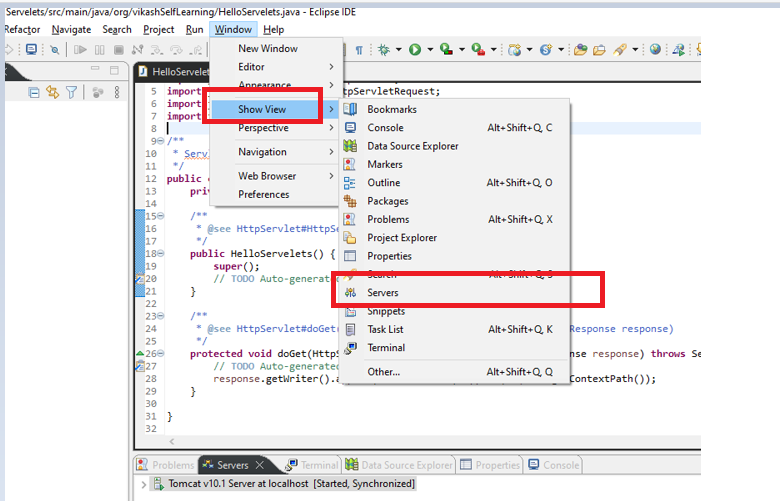
* Use Java code to handle requests and generate responses.
* Are more suitable for complex business logic and processing tasks.
* Typically involve more code for handling HTTP requests and responses directly.

**JSP**:

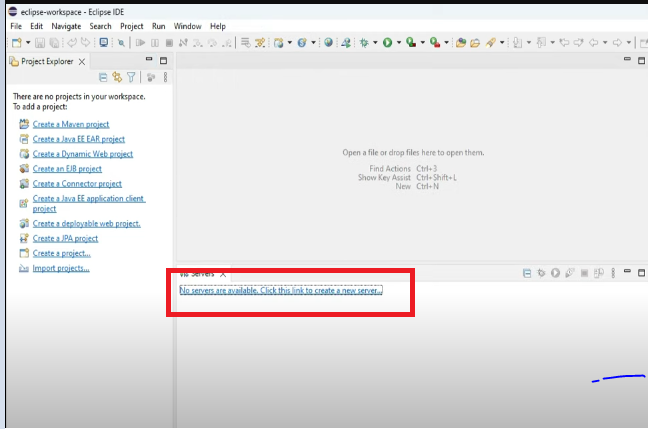
* Allows embedding Java code directly into HTML pages.
* Simplifies the process of creating dynamic web pages by blending HTML and Java code.
* Makes it easier for web designers and developers to work together, as designers can work on HTML/CSS while developers focus on Java logic.

In many applications, Servlets and JSP are used together. Servlets handle the business logic and interaction with the client, while JSP is used to generate the dynamic content to be displayed to the user, often by accessing data provided by the Servlets.

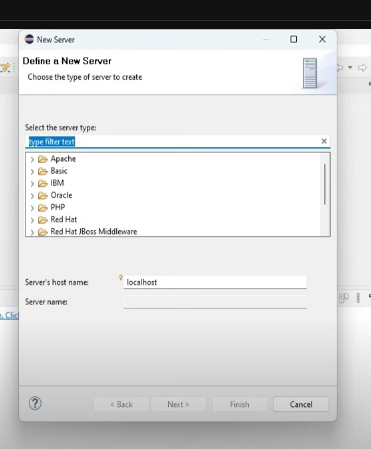
1. **How to Configure Tomcat Web Server in Eclipse IDE [2024] | Create & Run Web Project in Eclipse IDE.**



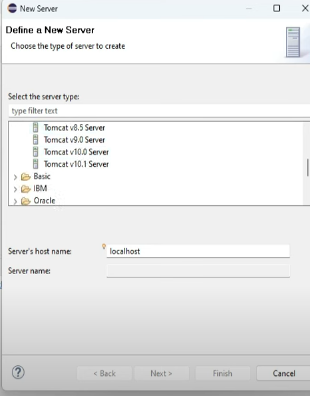
We can see it is showing no Server Available.

