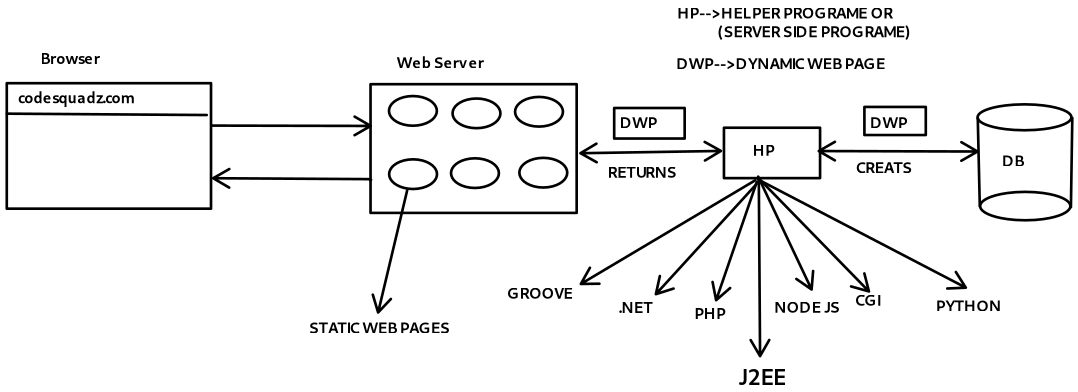
**Java class - 85, 22 Apr 2024**

**JEE**

**Why we need to make dynamic pages in web based application**

Whenever we have to show the current data from database on our web page. For that we have to create that page on the spot, that is why we make dynamic pages



**JEE Technologies**

JEE provide us three types of technologies to create server side programe  
 **1. Component technology**

a) Web component

i) Servlet

ii) JSP

b) EJB component[ It is removed from JEE, now JPA(Java Persistent API) has took the place of

EJB ]

i) Session Bean

ii) Entity Bean

iii) Message Driven Bean

**2. Service technology**

JEE provides so many services and these services are used by component (Web or EJB)

a) JDBC (Java Data Base Connectivity)

b) JNDI (Java Naming and Directory Interface)

c) Java Mail API

d) JMS (Java Message Service)

e) JAAS (Java Authentication and Authorization service)

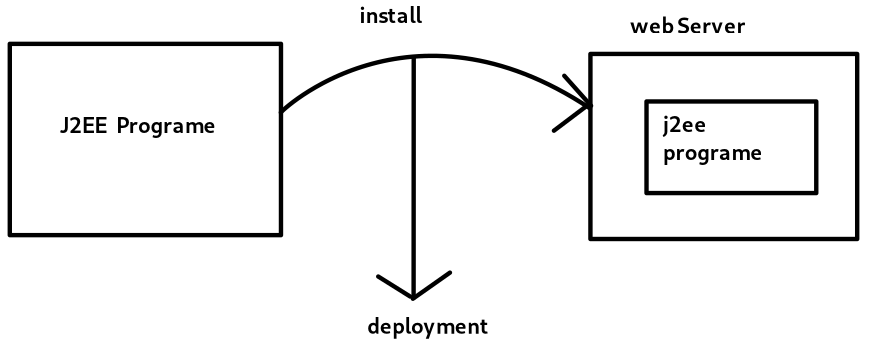
f) JTA (Java Transaction Service)

**3. Communication technology**

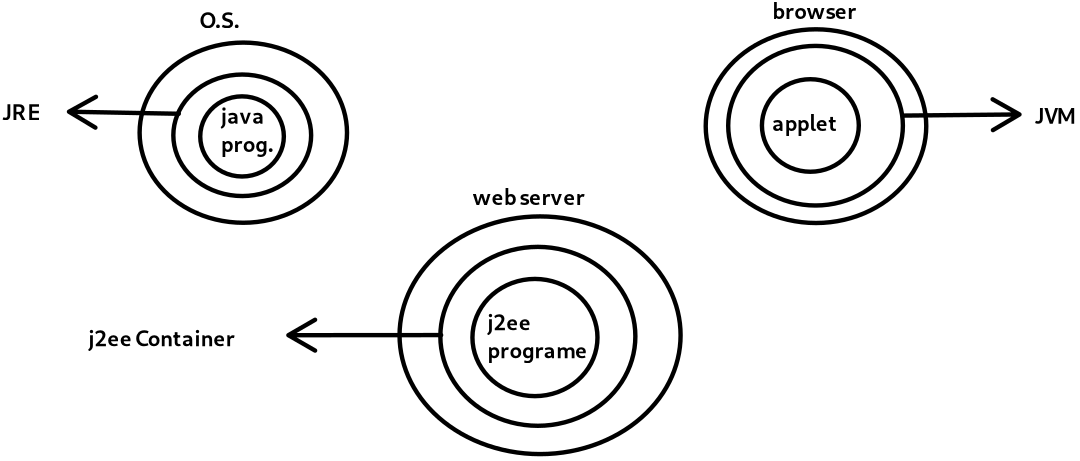
Used to make communication between server side programe and other technologies

a) Internet Protocol (HTTP, TCP/IP, SSL)

b) Remote Object Protocol (RMI)



**Java class - 86, 23 Apr 2024**

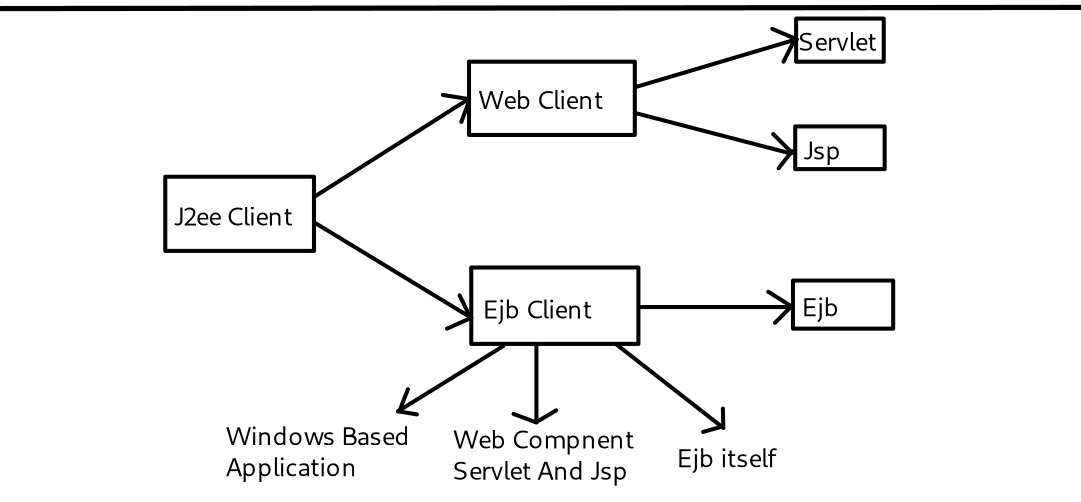
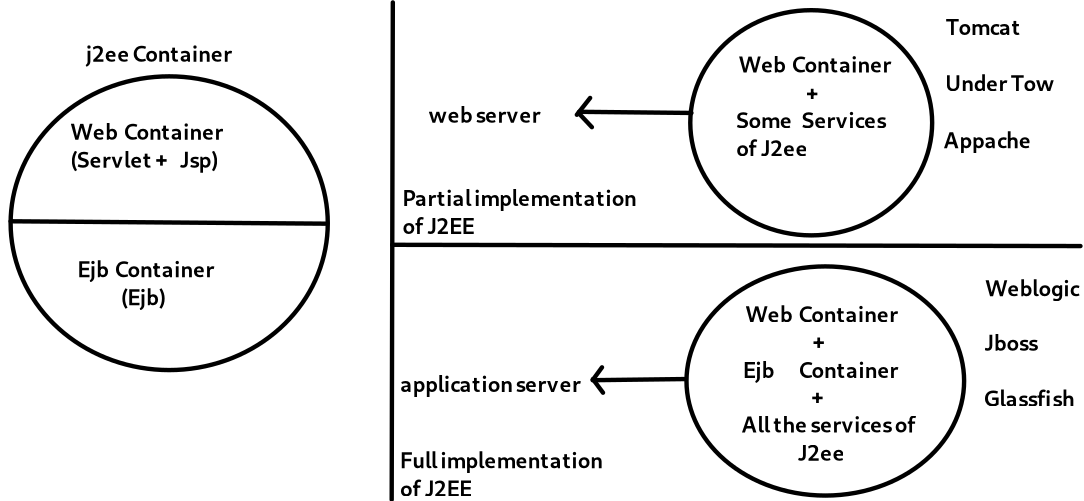
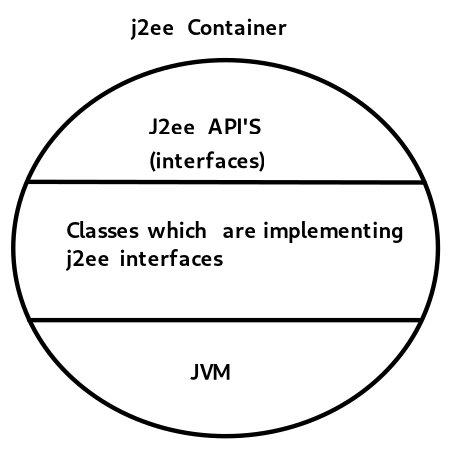


**Who makes this container and what is inside the container?**

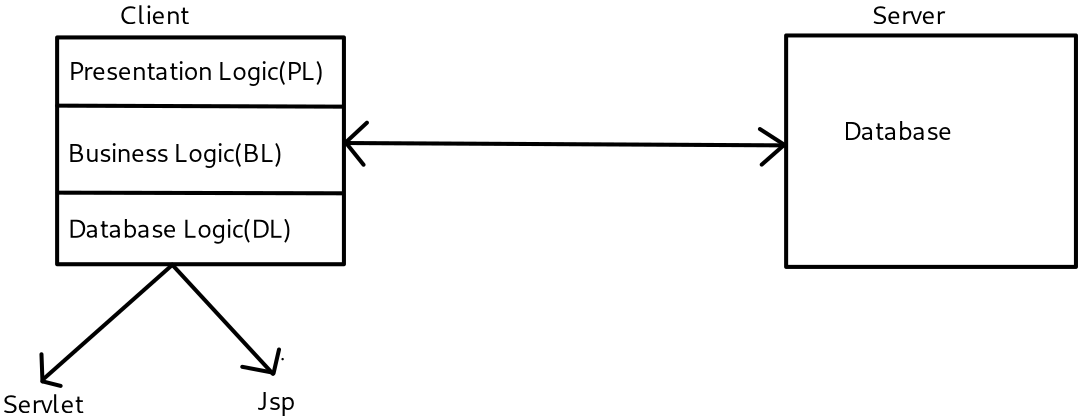
J2EE is a framework. It contains interface. Third party vendors provide the classes that implements.

Container contains three things first all the interfaces of J2EE, second classes that implements those interface and third is JVM. The combination of all three things is container. Container can not exist independently it always runs inside the web server but web server can exist without any container.

Java supports more than 45 web servers.

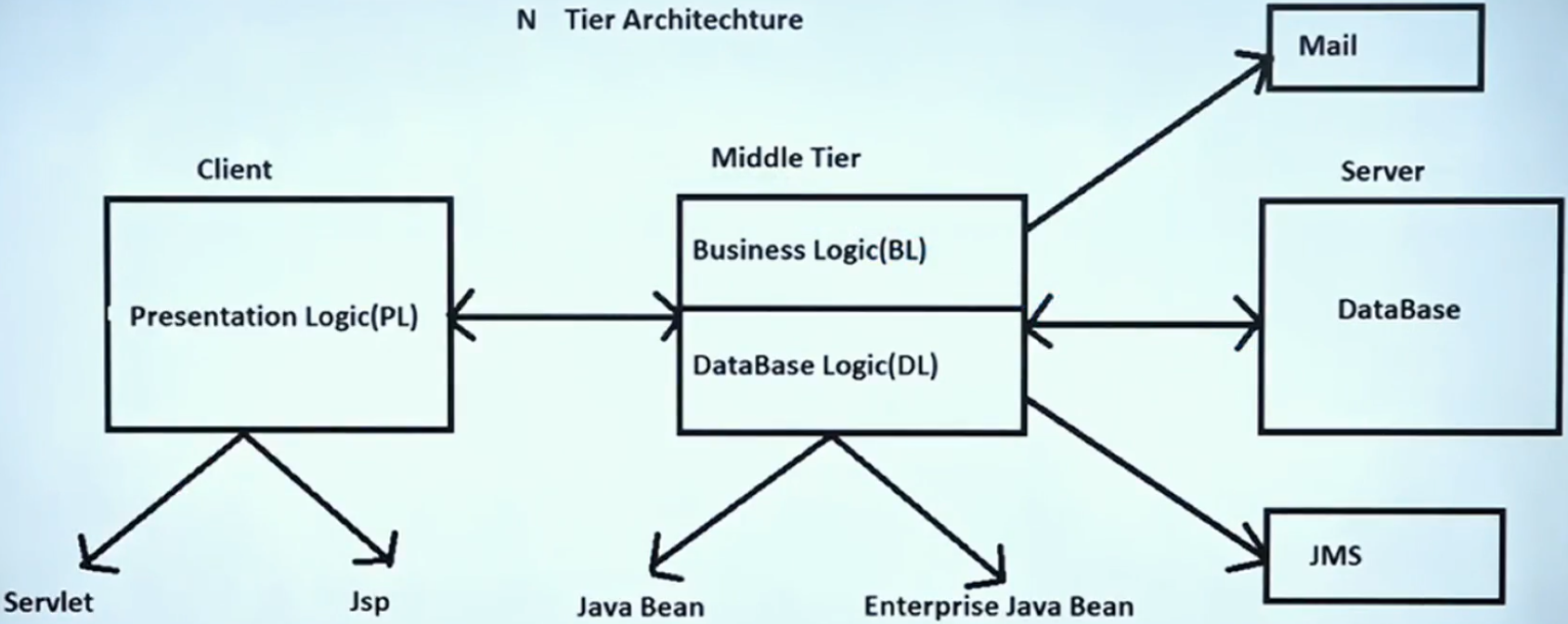
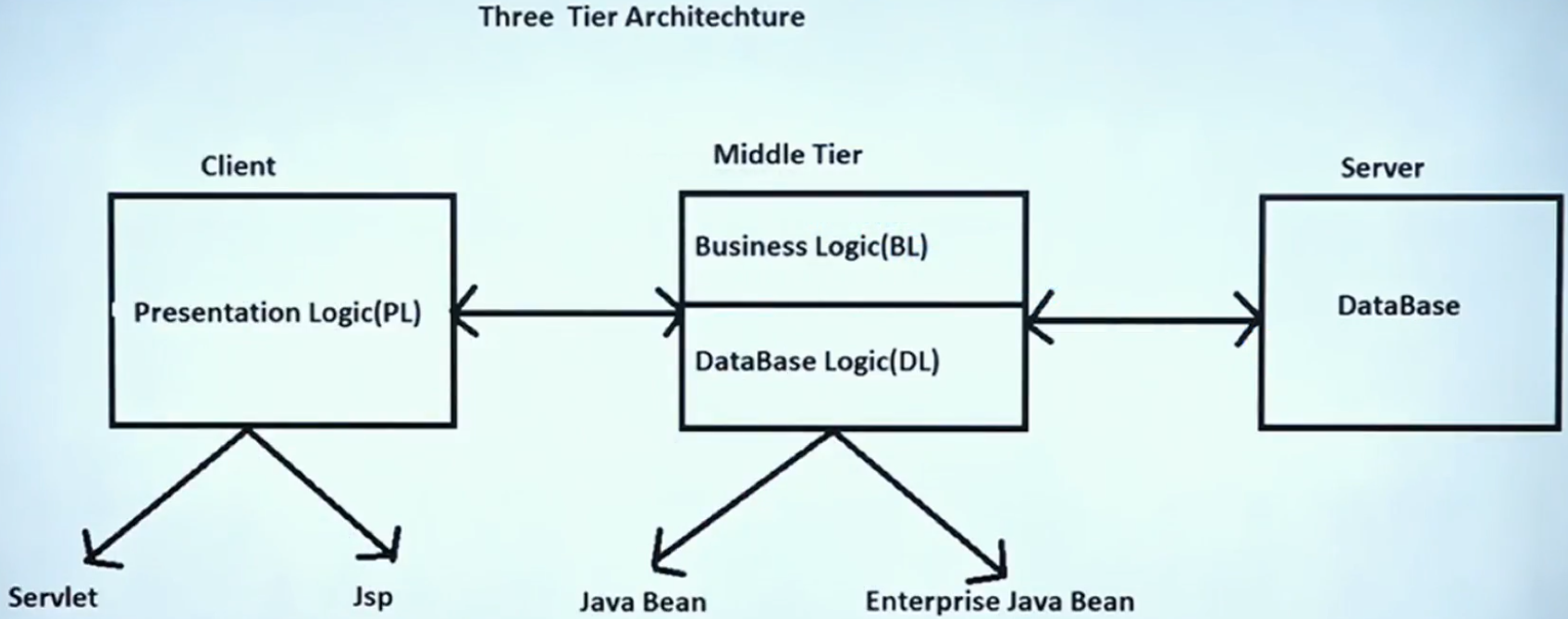


**Two tier architecture:** In two tier we have only two things first is client and second is server. Server provides only the database. Client has three things presentation logic, business logic, and database logic

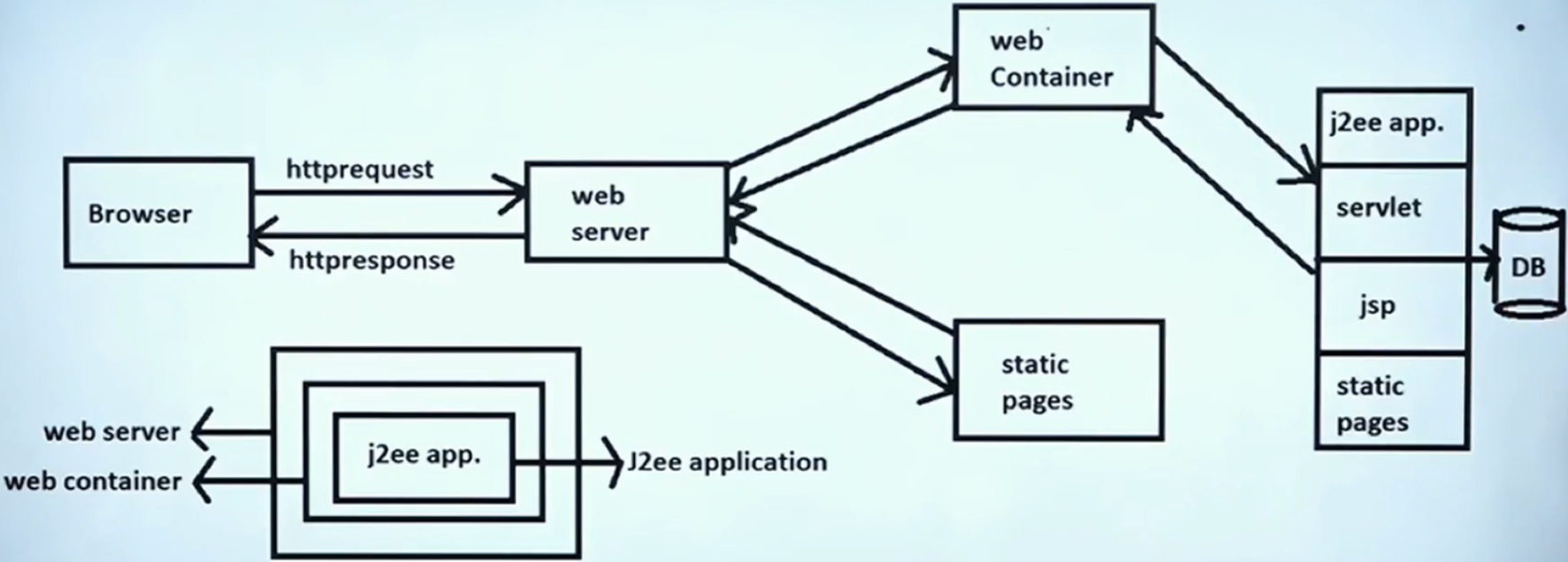


It is easy to deploy. But the biggest problem with two tier is making upgrades in application. In two tier architecture all the three things are on client side so if we make changes in presentation logic the business and database logic are going to break or if we make changes in database then other two are going to break. And if you want to upgrade you have close the entire application then you can make changes, because they are tightly couple with each other

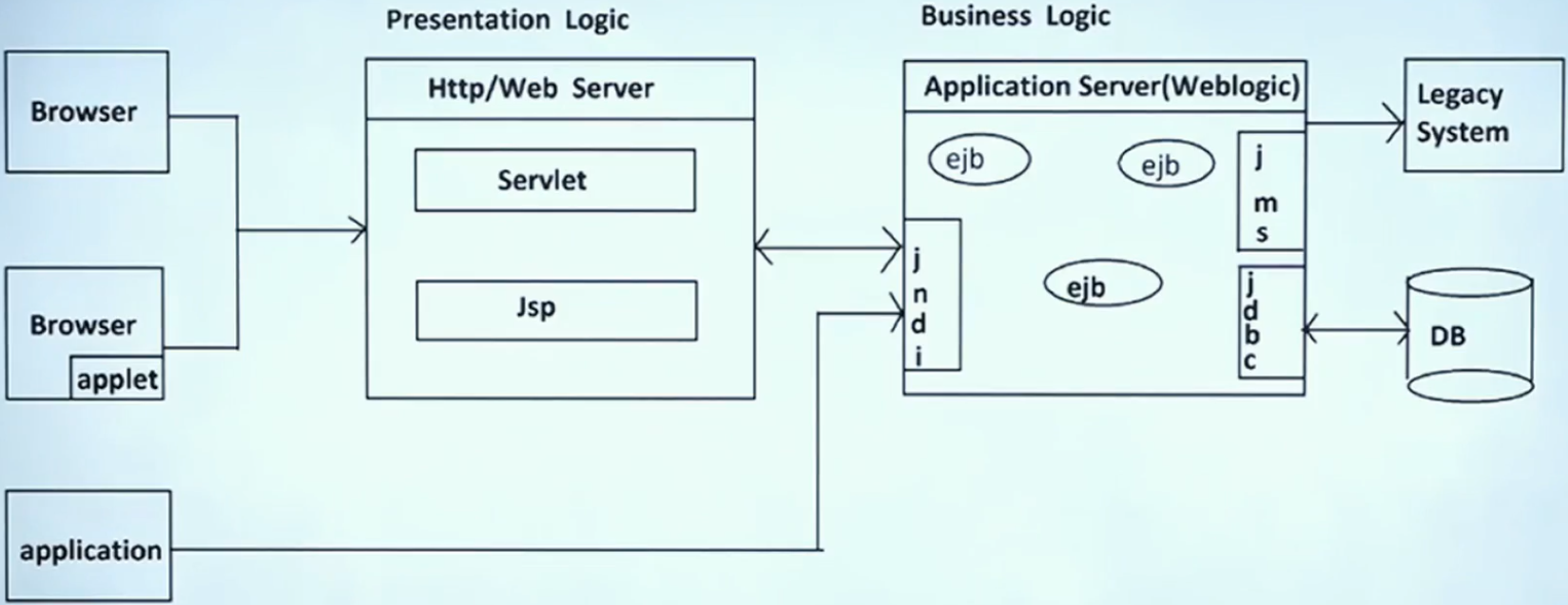
**Three tier architecture:** In this tier we have three things first is presentation logic, second is middle tier and third is server. We use EJB to make middle component.



