Servlets

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

By using java programing language we are able to prepare two types of applications

1. Standalone Applicationsà applications run on single server or without using client server arch.

|

|-----CUI[command prompt user interface]

|

|----GUI[Graphical user interface]

2. Enterprise Applications/ Distributed Applications

It we design and execute any application on the basis of client server architecture or by distributing application logic over multiple machines the that application is called as “enterprise application” or Distributed Application

There are two types of enterprise Application

1. Web Application/ web based Distributed Applications.

2. Distributed Applications/ Remote based Distributed Application

I.Q) What is the Difference between web applications And Distributed Applications.

1. Web Application is a Client-Server Application where the complete application logic is provided by the server machine.

Distributed applications is a Client-server application where complete application logic Distributed over multiple machines that are local machines or Remote machines.

2.in Web Applications client is Fixed that is Browser.

In Distributed Application client is not fixed. It may be normal java program with main method, it may be GUI Application with an applet, it may be a servlet, it may be frameworks, Structs,JSF,spring.

3.to prepare web applications we have to use a set of technologies is called “web Technologies”

Ex:CGI[common gateway interface],servlets,jsp’s,PHP,PERL.

To prepare Distributed applications we have to use a set of technologies is called “Distributed Technologies”

Ex: socket programing,RMI[remote method innvocation],CORBA[common object request broker architecureagent],[EJB]Enterprise java beans,Web services…

4.web applications are executed both webservers and application servers.

Distributed applications are executed by only application servers.

5. web applications are mainly for static response and little bit of dynamic response.

Distributed applications are having main concentration on the communication between local machine remote machine in order to consume remote services from local machines.

In Web applications , there are two types of responses.

1. Static Response

2. Dynamic Response

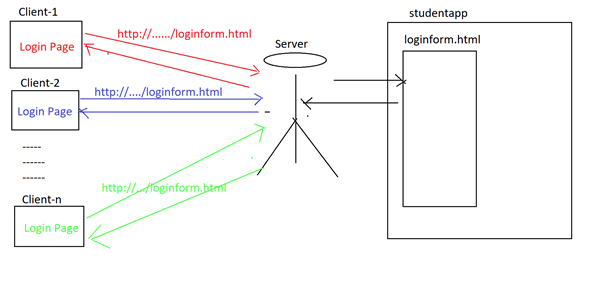
Q)What is the difference between Static Response and Dynamic Response?

—------------------------------------------------------------------------------

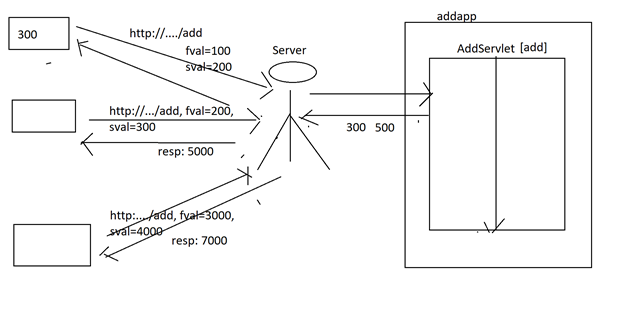
Static Response is the response from Server machine , it will be generated without executing any resource at Server machine and it will be generated from server without performing any action at server side application.

Static response is the response which must be common to every user of that server side application.

EX: Every User will get a login form when we use [www.gmail.com](http://www.gmail.com/).



Dynamic Response is the response from Server machine, it will be generated by executing a particular resource at server machine machine or by performing an action at server machine.



From the above representation, to generate dynamic response we have to use an application at server machine , that is web application, in web application we have to use some resources such as CGI, Servlets and JSPs.

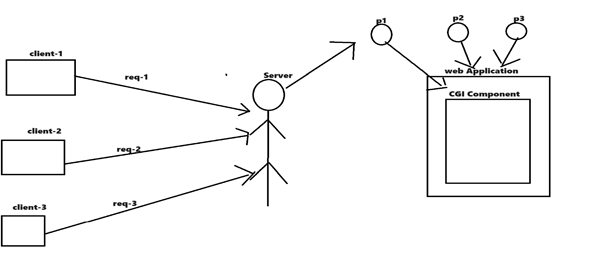
From the above representation, the main purpose of “Servlets and JSPs” is to prepare web applications in order to generate “Dynamic Response”.

IQ) To prepare web applications in order to generate dynamic response we have already CGI then what is the requirement to use servlets?

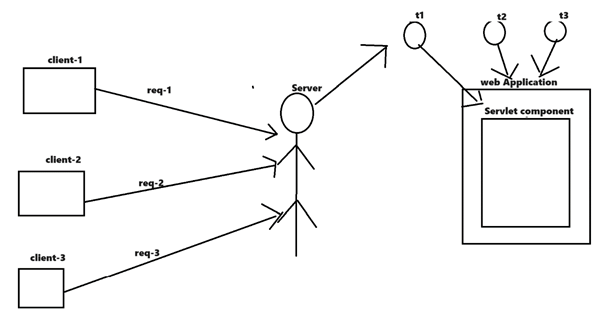
CGI is a server side technology, it can be used to prepare web Applications, it was designed on the basis of c and some other scripting languages, it is a process based technology, if we prepare web applications By using CGI, then container will create a separate process for each and every request coming from client.

Basically process is heavy weighted, so CGI is an Heavy weighted technology. It will take more memory and more execution time to process any request, it will take more response time it will be reduce server side application perfomence.

CGI is an error prone technology. More chances are available to get errors even in small piece of code also.



Servlet is a server side technology to prepare web applications. It was designed on the top of java, where java is a thread based programing language moreover multithreaded language so servlet is also a thread based server side technology if we design web application by using the servlets then container will create a separate thread instead of process for each and every request coming from client. Where thread is an lightweight component so servlet is also a lightweight server side technology. It will take less memory and less execution time to process request it will reduce overall response time. It will increase web applications perfomance.



Client-server Arch:

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

To prepare any enterprise applications, client-server arch minimum arch.

In client server arch there are 3 components are mainly

1. Client

2. Protocol

3. Server

1.client:- in client server application the main role of the client is to submit request to the server and to get response from the server.

At client machine to submit request to the client and to get response from the server we have to use a tool that is browser

2.protocol:- protocol is a set of rules and regulations to transfer data from one machine to another machine over the network

In client-server application the role of the protocol is to carry request data from client to server and carry response data from server to client.

Ex: HTTP,TCP,UDP,IP,FTP,RARP,ARP

3.Server:server is a software installed in a machine ,it is able to process the request.

In client-server applications the role of the server.

1.getting request from clients.

2. identifying that requested resources at server side applications.

3. executing the server side resources.

4.generating the dynamic response.

There are 2 types of servers.

1. Web servers

2. Application servers

1.web servers:- web servers are able to provide the an environment to execute web applications.

Ex: upto tomcat 5.x version web server. Ressign , oracle web,

2.Application servers: application servers are able to provide an environment to execute both web applications and distributed applications.

Ex: tomcat 6.x version, weblogic , wildfly , glassfish.

Steps to prepare first web[servlet] application

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

1. Download and install tomcat server.

2. Download and install java software IDE/ editor software.

3. Prepare web application architecture.

4. Prepare deployment descriptor[web.xml]

5. Prepare web resources like html, servlets, jsp’s.

6. Start the server and deploy the web application.

7. Access the web application

**Download And Install Server Software[Tomcat]:**

**—---------------------------------------------**

**Download the Tomcat 10 version from the following link.**

[**https://tomcat.apache.org/download-10.cgi**](https://tomcat.apache.org/download-10.cgi)

**Select “**[**32-bit/64-bit Windows Service Installer**](https://dlcdn.apache.org/tomcat/tomcat-10/v10.0.27/bin/apache-tomcat-10.0.27.exe) **(**[**pgp**](https://downloads.apache.org/tomcat/tomcat-10/v10.0.27/bin/apache-tomcat-10.0.27.exe.asc)**,** [**sha512**](https://downloads.apache.org/tomcat/tomcat-10/v10.0.27/bin/apache-tomcat-10.0.27.exe.sha512)**)”**

**Installation Process:**

**1.** **Double Click on the downloaded setup file.**

**2.** **Click on the “Next” button.**

**3.** **Click on the “I Agree” button.**

**4.** **Select “Host Manager” and “Examples” and Click on “Next” button.**

**5.** **Provide the following details.**

**Server Shutdown: 8976[Unknown 4-digit Number]**

**Http/1.1 connector : 1010[Changed from 8080]**

**User Name : malli**

**Password : malli**

**6.** **Click on the “Next” button.**

**7.** **Select “JRE8 or above” [i Selected JDK17].**

**8.** **Click on the “Next” button.**

**9.** **Change the installation location to “C:\Tomcat 10.0”.**

**10.** **Click on “Install” Button.**

**11.** **Un select “Run Apache Tomcat” and “Show Readme”.**

**12.** **Click on the “Finish” button.**

**To test whether Tomcat server is ready to run or not , execute the “Tomcat10” application under “C:\Tomcat 10.0\bin”, observe the Server command prompt where all the server logs.**

**Note: If we want to know or modify Server port numbers then use the following xml file.**

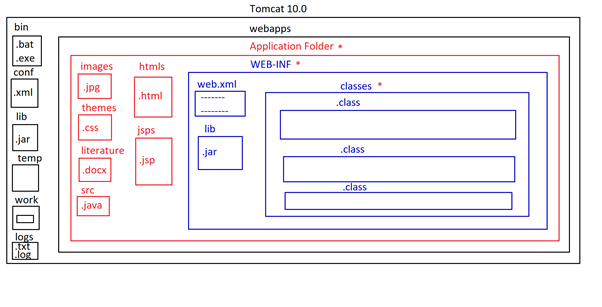
**C:\Tomcat 10.0\conf\server.xml.**

**Note: If we want to know or modify Tomcat Server user name and password then use the following xml file.**

**C:\Tomcat 10.0\conf\tomcat-users.xml**

**Prepare web application Architecture**

**—-------------------------------------**



**Prepare Deployment Descriptor[wb.xml]**

**—-------------------------------------**

**Deployment Descriptor is a web.xml file , it will provide description or meta data about our web application which is required by the container inorder to identify and execute server side resources.**

**In Web applications, Deployment Descriptor is able to provide the following configuration details of the present web application.**

1. **Welcome Files Configuration**
2. **Display Name Configuration**
3. **Servlets Configuration**
4. **Load on startup configuration**
5. **JSPs Configurations**
6. **Filters Configurations**
7. **Listeners Configurations**
8. **Session timeout configurations**
9. **Tag libraries configurations**
10. **Initialization parameters configurations**
11. **Context parameters Configurations**
12. **Security Configurations**
13. **Error pages Configurations**

—-----------

—-----------

**Servlet Configuration in Deployment Descriptor:**

**In general, we will keep servlets under “classes” folder that is private area, here to access Servlet classes we have to define an alias name or an url pattern or a locator , to define alias name for the servlet and to provide mapping between alias name and the servlet we must provide Servlet configuration in web.xml file.**

**To provide Servlets configuration in web.xml file we have to use the following tags in web.xml file.**

**<web-app>**

**—-----**

**<servlet>**

**<servlet-name>logicalName</servlet-name>**

**<servlet-class>Fully Qualified Name of the Servlet class</servlet-class>**

**—------**

**</servlet>**

**<servlet-mapping>**

**<servlet-name>logicalName</servlet-name>**

**<url-pattern>/patternName1</url-pattern>**

**<url-pattern>/patternName2</url-pattern>**

**—-----**

**—-----**

**</servlet-mapping>**

**—------**

**</web-app>**

**From the above web.xml file,**

1. **<web-app> is a root tag in a web.xml file that represents present web application configuration details.**
2. **<servlet> tag is able to provide servlet details.**
3. **<servlet-name> tag is able to provide the logical name of the servlet class.**
4. **<servlet-class> tag is able to provide a fully qualified name of the servlet class.**
5. **<servlet-mapping> tag is able to provide mapping details of the Servlet class with url patterns.**
6. **<url-pattern> tag is able to provide url pattern or alias name or locator for the servlet class.**

**Note: It is possible to define more than one url pattern for a single Servlet, it is possible right from Servlets2.5 version onwards, it is not possible up to Servlet 2.4 version.**

**EX:**

<web-app>

<servlet>

<servlet-name>loginServlet</servlet-name>

<servlet-class>com.dss.LoginServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>loginServlet</servlet-name>

<url-pattern>/login</url-pattern>

</servlet-mapping>

</web-app>

**Q)Is it possible to prepare web applications without a web.xml file?**

**—----------------------------------------------------------**

**Ans:**

**—--**

**In web applications, web.xml file is optional or mandatory depending on the following cases.**

**Case#1: Up to Servlet 2.5 version**

1. **If we prepare web applications without using the server side components like Servlets, Filters, Listeners,.... then the web.xml file is optional or mandatory completely depending on the server.**

**a)If we use the servers like Tomcat then a web.xml file is optional.**

**b)If we use the servers like weblogic then a web.xml file is mandatory.**

1. **If we prepare the web applications with the Server side components like Servlets, Filters and Listeners,....then a web.xml file is mandatory.**

**Case#2: Right from Servlets 3.x version**

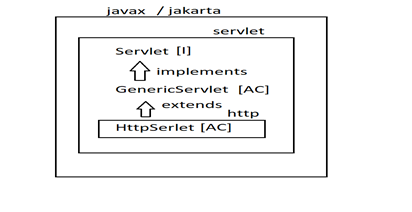
**In Servlets 3.x version web.xml file is optional irrespective of having the server side components like Servlets, Filters, Listeners,.... , because Servlets 3.x version has provided “Annotations” as an alternative for web.xml file.**

**Prepare Web Resources:**

—------------------------

In general, in web applications we will use web resources like Servlets, JSPs, html,..........

In web applications, To prepare Servlet classes Servlet API has provided the following predefined library.



**Q)What is Servlet and in how many ways are we able to prepare Servlets in web applications?**

**—---------------------------------------------------------**

**Ans:**

**—--**

Servlet is a Server side program , by executing this server side program some response will be generated by the container and server to the respective client.

Servlet is an object maintained by the container and it must implement a “Servlet” interface either directly or indirectly.

As per the Servlet API provided predefined library there are three ways to prepare Servlets.

1. Implementing “Servlet” interface:

public class WelcomeServlet implements javax.servelt. Servlet{

--- Implementation for all methods of servlet interface---

—-----

}

1. Extending “GenericServlet” abstract class:

public class WelcomeServlet extends GenericServlet{

—implementation for all services provided methods--

}

1. Extending “HttpServlet” abstract class:

public class WelcomeServlet extends HttpServlet{

--implementation for service provider methods---

—--------

}

**Servlets Design:**

**—-----------------**

There are three ways to prepare Servlets.

1. By implementing the “Servlet” interface.
2. By Extending “GenericServlet” abstract class.
3. By Extending “HttpServlet” abstract class.

