Servlet technology

---------------------------------

Tech stack?

* Desigining a web application involves the tasks like displaying the data sent from the server, handling client side operations, handling server side operations and handling database operations.
* To ensure the smooth execution of these operations, we use a combination of programming languages, tools and frameworks, called a tech stack.
* The tech stack, is visualized as two parts.

1. front end: refers to the technologies used for interaction with the user.
2. back end: refers to the technologies used on the server for processing the user inputs and interacting with the database.

* some of the tech stacks are,

MERN(MonogDB-Express.js-React JS-Node.js)

MEAN(MongoDB-Express.js-Angular-Node.js)

LAMP(Linux-Apache-MySQL-PHP)

MongoDB:

* MongoDB is an open source document oriented database written in C++.
* Data in MongoDB is stored in documents, with in a collection, as a set of name-value pairs.

Express.js:

* Express.js is a lightweight application development framework.
* Express.js is used for back-end development.

Angular:

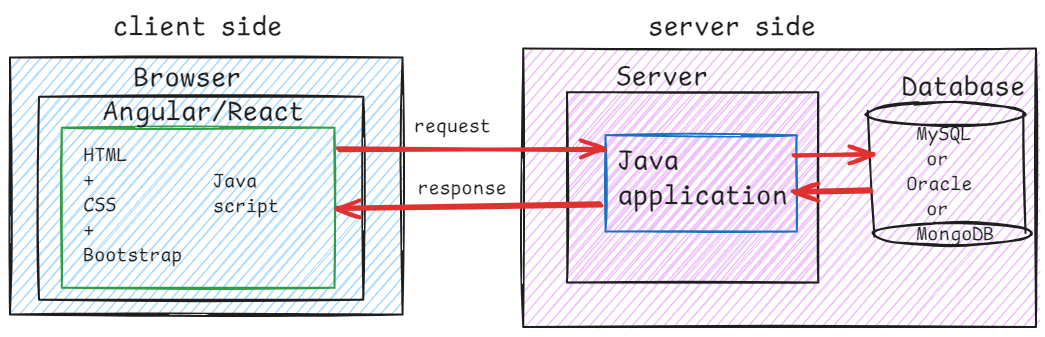
* Angular is an open source Javascript framework developed by Google, for building both mobile and web applications.
* It is a good framework for rapid front-end development.

Node.js:

* Node.js is a server-side, open source Javascript execution environment.
* It is built on Google chrome’s V8 Javascript runtime.

React JS:

* React is an open source Javascript library/framework developed by facebook, for building both mobile and web applications.
* It is a good framework for rapid front-end development.
* The MEAN / MERN stack, is good for developing client-side application development, not for server-side application development.
* client-side application runs on the user’s machine or mobile phone, which will communicate with the server to fetch the required information.
* server-side application contains the business logic and validations for processing the data sent by the client.
* For server-sde application development, we use the technologies like Java, Kotlin, .net, Python, etc..



Types of web applications:

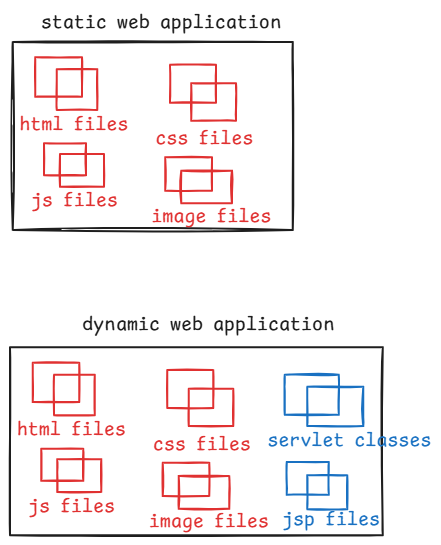
1. static web application
2. dynamic web application

* static web applications will deliver fixed content to every user.
* The webpages are pre-defined and they doesn’t change.
* If any changes are need to the page, the developer will do it manually.
* static web applications doesn’t contain any server-side logic and also they don’t need any database interactions.
* To Build static web application, you can use HTML, CSS, Basic Javascript.
* examples:

1. online tutotials
2. product launch websites
3. event promotion websites, etc..

* Dynamic web applications will deliver dynamic content to the users, based on user interactions.
* Dynamic web application’s contains server-side logics and often they interact with a database.
* To build dynamic web application, you can use HTML, CSS, JavaScript + server-side technologies like Java, PHP, etc..
* examples:

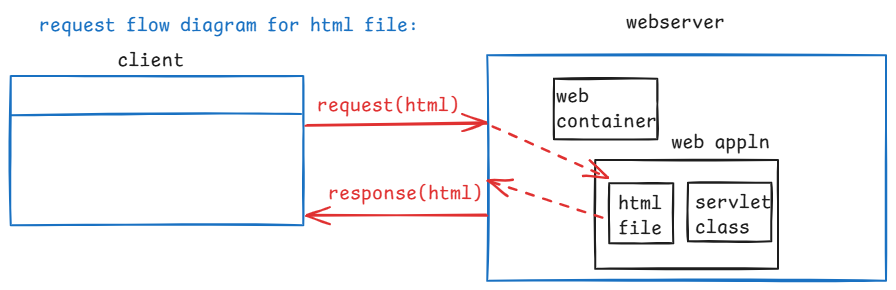
1. bookmyshow
2. irctc
3. facebook
4. youtube
5. amazon, etc..