**J2EE two tier architecture based application diagram:** We send a web page request using browser. And whenever we send a request it goes to the web server with the help of HTTP protocol called HTTP request, suppose you are requesting a static page then web server send a that page to you, if you are requesting a dynamic page then your request is forwarded to web container, and this web container contains the J2EE application( Servlet, JSP, static pages) that will fetch some data and create new page send to web container, web container to web server, web server to you

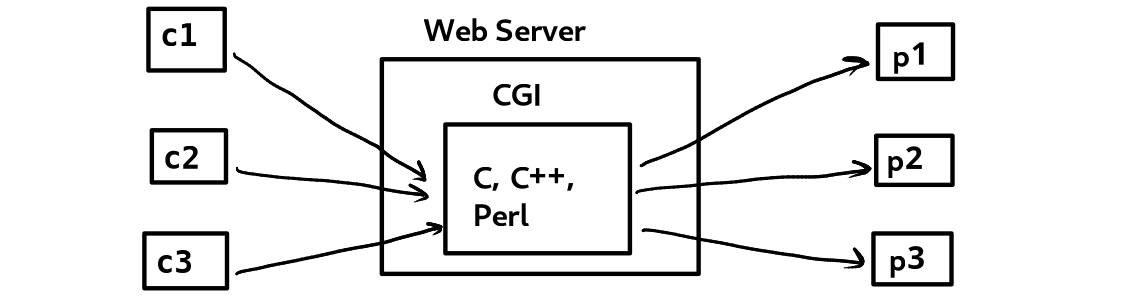


**JEE N tier architecture application diagram:** When we make a request using browser it goes to presentation logic which is present in HTTP/Web server(Servlet and JSP), Servlet and JSP calls the EJB(business logic) using JNDI service which is present in application server. Now, if EJB need some data from database it uses JDBC service or EJB needs to message then it uses other system. But now EJB is replaced by Java Bean

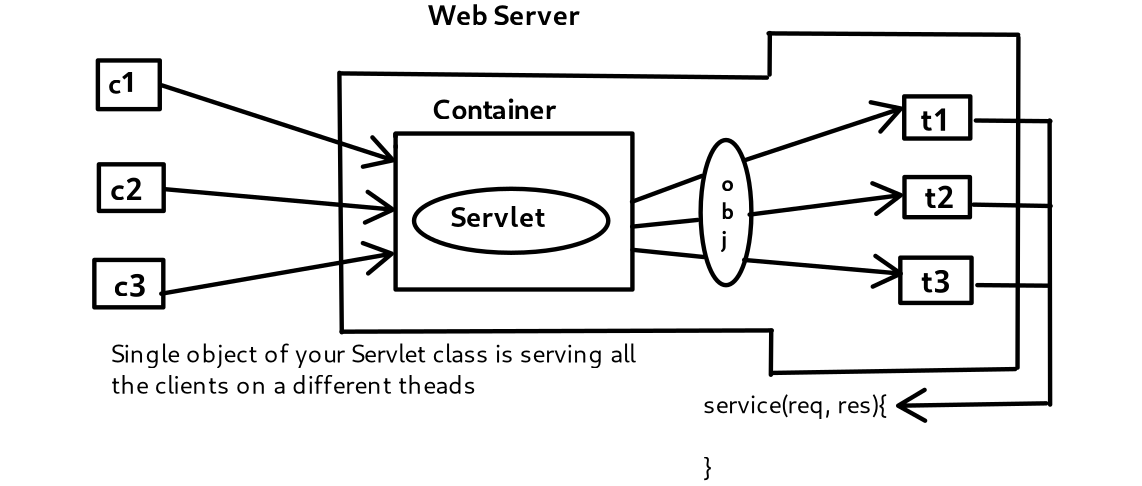


**Different different technologies to make server side programe and their problem**

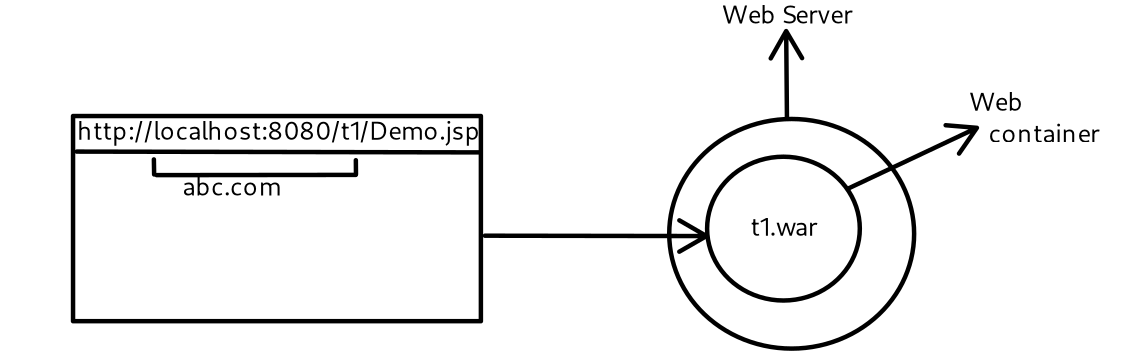
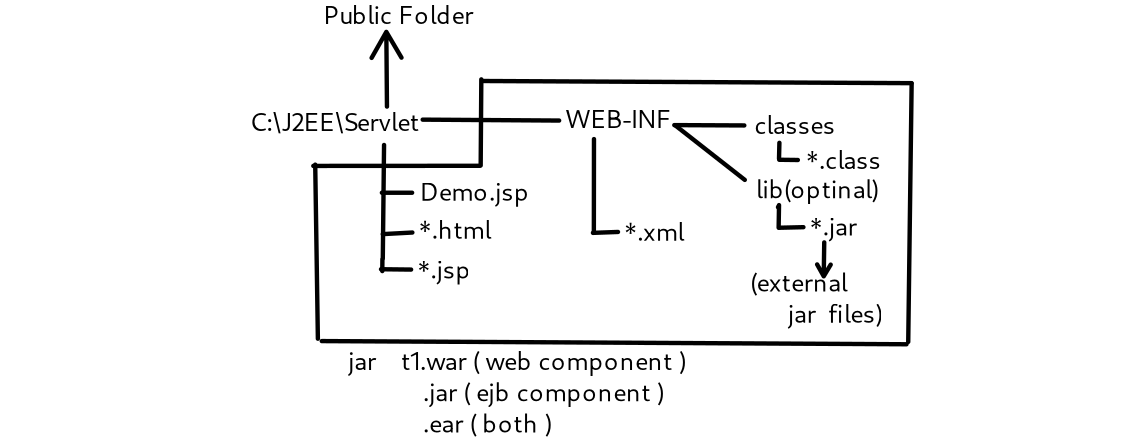
First we talk about **CGI**(Common Gateway Interface). It was a container inserted into the web server, and within this container, we would write server-side code in languages like C, C++, or Perl. When a request came from the browser for this server-side code, the web server would pass it to the CGI. CGI would then run this program as a separate process, meaning it would allocate separate memory in RAM for this program and start a new process for that particular client. This process would then interact with the client, handling the data exchange. However, if another client made a request to the same server-side program, which is quite common in web-based applications where multiple clients may access the server simultaneously, CGI would reload this program into memory again for the new client, starting another process, and so on for each new request. That is why CGI is slow



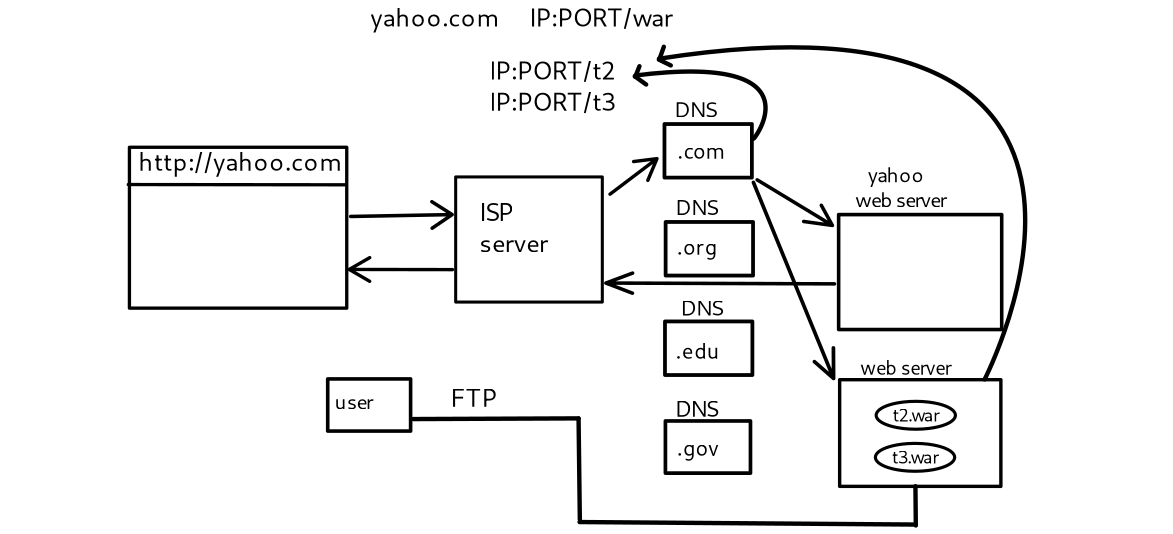
**Solution**



**Directory structure of web component**



**How internet work**



**Get method of HTTP request:** The method name tells the server the kind of request that is being made and how the rest of the message will be formatted. Get is the simplest http method and it’s main job in life is to ask the server to get a resources and send it back. That resources might be an html page, pdf file etc.

Limitation of get method:

1. The total amount of characters in GET method is really limited. If you type long passage into

search input box the GET might not work.

2. The data you send with GET is appended to the URL up in the browser, so whatever you send is

exposed. Better not put password or some other sensitive data as part of GET method.

**HTTP request Header**

Header name Header value

Host google.com

User Agent-IE/8.0

Accept text/xml

Accept.language en-US

Connection Keep-alive

**Post method of HTTP request:** Post is a more powerful request. With post method you can request something and at the same time send form data to the server.

**HTTP response Header:** An HTTP response has both a header and a body. The header info tells browser about the protocol being used,whether the request was successful, what kind of content is included in body. The body contains the contents for browser to display

Header name Header value

set-cookie name=value

content-type text/html

content-length 200

Date 15 may 2021 12:44:10 iST

server Tomcat

connection close

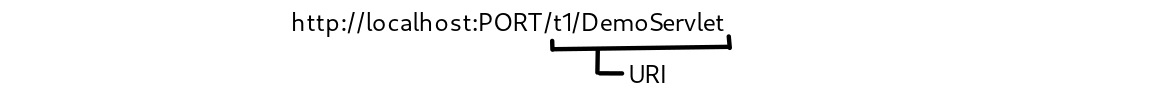
**Content-Type:** This header’s value is knows as MIME(Multi Purpose Internet Mail Extension). The MIME type tells the browser what kind of data the browser is about to receive so that the browser will know how to render it

**Other methods for HTTP request:** HEAD, TRACE, PUT, DELETE, CONNECT.

**Five sections of HTTP Request**

**Verb** – Indicates the HTTP methods such as GET, POST, DELETE, PUT, etc.

**URI** – Uniform Resource Identifier(URI) to identify the resource on the server.



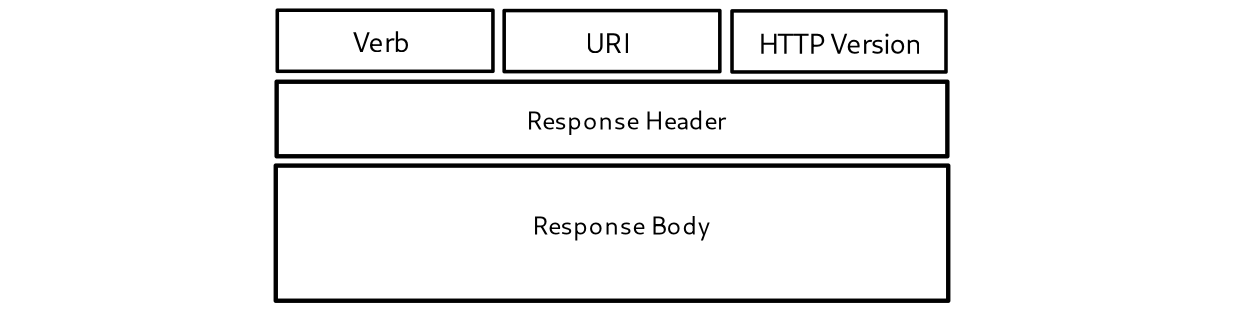
**Request Version** – Indicates the HTTP version. For example, HTTP v1.1.

**Request Header** – Contains meta-data for the HTTP Request message as key-value pairs. For

example, client(or browser) type, format supported by the client, format of the

message body, cache settings, etc

**Request Body** – Message content or Resource representation.



**Four sections of HTTP Response**

**Status/Response Code** – Indicates the Server status for the requested resource. For example, 404

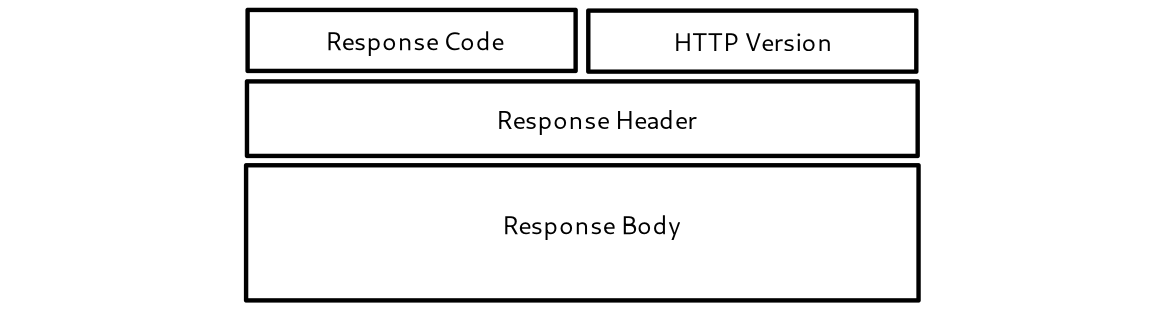
means resource not found and 200 means response is ok

**HTTP Version** – Indicates the HTTP version. For example, HTTP v1.1.

**Response Header** – Contains meta-data for the HTTP Response message as key-value pairs. For

example, content length, content type, response date, server type, etc.

**Response Body** – Response message content or Resource representation.



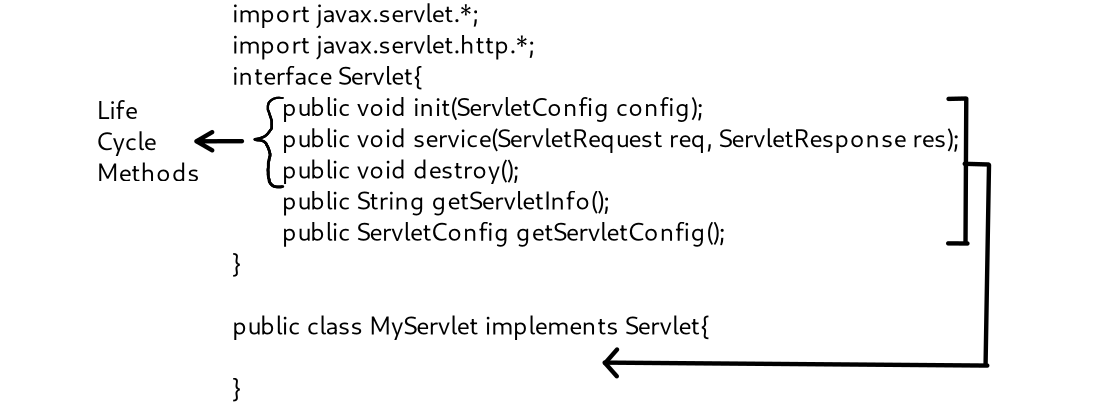
All the API related to servlet(classes and interfaces) are not available in rt.jar file, we have download any web server’s jar file and have to set the CLASSPATH of that file. These classes can be found in two packages of web server’s jar file first is javax.servlet.\* and second is javax.servlet.http.\*. The top most interface of the servlet api is Servlet interface can be found in javax.servlet.Servlet.