**By implementing the Servlet interface:**

**—------------------------------------------**

“Servlet” is an interface provided by Servlet API in the form of “javax.servlet.Servlet” or “jakarta.servlet.Servlet” and it has the following methods.

public interface Servlet{

public void init(ServletConfig config)throws

ServletException;

public void service(

ServletRequest request,

ServletResponse response)throws

ServletException, IOException;

public ServletConfig getServletConfig();

public String getServletInfo();

Public void destroy();

}

**From the above Servlet interface methods , the methods like init() method, service() method and destroy() are involved in Servlet lifecycle, so these methods are called “Lifecycle Methods”, the remaining methods like getServletConfig() method and getServletinfo() method are called “Non Lifecycle Methods”.**

**Where the “init()” method will be executed when the Container is performing Servlet Initialization.**

**Where the “service()” method will be executed when the container is performing the Request processing phase.**

**Where the “getServletConfig()” method will return a ServletConfig object.**

**Note: ServletConfig is an object created by the container as part of Servlet initialization and it is able to manage all the configuration details of a particular servlet which we provided in the web.xml file.**

**Where the “getServletInfo()” method is able to return some description about the Servlet.**

**Where the “destroy()” method will be executed when the container performs Servlet Deinstantion phase.**

**Implementation of service() method:**

—------------------------------------

**service() method is like the main() method in J2SE applications, it will have the application logic which we want to execute to process the request.**

**Container will execute the service() method automatically upon receiving requests from clients, because the container is aware about the service() method and its application logic in the body.**

**In the service() method implementation we have to use the following steps.**

1. **Set “ContentType” response header in order to give information to the client about the type of response that present Servlet is generating.**

**public void setContentType(String MIMEType)**

**Where MIME type may be “text/html”, “text/xml”, “img/jpg”, …….**

**Note: The default MIME type in web applications is “text/html”.**

**EX: response.setContentType(“text/html”);**

1. **Get PrintWriter object from response object inorder to send response to the Response object and to the Client.**

**public PrintWriter getWriter()**

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

1. **Send Response to the Response object:**

**public void println(String response)**

**EX: out.println(“<html>”);**

**out.println(“<body>”);**

**out.println(“Hello User!”);**

**out.println(“</body></html>”);**

**First servletprogram**

**------------------------------------**

EX:

web.xml

—-------

**<web-app>**

**<servlet>**

**<servlet-name>WelcomeServlet</servlet-name>**

**<servlet-class>WelcomeServlet</servlet-class>**

**</servlet>**

**<servlet-mapping>**

**<servlet-name>WelcomeServlet</servlet-name>**

**<url-pattern>/welcome</url-pattern>**

**</servlet-mapping>**

**</web-app>**

**WelcomeServlet.java**

**import javax.servlet.\*;**

**//import jakarta.servlet.\*;**

**import java.io.\*;**

**public class WelcomeServlet implements Servlet{**

**public void init(ServletConfig config)throws ServletException{**

**}**

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

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

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

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

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

**out.println("<h1 style='color: red' align='center'>");**

**out.println("Welcome To codegnan it solutions pvt ltd");**

**out.println("<hr></h1></body></html>");**

**}**

**public ServletConfig getServletConfig(){**

**return null;**

**}**

**public String getServletInfo(){**

**return "";**

**}**

**public void destroy(){**

**}**

**}**

**Application-2**

**--------------------------------------------**

**Using IDE:**

HelloServlet

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

package com.codegnan.web;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.GenericServlet;

import javax.servlet.ServletConfig;

import javax.servlet.ServletException;

import javax.servlet.ServletRequest;

import javax.servlet.ServletResponse;

**import** javax.servlet.annotation.WebServlet;

@WebServlet("/HelloServlet")

public class HelloServlet extends GenericServlet {

private static final long serialVersionUID = 1L;

public void init(ServletConfig config) throws ServletException {

}

public void destroy() {

}

public ServletConfig getServletConfig() {

return null;

}

public String getServletInfo() {

return null;

}

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

response.setContentType("text/html");

PrintWriter out=response.getWriter();

out.println("<html>");

out.println("<body>");

out.println("<h1 style='color:yellow' align='center'>");

out.println("Hello Jyoshna! Good Morning");

out.println("</h1>");

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

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

}

}

Drawback:

In this approach, we have to implement Servlet interface, where we must provide implementation for all five methods irrespective of the actual requirement of service() method, this approach will increase unnecessary methods implementation in servlet classes, it will increase burden to the developers.

To Overcome the above problem we have to use “GenericServlet” provided by Servlet API.

Servlets Design By Using GenericServlet:

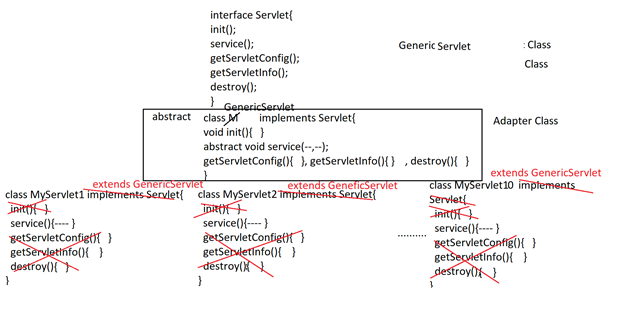
—-----------------------------------------

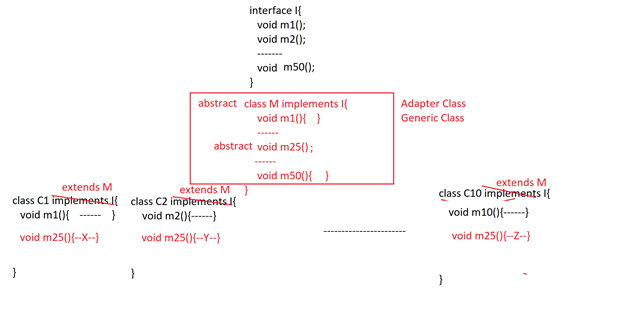
GenericServlet is provided by the Servlet API in the form of javax.servlet.GenericServlet or jakarta.servlet.GenericServlet

GenericServlet is an abstract class, it is an implementation abstract class to Servlet interface, it has provided all the servlet interface methods implementation except service() method.

In GenericServlet abstract class, still service() method is an abstract method that must be implemented by the developers as per their application requirement in the subclasses of GenericServlet abstract class.

GenericServlet is an idea that came from “Adapter Design Pattern”.

****

****

**GenericServlet example -1**

**------------------------------**

**package** com.codegnan.web;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** javax.servlet.GenericServlet;

**import** javax.servlet.Servlet;

**import** javax.servlet.ServletException;

**import** javax.servlet.ServletRequest;

**import** javax.servlet.ServletResponse;

**import** javax.servlet.annotation.WebServlet;

@WebServlet("/generic")

**public** **class** GenericServletDemo\_01 **extends** GenericServlet {

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

/\*\*

\* **@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();

out.println("<html>");

out.println("<body>");

out.println("<h1 style='color:red' align='center'>");

out.println("Welcome to GenericServlet");

out.println("</h1></body></html>");

}

}

**GenericServlet example -2**

**------------------------------**

**package** com.codegnan.web;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** java.time.LocalTime;

**import** javax.servlet.GenericServlet;

**import** javax.servlet.ServletException;

**import** javax.servlet.ServletRequest;

**import** javax.servlet.ServletResponse;

**import** javax.servlet.annotation.WebServlet;

@WebServlet("/gen")

**public** **class** GenericServletDemo\_02 **extends** GenericServlet {

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

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

response.setContentType("text/html");

PrintWriter out = response.getWriter();

LocalTime time = LocalTime.*now*();

**int** hour = time.getHour();

String wishMessage = "";

**if** (hour < 12) {

wishMessage = "Good Morning";

} **else** **if** (hour > 12 && hour < 17) {

wishMessage = "Good Afternoon...";

} **else** {

wishMessage = "Good Evening...";

}

out.println("<html>");

out.println("<body>");

out.println("<h1 style='color:yellow' align='center'>");

out.println("Hello Sunny" + wishMessage);

out.println("</h1></body></html>");

}

}

**Servlets Design Through HttpServlet:**

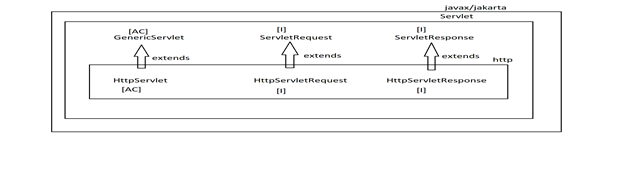
**—-------------------------------------**

**In Web application development, GenericServlet is able to process all the protocol requests as like normal requests, it is not providing specific implementation for any protocol conventions, this nature of the GenericServlet is not providing good compatibility between Protocols and Server side mechanisms, it will not provide effective implementation of the web applications.**

**To overcome the above problem, SUN Microsystems has provided a specific implementation for Http protocol at server side applications in the form of a separate predefined library in a separate package called “http” inside “servlet” package.**

**In Servlet API, “http” package is able to provide the following predefined library to support for Http protocol conventions.**

1. **HttpServlet**
2. **HttpServletRequest**
3. **HttpServletResponse**

****

**Why has Sun Microsystems provided Server side support for only Http protocol?**

**—-----------------------------------------------------------**

**Ans:**

**—----**

**SUN Microsystems has provided server side support for only http protocol , because**

1. **Http protocol is the frequently used protocol in web applications.**

**2. SUN Microsystems has given a model for implementing all the protocol conventions in the form of HttpServlet,...., with this model we can provide any type of protocol conventions at server side on our own.**

**Q)What HttpServlet and its related libraries are providing http protocol conventions over the GenericServlet and its conventions?**

**—-----------------------------------------------------------**

**Ans:**

**—-----**

**SUN Microsystems has provided HttpServlet and its related libraries to provide support for Http protocol in the following way.**

1. **HttpServlet is providing doGet(), doPost(), doHead(), ....doDelete() for each and every Http protocol requests like GET, POST, HEAD,....DELETE.**
2. **HttpServletResponse is providing support for all the Http protocol provided status codes in the form of the constants like SC\_OK, SC\_MOVED\_TEMPORARILY, SC\_MOVED\_PERMANENTLY,.....**

**—----------**

**—----------**

**—----------**

**jakarta.servlet.http.HttpServlet is an abstract class provided by SUN Microsystems as a sub abstract class to jakarta.servlet.GenericServlet.**

**SUN Microsystems has provided HttpServlet implementation like below.**

**public abstract class HttpServlet extends GenericServlet{**

**public void service(ServletRequest req,**

**ServletResponse res)throws ServletException,**

**IOException{**

**HttpServletRequest hreq = (HttpServletRequest) req;**

**HttpServletResponse hres = (HttpServletResponse) res;**

**service(hreq, hres);**

**}**

**Protected void service(HttpServletRequest hreq,**

**HttpServletResponse hres)throws ServletException,**

**IOException{**

**String method = hreq.getMethod();**

**if(method.equalsIgnoreCase(“GET”)){**

**doGet(hreq, hres);**

**}**

**if(method.equalsIgnoreCase(“POST”)){**

**doPost(hreq, hres);**

**}**

**—------**

**—------**

**—-------**

**if(method.equalsIgnoreCase(“DELETE”)){**

**doDelete(hreq, hres);**

**}**

**}**

**protected void doGet(HttpServletRequest hreq,**

**HttpServletResponse hres)throws ServletException,**

**IOException{**

**—---**

**}**

**protected void doPost(HttpServletRequest hreq,**

**HttpServletResponse hres)throws ServletException,**

**IOException{**

**—---**

**}**

**—-----**

**—------**

**—-------**

**protected void doDelete(HttpServletRequest hreq,**

**HttpServletResponse hres)throws ServletException,**

**IOException{**

**—---**

**}**

**}**

**To prepare HttpServlet in web applications, we have to declare an user defined class and it must be extended from jakarta.servlets.http.HttpServlet, where we must override doXxx() method on the basis of the request type.**

**public class WelcomeServlet extends HttpServlet{**

**protected void doXxx(HttpServletRequest hreq,**

**HttpServletResponse hres)throws ServletException,**

**IOException{**

**—--------**

**}**

**}**

**Http Methods:**

**The Http protocol is able to provide another advantage to the developers like to send different types of requests from client to Server due to the Http Methods.**

**Http Protocol has provided the following Http Methods for web applications.**

**Http 1.0 :**

1. **GET**
2. **POST**
3. **HEAD**

**Http 1.1 :**

1. **OPTIONS**
2. **TRACE**
3. **PUT**
4. **DELETE**

**Http protocol has provided some other reserved Http methods like CONNECT, TUNNEL,....**

**Q)What are the differences between GET and POST ?**

**Or**

**Q)What are the differences between doGet() and doPost()?**

**—-------------------------------------------------------**

**Ans:**

**—---**

1. **GET is a default request type in web applications, no need to specify GET to send GET requests in User Form.**

**POST is not a default request, to send POST requests from a client we must specify “POST” in the user form.**

1. **GET requests will not have body part in the Request Format.**

**POST requests will have body part in the Request Format.**

1. **GET requests are able to carry less data from Client to Server.**

**In GET Request, if we provide request parameters data then the request parameters data will be send to the Server through Request Formats Header part only due to the lack of Body part in GET request, here there is a memory limitation for Request Format header part, so it is able to carry only less data.**

**POST requests are able to carry more data from Client to server due to the availability of BODY part in POST request ,there is no memory limitation for POST request BODY part, it is able to carry any volume of data.**

1. **GET request is not providing security for the users data, because if we provide request parameters data in user form then GET request will provide that user provided data at client browsers address bar along with the URL as Query String, it will be open for every user.**

**POST request is able to provide security for the users data, because if we provide request parameters data along with POST request then POST request will not display that users parameters data at client browsers address bar as query string.**

1. **In case of GET request, Request parameters data will not be encrypted while transferring data from client to server in the network.**

**In case of POST request, Request parameters data will be encrypted while transferring data from Client to Server in the network.**

1. **GET request is mainly for Download operation, that is getting some resource from Server.**

**POST request is mainly for Upload operation, that is to post some resource from client to server.**

1. **GET request is supporting only character data.**

**POST Request is supporting Binary data.**

**8. GET requests are supporting bookmarks**

**POST requests are not supporting bookmarks**

**9. GET request data will be stored in Browsers history**

**POST request data will not be stored in browsers history**

**10. GET requests are cached requests.**

**POST requests are not cached requests**

**Demmy.html**

**----------**

**<!DOCTYPE html>**

**<html>**

**<head>**

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

**<title>Demmy</title>**

**</head>**

**<body>**

**<form action="demo" method="post">**

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

**</form>**

**</body>**

**</html>**

**HttpDemo\_01.java**

**-----------------**

**package com.codegnan.web;**

**import java.io.IOException;**

**import java.io.PrintWriter;**

**import javax.servlet.ServletException;**

**import javax.servlet.annotation.WebServlet;**

**import javax.servlet.http.HttpServlet;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

**@WebServlet("/demo")**

**public class HttpServletDemo\_01 extends HttpServlet {**

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

**protected void doGet(HttpServletRequest request, HttpServletResponse response)**

**throws ServletException, IOException {**

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

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

**out.println("inside of doGet()");**

**out.close();**

**}**

**protected void doPost(HttpServletRequest request, HttpServletResponse response)**

**throws ServletException, IOException {**

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

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

**out.println("inside of doPost()");**

**out.close();**

**}**

**}**

**we can run the above program on browser give the url on browser**

**http://localhost:8080/DynamicApp\_2/reg**

**we get doGet()**

**http://localhost:8080/DynamicApp\_2/Demmy.html**

**we get doPost() answer**

**Q)What is the difference between GET request and HEAD Request?**

**—----------------------------------------------------------**

**Ans:**

**—---**

**If we send a GET request for any resource that exists at server side application then Server will send both response data[Body] and response headers data[Header] in the form of Response Format to the client.**

**If we send a HEAD request for any resource that exists at Server side application then Server will send only the requested resources headers data to the client in the form of Response Format.**

**OPTIONS:**

**—--------**

**This request can be used to get the list of Http methods which are supported by the present server which we are using for web applications.**

**Q)What is the difference between POST request and PUT request?**

**—---------------------------------------------------------**

**Ans:**

**—----**

**POST request can be used to post a resource to the server machine , that is creating a resource at server side application.**

**PUT request can be used to update/override/replace the resource data which exists at server side application.**

**Note: As per the server security constraints , servers may not support PUT requests.**

**TRACE:**

**It can be used to get the Tracing path of the request from where the required response is generated.**

**Note: It can be used to get the working status of a particular server side resource, it is almost the same as Eco-Server.**

**DELETE:**

**It can be used to delete a particular resource from a server side application.**

**Note: As per the servers Security constraints , almost all the servers are not supporting this request type.**

**Note: All the above request types are supported or not supported completely depending on the servers, Some servers may support all the request types, some other servers may support some of the request types.**

**Note: Almost all the servers are not supporting DELETE request type, but almost all the servers are supporting GET and POST.**

**HttpServlet example:-2**

**============================**

**Dummy.html**

**==============  
<!DOCTYPE html>**

**<html>**

**<head>**

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

**<title>http</title>**

**</head>**

**<body>**

**<form action=*"reg"* method=*"post"*>**

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

**</form>**

**</body>**

**</html>**

**DateTimeServlet.java**

**=======================**

**package** com.codegnan.web;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** java.time.LocalDate;

**import** java.time.LocalTime;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

@WebServlet("/reg")

**public** **class** DateTimeServlet **extends** HttpServlet {

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

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

response.setContentType("text/html");

PrintWriter out = response.getWriter();

LocalDate localDate = LocalDate.*now*();

String date = localDate.getDayOfMonth() + "----------"

+ localDate.getMonthValue() + "--------"

+ localDate.getYear();

LocalTime localTime = LocalTime.*now*();

String time = localTime.getHour() + "----------"

+ localTime.getMinute() + "--------"

+ localTime.getSecond();

out.println("<html><body><h1 style='color:green' align='center'>");

out.println("Hello User!<br>");

out.println("Date : " + date);

out.println("<br>");

out.println("Time : " + time);

out.println("</h1></body></html>");

}

**protected** **void** doPost(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

response.setContentType("text/html");

PrintWriter out = response.getWriter();

LocalDate localDate = LocalDate.*now*();

String date = localDate.getDayOfMonth() + "----------"