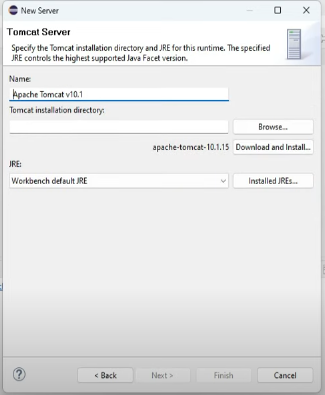


Click on the click to create new server



Expand Apache and select Tomcat10.1 and click on the next button.

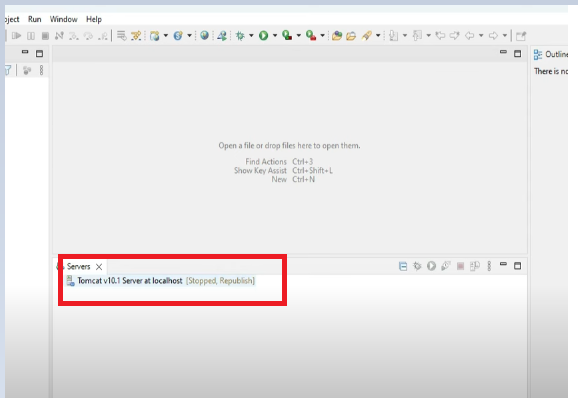




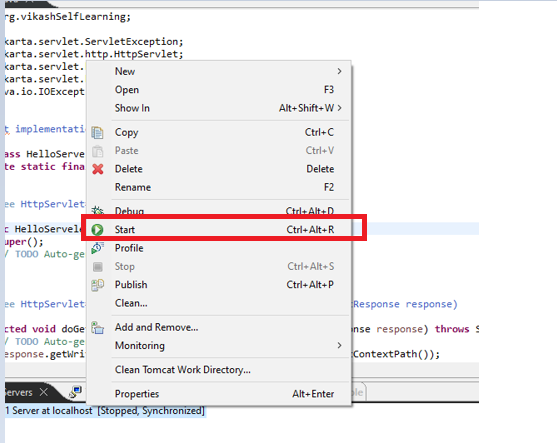
Click on Browse and go to location where tomcat is present generally It is available at the below location:

Local Disc C>> Program Files>>Apache Software Foundation >>Select the folder and give the permission.

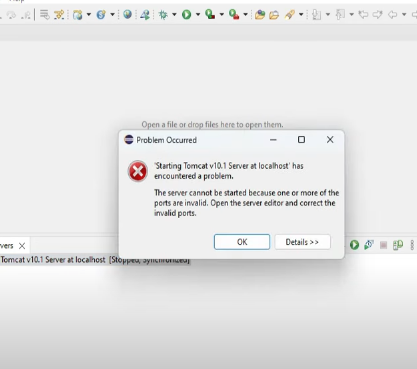
It will start showing as below:

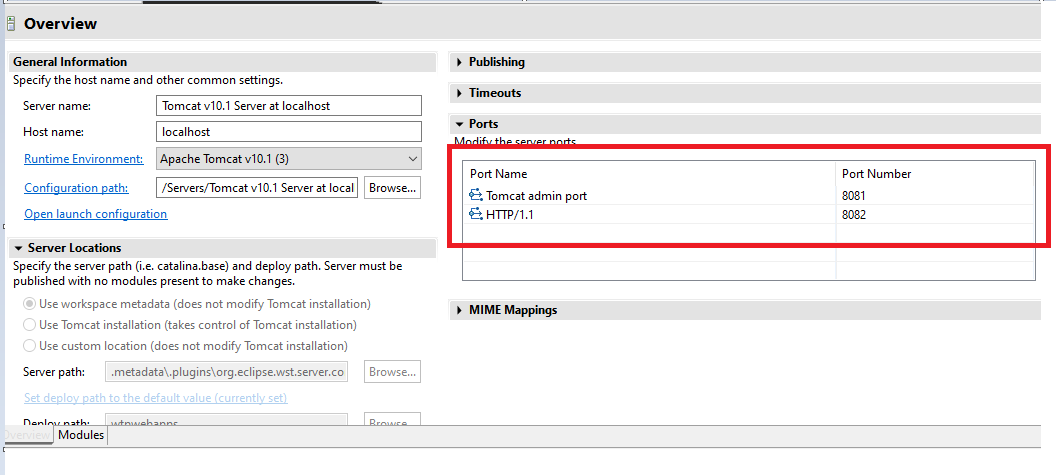


Now We need to start the server for that right Click on Tomcatv10.1 Server at localhost……… and click on Start button.

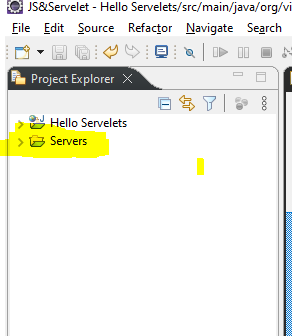


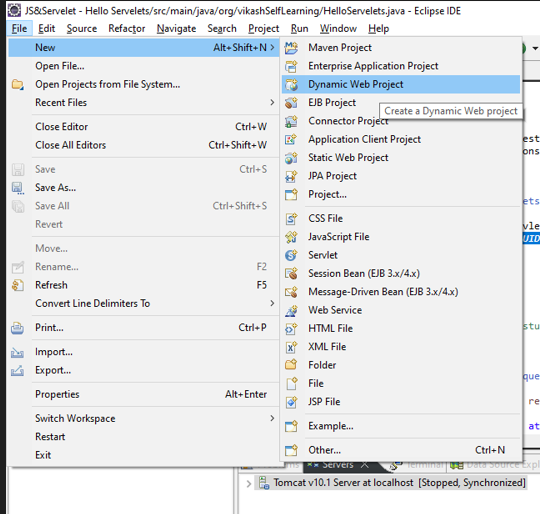
If you will get the below error Double click on the server and change the Port no from hyphen to 8080 or any number.





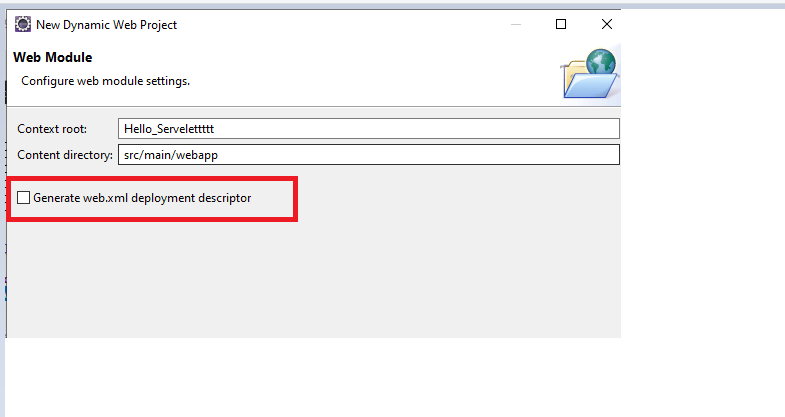
Again, Stat the server.