There are total five methods in Servlet interface and whenever we make our servlet programe, we have to override the all five methods of Servlet interface and we put our servlet programe in container, then we make a request of servlet from browser and container creates the object of our servlet programe and call all five method one by one actually container call the three methods and we have to call other two methods inside from any of those three methods otherwise they are not going to run itself

These three methods are called Life Cycle methods of servlet. There are two type of method in java first callback method and second is life cycle method. Callback methods are those methods which are defined by programer but they are called by the technology and all the interface methods are callback methods

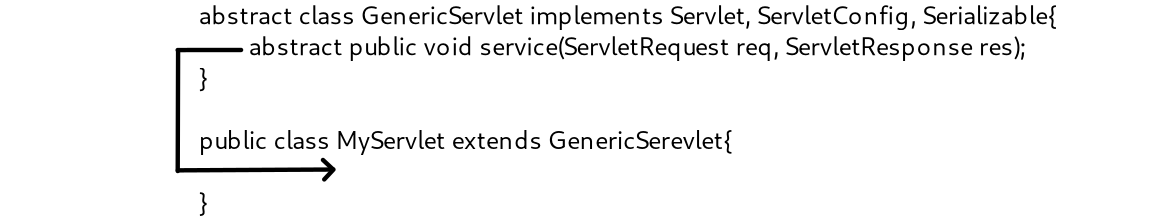
If these callback methods are executed by technology between the time object was created and object was destroyed then these callback methods are called life cycle methods

**1st way of creating servlet**



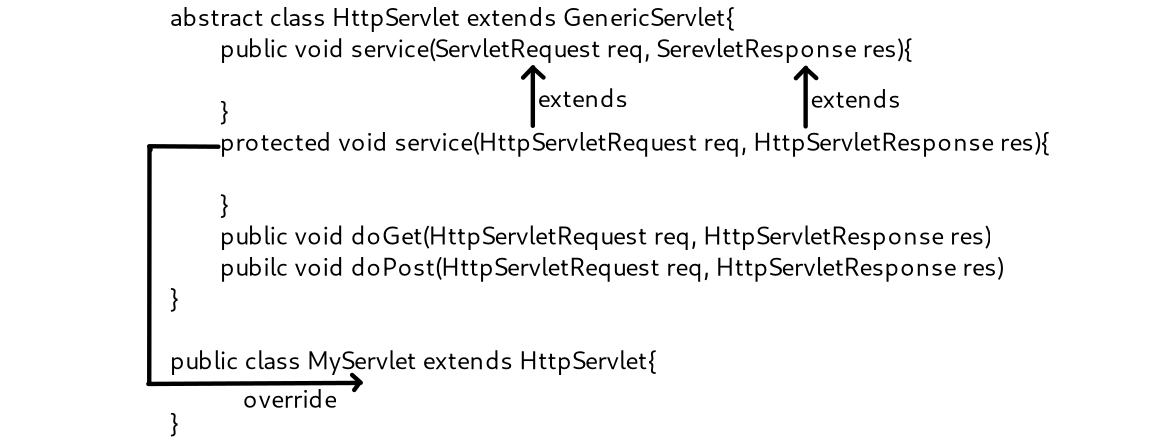
As we can see above the first method is public void init which accepts ServletConfig argument. Second is public void service which accepts ServletRequest and ServletResponse two interface argument. Third is public void destroy accepts no argument these three are life cycle method. Fourth is public String getServletInfo() accepts no argument and return a String value. Fifth is public ServletConfig getServletConfig()

**2nd way of creating servlet**



But there is a problem, when we create our servlet programe using this Servlet interface, we have to override all five methods. The solution is they gave us an adapter class GenericServlet and that adapter class already implement the Servlet interface and override its methods. Now we create servlet by inheriting GenericServlet class this class overrides all the methods of Servlet interface except service method and that’s why Generic class is a abstract class and this class does not only implement the Servlet interface it also implements the ServletConfig and Serializable

**3rd way of creating servlet**



Here we have third class HttpServlet and this class inherits the GenericServlet. there is only one method service from Generic which is abstract and this class also defined this method that means in HttpServlet

class there is no abstract method in it but this class is still abstract means you can not create object, only inheriting is possible. So first thing is the HttpServlet class has overridden the service method from GenericServlet which original service method and second thing is this class overloaded the service method which accepts two interface arguments, first argument is HttpServletRequest and second is HttpServletResponse and these two interfaces extend the ServletRequest and ServletResponse respectively, and this service is duplicate method.

And HttpServlet class also defined the all HTTP protocol methods like doGet, doPost.

**Difference between GenericServlet and HttpServlet**

Using GenericServlet class you can not perform following http protocol specific task.

1. You can not perform the task according to the method’s of HTTP protocol.

2. You can not track the session.

3. You can not handle cookies.

4. You can not handle request and response Headers.

**Steps for implementing a Servlet**

**1.** Set the class path of a jar file which is containing the J2EE API and their implementing classes.

**2.** Create the directory structure of Web Component.

**3.** Create the Servlet class by using one of the three ways then compile it and the .class file of this servlet

class into the classes folder of directory structure.

**4.** Create an XML file with a name web.xml and give your servlet class entry into this xml file and put this

xml file into WEB-INF

**5.** Create the .war file of this directory structure excluding the public folder.

**6.** Deploy the .war file into web server which is containing the J2EE Container.

**Programe 1: first servlet programe**

**public/DemoServlet1.java**

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

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

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

**public** **class** **DemoServlet1** **implements** **Servlet**{

**public** **void** **init**(ServletConfig con){}

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

        res.setContentType("text/html");

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

        out.println("<html><body>");

        out.println("hello servlet");

        out.println("<br>");

        out.println("Reference Id = "+this);

        out.println("<br>");

        out.println("Thread Name = "+Thread.currentThread().getName());

**try**{

*//Thread.sleep(15000);*

        }

**catch**(Exception e){}

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

    }

**public** **void** **destroy**(){}

**public** ServletConfig **getServletConfig**(){

**return** null;

    }

**public** String **getServletInfo**(){

**return** null;

    }

}

**public/index.html**

<html>

<body>

    <a href="demo1">DemoServlet1</a>

    <br>

</body>

</html>

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>DemoServlet1</servlet-name>

        <servlet-class>DemoServlet1</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>DemoServlet1</servlet-name>

        <url-pattern>/demo1</url-pattern>

    </servlet-mapping>

</web-app>

**Command of making application**

public> javac -d WEB-INF/classes DemoServlet1.java

public> jar cvf s1.war \*

**Three ways to deploy**

**1.** Cold deployment

**2.** Hot deployment

**3.** exploded form

**Seven ways for calling a servlet**

1. URL Pattern -> /demo1

2. Via Default Page -> index.html parallel to WEB-INF folder

3. Via any HTML -> /abc.html in web.xml in url-pattern

4. Giving the URL pattern of servlet as /index.html which will make you servlet as default page in url-

pattern

5. Giving the URL pattern of servlet as \*.lalu in url-pattern

6. Giving the URL pattern of servlet as /\* so it can be invoked by any URL in url-pattern

7. Giving the URL pattern of servlet as /test/\* in url-pattern

Now we are going to create servlet program using GenericServlet and by using this we don’t need to override other method we can just override the service method , that’s it and we good go. And it does not matter even if we override the other methods also.

**Que. What if we call destroy method, will it destroy the servlet’s object?**

Ans. No, it is the last method of servlet’s life cycle, when container calls the destroy method and after executing all the code inside the destroy() method then container destroy the servlet’s object , destroy() method does not destroy the servlet’s object itself. Whether you invoke it or container invokes it will execute normally. Container does not destroy the servlet’s object so easily because single object of servlet have to serve all the requests.

**Programe 2: servlet using generic**

**public/DemoServlet.java**

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

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

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

**public** **class** **DemoServlet** **extends** **GenericServlet**{

    PrintWriter out;

**static** **int** count=0;

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

        res.setContentType("text/html");

        out = res.getWriter();

        destroy();

        out.println("<html><body>");

        out.println("<br>");

        out.println("Hello servlet from generic servlet "+count);

        out.println("<br>");

        out.println(this);

**try**{

*//Thread.sleep(2000);*

            out.println("<br>");

        }

**catch**(Exception e){

*//empty*

        }

        out.println(Thread.currentThread().getName());

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

    }

**public** **void** **destroy**(){

        out.println("destroy");

        count++;

    }

}

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>DemoServlet</servlet-name>

        <servlet-class>DemoServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>DemoServlet</servlet-name>

        <url-pattern>/demo</url-pattern>

    </servlet-mapping>

</web-app>

**public/index.html**

<html>

<body>

<a href="demo">**DemoServlet**</a>

</body>

</html>

**How a container handles the first request of a servlet**

1. Container loads the servlet class. —> Class c = Class.forName(“DemoServlet”);

2. Container instantiate the servlet class. —> Servlet s = (Servlet)c.newInstance( );

3. Container creates the object of ServletConfig interface.

4. Container call the init() method of servlet class then passes the above object into this method as a

arguments.

5. Now container creates two objects

a) ServletResponse or HttpServletResponse

b) ServletRequest or HttpServletRequest

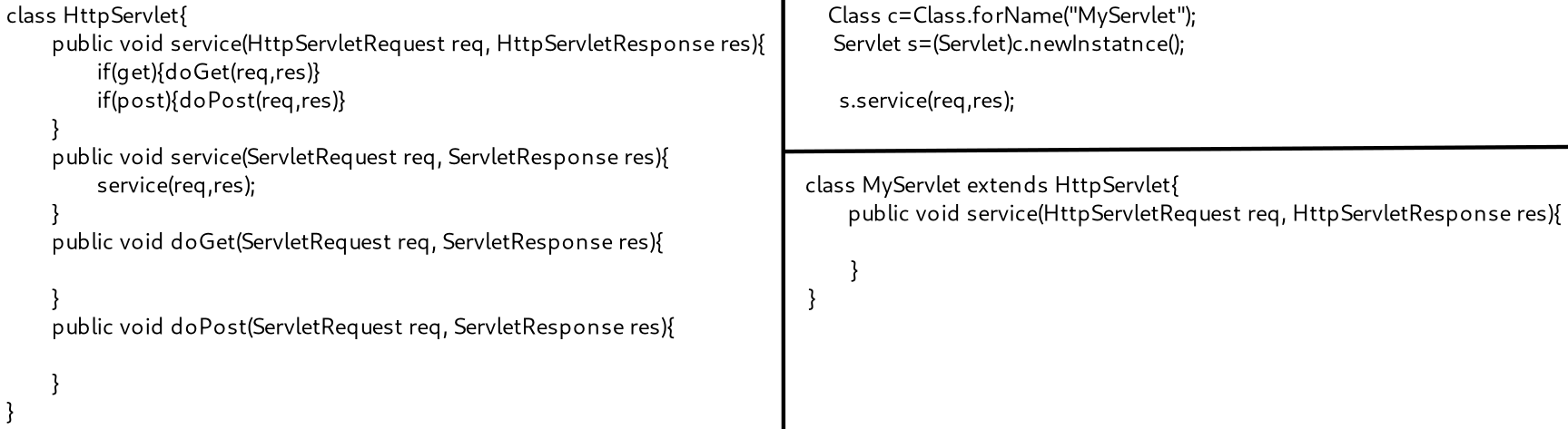
6. Container call the service() method and passes above two objects into this method as arguments.

7. service() process the client request.

8. If the container or web server gets another request of this servlet then process begin from step 5.

9. If the serve goes down or application goes down then container calls the destroy() method.

**Working of HttpServlet**

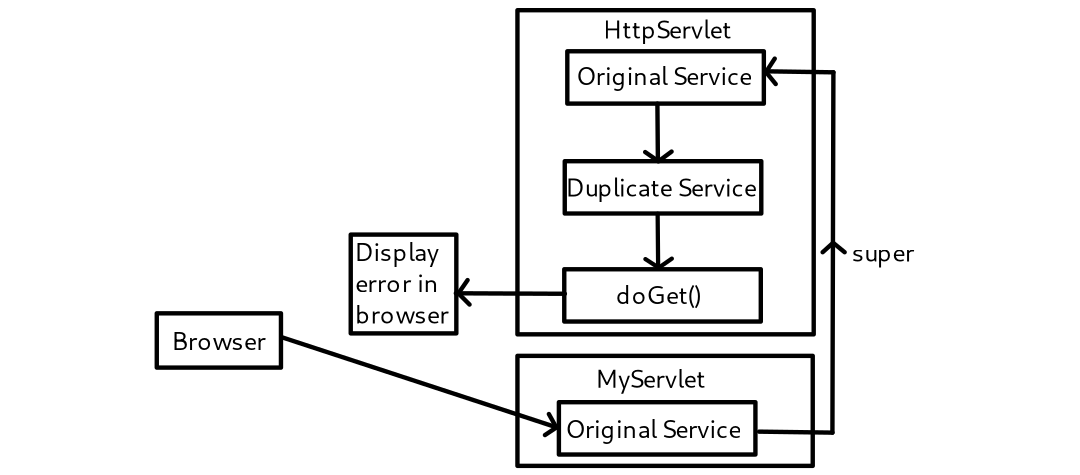


First, container create Class class object using forName() method then container calls newInstance method on Class class object and catch the reference id in parent(Servlet interface) and then it calls the original service method but in our MyServlet class we have overridden the duplicate service method now the question is how container will call duplicate service. MyServlet extends the HttpServlet that means MyServlet will get all the methods of HttpServlet and we have already overridden the duplicate service so if the container calls other methods the parent’s method will we executed and inside the original, duplicate will be invoked.

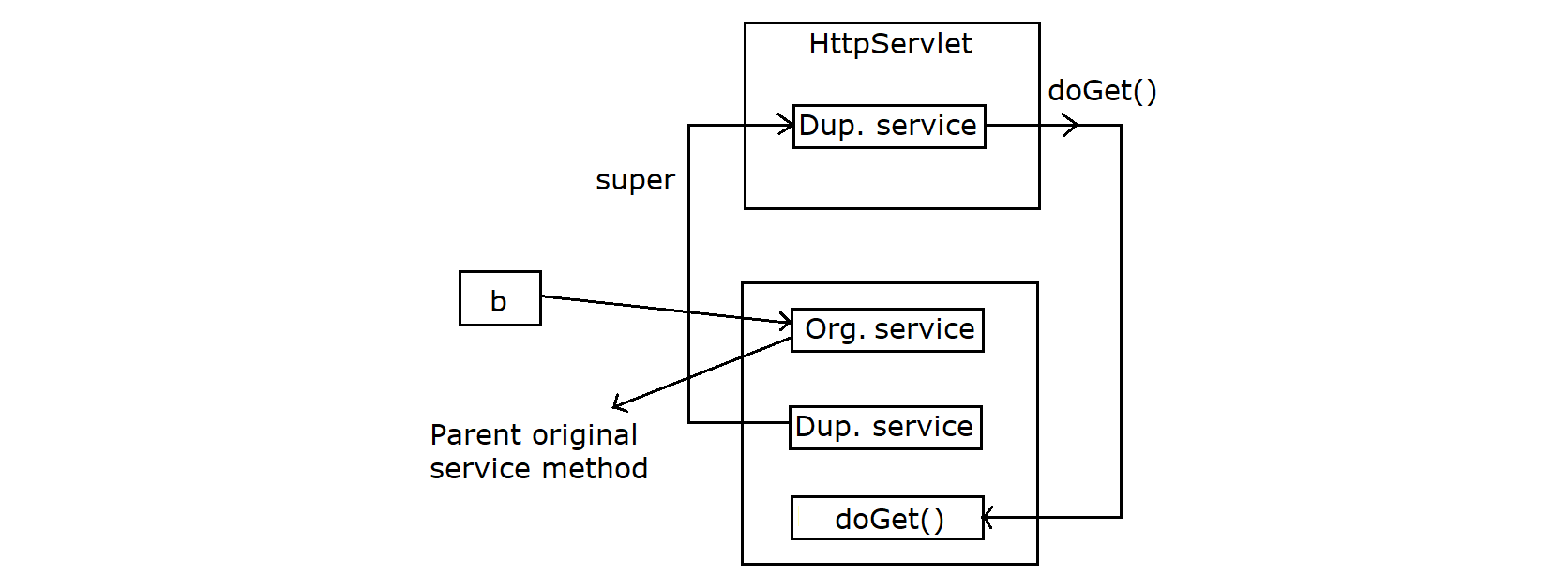
If you want doGet() and doPost() to be executed then do not override the any service method

Suppose we override the original service and what if we call original service using super the parent’s original service method and same of duplicate

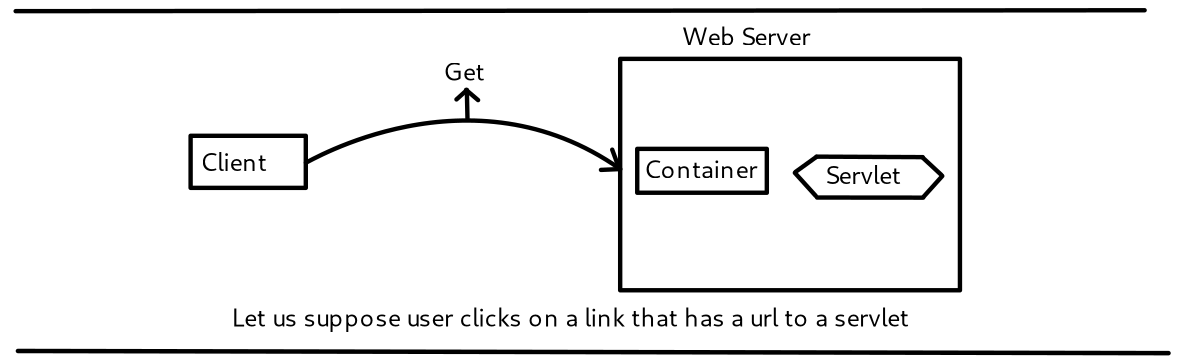
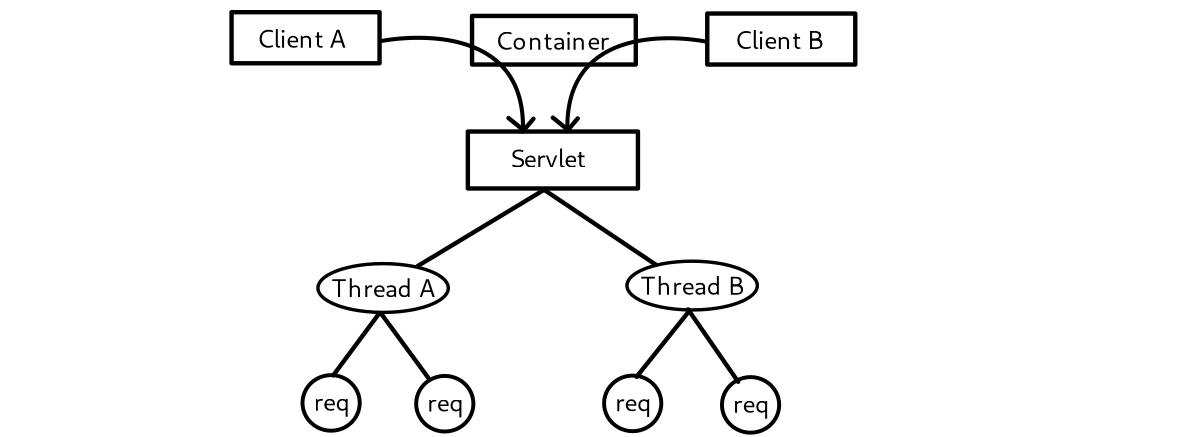
Overriding original service method