+ localDate.getMonthValue() + "--------"

+ localDate.getYear();

out.println("<html><body><h1 style='color:green' align='center'>");

out.println("Hello User!<br>");

out.println("Date : " + date);

out.println("<br>");

out.println("</h1></body></html>");

}

}

**User Interface In Web Applications:**

**—------------------------------------**

**Q)What is the requirement of User Interface in web applications?**

**—--------------------------------------------------**

**Ans:**

**—---**

**User Interface In Web Applications:**

**—------------------------------------**

**Q)What is the requirement of User Interface in web applications?**

**—--------------------------------------------------**

**Ans:**

**—---**

1. **User Interface is a top layer in enterprise applications.**
2. **User Interface Layer will provide a starting point to the users in order to interact with the web applications.**
3. **User Interface is able to take data from the users and it is able to submit data to the server side application in order to perform the respective server side action.**

**User interface is able to provide a very good environment to submit different types of requests from client to Server side application like GET,POST, HEAD, OPTIONS, PUT, TRACE,......**

1. **User Interface is able to provide a very good environment to define data validation logics in the form of Java script functions in order to perform Client Side Data validations.**
2. **User interface is able to improve look and feel to the web applications.**

**To prepare User interface in web applications we have to use technologies like Html, CSS, Java script, Bootstrap,........**

**There are two types of User Interfaces or View parts.**

1. **Informational User Interface**
2. **Form based User Interface**

**Informational User Interface:**

**This User interface is not taking data from Users, it is able to get data from Server side application and it is able to display data to the users.**

**EX: Displaying status of the Server side actions like Login Success, Login failure, User Added Successfully,.....**

**Form based User Interface:**

**This User interface includes forms to collect data from the users and to submit that data to the Server side application.**

**In web applications, to prepare Forms we will use basic html tags.**

**Request Parameters:**

**—---------------------**

**Request Parameters are the Name-Value pairs which are submitted by the user to the server side applications through the user forms at client browser.**

**To get the value of a particular request parameter we have to use the following method.**

**public String getParameter(String name)**

**If any request parameter contains more than one value then to get multiple values on the basis of the name / key then we have to use the following method.**

**public String[] getParameterValues(String name)**

**To get all parameter names we will use the following method.**

**public Enumeration getParameterNames()**

**Request Attributes:**

**—--------------------**

**Request Attributes data is the collection of Key-Value pairs provided by the Servlets at runtime in order to share data to some other web components where the same request exists.**

**To provide a request attribute in the request object we will use the following method.**

**public void setAttribute(String key, Object value)**

**To get an attribute value on the basis of the key then we have to `use the following method.**

**public Object getAttribute(String key)**

**To remove an attribute from the Request object we will use the following method.**

**public void removeAttribute(String key)**

**To get all attributes names from the request object we will use the following method.**

**public Enumeration getAttributeNames()**

**EX:**

**—---**

**Loginform.html**

**HTML CODE**

**-------------**

**<!DOCTYPE html>**

**<html>**

**<head>**

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

**<title>Insert title here</title>**

**</head>**

**<body>**

**<form action=*"login"* method=*"post"*>**

**<h2 style="color:*red*" align=*"center"*>Codegnan It solutions</h2>**

**<h3 style="color:*aqua*" align=*"center"*>userLogin</h3>**

**<table align=*"center"*>**

**<tr>**

**<td>User Name</td>**

**<td><input type=*"text"* name=*"uname"*></td>**

**</tr>**

**<tr>**

**<td>User Password</td>**

**<td><input type=*"password"* name=*"upwd"*></td>**

**</tr>**

**<tr>**

**<td>User emial</td>**

**<td><input type=*"email"* name=*"uemail"*></td>**

**</tr>**

**<tr>**

**<td><input type=*"submit"* value=*"register"*></td>**

**</tr>**

**<tr>**

**<td><input type=*"reset"* value=*"cancel"*></td>**

**</tr>**

**</table>**

**</form>**

**</body>**

**</html>**

**LoginServlet**

**----------------**

**package com.codegnan.web;**

**import java.io.IOException;**

**import java.io.PrintWriter;**

**import javax.servlet.ServletException;**

**import javax.servlet.annotation.WebServlet;**

**import javax.servlet.http.HttpServlet;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

**@WebServlet("/login")**

**public class LoginServlet extends HttpServlet {**

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

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

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

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

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

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

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

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

**out.println("User name"+uname+"<br>");**

**out.println("user Password"+upwd+"<br>");**

**out.println("user Email"+uemail+"<br>");**

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

**out.close();**

**}**

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

**}**

**}**

**EX:**

**—---**

**Loginform.html**

**HTML CODE**

**-------------**

**<html>**

**<head>**

**<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">**

**<title>Login Form</title>**

**</head>**

**<body>**

**<form action="./login">**

**<h2 style="color: red;" align="center">codegnan it Solutions</h2>**

**<h3 style="color: blue" align="center">User Login Page</h3>**

**<table align="center">**

**<tr>**

**<td>User Name</td>**

**<td><input type="text" name="uname"></td>**

**</tr>**

**<tr>**

**<td>Password</td>**

**<td><input type="password" name="upwd"></td>**

**</tr>**

**<tr>**

**<td><input type="submit" value="Login"></td>**

**</tr>**

**</table>**

**</form>**

**</body>**

**</html>**

**LoginServlet**

**----------------**

**package com.codegnan.web;**

**import java.io.IOException;**

**import java.io.PrintWriter;**

**import javax.servlet.ServletException;**

**import javax.servlet.annotation.WebServlet;**

**import javax.servlet.http.HttpServlet;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

**@WebServlet("/login")**

**public class LoginServlet extends HttpServlet {**

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

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

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

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

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

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

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

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

**out.println("<h1 style='color: red;' align='center'>");**

**if(uname.equals("malli") && upwd.equals("malli")) {**

**out.println("Login Success");**

**}else {**

**out.println("Login Failure");**

**}**

**out.println("</h1>");**

**out.println("<h3 align='center'>");**

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

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

**}**

**}**

**registrationform.html**

**------------------------------**

**<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Strict//EN" "http://www.w3.org/TR/html4/strict.dtd">**

**<html>**

**<head>**

**<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">**

**<title>Registration Form</title>**

**</head>**

**<body>**

**<h2 style="color: red;" align="center">codegnan it solutions</h2>**

**<h3 style="color: blue;" align="center">Student Course Registration Form</h3>**

**<form action="./reg">**

**<table align="center">**

**<tr>**

**<td>Student Name</td>**

**<td><input type="text" name="sname"></td>**

**</tr>**

**<tr>**

**<td>Student Qualification</td>**

**<td>**

**<input type="checkbox" name="squal" value="BSC">BSC**

**<input type="checkbox" name="squal" value="BTech">BTech**

**<input type="checkbox" name="squal" value="MCA">MCA**

**<input type="checkbox" name="squal" value="MTech">MTech**

**<input type="checkbox" name="squal" value="PHD">PHD**

**</td>**

**</tr>**

**<tr>**

**<td>Student Gender</td>**

**<td>**

**<input type="radio" name="sgender" value="Male">Male**

**<input type="radio" name="sgender" value="Female">Female**

**</td>**

**</tr>**

**<tr>**

**<td>Student Courses</td>**

**<td><select name="scourses" multiple="multiple" size="5">**

**<optgroup label="JAVA">**

**<option value="Core Java"> Core Java</option>**

**<option value="Advanced Java"> Advanced Java</option>**

**<option value="Spring"> Spring</option>**

**<option value="WebServices"> Webservices</option>**

**</optgroup>**

**<optgroup label="Python">**

**<option value="Core Python">Core Python</option>**

**<option value="Advanced Python">Advanced Python</option>**

**<option value="DJango">DJango</option>**

**<option value="WebServices">Webservices</option>**

**</optgroup>**

**<option value="Oracle">Oracle</option>**

**<option value="AWS">AWS</option>**

**<option value="DevOps">DevOps</option>**

**</select>**

**</td>**

**</tr>**

**<tr>**

**<td>Branch Location</td>**

**<td>**

**<select name="branchLocation">**

**<option value="Hyderabad"> Hyderabad</option>**

**<option value="vijayawada"> Pune</option>**

**<option value="Banglore">Banglore</option>**

**</select>**

**</td>**

**</tr>**

**<tr>**

**<td>Student Address</td>**

**<td>**

**<textarea name="saddr" rows="5" cols="30"></textarea>**

**</td>**

**</tr>**

**<tr>**

**<td><input type="submit" value="Register"></td>**

**</tr>**

**</table>**

**</form>**

**</body>**

**</html>**

**RegistrationServlet**

**------------------------------**

**package com.codegnan.web;**

**import java.io.IOException;**

**import java.io.PrintWriter;**

**import java.util.List;**

**import java.util.stream.Collectors;**

**import java.util.stream.Stream;**

**import javax.servlet.ServletException;**

**import javax.servlet.annotation.WebServlet;**

**import javax.servlet.http.HttpServlet;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

**/\*\***

**\* Servlet implementation class RegistrationServlet**

**\*/**

**@WebServlet("/reg")**

**public class RegistrationServlet extends HttpServlet {**

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

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

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

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

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

**String[] qual = request.getParameterValues("squal");**

**List<String> squal = Stream.*of*(qual).collect(Collectors.*toList*());**

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

**List<String> scourses = Stream.*of*(request.getParameterValues("scourses")).collect(Collectors.*toList*());**

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

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

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

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

**out.println("<h2 style='color: red;' align='center'>");**

**out.println("codegnan it solutions");**

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

**out.println("<h3 style='color: blue;' align='center'>");**

**out.println("Course Registration Details");**

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

**out.println("<table border='1' align='center'>");**

**out.println("<tr><th>Student Name</th><td>"+sname+"</td></tr>");**

**out.println("<tr><th>Student Qualification</th><td>"+squal+"</td></tr>");**

**out.println("<tr><th>Student Gender</th><td>"+sgender+"</td></tr>");**

**out.println("<tr><th>Student Courses</th><td>"+scourses+"</td></tr>");**

**out.println("<tr><th>Branch Location</th><td>"+branchLocation+"</td></tr>");**

**out.println("<tr><th>Student Address</th><td>"+saddr+"</td></tr>");**

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

**out.println("<h3 align='center'><a href='./registrationform.html'>|Registartion form|</a>");**

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

**}**

**}**

**Servlet Communication:**

**—----------------------**

**In General, in web application development, it is not suggestible to distribute the complete application logic within a single web resource like a single servlet, it is suggestible to distribute application logic over multiple server side resources like multiple Servlets, or JSps,.....**

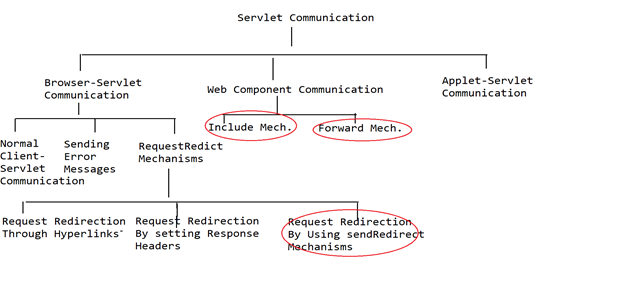
**In the above context, if we distribute application logic over multiple server side resources and if we want to process requests then we have to establish communication between server side resources, for this we have to use “Servlet Communication”.**

**To provide Servlet Communication we will use the following three approaches mainly.**

**1.** **Browser-Servlet Communication**

**2.** **Web COmponents Communication**

**3.** **Applet-Servlet Communication**

****

**Browser-Servlet Communication:**

**—--------------------------------**

**It is the communication between Client Browser and a servlet that exists at the Server side application.**

**There are three approaches to achieve Browser-Servlet communication.**

**1.** **Normal COmmunication between Browser and Servlet**

**2.** **Sending Error Messages**

**3.** **RequestRedirect Mechanisms**

**Normal COmmunication between Browser and Servlet:**

**—---------------------------------------------------**

**It is the communication between client browser and a servlet that exist at server side program, here if we submit request from client browser by providing url at client browser address bar, Servlet will take that request, Servlet will executes its lifecycle methods and servlet will generate some response to the client.**

**Sending Error Messages:**

**—------------------------**

**in general, when we access a particular servlet from a client browser, Server will process that request and Server will generate some response to the respective client, in this context, if any problem is identified in the request processing then Container or Server will generate some error response in a standard format.**

**In part of the web applications if we want to send error messages as per the application requirement then we have to use the following method.**

**public void sendError(int statusCode, String desc)**

**example program sending error messages**

**registrationform.html**

<!DOCTYPE html>

<html>

<head>

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

<title>Insert title here</title>

</head>

<body>

<font color=*'red'*>

<h2>codegnan it solutions</h2>

<h3>User Registration Form</h3>

</font>

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

<table>

<tr>

<td>User Name</td>

<td><input type=*"text"* name=*"uname"*/></td>

</tr>

<tr>

<td>User Age</td>

<td><input type=*"text"* name=*"uage"*/></td>

</tr>

<tr>

<td>User Email</td>

<td><input type=*"text"* name=*"uemail"*/></td>

</tr>

<tr>

<td>User Mobile</td>

<td><input type=*"text"* name=*"umobile"*/></td>

</tr>

<tr>

<td><input type=*"submit"* value=*"Registration"*/></td>

</tr>

</table>

</form>

</body>

</html>

RegistrationServlet

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

**package** com.codegnan.web;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** java.util.List;

**import** java.util.stream.Collectors;

**import** java.util.stream.Stream;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

/\*\*

\* Servlet implementation class RegistrationServlet

\*/

@WebServlet("/reg")

**public** **class** RegistrationServlet **extends** HttpServlet {

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

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

**try** {

response.setContentType("text/html");

PrintWriter out=response.getWriter();

String uname=request.getParameter("uname");

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

String uemail=request.getParameter("uemail");

String umobile=request.getParameter("umobile");

**if**(uage<18 || uage>30){

response.sendError(504, "User Age Is Not Sufficient for this Recruitment");

}**else**{

out.println("<html>");

out.println("<body>");

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

out.println("<h2>codegnan Services</h2>");

out.println("<h3>User Registration Details</h3>");

out.println("</font>");

out.println("<table border='1'>");

out.println("<tr><td>User Name</td><td>"+uname+"</td></tr>");

out.println("<tr><td>User Age</td><td>"+uage+"</td></tr>");

out.println("<tr><td>User Email</td><td>"+uemail+"</td></tr>");

out.println("<tr><td>User Mobile</td><td>"+umobile+"</td></tr>");

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

}

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

**Request Redirection**

The process of bypassing the request from one web application to another web application is called Request Redirection. In web applications, we are able to achieve Request Redirection in the following 3 ways:

1. **Request Redirection by using Hyperlinks.**
2. **Request Redirection by setting Response Headers.**
3. **Request Redirection by using Send Redirect Mechanism.**

**Request Redirection by using Hyperlinks:**

In this mechanism, when we send a request to the first application some resources will be executed and generated a hyperlink at the client browser as a response. By getting a hyperlink at the client browser we have to click on it and we have to send another request to the new web application. By executing some resources at the new web application, the required response will be generated to the client browser.

**Drawback:**

In this Request Redirection mechanism, the user may or may not click the generated hyperlink at the client browser after sending the first request. So that this mechanism won’t provide a guarantee to achieve Request Redirection.

**Request Redirection by setting Response Headers:**

In this mechanism, first, we will send a request to the first web application, where the first web application will set the Redirectional Status Code to Status Line field and new web application URI to Location Response Header.

When the Response Format reached the client, then the client will pick up Redirectional status code value from the Status-Line field, with this client browser will pick up Location Response Header value i.e. new web application URL then the client browser will send a new request to the new web application. By executing some resources at the new web application, the required response will be generated at the client machine.

To represent Request Redirection HttpServletResponse has introduced the following two constants:  
 **public static final int SC\_MOVED\_TEMPORARILY;** **public static final int SC\_MOVED\_PERMANENTLY;**

To set a particular status code value to Response Header we will use the following method.  
 **public void setStatus(int statuscode)**

To set a particular Response Header value in Response Format we have to use the following method.  
 **public void setHeader(String header\_name, String value)**

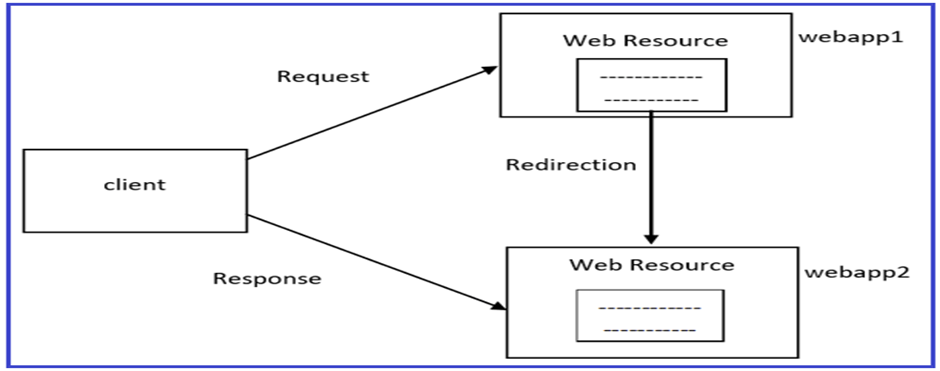
**Drawback:**

To perform Request Redirection, if we use this approach then every time, we have to set Redirectional status code and new web application URL to Location Response Header.