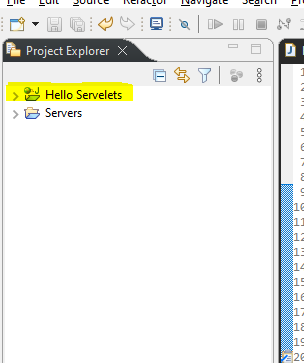


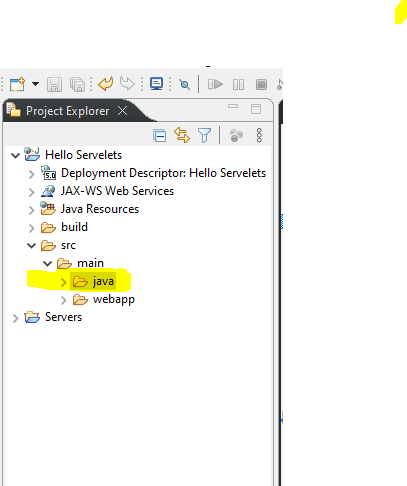


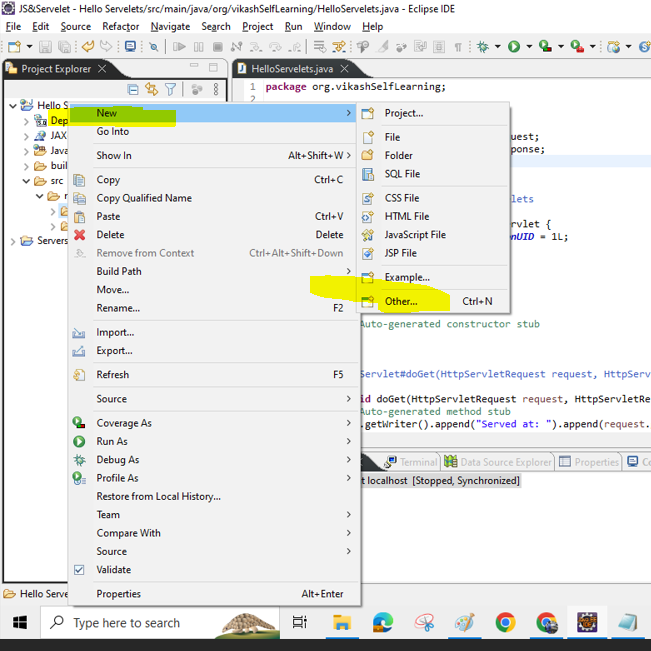
Click on next

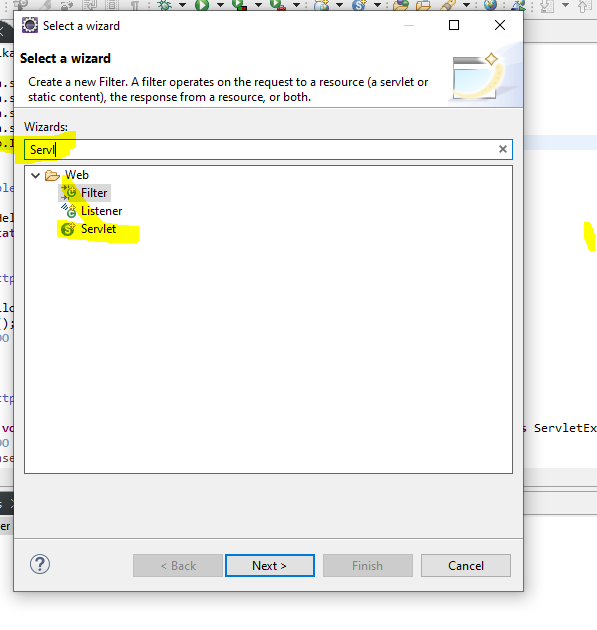
Mark the Generate web.xml deployment descriptor and click on next