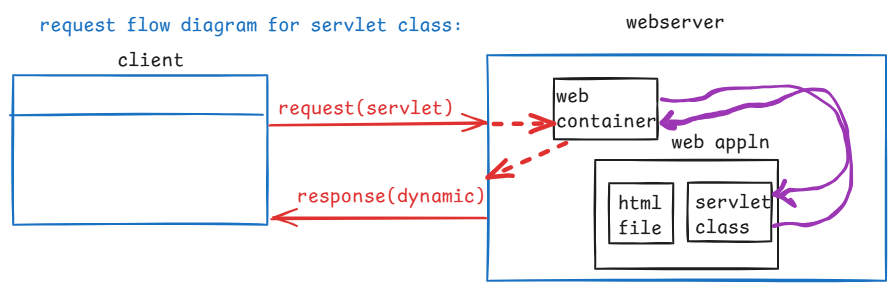


servlet definition:

* A servlet is a Java program, runs on a web server, handles the clients requests, generates response, sends it back to the client.
* servlet is the first Java web technology for developing dynamic web applications.
* A server is not a physical computer.
* A server is a software that has the capabilities of listening to the clients, handling the requests and sending back responses to the clients.
* A container(web container) is a part of server, it knows how to run servlets and how to manage their life cycles.

Server and Container will work together:





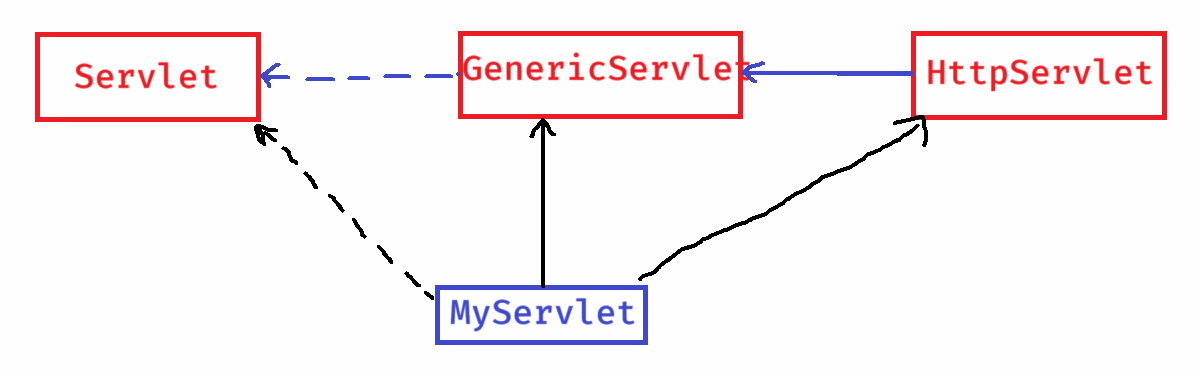
creating a servlet:

1. implement Servlet (interface)
2. extend GenericServlet (abstract class)
3. extend HttpServlet (abstract class)

* If you want to create a servlet from scratch then you can implement Servlet interface.
* jakarta.servlet.Servlet interface, has 5 abstract methods.

1. init(ServletConfig config)
2. getServletConfig()
3. service(ServletRequest req, ServletResponse resp)
4. getServletInfo()
5. destroy()

* GenericServlet is an abstract class that implements Servlet interface, and it is protocol-indepdent.
* protocol-independent means, it does not depend on any one specific protocol.
* It can handle any kind of request, like HTTP or FTP or SMTP, etc.. The only thing is you have to define the logic for processing the request.
* HttpServlet is a sub class of GenericServlet and it is specially designed for web applications, to handle HTTP protocol requests. So, it is protocol-dependent.