**Request Redirection by using Send Redirect Mechanisms**

To perform Request Redirection, if we use Send Redirect Mechanism no need to use Hyperlinks, not required to set status code and Response Header values to the Response Format, but we need to use the following method.

**public void sendRedirect(String url)**



**Program: Send Redirect App**

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

DemoServlet

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

**package** com.codegnan.web;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

/\*\*

\* Servlet implementation class DemoServlet

\*/

@WebServlet("/DemoServlet")

**public** **class** DemoServlet **extends** HttpServlet {

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

**public** **void** doGet(HttpServletRequest request,HttpServletResponse response)

**throws** ServletException,IOException

{

response.setContentType("text/html");

PrintWriter pw=response.getWriter();

response.sendRedirect("http://www.google.com");

pw.close();

}

}

Index.html

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

<!DOCTYPE html>

<html>

<head>

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

<title>Insert title here</title>

</head>

<body>

<form action=*"MySearcher"*>

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

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

</form>

</body>

</html>

MySearcher.java

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

package com.codegnan.web;

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

\* Servlet implementation class MySearcher

\*/

@WebServlet("/MySearcher")

public class MySearcher extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

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

\*/

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

String name=request.getParameter("name");

response.sendRedirect("https://www.google.co.in/#q="+name);

}

}

Web Component Communication

The process of providing communication between more than one web component available at the server machine is called Web Component Communication.

In general, web-component communication is available in between Servlet-Servlet, Servlet-Jsp, Servlet-HTML, Jsp-Jsp, Jsp-Servlet, Jsp-HTML, and so on. In web applications, we are able to achieve web-component communication in the following 2 ways:

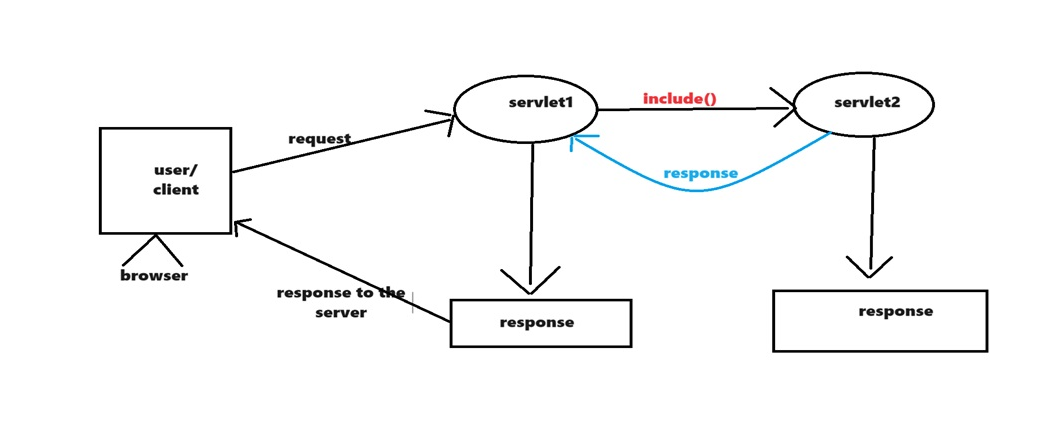
1. Include Mechanism
2. Forward Mechanism

Include Mechanism

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

Include(ServletRequest request,ServletResponse response)

Includes the content of a resource(servlet,jsp,html file) in the response. For example we have two pages servlet1 and servlet2.and we are sending request from Servlet1 toServlet2 by using include() the final response will be send to the client by using Servlet1 page only



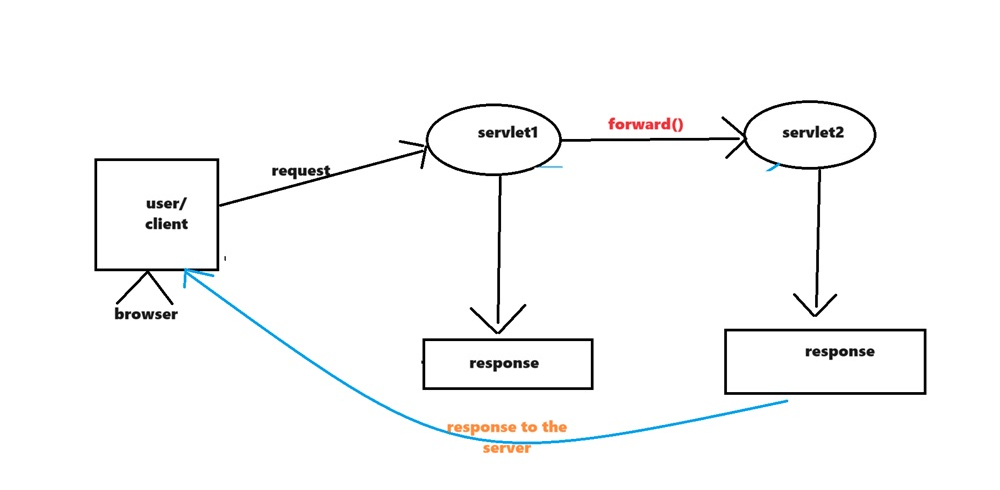
Forward Mechanism

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

Forwards(ServletRequest request, ServletResponse response)

Forwards a request from a servlet to another servlet(servlet,jsp or Html file)on the server. For example we have two pages servlet1 and servlet2. And we are sending request from servlet1 to servlet2 by using forward() the final response will send to the client by using servlet2 page.

If we want to perform the above mechanisms internally, we must use the RequestDispatcher object. So that both include and forward mechanisms are commonly called Request Dispatching Mechanisms.



To achieve web-component communication in web applications we have to use the following 2 steps:

1. Get the RequestDispatcher object.
2. Apply either include or forward mechanism by using the respective methods.

Step-1: Request Dispatcher object:

Request Dispatcher is an object, it will provide a very good environment either to include the target resource response into the present resource or to forward the request from the present resource to the target resource. To get the RequestDispatcher object we will use the following two ways:

ServletContext  
 getRequestDispatcher(\_)method  
 getNamedDispatcher(\_)method

ServletRequest  
 getRequestDispatcher(\_)method

Step-2: Apply either Include mechanism or Forward mechanism to achieve Web-Component Communication:

To represent Include and Forward mechanisms RequestDispatcher has provided the following methods:  
 public void include(ServletRequest req, servletResponse res) throws SE, IOE  
 public void forward(ServletRequest req, servletResponse res) throws SE, IOE

Forward mechanism example application

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

loginform.html

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

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Strict//EN" "http://www.w3.org/TR/html4/strict.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>Insert title here</title>

</head>

<body>

<h2 style="color: *red*;" align=*"center"*>codegnan it Solutions</h2>

<h3 style="color: *blue*;" align=*"center"*>User Login Form</h3>

<form method=*"post"* action=*"LoginServlet"*>

<table align=*"center"*>

<tr>

<td>User Name</td>

<td><input type=*"text"* name=*"uname"*></td>

</tr>

<tr>

<td>Password</td>

<td><input type=*"password"* name=*"upwd"*></td>

</tr>

<tr>

<td><input type=*"submit"* value=*"Login"*></td>

</tr>

</table>

</form>

</body>

</html>

success.html

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

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Strict//EN" "http://www.w3.org/TR/html4/strict.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>Insert title here</title>

</head>

<body>

<h2 style="color: *red*;" align=*"center"*>codegnan Solutions</h2>

<h3 style="color: *blue*;" align=*"center"*>User Login Status</h3>

<h1 style="color: *green*;" align=*"center"*>User Login Success</h1>

<h3 align=*"center"*>

<a href=*"./loginform.html"*>|Login Form|</a>

</h3>

</body>

</html>

failure.html

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

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Strict//EN" "http://www.w3.org/TR/html4/strict.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>Insert title here</title>

</head>

<body>

<h2 style="color: *red*;" align=*"center"*> codegnan Solutions</h2>

<h3 style="color: *blue*;" align=*"center"*>User Login Status</h3>

<h1 style="color: *green*;" align=*"center"*>User Login Failure</h1>

<h3 align=*"center"*>

<a href=*"./loginform.html"*>|Login Form|</a>

</h3>

</body>

</html>

LoginServlet.java

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

**package** com.codegnan.web;

**import** java.io.IOException;

**import** javax.servlet.RequestDispatcher;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

@WebServlet("/LoginServlet")

**public** **class** LoginServlet **extends** HttpServlet {

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

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

response.setContentType("text/html");

String uname = request.getParameter("uname");

String upwd = request.getParameter("upwd");

RequestDispatcher requestDispatcher = **null**;

**if**(uname.equals("malli") && upwd.equals("malli")) {

requestDispatcher = request.getRequestDispatcher("/success.html");

requestDispatcher.forward(request, response);

}**else** {

requestDispatcher = request.getRequestDispatcher("failure.html");

requestDispatcher.forward(request, response);

}

}

}

ServletConfig and ServletContext:

—----------------------------------

ServletConfig:

—-------------

1. ServletConfig is an object that is able to provide the complete view of a particular servlet in a web application.
2. ServletConfig is an object that is able to provide all the configuration details of a particular servlet which we provided in a web.xml file, where the servlet configuration details include the logical name of the servlet, initialization parameters of the servlet,......
3. In web applications, the ServletConfig object will be created by the container immediately after the servlet instantiation and just before accessing init() method in Servlet Initialization.
4. In web applications, the ServletConfig object will be destroyed by the container just before performing Servlet De-instantiation.
5. In web applications, ServletConfig object lifetime is almost all the same as the Servlet object lifetime.
6. The ServletConfig object is able to allow only parameters data , not able to allow attributes data.

1. If we provide data in the ServletConfig object then that data will be shared up to the respective Servlet only, not shared throughout the application.

1. The scope of the ServletConfig object is up to a particular Servlet.

To get the logical name of the Servlet from the ServletConfig object we have to use the following method.

public String getServletName()

In web applications, if we want to keep initialization parameters in ServletConfig objects, first we have to declare them in the web.xml file under the respective servlet configuration by using the following tags.

<web-app>

—----

<servlet>

—-----

<init-param>

<param-name>Name</param-name>

<param-value>value</param-value>

</init-param>

—------

</servlet>

—----

</web-app>

If we provide the initialization parameters in the web.xml file like above then the container will read all the initialization parameters from the web.xml file and the container will store them into the ServletConfig object at the time of creating ServletConfig object.

To get the value of a particular initialization parameter from the ServletConfig object we have to use the following method.

public String getInitParameter(String name)

To get all the names of the initialization parameters from the ServletConfig object we have to use the following method.

public Enumeration getInitParameterNames()

ServletConfig Example

Web.xml

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

<web-app id=*"WebApp\_ID"* version=*"2.4"*

xmlns=*"http://java.sun.com/xml/ns/j2ee"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app\_2\_4.xsd"*>

<display-name>webapp-11</display-name>

<welcome-file-list>

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

<welcome-file>index.htm</welcome-file>

<welcome-file>index.jsp</welcome-file>

<welcome-file>default.html</welcome-file>

<welcome-file>default.htm</welcome-file>

<welcome-file>default.jsp</welcome-file>

</welcome-file-list>

<servlet>

<servlet-name>MyServlet\_01</servlet-name>

<servlet-class>com.codegnan.web.MyServlet\_01</servlet-class>

<init-param>

<param-name>driver</param-name>

<param-value>com.mysql.cj.jdbc.Driver</param-value>

</init-param>

<init-param>

<param-name>url</param-name>

<param-value>jdbc:mysql://localhost:3306/adjava</param-value>

</init-param>

<init-param>

<param-name>userName</param-name>

<param-value>root</param-value>

</init-param>

<init-param>

<param-name>password</param-name>

<param-value>root</param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>MyServlet\_01</servlet-name>

<url-pattern>/config</url-pattern>

</servlet-mapping>

</web-app>

MyServlet\_01.java

========================

**package** com.codegnan.web;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** javax.servlet.ServletConfig;

**import** javax.servlet.ServletException;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**public** **class** MyServlet\_01 **extends** HttpServlet {

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

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

response.setContentType("text/html");

PrintWriter out = response.getWriter();

ServletConfig config = getServletConfig();

String logicalName = config.getServletName();

String driver = config.getInitParameter("driver");

String url = config.getInitParameter("url");

String userName = config.getInitParameter("userName");

String password = config.getInitParameter("password");

out.println("<html><body><h1>");

out.println("LogicalName: " + logicalName + "<br><br>");

out.println("driver : " + driver + "<br><br>");

out.println("URL :" + url + "<br><br>");

out.println("UserName : " + userName + "<br><br>");

out.println("Password : " + password + "<br><br>");

out.println("</h1></boody></html>");

}

}

# **SERVLETCONTEXT:**

|  |  |
| --- | --- |
|  |  |

### **Q: What are the differences between ServletConfig and ServletContext?**

1. ServletConfig is an object, it will manage all the configuration details of a particular servlet, where the configuration details include logical name of the servlet, initialization parameters and so on.

ServletContext is an object, it will manage all the context details of a particular web application, where the context details include logical name of web application and context parameters and so on.

2. ServletConfig is an object, it will provide the complete view of a particular servlet.

ServletContext is an object, it will provide the complete view of particular web application.

3. ServletConfig object will be prepared by the container immediately after servlet instantiation and just before calling init(\_) method in servlet initialization.

ServletContext object will be prepared by the container the moment when we start the server i.e. the time when we deploy the web application.

4. ServletConfig object will be destroyed by the container just before servlet deinstaniation. ServletContext object will be destroyed by the container when we shutdown the server i.e. the time when we undeploy the web application.

5. Due to the above reasons, the life of ServletConfig object is almost all the life of the respective servlet object.

The life of ServletContext object is almost all the life of the respective web application.

6. If we declare any data in ServletConfig object then that data will be shared upto respective servlet.

If we declare any data in ServletContext object then that data will be shared to all the no. of resources which are available in the present web application.

7. Due to the above reason, ServletConfig object will provide less sharability where as ServletContext object will provide more sharability.

8. In web applications, container will prepare ServletConfig object when it receives the request from client only except in load-on-startup case.

In web applications, container will prepare ServletContext object irrespective of getting request from client.

9. In web applications, ServletConfig object will allow only parameters data but ServletContext object will allow both parameters and attributes data.

10. Due to the above reason, ServletConfig object will allow only Static Inclusion of data where as ServletContext object will allow both Static Inclusion and Dynamic Inclusion of data.

To get the ServletContext object we have to use the following method from ServletConfig.

public ServletContext getServletContext();

EX: ServletContext context=config.getServletContext();

NOTE: In servlet3.0 version, it is possible to get ServletContext object even from ServletRequest.

EX: ServletContext context=req.getServletContext();

If we want to get the logical name of the web application from ServletContext object first of all we have to declare it in web.xml file.

To declare a logical name in web.xml file we have to use the following xml tag.

<web-app>

|  |  |
| --- | --- |
|  |  |

<display-name>logical\_name</display-name>

|  |  |
| --- | --- |
|  |  |

</web-app>

To get the logical name of web application from ServletContext object we will use the following method.

Public String getServletContextName()

EX: String lname=context.getServletContextName();

If we want to provide context parameters on ServletContext object first we have to declare them in web.xml file.

To declare a context parameter in web.xml file we have to use the following xml files.

<web-app>

|  |  |
| --- | --- |
|  |  |

<context-param>

<param-name>name</param-name>

<param-value>value</param-value>

</context -param>

|  |  |
| --- | --- |
|  |  |

</web-app>

When we start the server or at the time of application deployment the main job of the container is to recognize the web application and to prepare ServletContext object.

At the time of recognizing the application container will recognize web.xml file then perform web.xml file loading, parsing and reading the content.

While reading the content container will read all its context parameters and store them in ServletContext object at the time of creation.