Overriding duplicate service method

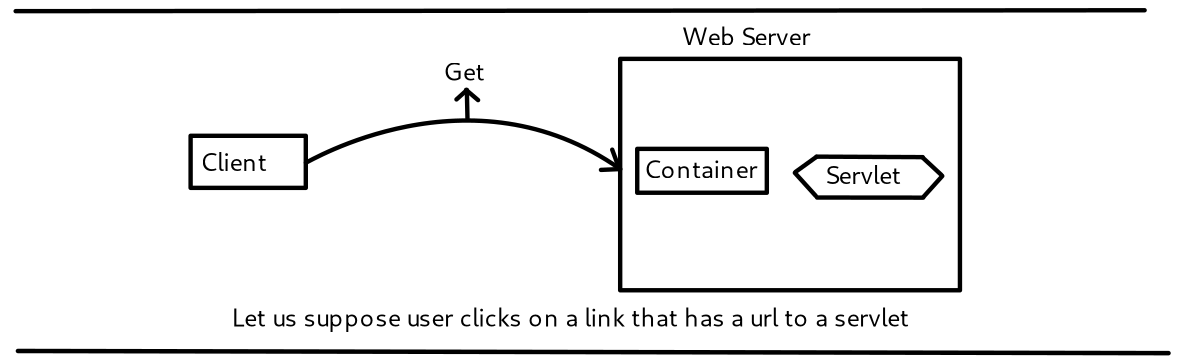


**First Life cycle of servlet**

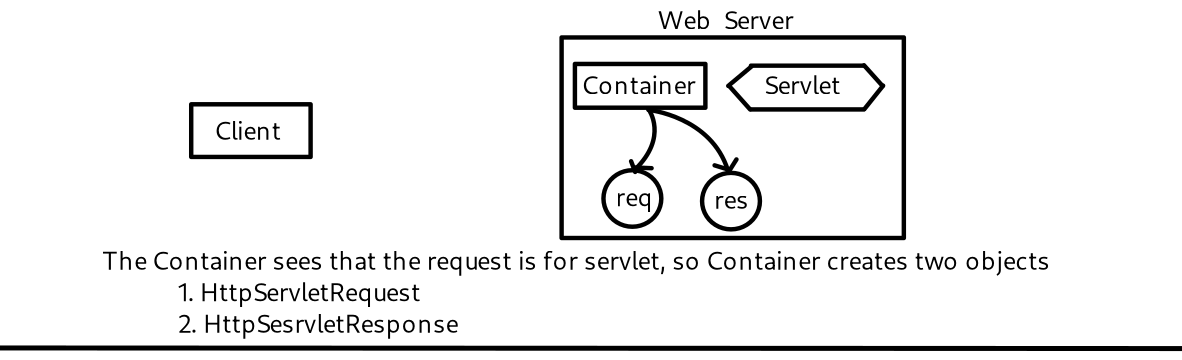


**How a container handles the frequent request of the same servlet**

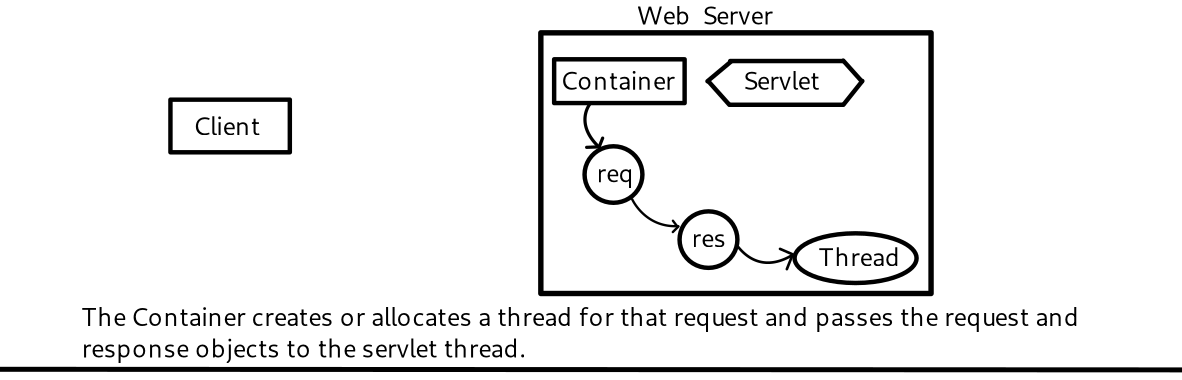
**Step 1:**



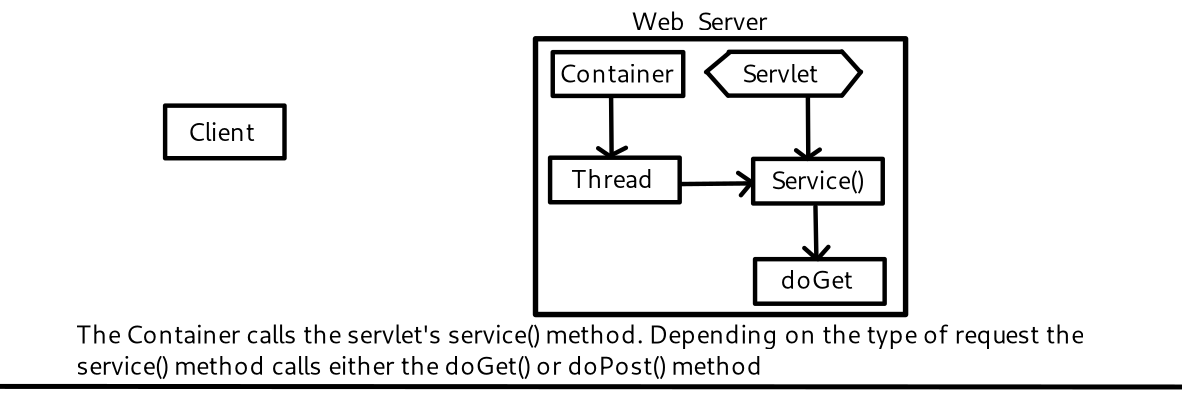
**Step 2:**



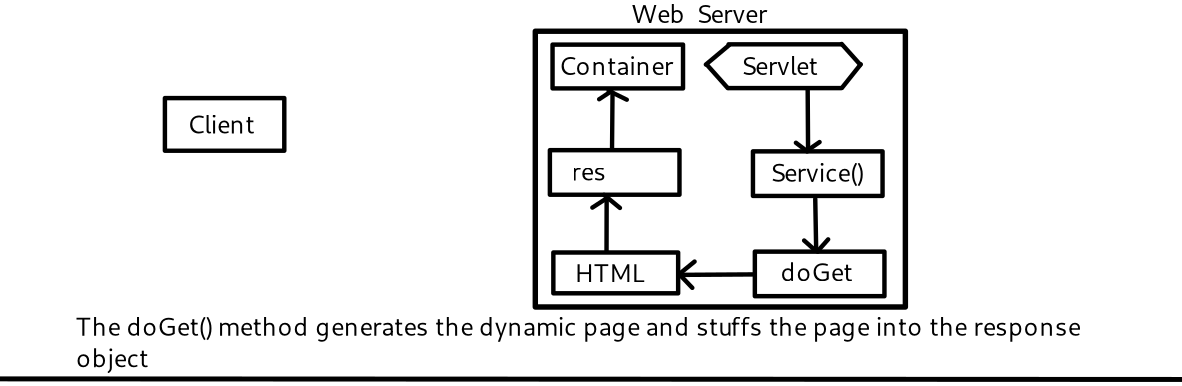
**Step 3:**



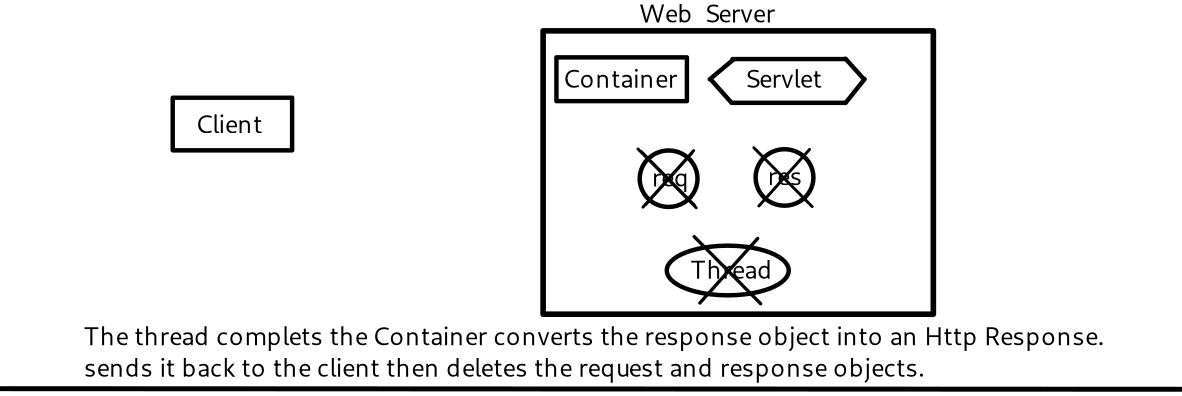
**Step 4:**



**Step 5:**



**Step 6:**



**Programe 3: HttpServlet, overriding actual service**

**public/ActualService.java**

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

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

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

**import** java.sql.\*;

**public** **class** **ActualService** **extends** **HttpServlet**{

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

        super.service(req,res);

        res.setContentType("text/html");

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

        out.println("<html><body>");

        out.println("Actual service");

    }

**public** **void** **service**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

        out.println("<html><body>");

        out.println("Duplicate service");

    }

}

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>ActualService</servlet-name>

        <servlet-class>ActualService</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>ActualService</servlet-name>

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

    </servlet-mapping>

</web-app>

**Programe 4: HttpServlet, overriding duplicate service**

**public/DuplicateService.java**

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

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

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

**import** java.sql.\*;

**public** **class** **DuplicateService** **extends** **HttpServlet**{

**public** **void** **service**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        super.service(req,res);

        res.setContentType("text/html");

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

        out.println("<html><body>");

        out.println("Duplicate service");

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

    }

**public** **void** **doGet**(HttpServletRequest req,HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

        out.println("<html><body>");

        out.println("Get Method");

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

    } }

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>DuplicateService</servlet-name>

        <servlet-class>DuplicateService</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>DuplicateService</servlet-name>

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

    </servlet-mapping>

</web-app>

**Programe 5: RequestServlet, login page**

**public/RequestServlet.java**

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

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

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

**import** java.sql.\*;

**public** **class** **RequestServlet** **extends** **HttpServlet**{

**public** **void** **service**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

        out.println("<html><body>");

**String** name = req.getParameter("name");

**String** pass = req.getParameter("pass");

**if**(name.equals("Codesquadz") && pass.equals("Codesquadz")){

            out.println("User is valid through service method");

        }

**else**{

            out.println("User is post invalid");

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

        }

    }

**public** **void** **doGet**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

        out.println("<html><body>");

**String** name = req.getParameter("name");

**String** pass = req.getParameter("pass");

**if**(name.equals("Codesquadz") && pass.equals("Codesquadz")){

            out.println("User is valid through get");

        }

**else**{

            out.println("User is invalid through get");

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

        }

    }

}

**public/login.html**

<html>

<body>

    <form action="login" method="get">

        Enter the name: <input type="TEXT" name="name">

        <br>

        Enter the password: <input type="password" name="pass">

        <br>

        <input type="SUBMIT">

    </form>

</body></html>

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>RequestServlet</servlet-name>

        <servlet-class>RequestServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>RequestServlet</servlet-name>

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

    </servlet-mapping>

</web-app>

**public/WEB-INF/lib/ojdbc14.jar**

Copy the jar file to lib folder so the web container can use it. We are copying because server

does not have ojdbc.jar file

**Programe 6: RequestServlet, login page with database**

**public/RequestServlet.java**

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

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

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

**import** java.sql.\*;

**public** **class** **RequestServlet** **extends** **HttpServlet**{

**public** **void** **doPost**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

        out.println("<html><body>");

**String** name = req.getParameter("name");

**String** pass = req.getParameter("pass");

**try**{

            Class.forName("oracle.jdbc.driver.OracleDriver");

**Connection** c = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","System",

"Oracle10g");

**Statement** s = c.createStatement();

**int** x = s.executeUpdate("insert into emp101 values('"+name+"','"+pass+"')");

**ResultSet** rs = s.executeQuery("select \* from emp101");

**ResultSetMetaData** rsmd = rs.getMetaData();

            out.println("<table bgcolor='yellow' border=1 width=200>");

            out.println("<tr>");

**for**(**int** i=1;i<=rsmd.getColumnCount(); i++){

                out.println("<th>"+rsmd.getColumnName(i)+"</th>");

            }

**while**(rs.next()){

                out.println("<tr>");

                out.println("<td>"+rs.getString(1)+"</td>");

                out.println("<td>"+rs.getString(2)+"</td>");

                out.println("</tr>");

            }

        }

**catch**(Exception e){}

        out.println("</table>");

**if**(name.equals("Codesquadz") && pass.equals("Codesquadz")){

            out.println("User is post valid");

        }

**else**{

            out.println("User is post invalid");

        }

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

    }

}

**public/login.html**

<html>

<body>

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

        Enter the name: <input type="TEXT" name="name">

        <br>

        Enter the password: <input type="password" name="pass">

        <br>

        <input type="SUBMIT">

    </form>

</body>

</html>

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>RequestServlet</servlet-name>

        <servlet-class>RequestServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>RequestServlet</servlet-name>

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

    </servlet-mapping>

</web-app>

**Programe 7: calling get and post from duplicate service**

**public/RequestServlet.java**

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

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

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

**import** java.sql.\*;

**public** **class** **RequestServlet** **extends** **HttpServlet**{

**public** **void** **service**(HttpServletRequest req,HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

        out.println("<html><body>");

**String** s = req.getMethod();

        out.println(s);

**if**(s.equals("POST")){

            doPost(req,res);

        }

**if**(s.equals("GET")){

            doGet(req,res);

        }

    }

**public** **void** **doGet**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

        out.println("<html><body>");

**String** name = req.getParameter("name");

**String** pass = req.getParameter("pass");

**if**(name.equals("Codesquadz") && pass.equals("Codesquadz")){

            out.println("User is valid through get");

        }

**else**{

            out.println("User is invalid through get");

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

        }

    }

**public** **void** **doPost**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

        out.println("<html><body>");

**String** name = req.getParameter("name");

**String** pass = req.getParameter("pass");

**if**(name.equals("Codesquadz") && pass.equals("Codesquadz")){

            out.println("User is post valid");

        }

**else**{

            out.println("User is post invalid");

        }

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

    }

}

**public/login.html**

<html>

<body>

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

        Enter the name: <input type="TEXT" name="name">

        <br>

        Enter the password: <input type="password" name="pass">

        <br>

        <input type="SUBMIT">

    </form>

</body>

</html>

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>RequestServlet</servlet-name>

        <servlet-class>RequestServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>RequestServlet</servlet-name>

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

    </servlet-mapping>

</web-app>

**Programe 8: verifying user from database**

**public/LoginServlet.java**

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

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

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

**import** java.sql.\*;

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

**public** **void** **service**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

        out.println("<html><body>");

**String** name = req.getParameter("name");

**String** pass = req.getParameter("pass");

**try**{

            Class.forName("oracle.jdbc.driver.OracleDriver");

**Connection** c = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","System",

"Oracle10g");

**Statement** s = c.createStatement();

**String** s1 = "select \* from emp101 where name='"+name+"' and pass='"+pass+"'";

**ResultSet** rs = s.executeQuery(s1);

**if**(rs.next()){

                out.println("User is Valid");

            }

**else**{

                out.println("User is Invalid");

            }

        }

**catch**(Exception e){

            out.println(e);

        }

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

    }

}

**public/login.html**

<html>

<body>

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

        Enter the name: <input type="TEXT" name="name">

        <br>

        Enter the password: <input type="PASSWORD" name="pass">

        <br>

        <input type="SUBMIT">

    </form>

</body>

</html>

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>LoginServlet</servlet-name>

        <servlet-class>LoginServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>LoginServlet</servlet-name>

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

    </servlet-mapping>

</web-app>

**Programe 9: proper form**

**public/FormServlet.java**

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

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

**import** java.io.IOException;

**public** **class** **FormServlet** **extends** **HttpServlet**{

**public** **void** **doPost**(HttpServletRequest req,HttpServletResponse res)**throws** ServletException,

IOException{

**ServletOutputStream** out = res.getOutputStream();

        res.setContentType("text/html");

        out.println("<html><head><title>Basic Form Processor Output</title><head>");

        out.println("<body>");

        out.println("<h1>Here is your Form Data</h1>");

*//extract the form data here*

**String** title = req.getParameter("title");

**String** name = req.getParameter("name");

**String** city = req.getParameter("city");

**String** country = req.getParameter("country");

**String** tel = req.getParameter("tel");

**String** age = req.getParameter("age");

*//extracting data from the checkbox field*

        String[] interests = req.getParameterValues("interests");

*//output the data into a web page*

        out.println("Your title is "+name);

        out.println("<br> Your name is "+name);

        out.println("<br> Your city is "+city);

        out.println("<br> Your country is "+country);

        out.println("<br> Your tel is "+tel);

        out.println("<br> Your interests include<ul>");

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

            out.println("<li>"+interests[i]+"</li>");

        }

        out.println("</ul>");

        out.println("<br>Your age is "+age);

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

    }

}

**public/form.html**

<html>

<head>

</head>

<body>

    <h1>Please enter your information</h1>

    <form method="POST" action="login">

        Title: <select size="1" name="title">

                    <option>Mr</option>

                    <option>Mrs</option>

                    <option>Miss</option>

                    <option>Ms</option>

                    <option>Other</option>

                </select><br>

        Name: <input type="text" name="name" size="20"><br>

        City: <input type="text" name="city" size="20"><br>

        Country: <input type="text" name="country" size="20"><br>

        Telephone: <input type="text" name="tel" size="20">

        <p>Please inform us of your interests:<br>

            <input type="checkbox" name="interests" value="Sport">Sport<br>

            <input type="checkbox" name="interests" value="Music">Music<br>

            <input type="checkbox" name="interests" value="Reading">Reading<br>

            <input type="checkbox" name="interests" value="TV and Film">TV and Film

        </p>

        <p>Your age

            <input type="radio" name="age" value="25orless" checked>Less than 25

            <input type="radio" name="age" value="26to40"> 26-40

            <input type="radio" name="age" value="41to65"> 41-65

            <input type="radio" name="age" value="over65"> Over 65

        </p>

        <p>

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

        </p>

    </form>

</body>

</html>

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>FormServlet</servlet-name>

        <servlet-class>FormServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>FormServlet</servlet-name>

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

    </servlet-mapping>

</web-app>

**Programe 10: getting parameters using enumeration**

**public/AllPServlet.java**

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

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

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

**import** java.sql.\*;

**import** java.util.\*;

**public** **class** **AllPServlet** **extends** **HttpServlet**{

**public** **void** **service**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

        out.println("<html><body>");

**Enumeration** e = req.getParameterNames();

**while**(e.hasMoreElements()){

**String** name = (String)e.nextElement();

**String** value = req.getParameter(name);

            out.println(name+" = "+value);

            out.println("<br>");

        }

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

    }

}

**public/form.html**

<html>

<head>

</head>

<body>

    <h1>Please enter your information</h1>

    <form method="POST" action="all">

        Title: <select size="1" name="title">

                    <option>Mr</option>

                    <option>Mrs</option>

                    <option>Miss</option>

                    <option>Ms</option>

                    <option>Other</option>

                </select><br>

        Name: <input type="text" name="name" size="20"><br>

        City: <input type="text" name="city" size="20"><br>

        Country: <input type="text" name="country" size="20"><br>

        Telephone: <input type="text" name="tel" size="20">

        <p>Please inform us of your interests:<br>

            <input type="checkbox" name="interests" value="Sport">Sport<br>

            <input type="checkbox" name="interests" value="Music">Music<br>

            <input type="checkbox" name="interests" value="Reading">Reading<br>

            <input type="checkbox" name="interests" value="TV and Film">TV and Film

        </p>

        <p>Your age

            <input type="radio" name="age" value="25orless" checked>Less than 25

            <input type="radio" name="age" value="26to40"> 26-40

            <input type="radio" name="age" value="41to65"> 41-65

            <input type="radio" name="age" value="over65"> Over 65

        </p>

        <p>

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

        </p>

    </form>

</body>

</html>

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>AllPServlet</servlet-name>

        <servlet-class>AllPServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>AllPServlet</servlet-name>

        <url-pattern>/all</url-pattern>

    </servlet-mapping>

</web-app>

**Programe 11: servlet via annotation**

**public/MyAnnotationServlet.java**

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

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

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

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

**@WebServlet(name = "MyAnnotationServlet", urlPatterns = {"/hello"})**

**public** **class** **MyAnnotationServlet** **extends** **HttpServlet**{

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

        response.setContentType("text/html");

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

        out.println("<h2>Hello World Servlet Annotation Example</h2>");

        out.close();

    }

}

**ServletConfig interface**

We had done the connectivity with database, suppose we have created a product sold to a client, but client wants to change database. For client requirement we have to change driver name, database url username, password in each class of java which will connect to database. A data that you don’t want to hard code then you can do this by supplying the data via web.xml. ServletConfig interface is given to fetching data from web.xml. getInitMethod()

Override the init method of Servlet interface

**ServletConfig Object ->**

**1.** One ServletConfig object per servlet.

**2.** Use it to pass deploy time information to servlet that you do not want to hard code into the servlet.

**3.** Parameters are configured in the deployment descriptor(web.xml).

**4.** ServletConfig object always holds reference to ServletContext for the servlet

**Programe 12: ConfigServlet, sending data via web.xml**

**public/ConfigServlet.java**

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

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

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

**public** **class** **ConfigServlet** **extends** **HttpServlet**{

    ServletConfig con;

**public** **void** **service**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        con=getServletConfig();

*//Enumeration e = con.getInitParameterNames();*

**String** image = con.getInitParameter("image");

**String** driver = con.getInitParameter("driver");

*//String image = getInitParameter("image");*

*//String driver = getInitParameter("driver");*

        res.setContentType("text/html");

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

        out.println("<html><body background = "+image+">");

        out.println(driver);

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

    }

}

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>ConfigServlet</servlet-name>

        <servlet-class>ConfigServlet</servlet-class>

        <init-param>

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

            <param-value>sun.jdbc.odbc.JdbcOdbcDriver</param-value>

        </init-param>

        <init-param>

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

            <param-value>linux.jpg</param-value>

        </init-param>

    </servlet>

    <servlet-mapping>

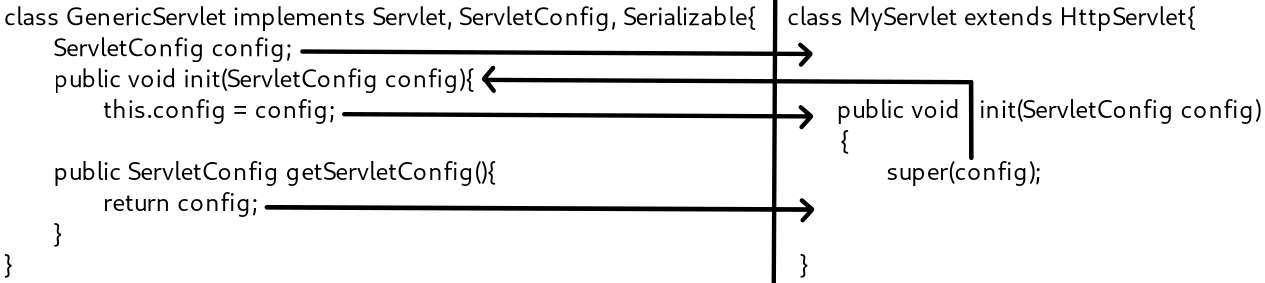
        <servlet-name>ConfigServlet</servlet-name>

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

    </servlet-mapping>

</web-app>

**Working of ServletConfig**



**Original GenericServlet class**

**package** javax.servlet;

**import** java.io.IOException;

**import** java.io.Serializable;

**import** java.util.Enumeration;

**public** **abstract** **class** **GenericServlet** **implements** **Servlet**, ServletConfig, Serializable{

**public** **GenericServlet**(){

    }

**public** **void** **destroy**(){

    }

**public** String **getInitParameter**(String name){

**return** getServletConfig().getInitParameter(name);

    }

**public** Enumeration **getInitParamterNames**(){

**return** getServletConfig().getInitParameterNames();

    }

**public** ServletConfig **getServletConfig**(){

**return** config;

    }

**public** ServletContext **getServletContext**(){

**return** getServletConfig().getServletContext();

    }

**public** String **getServletInfo**(){

**return** "";

    }

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

        this.config = config;

        init();

    }

**public** **void** **log**(String msg){

        getServletContext().log((**new** **StringBuilder**()).append(getServletName()).append(": ").append(msg).toString());

    }

**public** **void** **log**(String message, Throwable t){

        getServletContext().log((**new** **StringBuilder**()).append(getServletName()).appen(": ").appen(message).toString(),t);

    }

**public** **abstract** **void** **service**(ServletRequest servletrequest, ServletResponse servletresponse)**throws** ServletException, IOException:

**public** String **getServetName**(){

**return** config.getServletName();

    }

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

**private** **transient** ServletConfig config;

}

**Programe 13: ServletConfig via annotation**

**public/ServletInitParamDemo.java**

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** javax.servlet.ServletConfig;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebInitParam;

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

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

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

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

**@WebServlet(**

    urlPatterns = {"/initparam"},

    initParams = {

        @WebInitParam(name = "email", value = "abc@gmail.com"),

        @WebInitParam(name = "phone", value = "123456")

    }

)

**public** **class** **ServletInitParamDemo** **extends** **HttpServlet**{

**private** String email= "", phone = "";

**public** **ServletInitParamDemo**(){

        super();

    }

**@Override**

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

        super.init(config);

        email = config.getInitParameter("email");

        phone = config.getInitParameter("phone");

    }

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

ServletException,IOException{

        response.setContentType("text/html");

**String** html = "<h2>Access data using @WebInitParam</h2>";

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

        html += "<h3>Email : "+email+"<br>Phone no. : "+phone+"</h3>";

        out.println(html);

    }

}

**ServletContext object->**

**1.** When we deploy the application(.war) into the web server(Container) then Container one

ServletContext Object for that we application

**2.** ServletContext parameters work just like init parameters except ServletContext parameters are

available to the entire web application not just a single servlet. So that means any servlet and jsp in

the application automatically has access to the ServletContext parameters.