* When extending GenericServlet class, you have to override its abstract method called service method.

ex:

public class MyServlet extends GenericServlet {

@Override

public void service(ServletRequest req, ServletResponse resp)

throws ServletException, IOException

{

//define the logic

}

}

* When you are extending HttpServlet, you have to override the methods like doGet(), doPost(), …

ex:

public class MyServlet extends HttpServlet

{

@Override

public void doGet(HttpServletRequest req,HttpServletResponse resp) throws ServletException, IOException

{

//define the logic

}

@Override

public void doPost(HttpServletRequest req,HttpServletResponse resp) throws ServletException, IOException

{

//define the logic

}

}

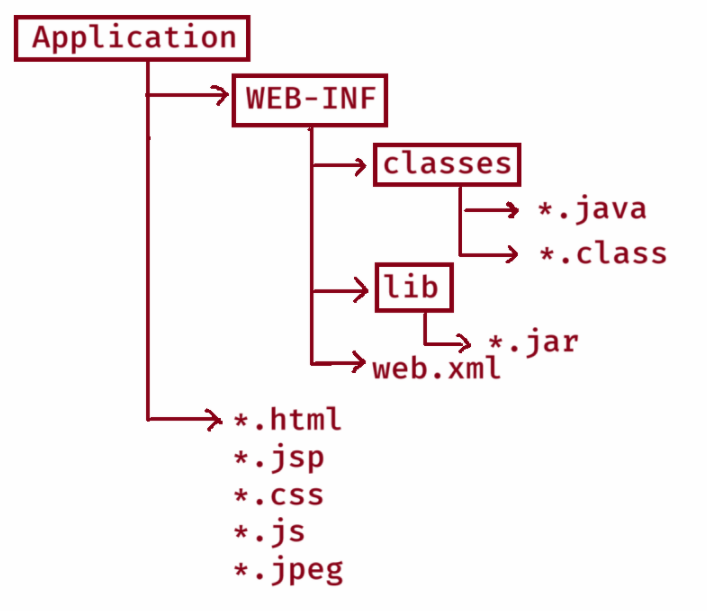
servlet configuration:

* It means, you should provide some information to the web container like which servlet class should be executed for which url.
* To provide this information, we have to define the servlet configuration in the web.xml file.

for example:



Folder structure of web application:



Install Tomcat server:

1. visit tomcat.apache.org
2. download Tomcat10
3. download 32bit/64bit Windows Service installer
4. double click on the downloaded apache-tomcat-10.1.40 file and click on next buttons.
5. change shutdown port: 8085

HTTP connector port : 2025

username : admin

password : admin

1. click Next 🡪 before click on finish, uncheck the boxes.
2. Now the location where tomcat is installed is,

C:\Program Files\Apache Software Foundation\Tomcat 10.1

request parameters:

* request parameters are used to send a user’s input to the server.
* request parameters can be sent in the url or can be sent by designing a html form.
* ex:
* <http://localhost:2025/Application1/srv?username=admin&email=admin@gmail.com>
* In servlet class, you can receive the parameters like below.

ex:

String user = req.getParameter(“username”);

String email= req.getParameter(“email”);



index.html

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset=*"UTF-8"*>**

**<title>Index Page</title>**

**<style>**

**body {**

**margin: *0*;**

**align-items: *center*;**

**justify-items: *center*;**

**}**

**a {**

**font-size: *24px*;**

**}**

**</style>**

**</head>**

**<body>**

**<a href = *"./WelcomeSrv"*>click here</a>**

**</body>**

**</html>**

WelcomeServlet.java

**package com.ashokit;**

**import java.io.IOException;**

**import java.io.PrintWriter;**

**import jakarta.servlet.GenericServlet;**

**import jakarta.servlet.ServletException;**

**import jakarta.servlet.ServletRequest;**

**import jakarta.servlet.ServletResponse;**

**public class WelcomeServlet extends GenericServlet {**

**@Override**

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

**resp.setContentType("text/html");**

**PrintWriter out = resp.getWriter();**

**out.println("<html>");**

**out.println("<body>");**

**out.println("<h3>");**

**out.println("Welcome to Servlets");**

**out.println("</h3");**

**out.println("</body>");**

**out.println("</html>");**

**out.close();**

**}**

**}**

web.xml

**<?xml version=*"1.0"* encoding=*"UTF-8"*?>**

**<web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns=*"https://jakarta.ee/xml/ns/jakartaee"* xsi:schemaLocation=*"https://jakarta.ee/xml/ns/jakartaee https://jakarta.ee/xml/ns/jakartaee/web-app\_5\_0.xsd"* id=*"WebApp\_ID"* version=*"5.0"*>**