To get the particular context parameter value from ServletContext object we will use the following method.

public String getInitParameter(String name)

EX: String value=context.getInitParameter(“name”);

To get all the context parameter names from ServletContext object we will use the following method.

public Enumeration getInitParameterNames()

EX: Enumeration e=context.getInitParameterNames();

In web application, ServletContext object is able to allow attributes.

To set an attribute on ServletContext object we will use the following method.

public void setAttribute(String name, Object value)

EX: context.setAttribute(“location”, “Hyd”);

To get an attribute from ServletContext object we will use the following method.

public Object getAttribute(String name)

EX: String location=(String)context.getAttribute(“location”);

To remove an attribute from ServletContext object we will use the following method.

public void removeAttribute(String name)

EX: context.removeAttribute(“location”);

To get all the names of attributes from ServletContext object we will use the following method.

public Enumeration getAttributeNames()

EX: Enumeration e=context.getAttributeNames();

### **Q: What is ForeignContext?**

ANS: ForeignContext is a ServletContext object of another web application being executed in the same server.

To get ForeignContext object we have to use the following method.

public ServletContext getContext(String path)

--------------Application by using ServletContext---------------

Web.xml file:-

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

<web-app id=*"WebApp\_ID"* version=*"2.4"*

xmlns=*"http://java.sun.com/xml/ns/j2ee"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app\_2\_4.xsd"*>

<display-name>webapp-12</display-name>

<context-param>

<param-name>a</param-name>

<param-value>AAA</param-value>

</context-param>

<context-param>

<param-name>b</param-name>

<param-value>BBB</param-value>

</context-param>

<servlet>

<description>

</description>

<display-name>FirstServlet</display-name>

<servlet-name>FirstServlet</servlet-name>

<servlet-class>com.codegnan.web.FirstServlet</servlet-class>

<init-param>

<param-name>c</param-name>

<param-value>CCC</param-value>

</init-param>

<init-param>

<param-name>d</param-name>

<param-value>DDD</param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>FirstServlet</servlet-name>

<url-pattern>/first</url-pattern>

</servlet-mapping>

<servlet>

<description>

</description>

<display-name>SecondServlet</display-name>

<servlet-name>SecondServlet</servlet-name>

<servlet-class>com.codegnan.web.SecondServlet</servlet-class>

<init-param>

<param-name>e</param-name>

<param-value>EEE</param-value>

</init-param>

<init-param>

<param-name>f</param-name>

<param-value>FFF</param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>SecondServlet</servlet-name>

<url-pattern>/second</url-pattern>

</servlet-mapping>

</web-app>

FirstServlet.java

—-------------------

**package** com.codegnan.web;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** javax.servlet.GenericServlet;

**import** javax.servlet.ServletConfig;

**import** javax.servlet.ServletContext;

**import** javax.servlet.ServletException;

**import** javax.servlet.ServletRequest;

**import** javax.servlet.ServletResponse;

**public** **class** FirstServlet **extends** GenericServlet {

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

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

response.setContentType("text/html");

PrintWriter out = response.getWriter();

ServletConfig config = getServletConfig();

ServletContext context = getServletContext();

out.println("<html><body><h1>");

out.println("a---->" + context.getInitParameter("a") + "<br>");

out.println("b---->" + context.getInitParameter("b") + "<br>");

out.println("c---->" + config.getInitParameter("c") + "<br>");

out.println("d---->" + config.getInitParameter("d") + "<br>");

out.println("e---->" + config.getInitParameter("e") + "<br>");

out.println("f---->" + config.getInitParameter("f") + "<br>");

out.println("</h1></body></html>");

}

}

SecondServlet.java

—-----------------------

**package** com.codegnan.web;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** javax.servlet.GenericServlet;

**import** javax.servlet.ServletConfig;

**import** javax.servlet.ServletContext;

**import** javax.servlet.ServletException;

**import** javax.servlet.ServletRequest;

**import** javax.servlet.ServletResponse;

**public** **class** SecondServlet **extends** GenericServlet {

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

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

response.setContentType("text/html");

PrintWriter out = response.getWriter();

ServletConfig config = getServletConfig();

ServletContext context = getServletContext();

out.println("<html><body><h1>");

out.println("a---->" + context.getInitParameter("a") + "<br>");

out.println("b---->" + context.getInitParameter("b") + "<br>");

out.println("c---->" + config.getInitParameter("c") + "<br>");

out.println("d---->" + config.getInitParameter("d") + "<br>");

out.println("e---->" + config.getInitParameter("e") + "<br>");

out.println("f---->" + config.getInitParameter("f") + "<br>");

out.println("</h1></body></html>");

}

}

Q)Consider the following application and its deployment descriptor?

[http://localhost:8080/webapp-12/](http://localhost:8080/webapp-12/second)first

# **a---->AAA**

# **b---->BBB**

# **c---->CCC**

# **d---->DDD**

# **e---->null**

# **f---->null**

<http://localhost:8080/webapp-12/second>

# **a---->AAA**

# **b---->BBB**

# **c---->null**

# **d---->null**

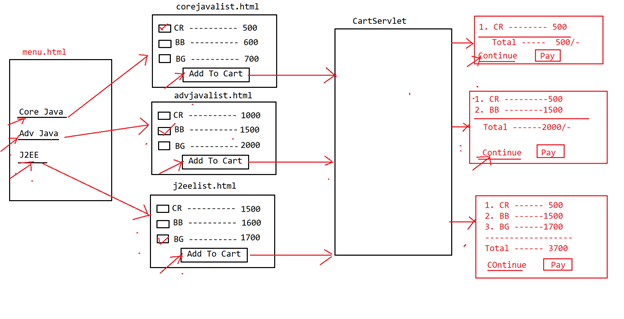
# **e---->EEE**

# **f---->FFF**

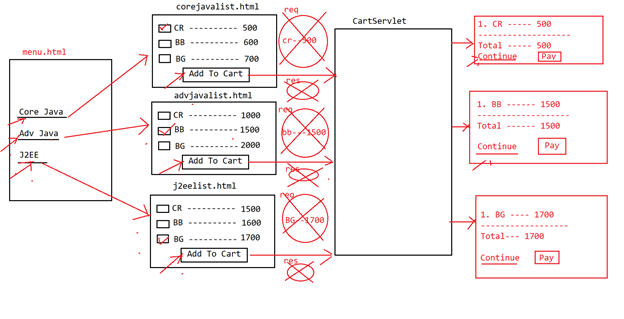
Session Tracking Mechanisms:

—-----------------------------

In web applications we will use http protocol, it is a stateless protocol, it is unable to manage clients previous requests data at the time of processing later requests, but as per the application requirements like online shopping, E-Banking,.... We have to manage clients' previous requests data at the time of processing later requests.

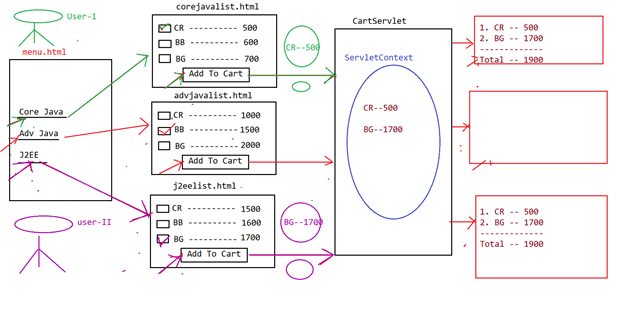
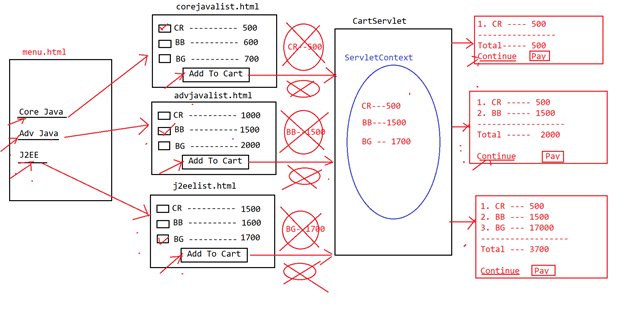


To manage the client’s previous requests data at the time of processing later requests if we use request object then we are unable to achieve the application requirement, because in web applications, container will create request object the moment when it receives request from the client for a particular servlet, Container will destroy request object the moment when response dispatched to the respective client, so request object is able to manage only current request data , it is unable to manage client’s previous request data, so request objects are not sufficient to achieve our application requirement like to manage client’s previous request data at the time of processing later requests.



In the above application, every time we are able to get only current request data that is only one time, not possible to get previously selected items through previous requests

To achieve the above requirement if we use ServletContext object then we are able to manage client’s previous requests data at the time of processing later requests, because ServletContext object is able to share its data through our the application irrespective of the number of requests which we submitted but it is unable to provide clear cut separation between the data which is selected by multiple users, it is able to provide all the users selected data to every user, it is not providing security for the users data.

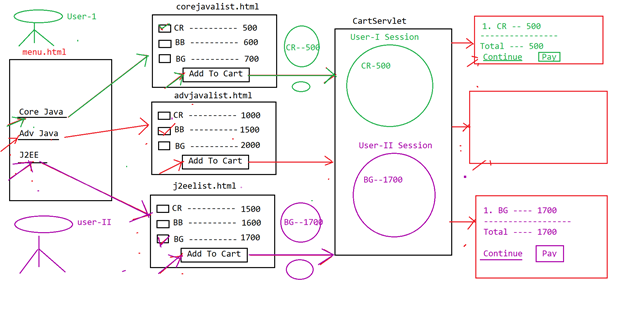


To achieve the application requirements like

1. To manage clients previous request data at the time of processing later requests.
2. To provide clear cut separate between multiple users data at server side and to provide security for the users data

We have to use “Session Tracking Mechanisms” over request objects and ServletContext objects.

In Session Tracking Mechanisms , Server side applications are able to manage a separate object for each and every user in order to store users specific data through multiple requests,where a user specific object which is created by the container is called “Session Object”.



|  |  |
| --- | --- |
|  |  |

Session is a time duration, it will start from the starting point of client conversation with server and will terminate at the ending point of client conversation with the server.

The data which we transferred from client to server through multiple number of requests during a particular session then that data is called State of the Session.

In general in web applications, container will prepare a request object similarly to represent a particular user we have to prepare a separate session.

If we allow multiple number of users on a single web application then we have to prepare multiple number of session objects.

In this context, to keep track of all the session objects at server machine we need a set of explicit mechanisms called as Session Tracking Mechanisms.

There are four session tracking mechanisms .

1. HttpSession Session Tracking Mechanism
2. Cookies Session Tracking Mechanism
3. URL-Rewriting Session Tracking Mechanism
4. Hidden Form Fields Session Tracking Mechanisms

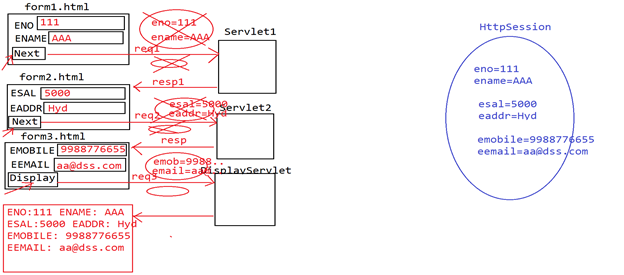
Note: In the above Session tracking mechanisms, first three mechanisms are provided by SUN Microsystems as part of the Servlet API,l but the last Session Tracking Mechanism [ Hidden Form Fields Session tracking Mechanisms] was not provided by SUN Microsystems, it is purely developers creation.

HttpSession Session Tracking Mechanism

—----------------------------------------

**In this mechanism, Server side application will perform the following actions at each and every request.**

1. **Web application[Servlet] will check whether any HttpSession object is created or not w.r.t present user user previously.**
2. **If no HttpSession object is created w.r.t the present user user previously then create new HttpSession object w.r.t the present User.**
3. **If any HttpSession object existed previously w.r.t the present user then get that existing HttpSession object.**
4. **Get request parameters from the request object.**
5. **Store all the request parameters in user specific HttpSession object.**
6. **Repeat the above steps for all the requests, at the end get all the data from user specific HttpSession object and utilize that data in our application as per the requirement.**

****

**To represent HttpSession objects, Servlet API has provided a predefined interface in the form of “/ javax.servlet.http.HttpSession”.**

**javax.servlet.http.HttpSession is an interface provided by SUN Microsystems as part of Servlet's abstraction and its implementation classes are provided by all the server vendors.**

**To create HttpSession objects we have to use the following methods from HttpServletRequest.**

1. **public HttpSession getSession()**
2. **public HttpSession getSession(boolean state)**

**HTTPSESSIONAPP:-**

**------------------------------**

**Form1.html**

**<!DOCTYPE html>**

**<html>**

**<head>**

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

**<title>Insert title here</title>**

**</head>**

**<body bgcolor=*"pink"*>**

**<center>**

**<form method=*"get"* action=*"first"*>**

**Name : <input type=*"text"* name=*"uname"* /><br> <br>**

**Age :<inputtype ="text" name=*"uage"* /><br> <br>**

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

**</form>**

**</center>**

**</body>**

**</html>**

**Form2.html:**

**-----------------------**

**<!DOCTYPE html>**

**<html>**

**<head>**

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

**<title>Insert title here</title>**

**</head>**

**<body bgcolor=*"pink"*>**

**<center>**

**<form method=*"get"* action=*"second"*>**

**Qualification : <input type=*"text"* name=*"uqual"* /><br>**

**<br> Designation : <input type=*"text"* name=*"udesig"* /><br>**

**<br> <input type=*"submit"* value=*"Next"* />**

**</form>**

**</center>**

**Form3.html**

**----------------------**

**<!DOCTYPE html>**

**<html>**

**<head>**

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

**<title>Insert title here</title>**

**</head>**

**<body bgcolor=*"pink"*>**

**<center>**

**<form method=*"get"* action=*"display"*>**

**Email : <input type=*"text"* name=*"email"* /><br> <br>**

**Mobile: <input type=*"text"* name=*"mobile"* /><br> <br>**

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

**</form>**

**</center>**

**</body>**

**</html>**

**FirstServlet.java**

**------------------------------**

**package com.codegnan.web;**

**import java.io.IOException;**

**import javax.servlet.RequestDispatcher;**

**import javax.servlet.ServletException;**

**import javax.servlet.annotation.WebServlet;**

**import javax.servlet.http.HttpServlet;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

**import javax.servlet.http.HttpSession;**

**@WebServlet("/first")**

**public class FirstServlet extends HttpServlet {**

**protected void doGet(HttpServletRequest request, HttpServletResponse response)**

**throws ServletException, IOException {**

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

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

**HttpSession hs = request.getSession();**

**hs.setAttribute("uname", uname);**

**hs.setAttribute("uage", uage);**

**RequestDispatcher rd = request.getRequestDispatcher("form2.html");**

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

**}**

**}**

**SecondServlet.java**

**—-----------------------**

**package com.codegnan.web;**

**import java.io.IOException;**

**import javax.servlet.RequestDispatcher;**

**import javax.servlet.ServletException;**

**import javax.servlet.annotation.WebServlet;**

**import javax.servlet.http.HttpServlet;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

**import javax.servlet.http.HttpSession;**

**@WebServlet("/second")**

**public class SecondServlet extends HttpServlet {**

**protected void doGet(HttpServletRequest request, HttpServletResponse response)**

**throws ServletException, IOException {**

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

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

**HttpSession hs = request.getSession();**

**hs.setAttribute("uqual", uqual);**

**hs.setAttribute("udesig", udesig);**

**RequestDispatcher rd = request.getRequestDispatcher("form3.html");**

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

**}**

**}**

**DisplayServlet.java**

**—-----------------------------**

**package com.codegnan.web;**

**import java.io.IOException;**

**import java.io.PrintWriter;**

**import javax.servlet.ServletException;**

**import javax.servlet.annotation.WebServlet;**

**import javax.servlet.http.HttpServlet;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

**import javax.servlet.http.HttpSession;**

**@WebServlet("/display")**

**public class DisplayServlet extends HttpServlet {**

**protected void doGet(HttpServletRequest request, HttpServletResponse response)**

**throws ServletException, IOException {**

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

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

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

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

**HttpSession hs = request.getSession();**

**String uname = (String) hs.getAttribute("uname");**

**String uage = (String) hs.getAttribute("uage");**

**String uqual = (String) hs.getAttribute("uqual");**

**String udesig = (String) hs.getAttribute("udesig");**

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

**out.println("<body bgcolor='lightgreen'>");**

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

**out.println("<table bgcolor='lightyellow'>");**

**out.println(**

**"<tr><td colspan='2'><center><b><font size='5' color='red'><u>Registration Details</u></font></b></center></td></tr>");**