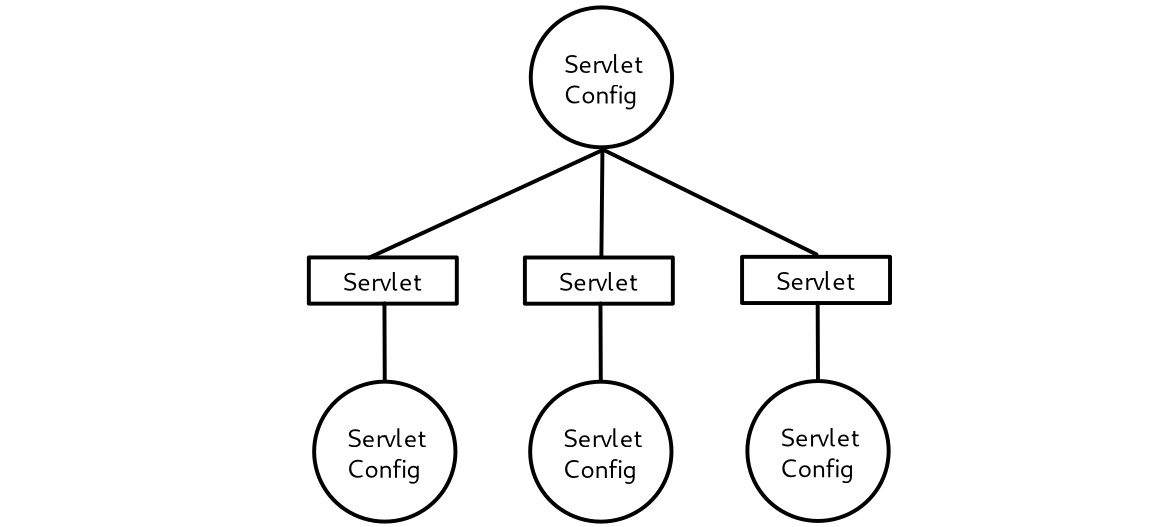
**3.** One ServletContext per web application

**4.** Parameters are also configured in web.xml

**5.** Use it as kind of bulletin board where you can put up message that other parts of application can

access.

Whenever you want combination of annotation and web.xml you have to include schema or DTD file in your web.xml



**Programe 14: servlet context**

**public/OracleServlet.java**

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

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

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

**import** java.sql.\*;

**public** **class** **OracleServlet** **extends** **HttpServlet**{

**public** **void** **service**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

**ServletContext** ctx = getServletContext();

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

        out.println("<html><body>");

*//Enumeration e = ctx.getInitParameterNames();*

**String** driver = ctx.getInitParameter("driver");

**String** url = ctx.getInitParameter("url");

**String** user = ctx.getInitParameter("user");

**String** pass = ctx.getInitParameter("pass");

**try**{

            Class.forName(driver);

**Connection** c = DriverManager.getConnection(url,user,pass);

**Statement** s = c.createStatement();

**ResultSet** rs = s.executeQuery("select \* from emp101");

**while**(rs.next()){

                out.println(rs.getString(1));

                out.println(rs.getString(2));

                out.println("<br>");

            }

        }

**catch**(Exception e){

        }

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

    }

}

**public/WEB-INF/web.xml**

<web-app>

    <context-param>

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

        <param-value>oracle.jdbc.driver.OracleDriver</param-value>

    </context-param>

    <context-param>

        <param-name>url</param-name>

        <param-value>jdbc:oracle:thin:**@localhost**:1521:xe</param-value>

    </context-param>

    <context-param>

        <param-name>user</param-name>

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

    </context-param>

    <context-param>

        <param-name>pass</param-name>

        <param-value>Oracle10g</param-value>

    </context-param>

    <servlet>

        <servlet-name>OracleServlet</servlet-name>

        <servlet-class>OracleServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>OracleServlet</servlet-name>

        <url-pattern>/oracle</url-pattern>

    </servlet-mapping>

</web-app>

**Programe 15: servlet context via annotation and web.xml**

**public/OracleServlet.java**

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

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

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

**import** java.sql.\*;

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

**@WebServlet(name = "OracleServlet" ,urlPatterns = {"/oracle"})**

**public** **class** **OracleServlet** **extends** **HttpServlet**{

**public** **void** **service**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

**ServletContext** ctx = getServletContext();

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

        out.println("<html><body>");

*//Enumeration e = ctx.getInitParameterNames();*

**String** driver = ctx.getInitParameter("driver");

**String** url = ctx.getInitParameter("url");

**String** user = ctx.getInitParameter("user");

**String** pass = ctx.getInitParameter("pass");

**try**{

            Class.forName(driver);

**Connection** c = DriverManager.getConnection(url,user,pass);

**Statement** s = c.createStatement();

**ResultSet** rs = s.executeQuery("select \* from emp101");

**while**(rs.next()){

                out.println(rs.getString(1));

                out.println(rs.getString(2));

                out.println("<br>");

            }

        }

**catch**(Exception e){

        }

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

    }

}

**public/WEB-INF/web.xml**

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http:/java.sun.com/xml/ns/javaee" xsi:schemaLocation="http://java.sun.com/xml/ns/javaeehttp://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd" version="3.0">

    <context-param>

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

        <param-value>oracle.jdbc.driver.OracleDriver</param-value>

    </context-param>

    <context-param>

        <param-name>url</param-name>

        <param-value>jdbc:oracle:thin:**@localhost**:1521:xe</param-value>

    </context-param>

    <context-param>

        <param-name>user</param-name>

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

    </context-param>

    <context-param>

        <param-name>pass</param-name>

        <param-value>Oracle10g</param-value>

    </context-param>

</web-app>

**Servlet Collaboration**

Servlet Collaboration means calling a servlet from another servlet. In past this was achieved via Servlet

Chaining. Container creates chain of all servlet and we ask to web server first run this servlet then that and then that servlet. But our requirement is different we want to take username & password and verify on a servlet if user is valid than we call welcome servlet from this servlet if not than not valid

But the Servlet Chaining concept is removed so we have two concepts instead of Servlet chaining first sendRedirect method of ServletResponse interface second is RequestDispatcher interface

In servlet chaining we create multiple servlets and set the sequence in server to execute first this, then this and then this. Now to the requirement is to take username and password from user if the user is valid then call the welcome servlet and if the user is invalid then call err servlet but this can not be done using servlet chaining.

**Rules for sendRedirect() Method**

**1.** Whatever URL u pass into the parentheses of sendRedirect() method that URL will be pasted into the

address bar of the browser by this method and ask the browser to send the fresh request of this url.

**2.** In case of sendRedirect() method URL of the browser will be changed.

**3.** In case of sendRedirect() method always the fresh request is generated.

**4.** Using sendRedirect() method you can not share the data via ServletRequest Object.

**5.** Using sendRedirect() method you can go beyond the one context that you can call the web component

(servlet,jsp etc) of another web application.

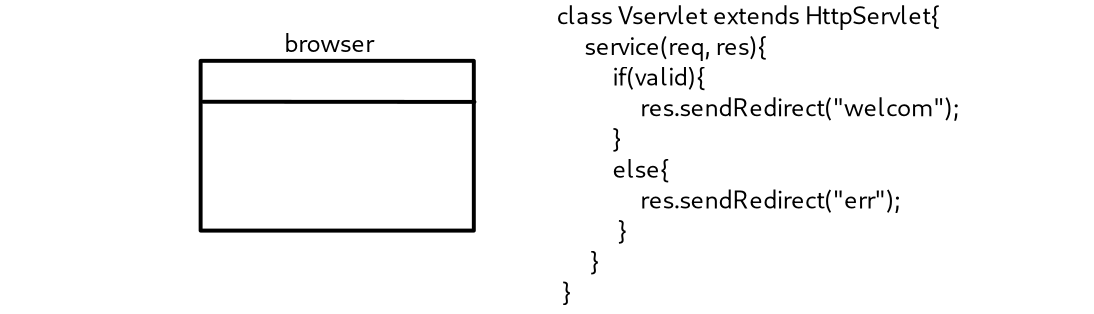
**6.** Never write any statement after the sendRedirect() method.

**7.** Never write any printing statement before the sendRedirect() method.

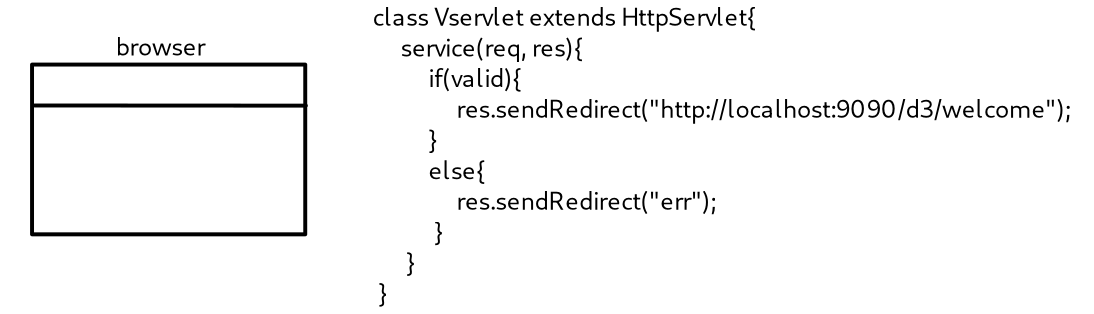
**8.** Called servlet must be having a either a doGet() method or service() method it must not be having a

doPost() method.

**Working of sendRedirect**



sendRedirect method, I am a powerful method. I can call web content(servlet, jsp or html) of same application and if you want to call web content of another application, i can also do that. For example, i have installed two war file in server s1 and s2, s1 is a validate servlet and once validation is completed then call the welcome page of s2 or error page of s2. Even you can put the s2 war file in tomcat and call it from weblogic



**What it(sendRedirect) does behind the screen**

Nothing!. It just paste the given url in browser ask to browser send its fresh request

**Programe 16: Servlet collaboration same application**

**public/SendRedirect.java**

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

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

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

**import** java.sql.\*;

**public** **class** **SendRedirect** **extends** **HttpServlet**{

**public** **void** **service**(HttpServletRequest req,HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

        out.println("<html><body>");

        out.println("before");

**String** name = req.getParameter("name");

        String pass=req.getParameter("pass");

**try**{

            Class.forName("oracle.jdbc.driver.OracleDriver");

**Connection** c = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system",

"Oracle10g");

**Statement** s = c.createStatement();

**String** s1 = "select \* from emp101 where name='"+name+"' and pass='"+pass+"'";

**ResultSet** rs = s.executeQuery(s1);

**if**(rs.next()){

*//res.sendRedirec("wel.html");*

                res.sendRedirect("wel");

*//res.sendRedirect("http://google.com");*

*//res.sendRedirec("http://localhost:9090/tom17/wel");*

            }

**else**{

                res.sendRedirect("err");

            }

        }

**catch**(Exception e){

            out.println(e);

        }

        out.println("after");

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

    }

}

**public/WelServlet.java**

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

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

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

**public** **class** **WelServlet** **extends** **HttpServlet**{

**public** **void** **service**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

        out.println("<html><body>");

        out.println("welcome to Codesquadz web");

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

    }

}

**public/ErrServlet.java**

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

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

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

**public** **class** **ErrServlet** **extends** **GenericServlet**{

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

        res.setContentType("text/html");

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

        out.println("<html><body>");

        out.println("bavari pooch invalid user");

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

    }

}

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>SendRedirect</servlet-name>

        <servlet-class>SendRedirect</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>SendRedirect</servlet-name>

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

    </servlet-mapping>

    <servlet>

        <servlet-name>WelServlet</servlet-name>

        <servlet-class>WelServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>WelServlet</servlet-name>

        <url-pattern>/wel</url-pattern>

    </servlet-mapping>

    <servlet>

        <servlet-name>ErrServlet</servlet-name>

        <servlet-class>ErrServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>ErrServlet</servlet-name>

        <url-pattern>/err</url-pattern>

    </servlet-mapping>

</web-app>

**Programe 17: servlet collaboration diff application**

**public/SendRedirect.java[In weblogic]**

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

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

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

**import** java.sql.\*;

**public** **class** **SendRedirect** **extends** **HttpServlet**{

**public** **void** **service**(HttpServletRequest req,HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

        out.println("<html><body>");

        out.println("before");

**String** name = req.getParameter("name");

        String pass = req.getParameter("pass");

**try**{

            Class.forName("oracle.jdbc.driver.OracleDriver");

**Connection** c = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system",

"Oracle10g");

**Statement** s = c.createStatement();

**String** s1 = "select \* from emp101 where name='"+name+"' and pass='"+pass+"'";

**ResultSet** rs = s.executeQuery(s1);

**if**(rs.next()){

                res.sendRedirect("http://localhost:9090/tom17/wel");

            }

**else**{

                res.sendRedirect("err");

            }

        }

**catch**(Exception e){

            out.println(e);

        }

        out.println("after");

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

    }

}

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>SendRedirect</servlet-name>

        <servlet-class>SendRedirect</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>SendRedirect</servlet-name>

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

    </servlet-mapping>

</web-app>

**public/WelServlet.java[tomcat]**

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

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

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

**public** **class** **WelServlet** **extends** **HttpServlet**{

**public** **void** **service**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

        out.println("<html><body>");

        out.println("welcome to Codesquadz web via Tomcat");

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

    }

}

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>WelServlet</servlet-name>

        <servlet-class>WelServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>WelServlet</servlet-name>

        <url-pattern>/wel</url-pattern>

    </servlet-mapping>

</web-app>

**Request dispatcher**

public void service(ServletRequest req, ServletResponse res){

RequestDispatcher rd = req.getRequestDispatcher(“wel”); //.jsp or .html

rd.include(req,res);

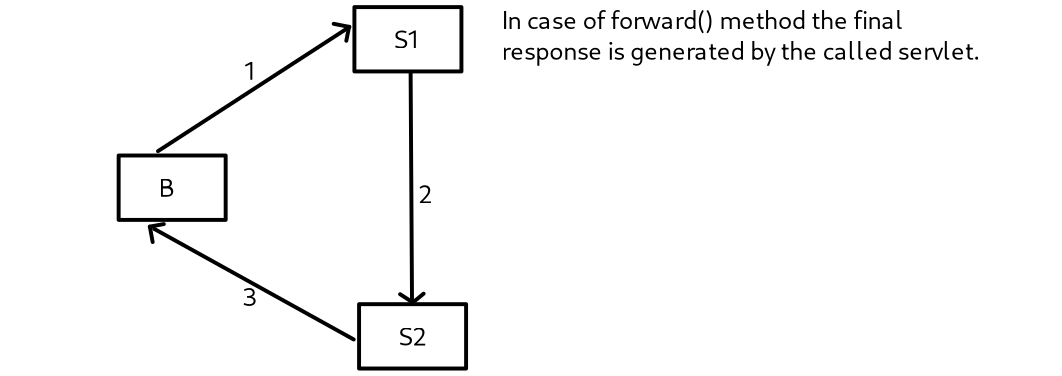
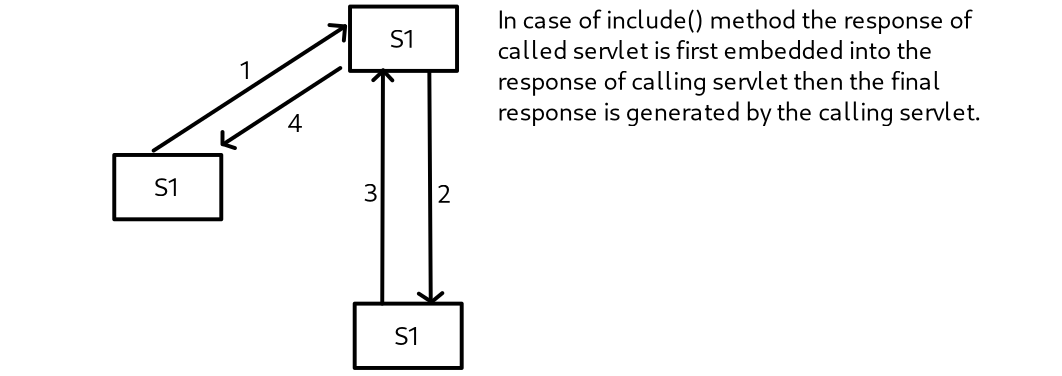
//OR

rd.forward(req,res);

}



**Difference between include and forward**



**1.** In case of RequestDispatcher interface URL of the browser will not be changed.

**2.** In case of RequestDispatcher interface no new request is going to be generated same request and

response are forwarded to the called servlet.

**3.** Using a RequestDispatcher interface you can share the data via ServletRequest Object.

**4.** Using RequestDispatcher interface you can not go beyond the one context that means you can not call

the web compenent(servlet, jsp etc) of another web application.