**<display-name>ServletApplication1</display-name>**

**<welcome-file-list>**

**<welcome-file>index.html</welcome-file>**

**</welcome-file-list>**

**<servlet>**

**<servlet-name>welcome</servlet-name>**

**<servlet-class>com.ashokit.WelcomeServlet</servlet-class>**

**</servlet>**

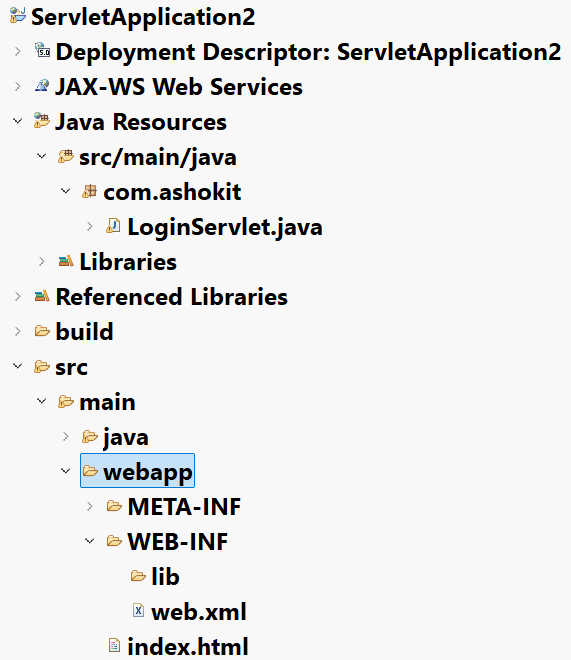
**<servlet-mapping>**

**<servlet-name>welcome</servlet-name>**

**<url-pattern>/WelcomeSrv</url-pattern>**

**</servlet-mapping>**

**</web-app>**



index.html

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset=*"UTF-8"*>**

**<title>Login Page</title>**

**</head>**

**<body>**

**<form action = *"./LoginSrv"* method = *"post"*>**

**Username : <input type=*"text"* name=*"username"*> <br>**

**Password : <input type=*"password"* name=*"password"*> <br>**

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

**</form>**

**</body>**

**</html>**

LoginServlet.java

package com.ashokit;

import java.io.IOException;

import java.io.PrintWriter;

import jakarta.servlet.GenericServlet;

import jakarta.servlet.Servlet;

import jakarta.servlet.ServletException;

import jakarta.servlet.ServletRequest;

import jakarta.servlet.ServletResponse;

import jakarta.servlet.annotation.WebServlet;

/\*\*

\* Servlet implementation class LoginServlet

\*/

public class LoginServlet extends GenericServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* @see GenericServlet#GenericServlet()

\*/

public LoginServlet() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see Servlet#service(ServletRequest request, ServletResponse response)

\*/

public void service(ServletRequest request, ServletResponse response) throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out = response.getWriter();

String user = request.getParameter("username");

String pwd = request.getParameter("password");

if(user!=null && user.equals("admin") && pwd !=null && pwd.equals("admin"))

{

out.println("<html> <body>");

out.println("<font color = 'green'> Login Success </font>");

out.println("</body> </html>");

}

else

{

out.println("<html> <body>");

out.println("<font color = 'red'> Login Failed </font>");

out.println("</body> </html>");

}

out.close();

}

}

web.xml

**<?xml version=*"1.0"* encoding=*"UTF-8"*?>**

**<web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns=*"https://jakarta.ee/xml/ns/jakartaee"* xsi:schemaLocation=*"https://jakarta.ee/xml/ns/jakartaee https://jakarta.ee/xml/ns/jakartaee/web-app\_5\_0.xsd"* id=*"WebApp\_ID"* version=*"5.0"*>**