**out.println("<tr><td>User Name</td><td>" + uname + "</td></tr>");**

**out.println("<tr><td>User Age</td><td>" + uage + "</td></tr>");**

**out.println("<tr><td>Qualification</td><td>" + uqual + "</td></tr>");**

**out.println("<tr><td>Designation</td><td>" + udesig + "</td></tr>");**

**out.println("<tr><td>Email</td><td>" + email + "</td></tr>");**

**out.println("<tr><td>Mobile</td><td>" + mobile + "</td></tr>");**

**out.println("<tr><td>Status</td><td>Success</td></tr>");**

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

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

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

**}**

**}**

**Cookies Session Tracking Mechanism:**

**—------------------------------------**

**Cookie is an object , it is able to manage a single key-value pair and it will be transferred from server to client along with response and from client to server along with request automatically and it will be managed at client browser.**

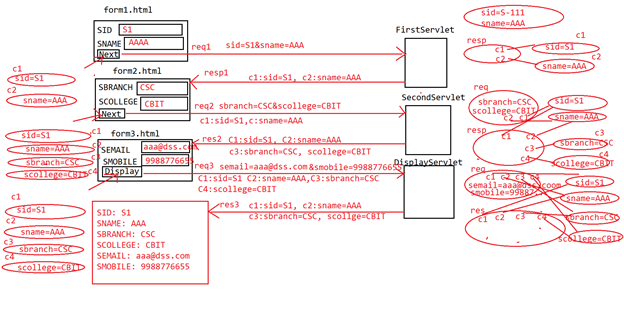
**In the Cookies Session tracking mechanism we have to perform the following activities at each and every request processing.**

1. **Get all the request parameters.**
2. **Create a separate cookie for each and every request parameter.**
3. **Add new cookies to the response object explicitly.**

**Note: The cookies which are coming along with requests from clients are added to the response object automatically, no need to add them explicitly.**

**Forward request to the next form.**

**At the end of the application, get all cookies from the request object and use the client's previous request data at the time of processing later requests.**

****

**In Web applications, to represent cookies , Servlet API has provided a predefined class in the form of javax.servlet.http.Cookie or jakarta.servlet.http.Cookie**

**To create a Cookie object we have to use the following constructor from Cookie class.**

**public Cookie(String name, String value)**

**EX: Cookie c1 = new Cookie(“sid”, “S1”);**

**Cookie c2 = new Cookie(“sname”, “AAA”);**

**To add a Cookie to the response object we have to use the following method.**

**public void addCookie(Cookie c)**

**EX: response.addCookie(c1);**

**response.addCookie(c2);**

**To get all Cookies from the request object we will use the following method from HttpServletRequest.**

**public Cookie[] getCookies()**

**To get Cookie name and cookie value from Cookie object we will use the following methods.**

**public String getName()**

**public String getValue()**

**In general, in web applications, we are able to provide some description to the cookies and we are able to get that description from the cookies by using the following methods.**

**public void setComment(String description)**

**public String getComment()**

**In general, in web applications, we are able to provide version numbers to the cookies and we are able to get the same version number from the cookies by using the following methods.**

**public void setVersion(int version)**

**public int getVersion()**

**IN general, in web applications, we are able to set age to the cookies and we are able to get the same age value from the cookies by using the following methods.**

**public void setMaxAge(int age)**

**public int getMaxAge()**

**In general, in web applications, we are able to set domain name to the cookies and we are able to get the same domain name from the cookie by using the following methods.**

**public void setDomain(String domainName)**

**public String getDomain()**

**In general, in web applications, we are able to set path values to the cookie in order to store them at client harddisk permanently and we are able to get the same path value from Cookie by using the following methods.**

**public void setPath(String path)**

**public String getPath()**

**cookies example application1:-**

**—------------------------------------------**

**CREATE TABLE PRODUCT (**

**pid VARCHAR(50) PRIMARY KEY,**

**pname VARCHAR(100),**

**pcost INT,**

**pquantity INT,**

**man\_Date DATE,**

**exp\_Date DATE**

**);**

**INSERT INTO PRODUCT (pid, pname, pcost, pquantity, man\_Date, exp\_Date) VALUES**

**('P001', 'Product A', 100, 50, '2024-01-01', '2025-01-01'),**

**('P002', 'Product B', 200, 30, '2024-02-15', '2025-02-15'),**

**('P003', 'Product C', 150, 20, '2024-03-10', '2025-03-10'),**

**('P004', 'Product D', 250, 10, '2024-04-20', '2025-04-20'),**

**('P005', 'Product E', 300, 15, '2024-05-25', '2025-05-25');**

**form1.html**

**---------------------**

**<!DOCTYPE html>**

**<html>**

**<head>**

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

**<title>Insert title here</title>**

**</head>**

**<body bgcolor=*"lightgreen"*>**

**<center>**

**<form action=*"./first"*>**

**<br>**

**<br>**

**<br>**

**<h1>Product Registration Form</h1>**

**<br>**

**<br>**

**<h2>**

**Product Id <input type=*"text"* name=*"pid"* /><br>**

**<br> Product Name <input type=*"text"* name=*"pname"* /><br>**

**<br> <input type=*"submit"* value=*"Next"* />**

**</h2>**

**</form>**

**</center>**

**</body>**

**</html>**

**form2.html**

**-------------------**

**<!DOCTYPE html>**

**<html>**

**<head>**

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

**<title>Insert title here</title>**

**</head>**

**<body bgcolor=*"lightgreen"*>**

**<center>**

**<form action=*"./second"*>**

**<br>**

**<br>**

**<br>**

**<h1>Product Registration Form(Cont..)</h1>**

**<br>**

**<br>**

**<h2>**

**Product Cost <input type=*"text"* name=*"pcost"* /><br>**

**<br> Product Quantity <input type=*"text"* name=*"pquantity"* /><br>**

**<br> <input type=*"submit"* value=*"Next"* />**

**</h2>**

**</form>**

**</center>**

**</body>**

**</html>**

**form3.html**

**--------------------------**

**<!DOCTYPE html>**

**<html>**

**<head>**

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

**<title>Insert title here</title>**

**</head>**

**<body bgcolor=*"lightgreen"*>**

**<center>**

**<form action=*"./reg"*>**

**<br>**

**<br>**

**<br>**

**<h1>Product Registration Form(Cont..)</h1>**

**<br>**

**<br>**

**<h2>**

**Manufactured Date <input type=*"text"* name=*"man\_Date"* /><br>**

**<br> Expiry Date <input type=*"text"* name=*"exp\_Date"* /><br>**

**<br> <input type=*"submit"* value=*"Registration"* />**

**</h2>**

**</form>**

**</center>**

**</body>**

**</html>**

**FirstServlet.java**

**-------------------------------**

**package com.codegnan.web;**

**import java.io.IOException;**

**import javax.servlet.RequestDispatcher;**

**import javax.servlet.ServletException;**

**import javax.servlet.annotation.WebServlet;**

**import javax.servlet.http.Cookie;**

**import javax.servlet.http.HttpServlet;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

**@WebServlet("/first")**

**public class FirstServlet extends HttpServlet {**

**protected void doGet(HttpServletRequest request, HttpServletResponse response)**

**throws ServletException, IOException {**

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

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

**Cookie c1 = new Cookie("pid", pid);**

**Cookie c2 = new Cookie("pname", pname);**

**response.addCookie(c1);**

**response.addCookie(c2);**

**RequestDispatcher rd = request.getRequestDispatcher("form2.html");**

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

**}**

**}**

**SecondServlet.java**

**---------------------------------**

**package com.codegnan.web;**

**import java.io.IOException;**

**import javax.servlet.RequestDispatcher;**

**import javax.servlet.ServletException;**

**import javax.servlet.annotation.WebServlet;**

**import javax.servlet.http.Cookie;**

**import javax.servlet.http.HttpServlet;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

**/\*\***

**\* Servlet implementation class SecondServlet**

**\*/**

**@WebServlet("/second")**

**public class SecondServlet extends HttpServlet {**

**protected void doGet(HttpServletRequest request, HttpServletResponse response)**

**throws ServletException, IOException {**

**try {**

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

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

**Cookie c3 = new Cookie("pcost", "" + pcost);**

**Cookie c4 = new Cookie("pquantity", "" + pquantity);**

**response.addCookie(c3);**

**response.addCookie(c4);**

**RequestDispatcher rd = request.getRequestDispatcher("form3.html");**

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

**} catch (Exception e) {**

**e.printStackTrace();**

**}**

**}**

**}**

**RegistrationServlet.java**

**--------------------------------------**

**package com.codegnan.web;**

**import java.io.IOException;**

**import java.io.PrintWriter;**

**import javax.servlet.ServletException;**

**import javax.servlet.annotation.WebServlet;**

**import javax.servlet.http.Cookie;**

**import javax.servlet.http.HttpServlet;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

**import com.codegnan.beans.ProductBean;**

**@WebServlet("/reg")**

**public class RegistrationServlet extends HttpServlet {**

**protected void doGet(HttpServletRequest request, HttpServletResponse response)**

**throws ServletException, IOException {**

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

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

**String man\_Date = request.getParameter("man\_Date");**

**String exp\_Date = request.getParameter("exp\_Date");**

**Cookie[] c = request.getCookies();**

**String pid = c[0].getValue();**

**String pname = c[1].getValue();**

**int pcost = Integer.parseInt(c[2].getValue());**

**int pquantity = Integer.parseInt(c[3].getValue());**

**ProductBean pb = new ProductBean();**

**String status = pb.register(pid, pname, pcost, pquantity, man\_Date, exp\_Date);**

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

**out.println("<body bgcolor='pink'>");**

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

**out.println("<u>Product Registration Details</u><br><br>");**

**out.println("Product Id. " + pid + "<br><br>");**

**out.println("Product Name. " + pname + "<br><br>");**

**out.println("Product Cost. " + pcost + "<br><br>");**

**out.println("Product Quantity " + pquantity + "<br><br>");**

**out.println("Product Manufactured Date. " + man\_Date + "<br><br>");**

**out.println("Product Expiry Date. " + exp\_Date + "<br><br>");**

**out.println("Status. " + status);**

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

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

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

**}**

**}**

**ProductBean.java**

**---------------------------------**

**package com.codegnan.beans;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.ResultSet;**

**import java.sql.SQLException;**

**import java.sql.Statement;**

**public class ProductBean {**

**Connection con;**

**Statement st;**

**ResultSet rs;**

**String status = "";**

**public ProductBean() {**

**try {**

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

**con = DriverManager.*getConnection*("jdbc:mysql://localhost/adjava", "root", "root");**

**st = con.createStatement();**

**} catch (Exception e) {**

**e.printStackTrace();**

**}**

**}**

**public String register(String pid, String pname, int pcost, int pquantity, String man\_Date, String exp\_Date) {**

**try {**

**rs = st.executeQuery("SELECT \* FROM product WHERE pid='" + pid + "'");**

**boolean b = rs.next();**

**if (b) {**

**status = "Product Already Exists";**

**} else {**

**int rowCount = st.executeUpdate("INSERT INTO product VALUES('" + pid + "','" + pname + "'," + pcost**

**+ "," + pquantity + ",'" + man\_Date + "','" + exp\_Date + "')");**

**if (rowCount == 1) {**

**status = "SUCCESS";**

**} else {**

**status = "FAILURE";**

**}**

**}**

**} catch (Exception e) {**

**status = "FAILURE";**

**e.printStackTrace();**

**} finally {**

**try {**

**st.close();**

**} catch (SQLException e) {**

**// TODO Auto-generated catch block**

**e.printStackTrace();**

**}**

**try {**

**rs.close();**

**} catch (SQLException e) {**

**// TODO Auto-generated catch block**

**e.printStackTrace();**

**}**

**try {**

**con.close();**

**} catch (SQLException e) {**

**// TODO Auto-generated catch block**

**e.printStackTrace();**

**}**

**}**

**return status;**

**}**

**}**

**URL-Rewriting Session Tracking mechanism:**

**—------------------------------------------**

**In Cookies Session Tracking mechanism , client’s Conversation is maintained at the same client browser only in the form of Cookies, here there is no security for the clients conversation in cookies session tracking mechanism, here to provide security for the users data we have to maintain users data at server side only , for this we have to use HttpSession Session Tracking mechanism.**

**In HttpSession Session Tracking Mechanism , the total session tracking mechanism is dependent on JSESSIONID cookie only, if the client browser disables cookies then there is no chance to perform Session Tracking mechanism.**

**In the above context , we need an alternative where we will use HttpSession objects to manage Users data at server machine and we want to execute session tracking mechanism even though the client browser disables cookies.**

**In the above context, the required alternative is “URL-Rewriting Session Tracking Mechanism”.**

**URL-Rewriting Session Tracking Mechanism is using HttpSession Session tracking Mechanism internally , but it is not depending on a cookie to manage SessionId value in order to identify user respective HttpSession object, but it is able to manage session Id value by rewriting url in the next form generation.**

**IN URL-Rewriting Session Tracking Mechanism, we have to perform the following actions at each and every Request Processing or Servlet.**

1. **Get Request Parameters**
2. **Get HttpSession Object[Either new or Existed]**
3. **Store Request parameters data in HttpSession object.**
4. **Prepare Dynamic form , where attach SessionId value to the URL in the form definition[in <form> tag].**

**In the dynamic form creation, to rewrite the URL with Session Id value we have to use the following method from HttpServletResponse.**

**public String encodeURL(String url)**

**This method will get SessionId value , it will append sessionId value to the url which we provided as parameter in the form of ‘JSESSIONID=value’.**

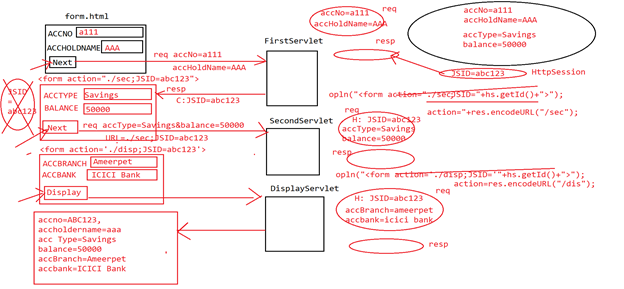
**EX:**

**out.println(“<form action=/sec;JSESSIONID=’”+hs.getId()+”’>”);**

**or**

**out.println(“<form action=’”+res.encodeURL(“/sec”)+”’>”);**

**Note: The main limitation of URL-Rewriting Session Tracking mechanism is to prepare dynamic forms, it is not suggestible in Servlet applications.**