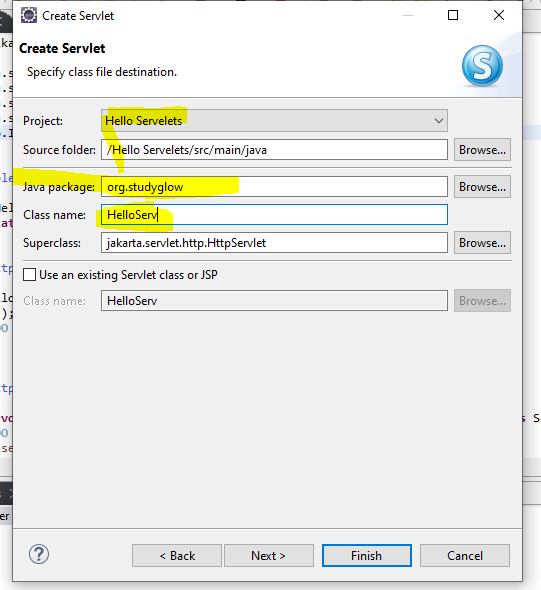












Once code is automatically generated

**package** org.vikashSelfLearning;

**import** jakarta.servlet.ServletException;

**import** jakarta.servlet.http.HttpServlet;

**import** jakarta.servlet.http.HttpServletRequest;

**import** jakarta.servlet.http.HttpServletResponse;

**import** java.io.IOException;

**public** **class** HelloServelets **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** HelloServelets() {

**super**();

// **TODO** Auto-generated constructor stub

}

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

// **TODO** Auto-generated method stub

response.getWriter().append("Served at: ").append(request.getContextPath()).append(" Hello Servlet");

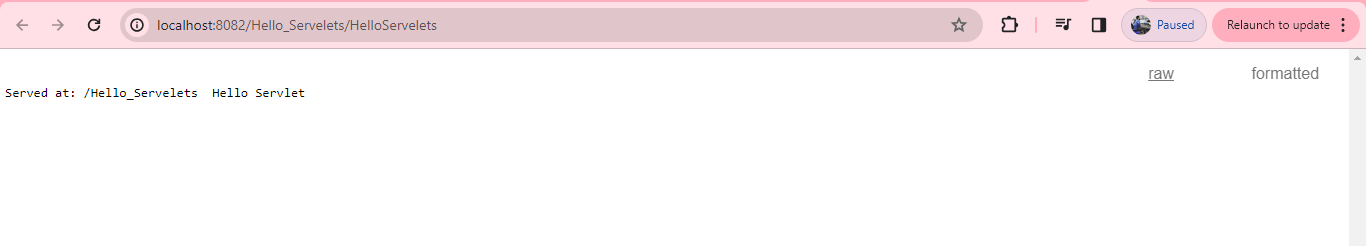
}

}

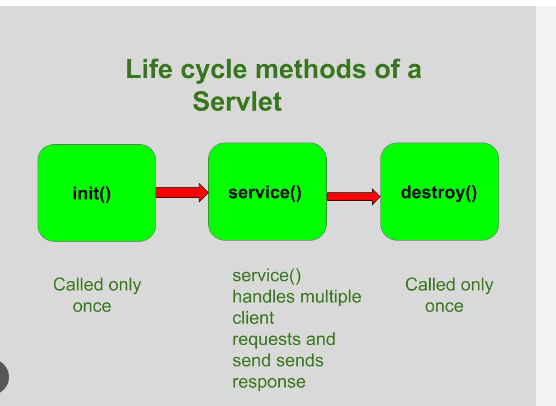
I have additionally appended ()).append(" Hello Servlet"); in the last line

Run the project on the server **like Run as server**.

On page got launched as below:



1. **Life Cycle of Servlets**



The life cycle of a servlet is managed by the servlet container, such **as Apache Tomcat or Jetty**. The container initializes the servlet, handles requests, and eventually destroys it. The life cycle of a servlet can be broken down into several stages:

**Loading**:

When a servlet container starts or when the first request for a servlet is received, the container loads the servlet class into memory.

The **init()** method of the servlet is called by the container to initialize the servlet. This method is called only once during the servlet's life cycle.

**Initialization:**

In the **init()** method, you can perform any necessary initialization tasks, such as opening a database connection or loading configuration settings.