**<display-name>ServletApplication2</display-name>**

**<welcome-file-list>**

**<welcome-file>index.html</welcome-file>**

**</welcome-file-list>**

**<servlet>**

**<servlet-name>login</servlet-name>**

**<servlet-class>com.ashokit.LoginServlet</servlet-class>**

**</servlet>**

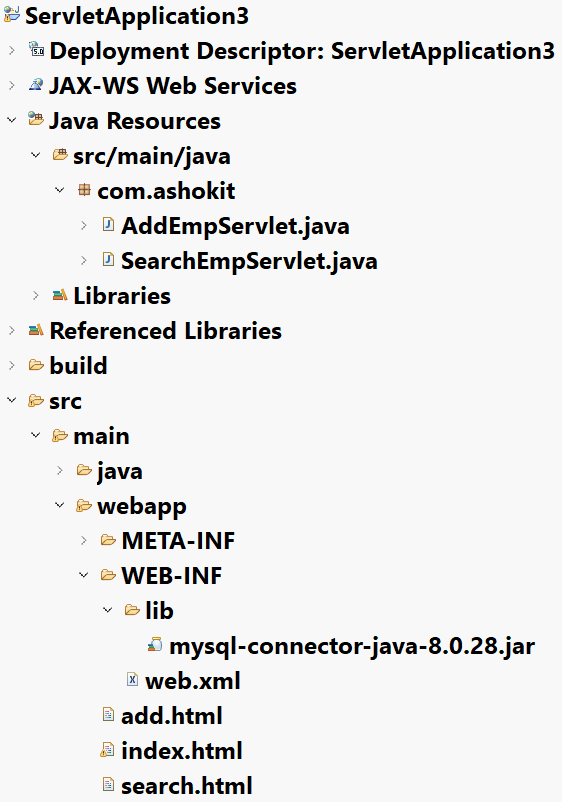
**<servlet-mapping>**

**<servlet-name>login</servlet-name>**

**<url-pattern>/LoginSrv</url-pattern>**

**</servlet-mapping>**

**</web-app>**



index.html

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset=*"UTF-8"*>**

**<title>index page</title>**

**</head>**

**<body>**

**<center>**

**<a href = *"./add.html"*>add employee</a> <br>**

**<a href = *"./search.html"*>search employee</a>**

**</center>**

**</body>**

**</html>**

**add.html**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset=*"UTF-8"*>**

**<title>add employee</title>**

**</head>**

**<body>**

**<form action =*"./addSrv"* method=*"post"*>**

**<fieldset style="width: *300px*; padding: *20px*;">**

**<legend style="font-weight: *bold*;">Employee Form</legend>**

**<label for=*"empno"*>Empno:</label> <br>**

**<input type=*"text"* name=*"empno"*> <br> <br>**

**<label for=*"ename"*>Ename:</label> <br>**

**<input type=*"text"* name=*"ename"*> <br> <br>**

**<label for=*"sal"*>Sal:</label> <br>**

**<input type=*"text"* name=*"sal"*> <br> <br>**

**<label for=*"deptno"*>Deptno:</label> <br>**

**<input type=*"text"* name=*"deptno"*> <br> <br>**

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

**</fieldset>**

**</form>**

**</body>**

**</html>**

**search.html**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset=*"UTF-8"*>**

**<title>Search Page</title>**

**</head>**

**<body>**

**<form action = *"./searchSrv"* method = *"post"*>**

**<fieldset style = "width: *300px*; padding: *20px*;">**

**<legend style ="font-weight: *bold*;">Search for Employee</legend>**

**<label for=*"empno"*>Empno:</label>**

**<input type=*"text"* name=*"empno"*> <br> <br>**

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

**</fieldset>**

**</form>**

**</body>**

**</html>**

**AddEmpServlet.java**

**package com.ashokit;**

**import java.io.IOException;**

**import java.io.PrintWriter;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.PreparedStatement;**

**import jakarta.servlet.ServletException;**

**import jakarta.servlet.annotation.WebServlet;**

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

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

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

**/\*\***

**\* Servlet implementation class AddEmpServlet**

**\*/**

**@WebServlet("/addSrv")**

**public class AddEmpServlet extends HttpServlet {**

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

**/\*\***

**\* @see HttpServlet#HttpServlet()**

**\*/**

**public AddEmpServlet() {**

**super();**

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

**}**

**/\*\***

**\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)**

**\*/**

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

**/\*\***

**\* read the request parameters**

**\*/**

**int empno = Integer.parseInt(request.getParameter("empno"));**

**String ename = request.getParameter("ename");**

**double sal = Double.parseDouble(request.getParameter("sal"));**

**int deptno = Integer.parseInt(request.getParameter("deptno"));**

**response.setContentType("text/html");**

**PrintWriter out = response.getWriter();**

**loadDriver();**

**try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/test", "root", "root");**

**PreparedStatement ps = conn.prepareStatement("INSERT INTO EMP VALUES(?,?,?,?)")**

**)**

**{**

**ps.setInt(1, empno);**

**ps.setString(2, ename);**

**ps.setDouble(3, sal);**

**ps.setInt(4, deptno);**

**ps.executeUpdate();**

**out.println("<html> <body> <h2> Employee added to the Database </h2> </body> </html>");**

**}**

**catch(Exception ex) {**

**out.println("<html> <body> <h3> Problem occurred </h3> </body> </html>");**

**ex.printStackTrace();**

**}**

**out.close();**

**}**

**private void loadDriver() {**

**try {**

**Class.forName("com.mysql.cj.jdbc.Driver");**

**} catch(Exception ex) {**

**ex.printStackTrace();**

**}**

**}**

**}**

**SearchEmpServlet.java**

**package com.ashokit;**

**import java.io.IOException;**

**import java.io.PrintWriter;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.PreparedStatement;**

**import java.sql.ResultSet;**

**import jakarta.servlet.ServletException;**

**import jakarta.servlet.annotation.WebServlet;**

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