****

**URL-ReWritting application:-**

**----------------------------------------**

**CREATE TABLE account (**

**acc\_no VARCHAR(20) PRIMARY KEY,**

**acc\_holder\_name VARCHAR(100),**

**acc\_type VARCHAR(50),**

**balance FLOAT,**

**acc\_branch VARCHAR(100),**

**acc\_bank VARCHAR(100)**

**);**

**INSERT INTO account (acc\_no, acc\_holder\_name, acc\_type, balance, acc\_branch, acc\_bank) VALUES**

**('A001', 'John Doe', 'Savings', 1500.75, 'Main Branch', 'Bank A'),**

**('A002', 'Jane Smith', 'Checking', 2500.00, 'West Branch', 'Bank B'),**

**('A003', 'Michael Brown', 'Savings', 3200.50, 'East Branch', 'Bank C'),**

**('A004', 'Emily Davis', 'Current', 4100.25, 'North Branch', 'Bank D'),**

**('A005', 'Daniel Wilson', 'Savings', 500.00, 'South Branch', 'Bank E');**

**Form1.html**

**—--------------------------**

**<!DOCTYPE html>**

**<html>**

**<head>**

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

**<title>Insert title here</title>**

**</head>**

**<body>**

**<h2 style="color: *red*;" align=*"center"*>Codegnan Banking Services</h2>**

**<h3 style="color: *blue*;" align=*"center"*>Account Creation Form</h3>**

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

**<table align=*"center"*>**

**<tr>**

**<td>Account Number</td>**

**<td><input type=*"text"* name=*"accNo"*></td>**

**</tr>**

**<tr>**

**<td>Account Holder Name</td>**

**<td><input type=*"text"* name=*"accHolderName"*></td>**

**</tr>**

**<tr>**

**<td><input type=*"submit"* value=*"Next"*></td>**

**</tr>**

**</table>**

**</form>**

**</body>**

**</html>**

**success.hml**

**---------------------**

**<!DOCTYPE html>**

**<html>**

**<head>**

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

**<title>Insert title here</title>**

**</head>**

**<body>**

**<h2 style="color: *red*;" align=*"center"*>codegnan Banking Services</h2>**

**<h3 style="color: *blue*;" align=*"center"*>Account Creation Status</h3>**

**<br>**

**<br>**

**<br>**

**<h1 style="color: *green*;" align=*"center"*>Account Created**

**Successfully</h1>**

**</body>**

**</html>**

**failure.html**

**---------------**

**<!DOCTYPE html>**

**<html>**

**<head>**

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

**<title>Insert title here</title>**

**</head>**

**<body>**

**<h2 style="color: *red*;" align=*"center"*>codegnan Banking Services</h2>**

**<h3 style="color: *blue*;" align=*"center"*>Account Creation Status</h3>**

**<br>**

**<br>**

**<br>**

**<h1 style="color: *green*;" align=*"center"*>Account Creation Failure**

**</h1>**

**</body>**

**</html>**

**FirstServlet**

**-----------------------**

**package com.codegnan.web;**

**import java.io.IOException;**

**import java.io.PrintWriter;**

**import javax.servlet.ServletException;**

**import javax.servlet.annotation.WebServlet;**

**import javax.servlet.http.HttpServlet;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

**import javax.servlet.http.HttpSession;**

**@WebServlet("/first")**

**public class FirstServlet extends HttpServlet {**

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

**protected void doPost(HttpServletRequest request, HttpServletResponse response)**

**throws ServletException, IOException {**

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

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

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

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

**HttpSession httpSession = request.getSession();**

**httpSession.setAttribute("accNo", accNo);**

**httpSession.setAttribute("accHolderName", accHolderName);**

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

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

**out.println("<h2 style='color: red;' align='center'>Codegnan Banking Services</h2>");**

**out.println("<h3 style='color: blue;' align='center'>Account Creation Form[Cont...]</h3>");**

**out.println("<form method='POST' action='" + response.encodeURL("./sec") + "'>");**

**out.println("<table align='center'>");**

**out.println("<tr><td>Account Type</td><td><input type='text' name='accType'></td></tr>");**

**out.println("<tr><td>Account Balance</td><td><input type='text' name='balance'></td></tr>");**

**out.println("<tr><td><input type='submit' value='Next'></td></tr>");**

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

**}**

**}**

**SecondServlet**

**-----------------------**

**package com.codegnan.web;**

**import java.io.IOException;**

**import java.io.PrintWriter;**

**import javax.servlet.ServletException;**

**import javax.servlet.annotation.WebServlet;**

**import javax.servlet.http.HttpServlet;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

**import javax.servlet.http.HttpSession;**

**/\*\***

**\* Servlet implementation class SecondServlet**

**\*/**

**@WebServlet("/sec")**

**public class SecondServlet extends HttpServlet {**

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

**protected void doPost(HttpServletRequest request, HttpServletResponse response)**

**throws ServletException, IOException {**

**try {**

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

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

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

**float balance = Float.*parseFloat*(request.getParameter("balance"));**

**HttpSession httpSession = request.getSession(false);**

**httpSession.setAttribute("accType", accType);**

**httpSession.setAttribute("balance", balance);**

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

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

**out.println("<h2 style='color: red;' align='center'>codegnan Banking Services</h2>");**

**out.println("<h3 style='color: blue;' align='center'>Account Creation Form[Cont...]</h3>");**

**out.println("<form method='POST' action='" + response.encodeURL("./disp") + "'>");**

**out.println("<table align='center'>");**

**out.println("<tr><td>Account Branch</td><td><input type='text' name='accBranch'></td></tr>");**

**out.println("<tr><td>Account Bank</td><td><input type='text' name='accBank'></td></tr>");**

**out.println("<tr><td><input type='submit' value='Display'></td></tr>");**

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

**} catch (Exception e) {**

**e.printStackTrace();**

**}**

**}**

**}**

**DisplayServlet**

**---------------------------**

**package com.codegnan.web;**

**import java.io.IOException;**

**import java.io.PrintWriter;**

**import javax.servlet.ServletException;**

**import javax.servlet.annotation.WebServlet;**

**import javax.servlet.http.HttpServlet;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

**import javax.servlet.http.HttpSession;**

**/\*\***

**\* Servlet implementation class DisplayServlet**

**\*/**

**@WebServlet("/disp")**

**public class DisplayServlet extends HttpServlet {**

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

**protected void doPost(HttpServletRequest request, HttpServletResponse response)**

**throws ServletException, IOException {**

**try {**

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

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

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

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

**HttpSession httpSession = request.getSession(false);**

**httpSession.setAttribute("accBranch", accBranch);**

**httpSession.setAttribute("accBank", accBank);**

**String accNo = (String) httpSession.getAttribute("accNo");**

**String accHolderName = (String) httpSession.getAttribute("accHolderName");**

**String accType = (String) httpSession.getAttribute("accType");**

**float balance = (float) httpSession.getAttribute("balance");**

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

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

**out.println("<h2 style='color: red;' align='center'>codegnan Banking Services</h2>");**

**out.println("<h3 style='color: blue;' align='center'>Account Details</h3>");**

**out.println("<table align='center' border='1'>");**

**out.println("<tr><td>Account Number</td><td>" + accNo + "</td></tr>");**

**out.println("<tr><td>Account Holder Name</td><td>" + accHolderName + "</td></tr>");**

**out.println("<tr><td>Account Type</td><td>" + accType + "</td></tr>");**

**out.println("<tr><td>Account Balance</td><td>" + balance + "</td></tr>");**

**out.println("<tr><td>Account Branch</td><td>" + accBranch + "</td></tr>");**

**out.println("<tr><td>Account Bank</td><td>" + accBank + "</td></tr>");**

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

**out.println("<form method='POST' action='" + response.encodeURL("./addAndEdit") + "'>");**

**out.println("<table align='center'>");**

**out.println("<tr><td><input type='submit' value='Confirm'></td></tr>");**

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

**} catch (Exception e) {**

**e.printStackTrace();**

**}**

**}**

**}**

**AddAndEditServlet**

**---------------------------**

**package com.codegnan.web;**

**import java.io.IOException;**

**import java.io.PrintWriter;**

**import javax.servlet.RequestDispatcher;**

**import javax.servlet.ServletException;**

**import javax.servlet.annotation.WebServlet;**

**import javax.servlet.http.HttpServlet;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

**import javax.servlet.http.HttpSession;**

**import com.codegnan.action.AccountAction;**

**import com.codegnan.beans.Account;**

**import com.codegnan.factories.AccountActionFactory;**

**/\*\***

**\* Servlet implementation class AddAndEditServlet**

**\*/**

**@WebServlet("/addAndEdit")**

**public class AddAndEditServlet extends HttpServlet {**

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

**protected void doPost(HttpServletRequest request, HttpServletResponse response)**

**throws ServletException, IOException {**

**try {**

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

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

**HttpSession httpSession = request.getSession(false);**

**AccountAction accountAction = AccountActionFactory.getAccountAction();**

**Account account = new Account();**

**account.setAccNo((String) httpSession.getAttribute("accNo"));**

**account.setAccHolderName((String) httpSession.getAttribute("accHolderName"));**

**account.setAccType((String) httpSession.getAttribute("accType"));**

**account.setBalance((float) httpSession.getAttribute("balance"));**

**account.setAccBranch((String) httpSession.getAttribute("accBranch"));**

**account.setAccBank((String) httpSession.getAttribute("accBank"));**

**String status = accountAction.addAccount(account);**

**RequestDispatcher requestDispatcher = null;**

**if (status.equalsIgnoreCase("success")) {**

**requestDispatcher = request.getRequestDispatcher("/success.html");**

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

**} else {**

**requestDispatcher = request.getRequestDispatcher("/failure.html");**

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

**}**

**} catch (Exception e) {**

**e.printStackTrace();**

**}**

**}**

**}**

**AccountAction**

**-----------------------**

**package com.codegnan.action;**

**import java.sql.Connection;**

**import java.sql.PreparedStatement;**

**import com.codegnan.beans.Account;**

**import com.codegnan.factories.ConnectionFactory;**

**public class AccountAction {**

**public String addAccount(Account account) {**

**String status = "";**

**try {**

**Connection connection = ConnectionFactory.*getConnection*();**

**PreparedStatement preparedStatement = connection**

**.prepareStatement("insert into account values(?,?,?,?,?,?)");**

**preparedStatement.setString(1, account.getAccNo());**

**preparedStatement.setString(2, account.getAccHolderName());**

**preparedStatement.setString(3, account.getAccType());**

**preparedStatement.setFloat(4, account.getBalance());**

**preparedStatement.setString(5, account.getAccBranch());**

**preparedStatement.setString(6, account.getAccBank());**

**int rowCount = preparedStatement.executeUpdate();**

**if (rowCount == 1) {**

**status = "success";**

**} else {**

**status = "failure";**

**}**

**} catch (Exception e) {**

**status = "failure";**

**e.printStackTrace();**

**}**

**return status;**

**}**

**}**

**Account**

**-------------------**

**package com.codegnan.beans;**

**public class Account {**

**private String accNo;**

**private String accHolderName;**

**private String accType;**

**private float balance;**

**private String accBranch;**

**private String accBank;**

**public String getAccNo() {**

**return accNo;**

**}**

**public void setAccNo(String accNo) {**

**this.accNo = accNo;**

**}**

**public String getAccHolderName() {**

**return accHolderName;**

**}**

**public void setAccHolderName(String accHolderName) {**

**this.accHolderName = accHolderName;**

**}**

**public String getAccType() {**

**return accType;**

**}**

**public void setAccType(String accType) {**

**this.accType = accType;**

**}**

**public float getBalance() {**

**return balance;**

**}**

**public void setBalance(float balance) {**

**this.balance = balance;**

**}**

**public String getAccBranch() {**

**return accBranch;**

**}**

**public void setAccBranch(String accBranch) {**

**this.accBranch = accBranch;**

**}**

**public String getAccBank() {**

**return accBank;**

**}**

**public void setAccBank(String accBank) {**

**this.accBank = accBank;**

**}**

**}**

**ConnectionFactory**

**-----------------------------**

**package com.codegnan.factories;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**public class ConnectionFactory {**