**Programe 18: RequestDispatcher, include and forward method**

**public/RdServlet.java**

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

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

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

**import** java.sql.\*;

**public** **class** **RdServlet** **extends** **HttpServlet**{

**public** **void** **service**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

        req.setAttribute("name","Codesquadz");

**String** name = req.getParameter("name");

**String** pass = req.getParameter("pass");

**try**{

            Class.forName("oracle.jdbc.driver.OracleDriver");

**Connection** c = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","System",

"Oracle10g");

**Statement** s = c.createStatement();

**String** s1 = "select \* from emp101 where name='"+name+"' and pass='"+pass+"'";

**ResultSet** rs = s.executeQuery(s1);

            out.println("before dispatcher");

            out.println("<br>");

**if**(rs.next()){

**RequestDispatcher** rd = req.getRequestDispatcher("/wel");

*//rd.include(req,res);*

                rd.forward(req,res);

            }

**else**{

*//RequestDispatcher rd = req.getRequestDispather("/err");*

*//rd.include(req,res);*

*//rd.forward(req,res);*

            }

        }

**catch**(Exception e){

            out.println(e);

        }

        out.println("<br>");

        out.println("after requestDispatcher");

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

    }

}

**public/WelServlet.java**

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

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

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

**public** **class** **WelServlet** **extends** **HttpServlet**{

**public** **void** **doPost**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

        out.println("<html><body>");

**String** s = (String) req.getAttribute("name");

        out.println("welcome to MyWeb of Codesquadz ="+s);

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

    }

}

**public/login.html**

<html>

<body>

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

        Enter the name: <input type="TEXT" name="name">

        <BR>

        Enter the password: <input type="PASSWORD" name="pass">

        <br>

        <input type="SUBMIT">

    </form>

</body>

</html>

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>RdServlet</servlet-name>

        <servlet-class>RdServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>RdServlet</servlet-name>

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

    </servlet-mapping>

    <servlet>

        <servlet-name>WelServlet</servlet-name>

        <servlet-class>WelServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>WelServlet</servlet-name>

        <url-pattern>/wel</url-pattern>

    </servlet-mapping>

</web-app>

**Attributes:**

We have learned how to call a servlet from another servlet within the same application and within the different application. Now we need is to send dynamic data from one servlet to another servlet, If the data is static and you want to share to all servlet it can be done using servlet context. Static data means which is predecided or compile time, But the requirement is dynamic. For example, the client make the servlet request and we did some calculation on behalf of client’s request then got the some result now we want to send the result on another servlet and the servlet will print the result in formatted manner

Assignment: emicalcultor.com

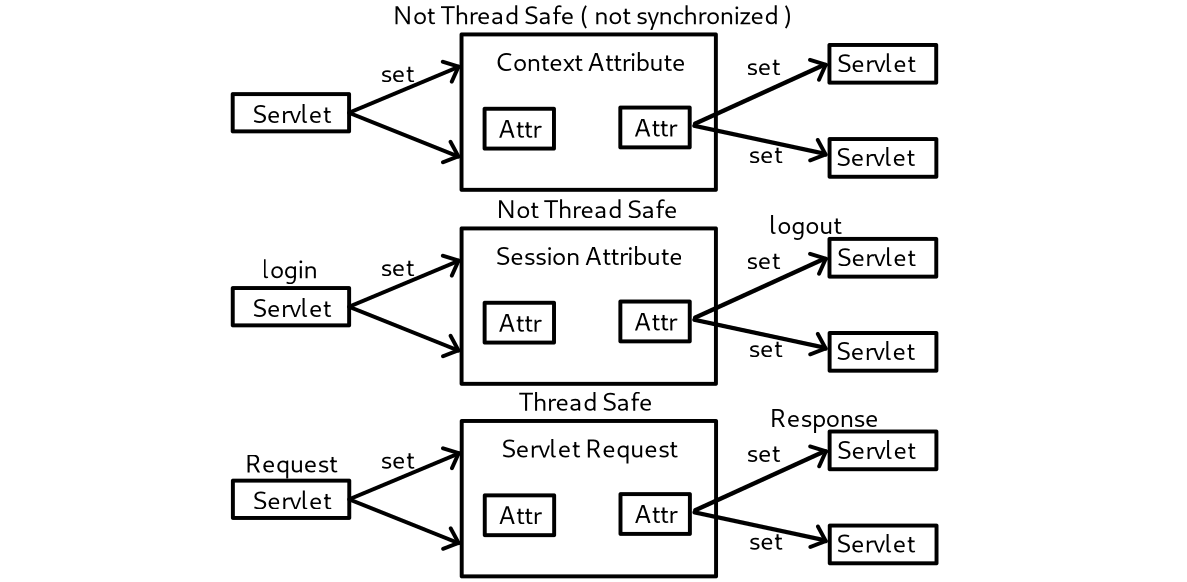
If you want to send data to another servlet you have put that in scope. Scope means how long you want to keep the data in your application, there are three types of scope ServletRequest, Session, and ServletContext. Session mean the time between login to logout. Last is ServletContext data will remain until application is closed.

There are three common methods to set , get and remove attribute in all scope interfaces

**1.** public void setAttribute(String name, Object value)

**2.** public Object getAttribute(String name)

**3.** public void removeAttribute(String name)



**Programe 19: Context attribute, count server hits**

**public/ContextAttribute.java**

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

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

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

**public** **class** **ContextAttribute** **extends** **HttpServlet**{

    PrintWriter out;

**public** **void** **service**(HttpServletRequest req,HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

        out = res.getWriter();

**ServletContext** ctx = getServletContext();

**Integer** count = (Integer)ctx.getAttribute("count");

**if**(count == null){

            count = **new** **Integer**(0);

        }

        count = **new** **Integer**(count.intValue()+1);

        ctx.setAttribute("count",count);

        out.println(count.intValue());

    }

}

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>ContextAttribute</servlet-name>

        <servlet-class>ContextAttribute</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>ContextAttribute</servlet-name>

        <url-pattern>/att</url-pattern>

    </servlet-mapping>

</web-app>

**HTTP status codes**

Status codes are used by the client application to know the results of a request processing in the server. The status code has 3 digit numbers, where the first digit defines the class of the response.

**The status code are of 5 categories namely:**

|  |  |
| --- | --- |
| **Informational codes ->**  **Code Message**  100 Continue  101 Switching Protocols | **Success code ->**  **Code Message**  200 OK  201 Created  202 Accepted  203 Non-Authoritative Information  204 No Content  205 Reset Content  206 Partial Content |
| **Redirection code ->**  **Code Message**  300 Multiple Choices  301 Moved Permanently  302 Found  304 See Other  305 Use Proxy  307 Temporary Redirect  **Server Error code ->**  **Code Message**  500 Internal Server Error  501 Not Implemented  502 Bad Gateway  503 Service Unavailable  504 Gateway Time-Out  504 HTTP Version not supported | **Client Error code ->**  **Code Message**  400 Bad Request  401 Unauthorized  402 Payment Required  403 Forbidden  404 Not Found  405 Methods Not Allowed  406 Not Acceptable  407 Proxy Authentication Required  408 Request Time-Out  409 Conflict  410 Gone  411 Length Required  412 Precondition Failed  413 Request Entity Too Large  414 Request-URI Too Large  415 Unsupported Media Type  416 Requested range not satisfy able  417 Exceptional Failed |

**Programe 20: sending error code to browser**

**public/ErrorcodeExample.java**

**import** java.io.IOException;

**import** javax.servlet.ServletException;

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

**public** **class** **ErrorcodeExample** **extends** **HttpServlet**{

**public** **void** **doGet**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.sendError(500);

    }

}

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>ErrorcodeExample</servlet-name>

        <servlet-class>ErrorcodeExample</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>ErrorcodeExample</servlet-name>

        <url-pattern>/ecode</url-pattern>

    </servlet-mapping>

    <error-page>

        <error-code>404</error-code>

        <location>/WEB-INF/pages/404.html</location>

    </error-page>

</web-app>

**public/WEB-INF/pages/404.html**

<html>

<body>

    bavari pooch sahi url enter kar

</body>

</html>

**Programe 21: all headers of a browser**

**public/HeaderServlet.java**

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

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

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

**import** java.util.\*;

**public** **class** **HeaderServlet** **extends** **HttpServlet**{

    PrintWriter out;

**public** **void** **service**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

        out = res.getWriter();

**Enumeration** e = req.getHeaderNames();

**while**(e.hasMoreElements()){

**String** name = (String)e.nextElement();

**String** value = req.getHeader(name);

            out.println(name+" : "+value);

            out.println("<br>");

        }

        out.println("protocol "+req.getProtocol());

        out.println("<br>");

        out.println("method "+req.getMethod());

        out.println("uri "+req.getRequestURI());

        out.println("<br>");

        out.println("Content type "+req.getContentType());

        out.println("<br>");

        out.println("Content length "+req.getContentLength());

    }

}

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>HeaderServlet</servlet-name>

        <servlet-class>HeaderServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>HeaderServlet</servlet-name>

        <url-pattern>/header</url-pattern>

    </servlet-mapping>

</web-app>

**Programe 22: auto refresh**

**public/RefreshServlet.java**

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

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

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

**import** java.util.\*;

**public** **class** **RefreshServlet** **extends** **HttpServlet**{

**static** **int** count = 0;

**public** **void** **doGet**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

*//res.setHeader("Refresh","2");*

        out.println("redirecting in 4 seconds to header servlet");

        res.setHeader("Refresh","4;http://localhost:7001/s21/header");

*//out.println(new Date().toString()+" time "+count++);*

    }

}

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>RefreshServlet</servlet-name>

        <servlet-class>RefreshServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>RefreshServlet</servlet-name>

        <url-pattern>/refresh</url-pattern>

    </servlet-mapping>

</web-app>

**Deploy application in exploded form**

Login to console -> deployments -> localhost -> select root folder in which your WEB-INF and other folder contains -> next -> next -> next -> finish

**Programe 23: send data to excel**

**public/ExcelServlet.java**

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

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

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

**public** **class** **ExcelServlet** **extends** **HttpServlet**{

    PrintWriter out;

**public** **void** **service**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("application/vnd.ms-excel");

        out = res.getWriter();

        out.println("Names\tHindi\tEnlish\tMaths\tTotal");

        out.println("Lalua\t35\t20\t95\t=sum(b2:d2)");

    }

}

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>ExcelServlet</servlet-name>

        <servlet-class>ExcelServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>ExcelServlet</servlet-name>

        <url-pattern>/excel</url-pattern>

    </servlet-mapping>

</web-app>

**Programe 24: send data from database to excel**

**public/ExcelServlet1.java**

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

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

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

**import** java.sql.\*;

**public** **class** **ExcelServlet1** **extends** **HttpServlet**{

**public** **void** **service**(HttpServletRequest req,HttpServletResponse res)**throws** ServletException,

IOException{

**try**{

            Class.forName("oracle.jdbc.driver.OracleDriver");

**Connection** c = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","System",

"Oracle10g");

**Statement** stmt = c.createStatement();

**ResultSet** rs = stmt.executeQuery("select \* from emp101");

**ResultSetMetaData** rd = rs.getMetaData();

            res.setContentType("application/vnd.ms-excel");

            res.setHeader("Content-Disposition","attachment; filename=sample.xls");

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

**int** count = rd.getColumnCount();

**for**(**int** i=1;i<=count;i++){

                out.println(rd.getColumnName(i)+"\t");

            }

            out.println();

**int** x = 2;

**while**(rs.next()){

                out.print(rs.getString(1)+"\t");

                out.print(rs.getString(2)+"\t");

                out.println();

            }

        }

**catch**(Exception e){

            e.printStackTrace();

        }

    }

}

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>ExcelServlet1</servlet-name>

        <servlet-class>ExcelServlet1</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>ExcelServlet1</servlet-name>

        <url-pattern>/excel1</url-pattern>

    </servlet-mapping>

</web-app>

**Programe 25: download image**

**public/ImageServlet.java**

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

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

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

**import** java.util.\*;

**import** java.text.DateFormat;

**public** **class** **ImageServlet** **extends** **HttpServlet**{

**public** **void** **doGet**(HttpServletRequest req,HttpServletResponse res)**throws** ServletException,

IOException{

**String** fileName = "Altron.jpg";

**ServletOutputStream** stream = null;

**BufferedInputStream** buf = null;

        stream = res.getOutputStream();

**String** s1 = getServletContext().getRealPath("/files/bg.png");

**File** doc = **new** **File**(s1);

        res.setContentType("image/jpeg");

        res.addHeader("Content-Disposition","attachment; filename="+fileName);

        res.setContentLength((**int**)doc.length());

**FileInputStream** input = **new** **FileInputStream**(doc);

        buf = **new** **BufferedInputStream**(input);

**int** readBytes = 0;

**while**((readBytes = buf.read())!=-1){

            stream.write(readBytes);

        }

    }

}

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>ImageServlet</servlet-name>

        <servlet-class>ImageServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>ImageServlet</servlet-name>

        <url-pattern>image</url-pattern>

    </servlet-mapping>

</web-app>

**File upload**

As we know that we have servlet output stream and we send the file’s data using that output stream

**Programe 26: upload file using oreilly jar**

**public/FileUploadServlet.java**

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

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

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

**import** com.oreilly.servlet.\*;

**public** **class** **FileUploadServlet** **extends** **GenericServlet**{

**public** **void** **service**(ServletRequest req,SerlvetResponse res){

        res.setContentType("text/html");

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

**String** path = getServletContext().getRealPath("/file");

**MultipartRequest** mpr = **new** **MultipartRequest**(req,path,500\*1024\*1024);

**String** s1 = mpr.getOriginalFileName("file123");

        out.println("<html><body>");

        out.println("File uploaded successfully = "+s1);

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

    }

}

**public/upload.html**

<html>

<body>

    <form action="upload" enctype="multipart/form-data" method="post">

        <input type="file name="file123">

        <input type="submit">

    </form>

<body>

<html>

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>FileUploadServlet</servlet-name>

        <servlet-class>FileUploadServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>FileUploadServlet</servlet-name>

        <url-pattern>/upload</url-pattern>

    </servlet-mapping>

</web-app>

**Programe 27: upload file using java jar**

**public/FileUploadServlet2.java**

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

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

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

**public** **class** **FileUploadServlet2** **extends** **HttpServlet**{

**public** **void** **service**(HttpServletRequest req,HttpServletResponse res)**throws** ServletException,

IOException{

**try**{

            res.setContentType("text/html");

**ServletInputStream** in = req.getInputStream();

**ServletOutputStream** out = res.getOutputStream();

**FileOutputStream** fout = **new** **FileOutputStream**("/home/sonu/Desktop/upload.txt");

**int** i=0;

**while**((i = in.read()) != -1){

                fout.write(i);

                System.out.print((**char**)i);

            }

            fout.close();

            out.println("file uploaded");

        }

**catch**(Exception e){

            System.out.println(e);

        }

    }

}

**pubilc/upload2.html**

<html>

<body>

    <form action="upload2" enctype="multipart/form-data" method="post">

        <input type="file" name="file">

        <input type="submit">

    </form>

</body>

</html>

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>FileUploadServlet2</servlet-name>

        <servlet-class>FileUploadServlet2</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>FileUploadServlet2</servlet-name>

        <url-pattern>/upload2</url-pattern>

    </servlet-mapping>

</web-app>

**Session Tracking**

There are no site that do not use session tracking even you have ever done the online transaction whether using mobile app and web app you have definitely heard a word **session expired**, you will see this specially in bank website. Maximum it takes four to five minute to say “your session has been expired login again” and we will also learn about why almost all of the websites ask for login

**What is session?**

Session means time interval. Session always represents a time interval, where beginning and ending both times are exist. For example, session in college, race session, parliament session, cricket match session. If you are measuring all the tasks that are performed in a particular time interval then that is called session tracking. In web application session is the time between “log in” to “log out”

HTTP is a state less protocol, what is state - current value of the data member of a class. Whatever data or information you are sending from your browser to the server (like header’s value or request parameter’s value) via HTTP protocol that is knows as state of HTTP protocol. What is mean by less – when http protocol reached to the server with some data says that i’ll give some you data and you have to keep this data util the response is not reached the browser after that delete all the data i gave you even it closes the TCP connection and after server do not have any data related t o client like name of browser,IP and etc and if the same client access the other page of that same website so the server will treat as new client

To identify the client that the client is same of or not then we have to give an id to that client known as session id. When a client makes a request then server checks for session id first if client does not have session id, a new id is generated for session and keeps a copy of the id and one copy added to the response and then response goes back to the browser with data and session id. And session id is stored in browser’s cache memory.

Now if you click on other link of the same website with the same browser session id is automatically attached to the request and you have already told the server check for session id and this is server will get the session id

Container will never check for the session id automatically we have force the container to do that using getSession() of HttpServletRequest we can force the container to check session id and if there is no session id then this method will create an object of HttpSession interface

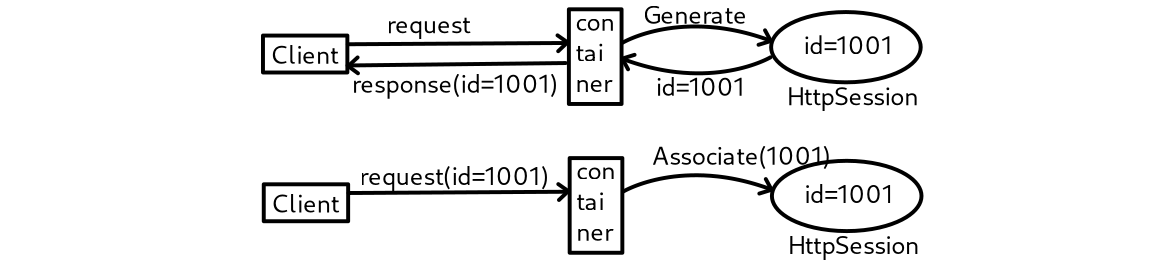
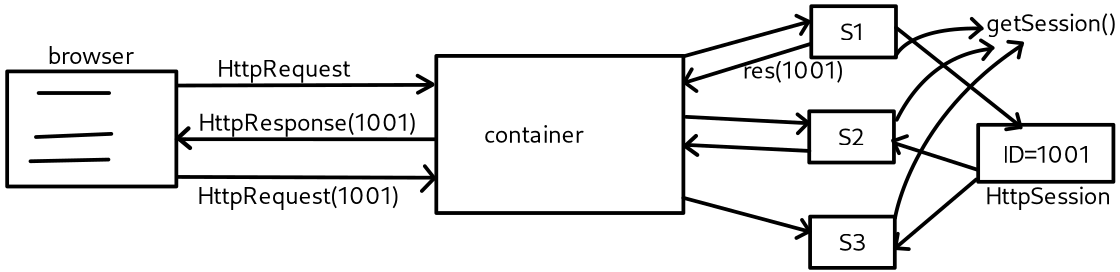
Step 1: generate session id

Step 2: create session object

Step 3: stuff the session id into response

Step 4: return object and give to you in HttpSession interface

Now you will extract the data from request object and set the data in session as a attribute after that session id is stuffed into the response’s object and send to the client now you click on the other link of the same website with the same browser now server will get the session id and extract the old data



**When to create session id and when to destroy session id?**

**Notes of session tracking**

**1.** Session always represents time interval.

**2.** Http protocol is a state less protocol.

**3.** What ever data and information of client you are sending from your browser to webserver is known as

state of Http protocol

**Servlet Session ->** Http is a stateless protocol. A client opens a connection and request some resources or information. The server responds with the requested resource if available or sends error status then close the connection. After closing the connection the server does not remember any information about the client. So the server considers the next request from the same client as fresh request with not remember any information about the client. So the server considers the next request from same client as fresh request with no relation to previous request. This is what makes Http a stateless protocol.

**Session ->** The server should be able to identify that a series of requests from a single client form a single working session.

**State ->** The server should be able to remember information related to previous requests and other business decision that are made for the request.

How does the server know who the client is and the answer is the client needs the unique session id.

The idea is simple on the client’s first request container generates a unique session id and gives it back to the client with response. The client sends back the session id with each subsequent request the container see the matching session and associates the session with the request.

**There are three ways to track session**

**1.** Cookies

**2.** URL rewriting

**3.** Hidden form field

**Cookies ->** **1.** Cookies are special objects which are used to store client’s related information on a client

machine.

**2.** They will be attached to that URL automatically through which they are send to the

browser whenever the client request the same URL.