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

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

**/\*\***

**\* Servlet implementation class SearchEmpServlet**

**\*/**

**@WebServlet("/searchSrv")**

**public class SearchEmpServlet extends HttpServlet {**

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

**/\*\***

**\* @see HttpServlet#HttpServlet()**

**\*/**

**public SearchEmpServlet() {**

**super();**

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

**}**

**/\*\***

**\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)**

**\*/**

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

**/\*\***

**\* read request parameter**

**\*/**

**int empno = Integer.parseInt(request.getParameter("empno"));**

**response.setContentType("text/html");**

**PrintWriter out = response.getWriter();**

**loadDriver();**

**try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/test", "root", "root");**

**PreparedStatement ps = conn.prepareStatement("SELECT \* FROM EMP WHERE EMPNO = ?")**

**)**

**{**

**ps.setInt(1, empno);**

**ResultSet rs = ps.executeQuery();**

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

**rs.next();**

**out.println("<html> <body>");**

**out.println("Empno : " + rs.getInt(1));**

**out.println("<br>");**

**out.println("Ename : " + rs.getString(2));**

**out.println("<br>");**

**out.println("Sal : " + rs.getDouble(3));**

**out.println("<br>");**

**out.println("Deptno : " + rs.getInt(4));**

**out.println("<br>");**

**out.println("</body> </html>");**

**}**

**else {**

**out.println("<html> <body>");**

**out.println("<h2> No records found! </h2>");**

**out.println("</body> </html>");**

**}**

**}**

**catch(Exception ex) {**

**out.println("<html> <body> Problem occurred </body> </html>");**

**ex.printStackTrace();**

**}**

**out.close();**

**}**

**private void loadDriver() {**

**try {**

**Class.forName("com.mysql.cj.jdbc.Driver");**

**} catch(Exception ex) {**

**ex.printStackTrace();**

**}**

**}**

**}**

Request dispatching in servlets:

* Request dispatching is a mechanism which can be used to delegate the work between the servlets or JSP files.
* Request dispatching is of two types.
* 1. forwarding
* 2. including
* forwarding is used to delegate a work from one servlet to another servlet.
* forwarding means, the current servlet pass the control to another servlet. Now, the current servlet stops working and the forwarded servlet stars working and creates the response.
* For example, we have OrderServlet and PaymentServlet, where OrderServlet will processes the cart, then forwards the request to PaymentServlet for receiving the payment.
* Another example, we have LoginServlet, HomeServlet and ErrorServlet. A LoginServlet performs security check, if success then forwards the request to HomeServlet. If failure, then forwards the request to ErrorServlet.
* including means, inserting the output of another servlet inside the current response.
* For example, we have ContentServlet, HeaderServlet and FooterServlet. Now the ContentServlet includes the output of header and footer.
* Another example, we have HomeServlet, WeatherServlet and NewsServlet. Here, HomeServlet includes the output from weather and news.
* RequestDispatcher interface is provided in the Servlet API, for forwading/including.
* For example, Servlet1 wants to forward a request to Servlet2. Now, in Servlet1, the below code is required.

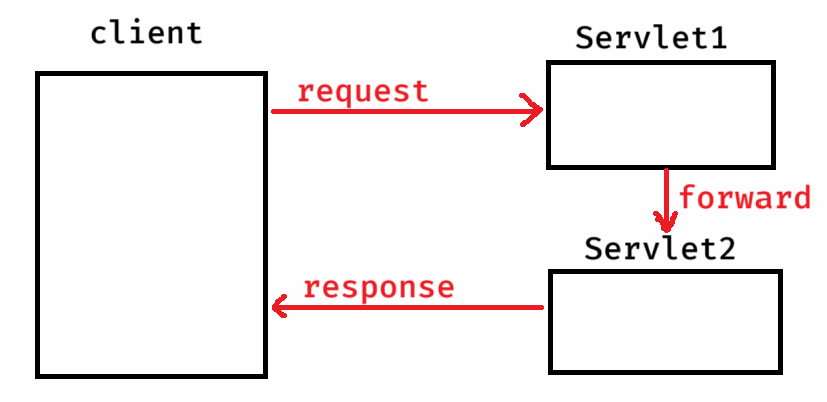
RequestDispatcher dispatcher = request.getRequestDispatcher(“/srv2”);

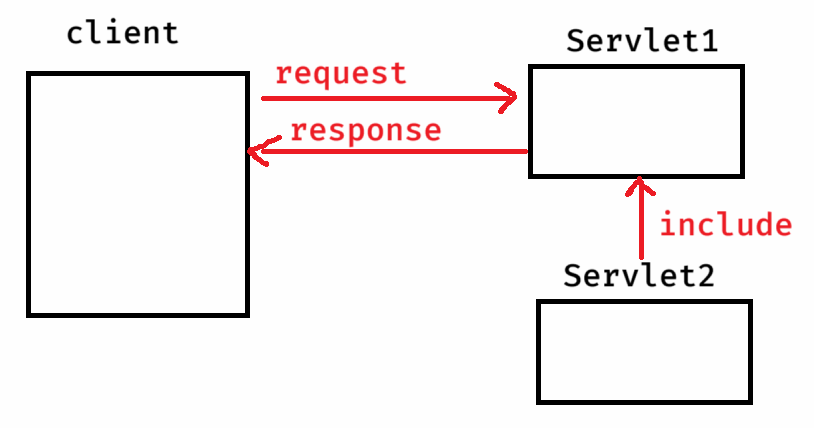
dispatcher.forward(request, response);

* For example, Servlet1 wants to insert the output of Servlet2 into its response. Now, in Servlet1, the below code is required.