**private static Connection *connection* = null;**

**static {**

**try {**

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

***connection* = DriverManager.*getConnection*("jdbc:mysql://localhost:3306/adjava", "root", "root");**

**} catch (Exception e) {**

**e.printStackTrace();**

**}**

**}**

**public static Connection getConnection() {**

**return *connection*;**

**}**

**}**

**AccountActionFactory**

**------------------------------------**

**package com.codegnan.factories;**

**import com.codegnan.action.AccountAction;**

**public class AccountActionFactory {**

**private static AccountAction *accountAction* = null;**

**static {**

***accountAction* = new AccountAction();**

**}**

**public static AccountAction getAccountAction() {**

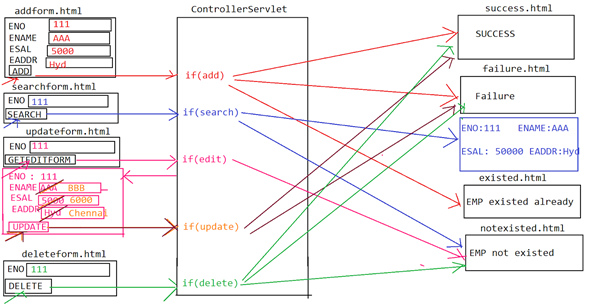
**return *accountAction*;**

**}**

**}`**

**Servlet Project**

**----------------------------------------------**

****

**Create database table in the above project requirement**

**------------------------------------------------------------------------------**

**CREATE TABLE emp1 (**

**eno INT PRIMARY KEY,**

**ename VARCHAR(255),**

**esal FLOAT,**

**eaddr VARCHAR(255)**

**);**

**Insert some sample records in the above table emp1**

**INSERT INTO emp1 (eno, ename, esal, eaddr)**

**VALUES (1, 'John Doe', 50000.00, '123 Main St, Anytown, USA');**

**INSERT INTO emp1 (eno, ename, esal, eaddr)**

**VALUES (2, 'Jane Smith', 60000.00, '456 Elm St, Othertown, USA');**

**INSERT INTO emp1 (eno, ename, esal, eaddr)**

**VALUES (3, 'Alice Johnson', 55000.00, '789 Oak St, Sometown, USA');**

**INSERT INTO emp1 (eno, ename, esal, eaddr)**

**VALUES (4, 'Bob Brown', 70000.00, '101 Maple St, Newcity, USA');**

**select\*from emp1;**

**header.html**

**---------------------------**

**<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Strict//EN" "http://www.w3.org/TR/html4/strict.dtd">**

**<html>**

**<head>**

**<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>**

**<title>Insert title here</title>**

**</head>**

**<body bgcolor=*"DF2E38"*>**

**<h1 style="color: *white*;" align=*"center"*>Codegnan it SOLUTIONS</h1>**

**</body>**

**</html>**

**welcome.html**

**------------------------**

**<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Strict//EN" "http://www.w3.org/TR/html4/strict.dtd">**

**<html>**

**<head>**

**<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>**

**<title>Insert title here</title>**

**</head>**

**<body bgcolor=*"DDF7E3"*>**

**<br>**

**<br>**

**<br>**

**<h1 style="color: *red*;" align=*"center"*>**

**<marquee>Welcome To codegnan Solutions</marquee>**

**</h1>**

**</body>**

**</html>**

**menu.html**

**------------------**

**<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Strict//EN" "http://www.w3.org/TR/html4/strict.dtd">**

**<html>**

**<head>**

**<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>**

**<title>Insert title here</title>**

**</head>**

**<body bgcolor=*"FFA3FD"*>**

**<h3 align=*"left"*>**

**<a href=*"./addform.html"* target=*"body"*>ADD</a><br>**

**<br> <a href=*"./searchform.html"* target=*"body"*>SEARCH</a><br>**

**<br> <a href=*"./updateform.html"* target=*"body"*>UPDATE</a><br>**

**<br> <a href=*"./deleteform.html"* target=*"body"*>DELETE</a>**

**</h3>**

**</body>**

**</html>**

**footer.html**

**--------------------**

**<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Strict//EN" "http://www.w3.org/TR/html4/strict.dtd">**

**<html>**

**<head>**

**<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>**

**<title>Insert title here</title>**

**</head>**

**<body bgcolor=*"658864"*>**

**<h3 style="color: *white*;" align=*"center"*>© Copyright 2018-2024 |**

**Codegnan IT Solutions PVT LTD. Hyderabad Kothwal Madhava Reddy Plaza,**

**Beside Indian Oil Petrol Bunk, JNTUH Metro Station, Nizampet X Roads,**

**Hyderabad, 500072.</h3>**

**</body>**

**</html>**

**home.html**

**---------------------**

**<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">**

**<html>**

**<head>**

**<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>**

**<title>Insert title here</title>**

**</head>**

**<frameset rows=*"20%, 65%, 15%"*>**

**<frame src=*"header.html"*>**

**<frameset cols=*"20%, 80%"*>**

**<frame src=*"menu.html"*>**

**<frame src=*"welcome.html"* name=*"body"*>**

**</frameset>**

**<frame src=*"footer.html"*>**

**</frameset>**

**</html>**

**addform.html**

**------------------------**

**<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Strict//EN" "http://www.w3.org/TR/html4/strict.dtd">**

**<html>**

**<head>**

**<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>**

**<title>Insert title here</title>**

**</head>**

**<body bgcolor=*"DDF7E3"*>**

**<br>**

**<br>**

**<br>**

**<form method=*"POST"* action=*"add.do"*>**

**<table align=*"center"*>**

**<tr>**

**<td>Employee Number</td>**

**<td><input type=*"text"* name=*"eno"*></td>**

**</tr>**

**<tr>**

**<td>Employee Name</td>**

**<td><input type=*"text"* name=*"ename"*></td>**

**</tr>**

**<tr>**

**<td>Employee Salary</td>**

**<td><input type=*"text"* name=*"esal"*></td>**

**</tr>**

**<tr>**

**<td>Employee Address</td>**

**<td><input type=*"text"* name=*"eaddr"*></td>**

**</tr>**

**<tr>**

**<td><input type=*"submit"* value=*"ADD"*></td>**

**</tr>**

**</table>**

**</form>**

**</body>**

**</html>**

**searchform.html**

**---------------------------**

**<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Strict//EN" "http://www.w3.org/TR/html4/strict.dtd">**

**<html>**

**<head>**

**<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>**

**<title>Insert title here</title>**

**</head>**

**<body bgcolor=*"DDF7E3"*>**

**<br>**

**<br>**

**<br>**

**<form method=*"POST"* action=*"search.do"*>**

**<table align=*"center"*>**

**<tr>**

**<td>Employee Number</td>**

**<td><input type=*"text"* name=*"eno"*></td>**

**</tr>**

**<tr>**

**<td><input type=*"submit"* value=*"SEARCH"*></td>**

**</tr>**

**</table>**

**</form>**

**</body>**

**</html>**

**updateform.html**

**--------------------------**

**<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Strict//EN" "http://www.w3.org/TR/html4/strict.dtd">**

**<html>**

**<head>**

**<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>**

**<title>Insert title here</title>**

**</head>**

**<body bgcolor=*"DDF7E3"*>**

**<br>**

**<br>**

**<br>**

**<form method=*"POST"* action=*"editform.do"*>**

**<table align=*"center"*>**

**<tr>**

**<td>Employee Number</td>**

**<td><input type=*"text"* name=*"eno"*></td>**

**</tr>**

**<tr>**

**<td><input type=*"submit"* value=*"GET EDIT FORM"*></td>**

**</tr>**

**</table>**

**</form>**

**</body>**

**</html>**

**deleteform.html**

**--------------------------**

**<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Strict//EN" "http://www.w3.org/TR/html4/strict.dtd">**

**<html>**

**<head>**

**<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>**

**<title>Insert title here</title>**

**</head>**

**<body bgcolor=*"DDF7E3"*>**

**<br>**

**<br>**

**<br>**

**<form method=*"POST"* action=*"delete.do"*>**

**<table align=*"center"*>**

**<tr>**

**<td>Employee Number</td>**

**<td><input type=*"text"* name=*"eno"*></td>**

**</tr>**

**<tr>**

**<td><input type=*"submit"* value=*"DELETE"*></td>**

**</tr>**

**</table>**

**</form>**

**</body>**

**</html>**

**existed.html**

**----------------**

**<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Strict//EN" "http://www.w3.org/TR/html4/strict.dtd">**

**<html>**

**<head>**

**<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>**

**<title>Insert title here</title>**

**</head>**

**<body bgcolor=*"DDF7E3"*>**

**<br>**

**<br>**

**<br>**

**<br>**

**<h1 style="color: *red*;" align=*"center"*>Employee Existed Already</h1>**

**</body>**

**</html>**

**notexisted.html**

**----------------**

**<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Strict//EN" "http://www.w3.org/TR/html4/strict.dtd">**

**<html>**

**<head>**

**<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>**

**<title>Insert title here</title>**

**</head>**

**<body bgcolor=*"DDF7E3"*>**

**<br>**

**<br>**

**<br>**

**<br>**

**<h1 style="color: *red*;" align=*"center"*>Employee Not Existed</h1>**

**</body>**

**</html>**

**success.html**

**-------------------**

**<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Strict//EN" "http://www.w3.org/TR/html4/strict.dtd">**

**<html>**

**<head>**

**<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>**

**<title>Insert title here</title>**

**</head>**

**<body bgcolor=*"DDF7E3"*>**

**<br>**

**<br>**

**<br>**

**<br>**

**<h1 style="color: *red*;" align=*"center"*>Success</h1>**

**</body>**

**</html>**

**failure.html**

**------------------------**

**<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Strict//EN" "http://www.w3.org/TR/html4/strict.dtd">**

**<html>**

**<head>**

**<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>**

**<title>Insert title here</title>**

**</head>**

**<body bgcolor=*"DDF7E3"*>**

**<br>**

**<br>**

**<br>**

**<br>**

**<h1 style="color: *red*;" align=*"center"*>Failure</h1>**

**</body>**

**</html>**

**Employee**

**------------------**

**package com.codegnan.dto;**

**public class Employee {**

**private int eno;**

**private String ename;**

**private float esal;**

**private String eaddr;**

**public int getEno() {**

**return eno;**

**}**

**public void setEno(int eno) {**

**this.eno = eno;**

**}**

**public String getEname() {**

**return ename;**

**}**

**public void setEname(String ename) {**

**this.ename = ename;**

**}**

**public float getEsal() {**

**return esal;**

**}**

**public void setEsal(float esal) {**

**this.esal = esal;**

**}**

**public String getEaddr() {**

**return eaddr;**

**}**

**public void setEaddr(String eaddr) {**

**this.eaddr = eaddr;**

**}**

**}**

**EmployeeDao**

**------------------------**

**package com.codegnan.dao;**

**import com.codegnan.dto.Employee;**

**public interface EmployeeDao {**

**public String add(Employee employee);**

**public Employee search(int eno);**

**public String update(Employee employee);**

**public String delete(int eno);**

**}**

**EmployeeDaoImpl**

**----------------------------**

**package com.codegnan.dao;**

**import java.sql.Connection;**

**import java.sql.PreparedStatement;**

**import java.sql.ResultSet;**

**import java.sql.Statement;**

**import com.codegnan.dto.Employee;**

**import com.codegnan.factories.ConnectionFactory;**

**public class EmployeeDaoImpl implements EmployeeDao {**

**@Override**

**public String add(Employee employee) {**

**String status = "";**

**try {**

**Employee emp = search(employee.getEno());**

**if (emp == null) {**

**Connection connection = ConnectionFactory.*getConnection*();**

**PreparedStatement preparedStatement = connection.prepareStatement("insert into emp1 values(?,?,?,?)");**

**preparedStatement.setInt(1, employee.getEno());**

**preparedStatement.setString(2, employee.getEname());**

**preparedStatement.setFloat(3, employee.getEsal());**

**preparedStatement.setString(4, employee.getEaddr());**

**int rowCount = preparedStatement.executeUpdate();**

**if (rowCount == 1) {**

**status = "success";**

**} else {**

**status = "failure";**

**}**

**} else {**

**status = "existed";**

**}**

**} catch (Exception e) {**

**e.printStackTrace();**

**}**

**return status;**

**}**

**@Override**

**public Employee search(int eno) {**

**Employee employee = null;**

**try {**

**Connection connection = ConnectionFactory.*getConnection*();**

**Statement statement = connection.createStatement();**

**ResultSet resultSet = statement.executeQuery("select \* from emp1 where ENO = " + eno);**

**boolean b = resultSet.next();**

**if (b == true) {**

**employee = new Employee();**

**employee.setEno(resultSet.getInt("ENO"));**

**employee.setEname(resultSet.getString("ENAME"));**

**employee.setEsal(resultSet.getFloat("ESAL"));**

**employee.setEaddr(resultSet.getString("EADDR"));**

**} else {**

**employee = null;**

**}**

**} catch (Exception e) {**

**e.printStackTrace();**

**}**

**return employee;**

**}**

**@Override**

**public String update(Employee employee) {**

**String status = "";**

**try {**

**Connection connection = ConnectionFactory.*getConnection*();**

**PreparedStatement preparedStatement = connection**

**.prepareStatement("update emp1 set ENAME = ?, ESAL = ?, EADDR = ? where ENO = ?");**

**preparedStatement.setString(1, employee.getEname());**

**preparedStatement.setFloat(2, employee.getEsal());**

**preparedStatement.setString(3, employee.getEaddr());**

**preparedStatement.setInt(4, employee.getEno());**

**int rowCount = preparedStatement.executeUpdate();**

**if (rowCount == 1) {**

**status = "success";**

**} else {**

**status = "failure";**

**}**

**} catch (Exception e) {**

**e.printStackTrace();**

**}**

**return status;**

**}**

**@Override**

**public String delete(int eno) {**

**String status = "";**

**try {**

**Connection connection = ConnectionFactory.*getConnection*();**

**PreparedStatement preparedStatement = connection.prepareStatement("delete from emp1 where ENO = ?");**

**preparedStatement.setInt(1, eno);**

**int rowCount = preparedStatement.executeUpdate();**

**if (rowCount == 1) {**

**status = "success";**

**} else {**

**status = "failure";**

**}**

**} catch (Exception e) {**

**e.printStackTrace();**

**}**

**return status;**

**}**

**}**

**EmployeeService**

**-----------------------------**

**package com.codegnan.service;**

**import com.codegnan.dto.Employee;**

**public interface EmployeeService {**

**public String addEmployee(Employee employee);**

**public Employee searchEmployee(int eno);**

**public String updateEmployee(Employee employee);**

**public String deleteEmployee(int eno);**

**}**

**EmployeeServiceImpl**

**------------------------------------**

**package com.codegnan.service;**

**import com.codegnan.dao.EmployeeDao;**

**import com.codegnan.dto.Employee;**

**import com.codegnan.factories.EmployeeDaoFactory;**

**public class EmployeeServiceImpl implements EmployeeService {**

**@Override**

**public String addEmployee(Employee employee) {**

**EmployeeDao employeeDao = EmployeeDaoFactory.*getEmployeeDao*();**

**String status = employeeDao.add(employee);**

**return status;**

**}**

**@Override**

**public Employee searchEmployee(int eno) {**

**EmployeeDao employeeDao = EmployeeDaoFactory.*getEmployeeDao*();**

**Employee employee = employeeDao.search(eno);**

**return employee;**

**}**

**@Override**

**public String updateEmployee(Employee employee) {**

**EmployeeDao employeeDao = EmployeeDaoFactory.*getEmployeeDao*();**

**String status = employeeDao.update(employee);**

**return status;**

**}**

**@Override**

**public String deleteEmployee(int eno) {**

**EmployeeDao employeeDao = EmployeeDaoFactory.*getEmployeeDao*();**

**String status = employeeDao.delete(eno);**

**return status;**

**}**

**}**

**ConnectionFactory**

**-------------------------**

**package com.codegnan.factories;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**public class ConnectionFactory {**

**private static Connection connection = null;**

**static {**

**try {**

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

**connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/adjava", "root", "root");**

**} catch (Exception e) {**

**e.printStackTrace();**

**}**

**}**

**public static Connection getConnection() {**

**return connection;**

**}**

**}**

**EmployeeDaoFactory**

**----------------------------------**

**package com.codegnan.factories;**

**import com.codegnan.dao.EmployeeDao;**

**import com.codegnan.dao.EmployeeDaoImpl;**

**public class EmployeeDaoFactory {**

**private static EmployeeDao *employeeDao* = null;**

**static {**

***employeeDao* = new EmployeeDaoImpl();**

**}**

**public static EmployeeDao getEmployeeDao() {**

**return *employeeDao*;**

**}**

**}**

**EmployeeServiceFactory**

**-----------------------------------------**

**package com.codegnan.factories;**

**import com.codegnan.service.EmployeeService;**

**import com.codegnan.service.EmployeeServiceImpl;**

**public class EmployeeServiceFactory {**

**private static EmployeeService *employeeService* = null;**

**static {**

***employeeService* = new EmployeeServiceImpl();**

**}**

**public static EmployeeService getEmployeeService() {**

**return *employeeService*;**

**}**

**}**

**ControllerServlet**

**----------------------------**

**import java.io.IOException;**

**import java.io.PrintWriter;**

**import javax.servlet.RequestDispatcher;**

**import javax.servlet.ServletException;**

**import javax.servlet.annotation.WebServlet;**

**import javax.servlet.http.HttpServlet;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

**import com.codegnan.dto.Employee;**

**import com.codegnan.factories.EmployeeServiceFactory;**

**import com.codegnan.service.EmployeeService;**

**@WebServlet("/.do")**

**public class ControllerServlet extends HttpServlet {**

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

**protected void doPost(HttpServletRequest request, HttpServletResponse response)**

**throws ServletException, IOException {**

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

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

**int eno = 0;**

**String ename = "";**

**float esal = 0.0f;**

**String eaddr = "";**

**Employee employee = null;**

**EmployeeService employeeService = EmployeeServiceFactory.getEmployeeService();**

**String status = "";**

**RequestDispatcher requestDispatcher = null;**

**String requestURI = request.getRequestURI();**

**// System.out.println(requestURI);**

**if (requestURI.endsWith("add.do")) {**

**try {**

**eno = Integer.parseInt(request.getParameter("eno"));**

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

**esal = Float.parseFloat(request.getParameter("esal"));**

**eaddr = request.getParameter("eaddr");**

**employee = new Employee();**

**employee.setEno(eno);**

**employee.setEname(ename);**

**employee.setEsal(esal);**

**employee.setEaddr(eaddr);**

**status = employeeService.addEmployee(employee);**

**if (status.equalsIgnoreCase("existed")) {**

**requestDispatcher = request.getRequestDispatcher("/existed.html");**

**}**

**if (status.equalsIgnoreCase("success")) {**

**requestDispatcher = request.getRequestDispatcher("/success.html");**

**}**

**if (status.equalsIgnoreCase("failure")) {**

**requestDispatcher = request.getRequestDispatcher("/failure.html");**

**}**

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

**} catch (Exception e) {**

**e.printStackTrace();**

**}**

**}**

**if (requestURI.endsWith("search.do")) {**

**eno = Integer.parseInt(request.getParameter("eno"));**

**employee = employeeService.searchEmployee(eno);**

**if (employee == null) {**

**requestDispatcher = request.getRequestDispatcher("/notexisted.html");**

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

**} else {**

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

**out.println("<body bgcolor='DDF7E3'><center>");**

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

**out.println("<table align='center' border='1'>");**

**out.println("<tr><td>Employee Number</td><td>" + employee.getEno() + "</td></tr>");**

**out.println("<tr><td>Employee Name</td><td>" + employee.getEname() + "</td></tr>");**

**out.println("<tr><td>Employee Salary</td><td>" + employee.getEsal() + "</td></tr>");**

**out.println("<tr><td>Employee Address</td><td>" + employee.getEaddr() + "</td></tr>");**

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

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

**}**

**}**

**if (requestURI.endsWith("editform.do")) {**

**try {**

**eno = Integer.parseInt(request.getParameter("eno"));**

**employee = employeeService.searchEmployee(eno);**

**if (employee == null) {**

**requestDispatcher = request.getRequestDispatcher("notexisted.html");**

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

**} else {**

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

**out.println("<body bgcolor='DDF7E3'><center>");**

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

**out.println("<form method='POST' action='update.do'>");**

**out.println("<table align='center'>");**

**out.println("<tr><td>Employee Number</td><td>" + employee.getEno() + "</td></tr>");**

**out.println("<input type='hidden' name='eno' value='" + employee.getEno() + "'/>");**

**out.println("<tr><td>Employee Name</td><td><input type='text' name='ename' value='"**

**+ employee.getEname() + "'></td></tr>");**

**out.println("<tr><td>Employee Salary</td><td><input type='text' name='esal' value='"**

**+ employee.getEsal() + "'></td></tr>");**

**out.println("<tr><td>Employee Address</td><td><input type='text' name='eaddr' value='"**

**+ employee.getEaddr() + "'></td></tr>");**

**out.println("<tr><td><input type='submit' value='UPDATE'></td></tr>");**

**out.println("</table></form>");**

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

**}**

**} catch (Exception e) {**

**e.printStackTrace();**

**}**

**}**

**if (requestURI.endsWith("update.do")) {**

**try {**

**eno = Integer.parseInt(request.getParameter("eno"));**

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

**esal = Float.parseFloat(request.getParameter("esal"));**

**eaddr = request.getParameter("eaddr");**

**employee = new Employee();**

**employee.setEno(eno);**

**employee.setEname(ename);**

**employee.setEsal(esal);**

**employee.setEaddr(eaddr);**

**status = employeeService.updateEmployee(employee);**

**if (status.equalsIgnoreCase("success")) {**

**requestDispatcher = request.getRequestDispatcher("success.html");**

**} else {**

**requestDispatcher = request.getRequestDispatcher("failure.html");**

**}**

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

**} catch (Exception e) {**

**e.printStackTrace();**

**}**

**}**

**if (requestURI.endsWith("delete.do")) {**

**eno = Integer.parseInt(request.getParameter("eno"));**

**employee = employeeService.searchEmployee(eno);**

**if (employee == null) {**

**requestDispatcher = request.getRequestDispatcher("/notexisted.html");**

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

**} else {**

**status = employeeService.deleteEmployee(eno);**

**if (status.equalsIgnoreCase("success")) {**

**requestDispatcher = request.getRequestDispatcher("/success.html");**

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

**} else {**

**requestDispatcher = request.getRequestDispatcher("/failure.html");**

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

**}**

**}**

**}**

**}**

**}**

**Web.xml**

**==========**

**<web-app>**

<servlet>

<description></description>

<display-name>ControllerServlet</display-name>

<servlet-name>ControllerServlet</servlet-name>

<servlet-class>com.codegnan.controller.ControllerServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>ControllerServlet</servlet-name>

<url-pattern>\*.do</url-pattern>

</servlet-mapping>

<display-name>app24-empapp</display-name>

<welcome-file-list>

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

<welcome-file>index.htm</welcome-file>

<welcome-file>index.jsp</welcome-file>

<welcome-file>default.html</welcome-file>

<welcome-file>default.htm</welcome-file>

<welcome-file>default.jsp</welcome-file>

</welcome-file-list>

</web-app>