**HttpSession s = request.getSession();**

Through this method we ask the request for session and container will create or use a session and take care of generating the session id, creating a new Cookie object, stuffing the session id into the cookie and setting the cookie as the part of response. Everything is done by the container.

|  |  |
| --- | --- |
| 1 You do not make the new HttpSession object yourself.  2 You do not generate the unique session id.  3 You do not make the new Cookie object.  4 You do not associate the session id with cookie.  5 You do not set cookie into the response. | HttpSession s = request.getSession();  if(the request includes the session id cookie)  find the session and match the id.  else  create a new session. |

**Programe 28: Session tracking first programe**

**public/LoginSessionServlet.java**

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

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

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

**import** java.sql.\*;

**public** **class** **LoginSessionServlet** **extends** **HttpServlet**{

**public** **void** **service**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

        out.println("<html><body>");

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

        String pass=req.getParameter("pass");

**try**{

            Class.forName("oracle.jdbc.driver.OracleDriver");

**Connection** c = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","System",

"Oracle10g");

**Statement** s = c.createStatement();

**String** s1 = "select \* from emp101 where name='"+name+"' and pass='"+pass+"'";

**ResultSet** rs = s.executeQuery(s1);

**if**(rs.next()){

**HttpSession** se = req.getSession();

                se.setAttribute("name",name);

                se.setAttribute("pass",pass);

                out.println("User is valid "+se.isNew());

                out.println("<br>");

                out.println("<br>");

                out.println("<a href='demo'>demo</a>");

                out.println("<br>");

                out.println("<a href='log'>Logout</a>");

            }

**else**{

                out.println("User is InValid");

            }

        }

**catch**(Exception e){

            out.println(e);

        }

    }

}

**public/DemoServlet1.java**

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

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

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

**public** **class** **DemoServlet1** **extends** **HttpServlet**{

**public** **void** **service**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

**HttpSession** s = req.getSession();

**String** s1 = (String)s.getAttribute("name");

**String** s2 = (String)s.getAttribute("pass");

        out.println("<html><body>");

        out.println(s1+" = "+s2+" "+s.isNew());

        out.println("<br>");

        out.println("<a href='log'>Logout</a>");

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

    }

}

**public/login1.html**

<html>

<body>

    <form action="login" method="get">

        Enter the name: <input type="TEXT" name="name">

        <br>

        Enter the password: <input type="password" name="pass">

        <br>

        <input type="submit">

    </form>

</body>

</html>

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>LoginSessionServlet</servlet-name>

        <servlet-class>LoginSessionServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>LoginSessionServlet</servlet-name>

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

    </servlet-mapping>

    <servlet>

        <servlet-name>DemoServlet1</servlet-name>

        <servlet-class>DemoServlet1</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>DemoServlet1</servlet-name>

        <url-pattern>/demo</url-pattern>

    </servlet-mapping>

    <servlet>

        <servlet-name>Logout</servlet-name>

        <servlet-class>Logout</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>Logout</servlet-name>

        <url-pattern>/log</url-pattern>

    </servlet-mapping>

</web-app>

Now, what if i access demo servlet(http://localhost:7001/s28/demo) directly without login. getSession will be executed and new session will be created because there is no old cookie to check the previous session and which is wrong.If user is being entered through login then we are allowing user to login and creating proper session id but if user bypassed the login page and access other page then new will be created from that page which is wrong.

But user wants to visit your website without login for that we have to allow user to visit our site without login. We can do this by using getSession in all application we have to write getSession(false); it will check for old session except in login page

**Programe 29: session tracking security patch**

**public/LoginSessionServlet.java**

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

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

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

**import** java.sql.\*;

**public** **class** **LoginSessionServlet** **extends** **HttpServlet**{

**public** **void** **service**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

        out.println("<html><body>");

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

        String pass=req.getParameter("pass");

**try**{

            Class.forName("oracle.jdbc.driver.OracleDriver");

**Connection** c = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","System",

"Oracle10g");

**Statement** s = c.createStatement();

**String** s1 = "select \* from emp101 where name='"+name+"' and pass='"+pass+"'";

**ResultSet** rs = s.executeQuery(s1);

**if**(rs.next()){

**HttpSession** se = req.getSession();

                se.setAttribute("name",name);

                se.setAttribute("pass",pass);

                out.println("User is valid "+se.isNew());

                out.println("<br>");

                out.println("<br>");

                out.println("<a href='demo'>demo</a>");

                out.println("<br>");

                out.println("<a href='log'>Logout</a>");

            }

**else**{

                out.println("User is InValid");

            }

        }

**catch**(Exception e){

            out.println(e);

        }

    }

}

**public/DemoServlet.java**

*// public HttpSession getSession(boolean b) if you pass false in above method then it will check existing*

*// or old session if it is found then return that session otherwise it will return null. But it will never create*

*// new session getSession()==getSession(true);*

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

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

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

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

**public** **void** **service**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

**HttpSession** s = req.getSession(false);

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

**String** s1 = (String)s.getAttribute("name");

**String** s2 = (String)s.getAttribute("pass");

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

            out.println(s1+" = "+s2+" "+s.isNew());

            out.println("<br>");

            out.println("<a href='log'>Logout</a>");

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

        }

**else**{

            out.println("bavari pooch login kar le");

        }

    }

}

**public/login1.html**

<html>

<body>

    <form action="login" method="get">

        Enter the name: <input type="TEXT" name="name">

        <br>

        Enter the password: <input type="password" name="pass">

        <br>

        <input type="submit">

    </form>

</body>

</html>

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>LoginSessionServlet</servlet-name>

        <servlet-class>LoginSessionServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>LoginSessionServlet</servlet-name>

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

    </servlet-mapping>

    <servlet>

        <servlet-name>DemoServlet</servlet-name>

        <servlet-class>DemoServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>DemoServlet</servlet-name>

        <url-pattern>/demo</url-pattern>

    </servlet-mapping>

    <servlet>

        <servlet-name>Logout</servlet-name>

        <servlet-class>Logout</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>Logout</servlet-name>

        <url-pattern>/log</url-pattern>

    </servlet-mapping>

</web-app>

**URL rewriting**

Now you can easily find that a user is crawling your website with login or without. All the things have been done using cookies. Now we are going to learn when does cookie fail? And because of that a new concept came to solve this problem named URL rewriting.

Every browser provides a functionality to disable cookies in your browser, If cookies are disabled and your have getSession() method in your servlet then servlet will do all the work related to session tracking. It will create session id, session object, cookie object and stuff that id in cookie and the cookie will definitely come to browser but this time browser won’t accept the cookie.

Now if you click on demo link that link will ask you for login again and again. Here we want to track the session when cookies are disabled

We track the session even cookies are in client’s browser we have ServletResponse’s method encodeURL you can pass any url in that method it will return encoded URL. Now think we have two url and two url needs to call encodeURL method two times think about gmail site that have a lot of URL we need to encode every URL on gmail page and here URL rewriting becomes headache.

If we try to print encodedURL object it will print is string which contains three things

1. Original url of servlet

2. ?jsessionid

3. Alphanumeric value

Like this : demo?jsessionid=<alphanumeric value>

jsessionid=<alphanumeric value> It is known as query string. If you are sending any url from browser so using that url you can send whatever data you want for example:

<http://localhost>:7001/s29/login1.html?name=meesa&pass=meesa&

Using this you can send data as many as you want

**Notes on URL Rewriting**

What will happen if cookies are disable in browser. If cookies are not enabled in browser that means

client will never join the session. In other words the session’s isNew() method will always return true. A client with cookies disabled will ignore “set-cookie” response header. If the client will not take cookie you can use URL rewriting as a backup. In case of URL rewriting we add the session to the end of all the URLs in the HTML we send back in the response. URL Rewriting kicks in only if cookies fail and only if you ask the response to encode the URL.

**Programe 30: session tracking using url rewriting**

**public/LoginSessionServlet.java**

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

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

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

**import** java.sql.\*;

**public** **class** **LoginSessionServlet** **extends** **HttpServlet**{

**public** **void** **service**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

        out.println("<html><body>");

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

        String pass=req.getParameter("pass");

**try**{

            Class.forName("oracle.jdbc.driver.OracleDriver");

**Connection** c = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","System",

"Oracle10g");

**Statement** s = c.createStatement();

**String** s1 = "select \* from emp101 where name='"+name+"' and pass='"+pass+"'";

**ResultSet** rs = s.executeQuery(s1);

**if**(rs.next()){

**HttpSession** se = req.getSession();

                se.setAttribute("name",name);

                se.setAttribute("pass",pass);

                out.println("User is valid "+se.isNew());

                out.println("<br>");

                out.println("<br>");

                out.println("<a href='demo'>demo</a>");

                out.println("<br>");

                out.println("<a href='log'>Logout</a>");

            }

**else**{

                out.println("User is InValid");

            }

        }

**catch**(Exception e){

            out.println(e);

        }

    }

}

**public/DemoServlet.java**

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

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

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

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

**public** **void** **service**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

**HttpSession** s = req.getSession(false);

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

**String** s1 = (String)s.getAttribute("name");

**String** s2 = (String)s.getAttribute("pass");

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

            out.println(s1+" = "+s2+" "+s.isNew());

            out.println("<br>");

            out.println("<a href='log'>Logout</a>");

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

        }

**else**{

            out.println("bavari pooch login kar le");

        }

    }

}

**public/Logout.java**

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

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

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

**public** **class** **Logout** **extends** **HttpServlet**{

**public** **void** **service**(HttpServletRequest req,HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

**HttpSession** s = req.getSession();

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

            s.invalidate();

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

            out.println("You have Logged out successfully");

            out.println("<br>");

            out.println("<a href='login1.html'>Login</a>");

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

        }

**else**{

        }

    }

}

**public/login1.html**

<html>

<body>

    <form action="login" method="get">

        Enter the name: <input type="TEXT" name="name">

        <br>

        Enter the password: <input type="password" name="pass">

        <br>

        <input type="submit">

    </form>

</body>

</html>

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>UrlSessionServlet</servlet-name>

        <servlet-class>UrlSessionServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>UrlSessionServlet</servlet-name>

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

    </servlet-mapping>

    <servlet>

        <servlet-name>DemoServlet</servlet-name>

        <servlet-class>DemoServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>DemoServlet</servlet-name>

        <url-pattern>/demo</url-pattern>

    </servlet-mapping>

    <servlet>

        <servlet-name>Logout</servlet-name>

        <servlet-class>Logout</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>Logout</servlet-name>

        <url-pattern>/log</url-pattern>

    </servlet-mapping>

</web-app>

**Hidden form field**

<form action=” “>

<input type=”hidden” value=”id” name=”jsessionid”>

</form>

It is similar to URL rewriting but in this case each request should include a for submission.

**Programe 31 session tracking using hidden form field**

**pubilc/HiddenFormFieldServlet.java**

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

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

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

**import** java.sql.\*;

**public** **class** **HiddenFormFieldServlet** **extends** **HttpServlet**{

**public** **void** **service**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

        out.println("<html><body>");

**String** name = req.getParameter("name");

**String** pass = req.getParameter("pass");

**try**{

            Class.forName("oracle.jdbc.driver.OracleDriver");

**Connection** c = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","System",

"Oracle10g");

**Statement** s = c.createStatement();

**String** s1 = "select \* from emp101 where name='"+name+"' and pass='"+pass+"'";

*//out.println(s1);*

**ResultSet** rs = s.executeQuery(s1);

**if**(rs.next()){

**HttpSession** se = req.getSession();

**String** id = se.getId();

                se.setAttribute("name",name);

                se.setAttribute("pass",pass);

                out.println("User is valid "+se.isNew());

                out.println("<br>");

                out.println(id);

                out.println("<br>");

                out.println("<form action='demo' method='get'>");

                out.println("<input type=hidden name=jsessionid value="+id+">");

                out.println("<input type='text' name='uname'>");

                out.println("<input type='submit'>");

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

            }

**else**{

                out.println("User is Invalid");

            }

        }

**catch**(Exception e){

            out.println(e);

        }

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

    }

}

**public/DemoServlet.java**

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

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

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

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

**public** **void** **service**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

**HttpSession** s = req.getSession(false);

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

**String** s1 = (String)s.getAttribute("name");

**String** s2 = (String)s.getAttribute("pass");

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

            out.println(s1+" = "+s2+" "+s.isNew());

            out.println("<br>");

            out.println("<a href='log'>Logout</a>");

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

        }

**else**{

            out.println("bavari pooch login kar le");

        }

    }

}

**public/login1.html**

<html>

<body>

    <form action="login2" method="get">

        Enter the name: <input type="TEXT" name="name">

        <br>

        Enter the password: <input type="PASSWORD" name="pass">

        <br>

        <input type="SUBMIT">

    </form>

</body></html>

**Methods of HttpSession interface**

**1. public long getCreationTime() :** returns creation time of session

**2. public String getId():** returns session id

**3. public long getLastAccessTime():** returns last active time of a session

**4. public void setMaxInactiveInterval(int interval):** set the time to automatically close the session.

**5. public int getMaxInactiveInterval():** get the time when session is automatically closed.

**6. Pubilc boolean isNew():** check is session new or old.

**7. public void invalidate():** use to close session.

There are three ways in which a session can die.

1. It is time out.

2. You have called invalidate() method on HttpSession object.

3. Application goes down.

We can also specify inactive interval using web.xml in root tag(web-app)

<session-config>

<session-timeout>15</session-timeout>

</session-config>

This time is considered as minute and set this using setMaxInactiveInterval() method then considered as seconds

**Cookies**

We have learn the cookie which is used by session internally, Now we are going to track cookies our self means we will set data in cookie, send the cookie to browser and accept cookie using servlet. If you don’t set cookie age then default cookie take space in browser’s cache memory and as soon as browser is closed, cookie is also deleted

First we will create cookie class object, one cookie object for one data and we will pass cookie name and its value in cookie constructor. For example Cookie c1 = new Cookie(username,lalu);

Cookie c2 = new Cookie(password,rabari);

c1.setMaxAge(60\*60\*24\*365);

c2.setMaxAge(60\*60\*24\*365);

res.addCookie(c1);

res.addCookie(20);

———————————————————

Cookie c[] = req.getCookies();

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

String name = c[i].getName();

String pass = c[i].getValue();

out.println(names+pass);

}

By default a cookie lives in the browser as long as it is opened. Once the client quits his browser the cookies disappears. That’s how the jsessionid cookie works. But you can tell a cookie to stay alive even after shuts down.

**Programe 32: cookies first programe**

**public/MyCookie.java**

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

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

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

**public** **class** **MyCookie** **extends** **HttpServlet**{

    PrintWriter out;

**public** **void** **service**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

        out = res.getWriter();

        Cookie c[] = req.getCookies();

**if**(c == null){

            req.getSession();

**Cookie** c1 = **new** **Cookie**("name","appsquadz");

**Cookie** c2 = **new** **Cookie**("add","noida");

            res.addCookie(c1);

            res.addCookie(c2);

            out.println("cookies added");

        }

**else**{

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

                out.println(c[i].getName());

                out.println(c[i].getValue());

                out.println("<br>");

            }

        }

    }

}

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>MyCookie</servlet-name>

        <servlet-class>MyCookie</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>MyCookie</servlet-name>

        <url-pattern>/cookie1</url-pattern>

    </servlet-mapping>

</web-app>

**Programe 33: cookies age**

**public/MyCookie1.java**

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

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

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

**public** **class** **MyCookie1** **extends** **HttpServlet**{

    PrintWriter out;

**public** **void** **service**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

        out = res.getWriter();

        Cookie c[] = req.getCookies();

**if**(c == null){

**Cookie** c1 = **new** **Cookie**("name","appsquadz");

**Cookie** c2 = **new** **Cookie**("add","noida");

            res.addCookie(c1);

            res.addCookie(c2);

            c1.setMaxAge(30\*60);

            c2.setMaxAge(30\*60);

            out.println("cookies added");

        }

**else**{

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

            out.println("<form>");

            out.println("<input type='text' value='"+c[0].getValue()+"'>");

            out.println("<br>");

            out.println("<input type='text' value='"+c[1].getValue()+"'>");

            out.println("<br>");

            out.println("<input type='submit'>");

            out.println("</form>");

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

        }

    }

}

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>MyCookie1</servlet-name>

        <servlet-class>MyCookie1</servlet-class>

    </servlet>

    <servlet-mapping>

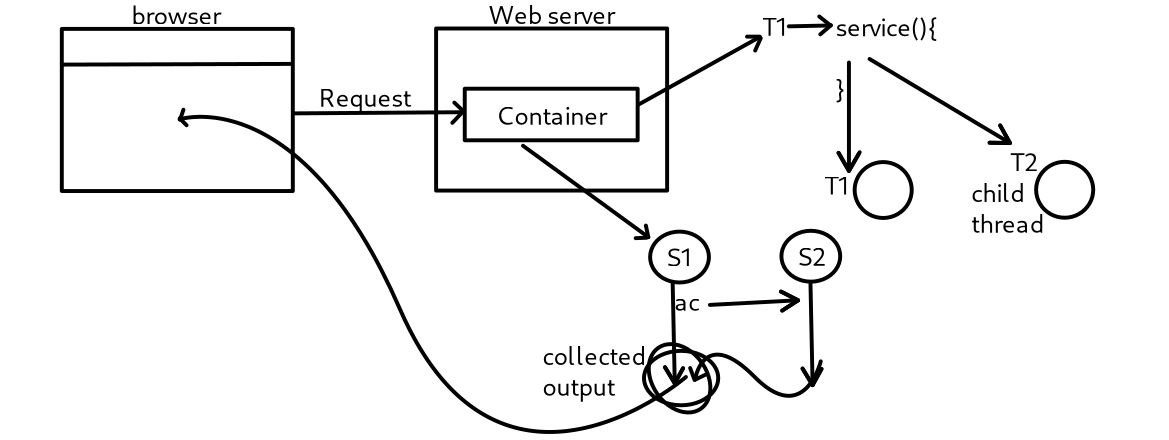
        <servlet-name>MyCookie1</servlet-name>

        <url-pattern>/cookie1</url-pattern>

    </servlet-mapping>

</web-app>

**Asynckness Context interface**



We make a request of a servlet from browser, now container will start a thread T1 and that thread runs service method, we are doing a lot of things on servlet which is being done one by one and takes more time so i thought, we should start a child thread and assign half of the work to T1 thread and the rest half work to its child thread. But we can do that because if T1 thread do its work before its child thread then child thread will become orphaned.

So we are not able to start a thread in servlet. To solve this problem, AsyncContext interface come into the story. Now what we can do we have two servlet S1 and S2 and container will start thread and S1 started to running, Now what we’ll do with help of AsyncContext i’ll call S2 servlet and this one will also start running on a separate thread, both servlet will do their job parallely. If S1 servlet has done its job so it will wait for S2 complete its work then output of both thread collectively send to browser.

**Notes on AsyncContext interface**

javax.servlet.AsyncContext is an interface introduced in Servlet 3.0. AsyncContext has the role to start asynchronous context within a servlet using HttpServletRequest.startAsync() method. We can divide our task for asynchronous execution. Within a servlet, some of the task can be performed by another servlet or jsp and some of the task can be done by parent servlet itself and finally response will be sent collectively. AsyncContext is started by Request object as below

AsyncContext asyncContext = request.startAsync();

We can also set the timeout for asynchronous context. For the example we scan set indefinite timeout by setting it as 0.