RequestDispatcher dispatcher = request.getRequestDispatcher(“/srv2”);

dispatcher.include(request, response);







Login.html

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset=*"UTF-8"*>**

**<title>Login Page</title>**

**</head>**

**<body>**

**<form action = *"./LoginSrv"* method = *"post"*>**

**<fieldset style = "width: *200px*; padding: *20px*;">**

**<legend style = "font-weight: *bold*;">Login Form</legend>**

**<label>Username</label> <br>**

**<input type=*"text"* name=*"username"*> <br><br>**

**<label>Password</label> <br>**

**<input type=*"password"* name=*"password"*> <br><br>**

**<button type=*"submit"*>submit</button>**

**</fieldset>**

**</form>**

**</body>**

**</html>**

LoginServlet.java

**@WebServlet("/LoginSrv")**

**public class LoginServlet extends HttpServlet {**

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

**/\*\***

**\* @see HttpServlet#HttpServlet()**

**\*/**

**public LoginServlet() {**

**super();**

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

**}**

**/\*\***

**\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)**

**\*/**

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

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

**//read the request parameters**

**String username = request.getParameter("username");**

**String password = request.getParameter("password");**

**if (username.equals("Ashokit") && password.equals("ashokit@123"))**

**{**

**//forward the request to CourseServlet**

**RequestDispatcher dispatcher = request.getRequestDispatcher("./CourseSrv");**

**dispatcher.forward(request, response);**

**}**

**else**

**{**

**PrintWriter out = response.getWriter();**

**out.println("<html> <body>");**

**out.println("<font color='red' size=18> Credentials are invalid! </font>");**

**out.println("<br>");**

**out.println("<a href='./Login.html'> Login again </a>");**

**out.println("</body> </html>");**

**out.close();**

**}**

**}**

**}**

CourseServlet.java

**@WebServlet("/CourseSrv")**

**public class CourseServlet extends HttpServlet {**

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

**/\*\***

**\* @see HttpServlet#HttpServlet()**

**\*/**

**public CourseServlet() {**

**super();**

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

**}**

**/\*\***

**\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)**

**\*/**

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

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

**// include the output of HeaderServlet**

**RequestDispatcher dispatcher = request.getRequestDispatcher("./HeaderSrv");**

**dispatcher.include(request, response);**

**// Generate the body content**

**PrintWriter out = response.getWriter();**

**out.println("<html> <body>");**

**out.println("<h2>");**

**out.println("Java FullStack Development");**

**out.println("<ol>");**

**out.println("<li>Front-end(HTML,CSS,JS,Bootstrap,React) </li>");**

**out.println("<li>Back-end(Java, Spring, Boot, Microservices) </li>");**

**out.println("<li>Database(Oracle/MySQL/MongoDB)</li>");**

**out.println("</ol> </h2> </body> </html>");**

**// include the output of FooterServlet**

**RequestDispatcher dispatcher2 = request.getRequestDispatcher("./FooterSrv");**

**dispatcher2.include(request, response);**

**}**

**}**

HeaderServlet.java

**@WebServlet("/HeaderSrv")**

**public class HeaderServlet extends HttpServlet {**

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

**/\*\***

**\* @see HttpServlet#HttpServlet()**

**\*/**

**public HeaderServlet() {**

**super();**

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

**}**

**/\*\***

**\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)**

**\*/**

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

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

**PrintWriter out = response.getWriter();**

**out.println("<html> <body>");**

**out.println("<center>");**

**out.println("<h2> AshokIT technologies </h2>");**

**out.println("</center>");**

**out.println("</body> </html>");**

**}**

**}**

FooterServlet.java

**@WebServlet("/FooterSrv")**

**public class FooterServlet extends HttpServlet {**

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

**/\*\***

**\* @see HttpServlet#HttpServlet()**

**\*/**

**public FooterServlet() {**

**super();**

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

**}**

**/\*\***

**\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)**

**\*/**

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

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

**PrintWriter out = response.getWriter();**

**out.println("<html> <body>");**

**out.println("<center>");**

**out.println("&copy; All rights reserved, Hyderabad");**

**out.println("</center>");**

**out.println("</body> </html>");**

**}**

**}**

Session Management

* HTTP is a stateless protocol.
* stateless means, every request from a client to a server is treated as an independent request. The server does not rememeber previous interactions with the client.
* Suppose, if a user is sending multiple requests, the server treats like each request is coming from a new user.
* In most web applications, keeping track of information about the user across the mulitple requests is required.
* So, session management techniques are used.
* Session management is a process, used to maintain stateful interactions between the client and server in web applications, despite of the stateless nature of HTTP.
* Suppose, in E-commerce applications, if session management technique is not used, then when a user is moving from one page another page, the server treats that user as a new user and the cart will be reset with every request. So, a user can not purchase multiple items at a time.
* If session management technique is used, the server can track and manage the user and the items added to the cart when a user is moving between pages. Which means, the server remembers the user’s data across mulitple requests.
* The servlets technology has provided HTTP Cookies and HttpSession API to manage the sessions.