The **init()** method is typically overridden by the servlet developer to perform custom initialization.

**Request Handling:**

Once initialized, the servlet is ready to handle client requests.

For each request received by the servlet container, the container invokes the **service()** method of the servlet.

The **service()** method then determines the type of request (GET, POST, etc.) and dispatches it to the appropriate method **(doGet(), doPost(),** etc.) for processing.

**Request Processing:**

In the methods such as **doGet()** and **doPost(),** the servlet processes the client request, performs any necessary operations, and generates a response.

These methods are overridden by the servlet developer to implement the specific functionality of the servlet.

**Response Generation:**

After processing the request, the servlet generates a response, typically by writing content to the response stream using methods provided by the **ServletResponse** object.

**Destruction**:

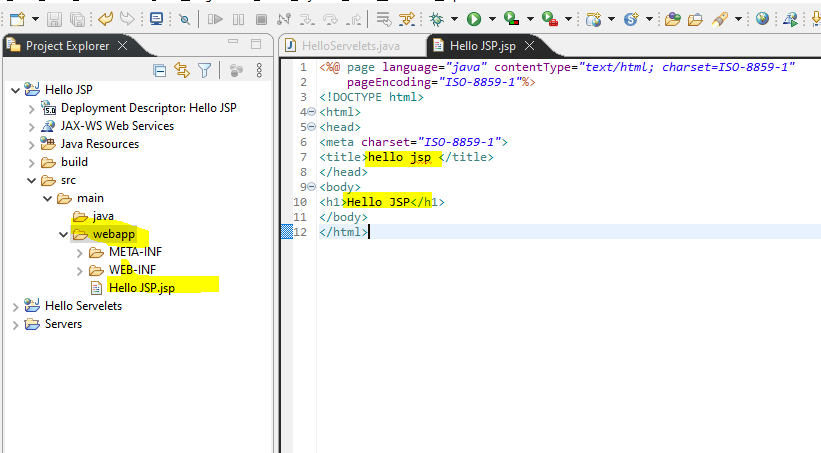
When the servlet container decides to remove the servlet from service (e.g., when the container shuts down or the servlet is no longer needed), it calls the **destroy()** method of the servlet.

The **destroy()** method allows the servlet to perform any cleanup tasks, such as releasing resources or closing database connections.

Like **init(),** the **destroy()** method is called only once during the servlet's life cycle.

Throughout its life cycle, a servlet instance may handle multiple requests, with the container invoking the **service()** method each time a request is received. Finally, when the servlet container is shut down or decides to unload the servlet, the **destroy()** method is called to allow the servlet to release any held resources before it is removed from memory.

1. **Create JSP File**



**JSP Scripting Elements**

JSP (JavaServer Pages) scripting elements allow you to embed Java code within your HTML content. There are three main types of scripting elements in JSP:

1. **Declaration**:

* Declaration tags are used to declare variables and methods in the JSP page.
* **Syntax: <%! ... %>**

**Example**:

<%!

int num = 10;

public void display() {

System.out.println("Hello from JSP!");

}

%>

1. **Scriptlet**:

* Scriptlet tags are used to write Java code snippets directly into the JSP page.
* Syntax: <% ... %>
* Example:

<%

String name = "John";

out.println("Hello, " + name);

%>

1. **Expression**

* Expression tags are used to evaluate Java expressions and print their result directly into the JSP output.
* Syntax: <%= ... %>
* Example:

<%

int x = 10;

int y = 20;

%>

<%= x + y %>

1. **Directive Elements:**

* Directive elements are used to provide special instructions to the JSP container, such as specifying page settings or including external resources.
* There are three types of directive elements:
* page: Defines attributes for the entire JSP page.
* include: Specifies files to be included during the translation phase.
* taglib: Declares a tag library for use in the JSP page.
* Syntax: <%@ ... %>
* Example:

<%@ page language="java" contentType="text/html; charset=UTF-8" %>

1. **Comment Elements:**

* JSP comment elements are used to add comments to JSP code. These comments are not visible in the generated HTML output.
* There are two types of comment elements:
* <!-- ... -->: HTML-style comments that are ignored by the JSP container.
* <%-- ... --%>: JSP-specific comments that are also ignored by the JSP container.
* Example:

<%-- This is a JSP comment --%>

<!-- This is an HTML comment -->

1. **JSP Servlets Forms**

**Form.jsp**

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Document</title>

</head>

<body>

<!DOCTYPE html>

<html>

<body>

<form action=*"submit.jsp"* method=*"get"*>

Full name: <input type=*"text"* name=*"fullname"*><br>

Gender:

<input type=*"radio"* name=*"gender"* value=*"male"*> Male

<input type=*"radio"* name=*"gender"* value=*"female"*> Female<br>

Language known:

<input type=*"checkbox"* name=*"language"* value=*"hindi"*> Hindi

<input type=*"checkbox"* name=*"language"* value=*"english"*> English

<input type=*"checkbox"* name=*"language"* value=*"french"*> French<br>

Country:

<select name=*"country"*>

<option value=*"india"*>India</option>

<option value=*"us"*>US</option>

<option value=*"england"*>England</option>

<option value=*"france"*>France</option>

<option value=*"germany"*>Germany</option>

<option value=*"canada"*>Canada</option>

<option value=*"china"*>China</option>

<option value=*"russia"*>Russia</option>

</select><br>

<input type=*"submit"* value=*"Submit"*>

</form>

</body>

</html>

</body>

</html>

**Submit.jsp**

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>submit</title>

</head>

<body>

Name: <%= request.getParameter("fullname") %><br>

Gender :<%= request.getParameter("gender") %><br>

Language:

<%

String[] languages =request.getParameterValues("language");

**if**(languages!=**null**){

**for**(**int** i=0;i<languages.length;i++){

out.print("<br>");

out.print(languages[i]);

}

}

%>

<br>

Country : <%= request.getParameter("country") %><br>

</body>

</html>

1. **Forms Under Servlets.**

**Forms.jsp**

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Document</title>

</head>

<body>

<!DOCTYPE html>

<html>

<body>

<form action=<%= request.getContextPath() %>/controller method=*"get"*>

Full name: <input type=*"text"* name=*"fullname"*><br>

Gender:

<input type=*"radio"* name=*"gender"* value=*"male"*> Male

<input type=*"radio"* name=*"gender"* value=*"female"*> Female<br>

Language known:

<input type=*"checkbox"* name=*"language"* value=*"hindi"*> Hindi

<input type=*"checkbox"* name=*"language"* value=*"english"*> English

<input type=*"checkbox"* name=*"language"* value=*"french"*> French<br>

Country:

<select name=*"country"*>

<option value=*"india"*>India</option>

<option value=*"us"*>US</option>

<option value=*"england"*>England</option>

<option value=*"france"*>France</option>

<option value=*"germany"*>Germany</option>

<option value=*"canada"*>Canada</option>

<option value=*"china"*>China</option>

<option value=*"russia"*>Russia</option>

</select><br>

<input type=*"submit"* value=*"Submit"*>

</form>

</body>

</html>

</body>

</html>

**Controller.java**

**package** org.vikashlearning;

**import** java.io.IOException;

**import** jakarta.servlet.ServletException;

**import** jakarta.servlet.http.HttpServlet;

**import** jakarta.servlet.http.HttpServletRequest;

**import** jakarta.servlet.http.HttpServletResponse;

**public** **class** controller **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

controller() {

**super**();

// **TODO** Auto-generated constructor stub

}

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

// **TODO** Auto-generated method stub

response.getWriter().append("Served at: ").append(request.getContextPath());

}

**protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

// **TODO** Auto-generated method stub

doGet(request, response);

}

}