HTTP Cookies:

-------------

* Cookie class is provided in Servlet API, to create cookies.
* A cookie is an object, which can store a key-value pair, created at server and the server sends to the client, where client will store and sends back to the server with the subsequent requests.
* The server can create multiple cookies to keep track of user information across multiple requests.
* If there are more cookies, then sending from server to client and then client to server will increase network traffic and the cookies may lost due to network errors.
* So, HTTP cookies are good for managing small piece of information across multiple requests and it is not a good technique to manage large amount of information.
* A cookie can be created in a servlet class like below.

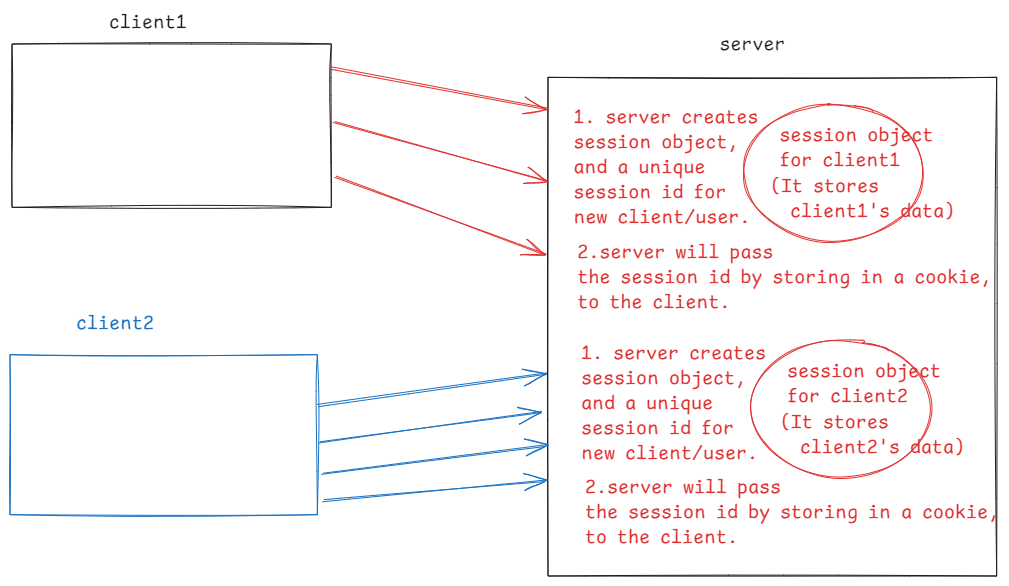
Cookie loginCookie = new Cookie(“username”, username);

loginCookie.setMaxAge(30 \* 60); //expires in 30 minutes

resp.addCookie(loginCookie); //add cookie to the response

HttpSession API

* HttpSession API is the most common way to handle session management in servlets.
* A HttpSession object is created at server, to store the user data across multiple requests, for each user/client.
* The server also creates a unique session id for each user/client and this id will be passed in a cookie to the client.
* When the client is sending subsequent requests, the client will also send the cookie with session id to the server. With this session id, the server will recognize the client as an existing user.
* The session object is created at server, for each client, but not for each request.
* Suppose, there are 2 clients, where client1 is sending 3 requests to server and client2 is sending 5 requests to server. So, the number of session objects created by the server is 2.



creating a session object:

. a session object can be created by calling getSession() method.

. when a user has sent first request, it means, the user is a new user then this getSession() method creates a new session object.

. when a user has sent next request, it means, the user is an existing user then this getSession() method retrieves the session object associated with that user.

HttpSession session = request.getSession();

. we can also pass boolean parameter to the getSession() method

. getSession(true) works like getSession() only.

. getSession(false) returns the session object of a user. If not exist then returns null.

storing or retrieving the data:

. session object stores data as key-value pairs.

. A user data can be stored in a session object, by calling

setAttribute(key, value) method.

. The data can be retrieved from a session object by calling

getAttribute(key) method.

. The data can be removed from a session object by calling

removeAttribute(key) method.

expiring a session object:

. a session object should be expired from a server, when a user

logout of the application or when a user is not coming back to

the server for some period.

. To remove the session object, when a user logout then call

invalidate() method.

. The inactivity period can be set for a session, by calling

setMaxInactiveInterval() method.

for example,

session.setMaxInactiveInterval(120); // 2 minutes