—————————————————————————————————

To divide the task of parent servlet I am calling a jsp which will execute other task. For this AsyncContext provides dispatch() method. We can use it as below.

asyncContext.dispatch(“asynctext.jsp”)

**Programe 34: AsyncContext servlet**

**public/AsyncContextExample.java**

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** javax.servlet.AsyncContext;

**import** javax.servlet.ServletRequest;

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

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

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

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

**@WebServlet(asyncSupported = true, value = "/ace", loadOnStartup = 1)**

**public** **class** **AsyncContextExample** **extends** **HttpServlet**{

**public** **void** **doPost**(HttpServletRequest req, HttpServletResponse res)**throws** IOException{

        doGet(req,res);

    }

**public** **void** **doGet**(HttpServletRequest req, HttpServletResponse res)**throws** IOException{

        res.setContentType("text/html");

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

        out.println(Thread.currentThread().getName());

**AsyncContext** asyncContext = req.startAsync();

        asyncContext.setTimeout(0);

**ServletRequest** servReq = asyncContext.getRequest();

**boolean** b = servReq.isAsyncStarted();

        out.println("isAsyncStarted: "+b);

        asyncContext.dispatch("/hello");

**try**{

            Thread.sleep(1000);

        }

**catch**(Exception e){

        }

        out.println("<br/>asynchronous task finished.");

    }

}

**Public/MyAnnotationServlet.java**

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

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

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

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

**@WebServlet(name = "MyAnnotationServlet", urlPatterns = {"/hello"})**

**public** **class** **MyAnnotationServlet** **extends** **HttpServlet**{

**public** **void** **service**(HttpServletRequest req,HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

        out.println("<h2>Hello World Servlet Annotation Example</h2>");

        out.println(Thread.currentThread().getName());

**try**{

            Thread.sleep(3000);

        }

**catch**(Exception e){

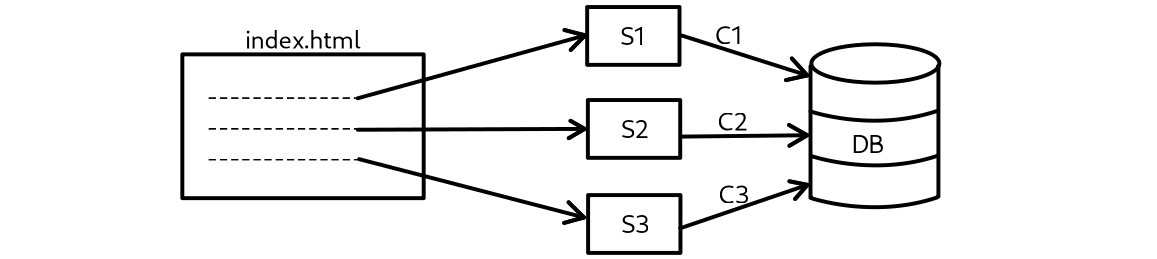
        }

        out.close();

    }

}

**Listeners**



We open a index.html in browser as we can see and in index.html there are three hyper link and each link is connected to a servlet and all three hit the database first servlet inserts data ,second deletes data and third modifies a data so when i make connection with database from first servlet, I’ll load the driver class using Class.forName() then getConnection and same for second and third servlet. We can see here we are making database connection again and again, which i don’t want to do. They gave us event ServletContext event and it will fire when ServletContext object is created ServletContext is created when you deploy application in container. And we have session event as soon as session is created event will be fire

**Programe 35: ServletContext event**

**public/ListenerServlet.java**

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

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

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

**import** java.sql.\*;

**public** **class** **ListenerServlet** **extends** **HttpServlet**{

**public** **void** **service**(HttpServletRequest req,HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

        out.println("<html><body>");

**ServletContext** ctx = getServletContext();

**Connection** c = (Connection)ctx.getAttribute("con");

**try**{

**Statement** s = c.createStatement();

**ResultSet** result1 = s.executeQuery("select \* from emp101");

**while**(result1.next()){

                out.println("id: "+result1.getString(1));

                out.println("name: "+result1.getString(2));

                out.println("<br>");

            }

        }

**catch**(Exception e){}

    }

}

**public/MyListener.java**

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

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

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

**import** java.sql.\*;

**public** **class** **MyListener** **implements** **ServletContextListener**{

    Connection c;

**public** **void** **contextInitialized**(ServletContextEvent e){

**try**{

**ServletContext** ctx = e.getServletContext();

            Class.forName("oracle.jdbc.driver.OracleDriver");

            c = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","System","Oracle10g");

            ctx.setAttribute("con",c);

            System.out.println("context created");

        }

**catch**(Exception e1){}

    }

**public** **void** **contextDestroyed**(ServletContextEvent e){

**try**{

            c.close();

        }

**catch**(Exception e1){}

    }

}

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>ListenerServlet</servlet-name>

        <servlet-class>ListenerServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>ListenerServlet</servlet-name>

        <url-pattern>/listener</url-pattern>

    </servlet-mapping>

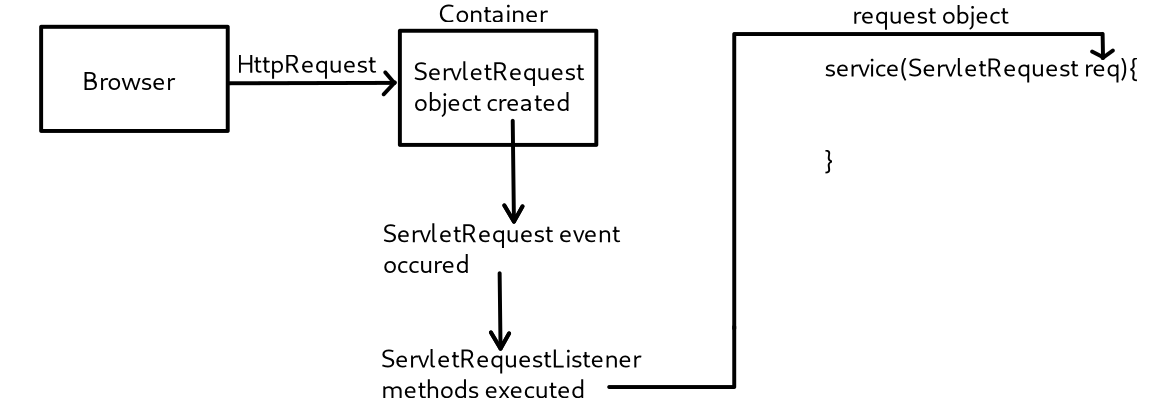
    <listener>

        <listener-class>MyListener</listener-class>

    </listener>

</web-app>

**How RequestListener works**



HttpRequest goes to container from browser and then container will create two objects as the ServletRequest object will be created immediately ServletRequest event will be occurred and then ServletRequestListener method is executed

**Programe 36: ServletRequest event**

**public/RequestListenerServlet.java**

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

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

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

**import** java.sql.\*;

**public** **class** **RequestListenerServlet** **extends** **HttpServlet**{

**public** **void** **service**(HttpServletRequest req,HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

        out.println("<html><body>");

        System.out.println("after via servlet");

**String** s1 = (String)req.getAttribute("name");

        out.println(s1);

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

    }

}

**public/MyRequestListener.java**

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

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

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

**import** java.sql.\*;

**public** **class** **MyRequestListener** **implements** **ServletRequestListener**{

**public** **void** **requestInitialized**(ServletRequestEvent e){

**try**{

**ServletRequest** s = e.getServletRequest();

            s.setAttribute("name","appsquadz");

            System.out.println("request created");

        }

**catch**(Exception e1){}

    }

**public** **void** **requestDestroyed**(ServletRequestEvent e){

        System.out.println("request destroyed");

    }

}

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>RequestListenerServlet</servlet-name>

        <servlet-class>RequestListenerServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>RequestListenerServlet</servlet-name>

        <url-pattern>/listener</url-pattern>

    </servlet-mapping>

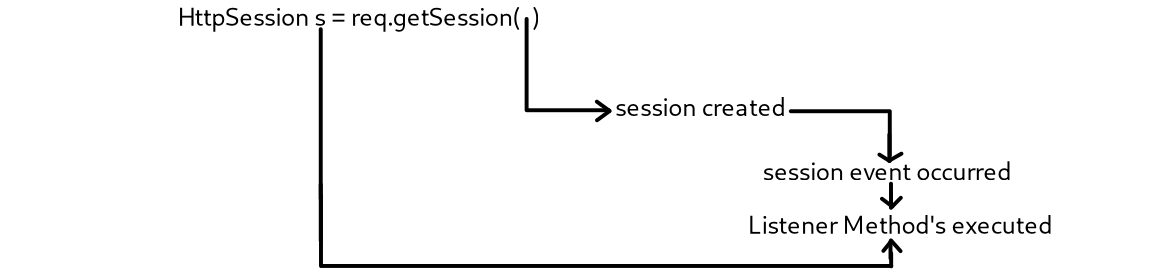
    <listener>

        <listener-class>MyRequestListener</listener-class>

    </listener>

</web-app>

**How SessionListener works**



**Programe 37: HttpSession event**

**public/SessionListenerServlet.java**

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

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

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

**import** java.sql.\*;

**public** **class** **SessionListenerServlet** **extends** **HttpServlet**{

**public** **void** **service**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

        out.println("<html><body>");

**HttpSession** s = req.getSession();

**String** s1 = (String)s.getAttribute("name");

        out.println(s1+" "+s.isNew());

        out.println("<br>");

        out.println("<a href='log'>Logout</a>");

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

    }

}

**public/MySessionListener.java**

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

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

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

**import** java.sql.\*;

**public** **class** **MySessionListener** **implements** **HttpSessionListener**{

**public** **void** **sessionCreated**(HttpSessionEvent e){

**try**{

**HttpSession** s = e.getSession();

            s.setAttribute("name","appsquadz");

            System.out.println("session created");

        }

**catch**(Exception e1){

        }

    }

**public** **void** **sessionDestroyed**(HttpSessionEvent e){

        System.out.println("session destroyed");

    }

}

**public/Logout.java**

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

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

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

**public** **class** **Logout** **extends** **HttpServlet**{

**public** **void** **service**(HttpServletRequest req,HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

**HttpSession** s = req.getSession();

*//HttpSession s = req.getSession(false);*

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

            s.invalidate();

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

            out.println("You have Logged out successfully");

            out.println("<br>");

            out.println("<a href='login1.html'>Login</a>");

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

        }

**else**{

        }

    }

}

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>SessionListenerServlet</servlet-name>

        <servlet-class>SessionListenerServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>SessionListenerServlet</servlet-name>

        <url-pattern>/listener</url-pattern>

    </servlet-mapping>

    <servlet>

        <servlet-name>Logout</servlet-name>

        <servlet-class>Logout</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>Logout</servlet-name>

        <url-pattern>/log</url-pattern>

    </servlet-mapping>

</web-app>

**Programe 38: ServletContext event via annotation**

**public/MyAnnotationServlet.java**

**import** java.io.IOException;

**import** java.io.PrintWriter;

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

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

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

**@WebServlet(name = "MyAnnotationServlet", urlPatterns = {"/listener"})**

**public** **class** **MyAnnotationServlet** **extends** **HttpServlet**{

**public** **void** **doGet**(HttpServletRequest req,HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

**ServletContext** ctx = getServletContext();

**String** s = (String)ctx.getAttribute("name");

        out.println("<h2>Hello World Servlet Annotation Example</h2>");

        out.close();

    }

}

**public/MyListener.java**

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

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

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

**import** java.sql.\*;

**import** javax.servlet.ServletContextEvent;

**import** javax.servlet.ServletContextListener;

**import** javax.servlet.annotation.WebListener;

**@WebListener**

**public** **class** **MyListener** **implements** **ServletContextListener**{

**public** **void** **contextInitialized**(ServletContextEvent e){

**try**{

**ServletContext** ctx = e.getServletContext();

            ctx.setAttribute("name","appsquadz");

            System.out.println("context created");

        }

**catch**(Exception e1){}

    }

**public** **void** **contextDestroyed**(ServletContextEvent e){

        System.out.println("context destroyed");

    }

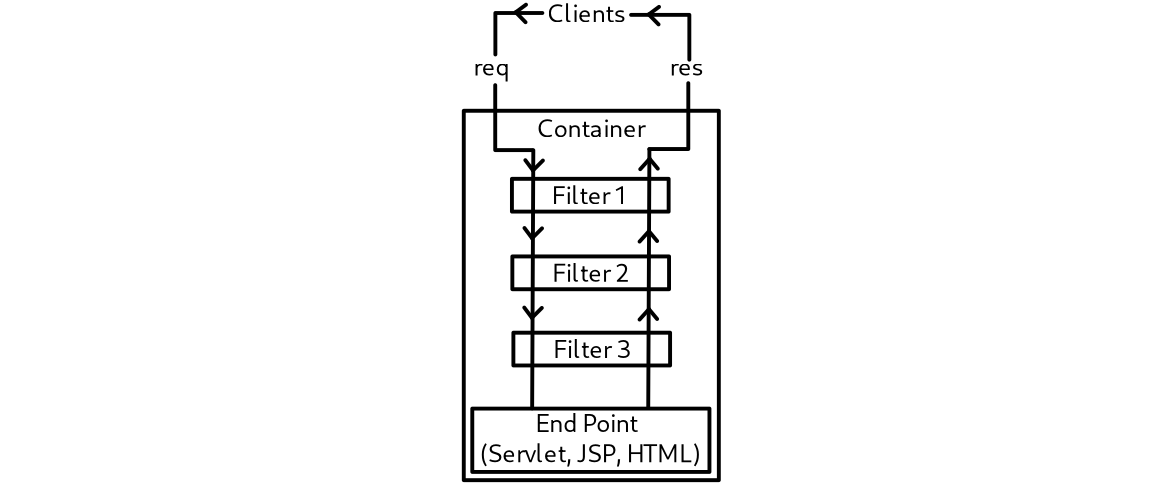
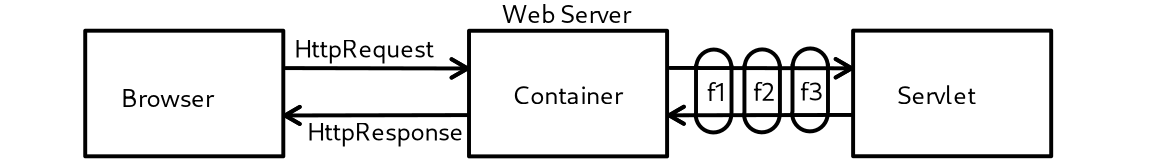
}

**Filters**

When we make request from browser for a web page. The request goes to container and container makes two object first is request and second is response corresponding to that request and these two objects pass to the service but here we will put filter between container and servlet. And when response goes to browser it will also go through filter.

Filters are just like a servlet, they are helper object of your servlet. For example, we can check user-name and password validity in filter and if valid then request pass to servlet and if not then do not need to.

We can connect one filter with one servlet, one filter with 50 servlet or you can connect single filter with application. We can connect filter with servlet, JSP, you can give its own url



**Notes on Filter**

Filters are the helper objects of your servlet. They are used to make your servlet light weight. A filter is servlet like container managed object. They can be inserted with the request and response. Filter are designed to be able to manipulate a request must go through that is sent to web application. A good way to think of servlet filters is a chain of steps that a request must go through before servlet, JSP or an HTML page in web application.

**Functionalities provided by filters**

**1.** Validation of HttpRequest.

**2.** Logging HttpRequest.

**3.** Authorization of HttpRequest.

**4.** Content management.

**5.** Provide custom http environment for servlet and JSP.

**Difference between Servlet and Filter**

|  |  |
| --- | --- |
| **Servlet** | **Filter** |
| Web Resources | Not Web Resources |
| Dynamic | Can be invoked by static pages |

**Filter API**

**javax.servlet.Filter interface**

public void init(FilterConfig config)throws ServletException

public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain)throws ServletException,

IOException

public void destroy()

|  |  |
| --- | --- |
| **Javax.servlet.FilterConfig interface ->**  public String getFilterName()  public String getInitParameter(String name)  public ServletContext getServletContext() | **Javax.servlet.FilterChain**  public void doFilter(ServletRequest req,  ServletResponse res) |

We write filtration code in doFilter method of Filter interface and call doFilter of FilterChain to forward request and response object to next filter or servlet

public void doFilter(req,res,chain){

- - - - - - - - - - -

- - - - - - - - - - -

chain.doFilter(req,res);

- - - - - - - - - - -

- - - - - - - - - - -

}

**Programe 39: Filter first programe**

**public/MyFilterServlet.java**

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

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

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

**public** **class** **MyFilterServlet** **extends** **HttpServlet**{

    PrintWriter out;

**public** **void** **service**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

        out = res.getWriter();

**ServletContext** ctx = getServletContext();

**Integer** count = (Integer)ctx.getAttribute("count");

        out.println("via servlet = "+count.intValue());

        out.println("<br>");

    }

}

**public/MyFilter.java**

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

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

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

**public** **class** **MyFilter** **implements** **Filter**{

    FilterConfig config;

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

        this.config = config;

    }

**public** **void** **doFilter**(ServletRequest req,ServletResponse res,FilterChain chain)**throws**

ServletException, IOException{

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

**ServletContext** ctx = config.getServletContext();

**Integer** count = (Integer)ctx.getAttribute("count");

**if**(count==null){

            count = **new** **Integer**(0);

        }

        count = **new** **Integer**(count.intValue()+1);

        ctx.setAttribute("count",count);

        out.println("myfilter jate hua");

        out.println("<br>");

        chain.doFilter(req,res);

        out.println("MyFilter response aate hua");

        out.println("<br>");

    }

**public** **void** **destroy**(){

    }

}

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>MyFilterServlet</servlet-name>

        <servlet-class>MyFilterServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>MyFilterServlet</servlet-name>

        <url-pattern>/filter1</url-pattern>

    </servlet-mapping>

    <filter>

        <filter-name>MyFilter</filter-name>

        <filter-class>MyFilter</filter-class>

    </filter>

    <filter-mapping>

        <filter-name>MyFilter</filter-name>

        <url-pattern>/filter1</url-pattern>

    </filter-mapping>

    <filter>

        <filter-name>MyFilter1</filter-name>

        <filter-class>MyFilter1</filter-class>

    </filter>

    <filter-mapping>

        <filter-name>MyFilter1</filter-name>

        <url-pattern>/filter1</url-pattern>

    </filter-mapping>

</web-app>

**Note:** filter will execute in order as they are mapped in web.xml

Now I want to connect a filter with any three servlets out of 100

**Que.** Is there any way to force container to create more than one object of servlet (CapGemini)

**Ans.** Yes, we can do that by mapping same servlet with three different name and url pattern.

**Programe 40: one filter with three servlet**

**public/MyFilterServlet1.java**

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

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

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

**public** **class** **MyFilterServlet1** **extends** **HttpServlet**{

    PrintWriter out;

**public** **void** **service**(HttpServletRequest req,HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

        out = res.getWriter();

        out.println("MyFilterServlet1");

    }

}

**public/MyFilterServlet2.java**

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

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

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

**public** **class** **MyFilterServlet2** **extends** **HttpServlet**{

    PrintWriter out;

**public** **void** **service**(HttpServletRequest req,HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

        out = res.getWriter();

        out.println("MyFilterServlet2");

    }

}

**public/MyFilterServlet3.java**

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

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

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

**public** **class** **MyFilterServlet3** **extends** **HttpServlet**{

    PrintWriter out;

**public** **void** **service**(HttpServletRequest req,HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

        out = res.getWriter();

        out.println("MyFilterServlet3");

    }

}

**public/MyFilter1.java**

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

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

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

**public** **class** **MyFilter1** **implements** **Filter**{

    FilterConfig config;

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

        this.config = config;

    }

**public** **void** **doFilter**(ServletRequest req,ServletResponse res,FilterChain chain)**throws**

ServletException,IOException{

        res.getWriter().println("MyFilter1");

        chain.doFilter(req,res);

    }

**public** **void** **destroy**(){

    }

}

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>MyFilterServlet1</servlet-name>

        <servlet-class>MyFilterServlet1</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>MyFilterServlet1</servlet-name>

        <url-pattern>/filter1</url-pattern>

    </servlet-mapping>

    <servlet>

        <servlet-name>MyFilterServlet2</servlet-name>

        <servlet-class>MyFilterServlet2</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>MyFilterServlet2</servlet-name>

        <url-pattern>/filter2</url-pattern>

    </servlet-mapping>

    <servlet>

        <servlet-name>MyFilterServlet3</servlet-name>

        <servlet-class>MyFilterServlet3</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>MyFilterServlet3</servlet-name>

        <url-pattern>/filter3</url-pattern>

    </servlet-mapping>

    <filter>

        <filter-name>MyFilter1</filter-name>

        <filter-class>MyFilter1</filter-class>

    </filter>

    <filter-mapping>

        <filter-name>MyFilter1</filter-name>

        <url-pattern>/filter1</url-pattern>

    </filter-mapping>

    <filter>

        <filter-name>MyFilter2</filter-name>

        <filter-class>MyFilter1</filter-class>

    </filter>

    <filter-mapping>

        <filter-name>MyFilter2</filter-name>

        <url-pattern>/filter2</url-pattern>

    </filter-mapping>

    <filter>

        <filter-name>MyFilter3</filter-name>

        <filter-class>MyFilter1</filter-class>

    </filter>

    <filter-mapping>

        <filter-name>MyFilter3</filter-name>

        <url-pattern>/filter3</url-pattern>

    </filter-mapping>

</web-app>

**Programe 41: login filter**

**public/LoginFilter.java**

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

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

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

**import** java.sql.\*;

**public** **class** **LoginFilter** **implements** **Filter**{

    FilterConfig config;

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

        this.config = config;

    }

**public** **void** **doFilter**(ServletRequest req,ServletResponse res,FilterChain chain)**throws**

ServletException,IOException{

**String** name = req.getParameter("name");

**String** pass = req.getParameter("pass");

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

**try**{

            Class.forName("oracle.jdbc.driver.OracleDriver");

**Connection** c = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","System",

"Oracle10g");

**Statement** s = c.createStatement();

**String** s1 = "select \* from emp101 where name='"+name+"' and pass='"+pass+"'";

**ResultSet** rs = s.executeQuery(s1);

**if**(rs.next()){

                chain.doFilter(req,res);

            }

**else**{

                out.println("User is InValid");

            }

        }

**catch**(Exception e){

            out.println(e);

        }

        out.println("loginfilter");

    }

**public** **void** **destroy**(){

    }

}

**public/LogFilter.java**

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

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

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

**import** java.util.\*;

**public** **class** **LogFilter** **implements** **Filter**{

**private** **FilterConfig** filterConfig = null;

**public** **void** **doFilter**(ServletRequest req,ServletResponse res, FilterChain chain)**throws**

ServletException,IOException{

**long** start = System.currentTimeMillis();

**String** address = req.getRemoteAddr();

**String** file = ((HttpServletRequest)req).getRequestURI();

        filterConfig.getServletContext().log(

            "User access! \n"+

            "User IP: "+address+"\n"+

            "Resource: "+file+"\n"+

            "Milliseconds used: "+(System.currentTimeMillis() - start)

        );

        chain.doFilter(req,res);

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

        out.println("log filter");

    }

**public** **void** **destroy**(){

    }

**public** **void** **init**(FilterConfig filterConfig){

        this.filterConfig = filterConfig;

    }

}

**public/Welcome.java**

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

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

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

**import** java.sql.\*;

**public** **class** **WelcomeServlet** **extends** **HttpServlet**{

**public** **void** **service**(HttpServletRequest req,HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

        out.println("<html><body>");

**String** name = req.getParameter("name");

        out.println("welcome = "+name);

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

    }

}

**public/login.html**

<html>

<body>

    <form action="login">

        Enter the name: <input type="text" name="name">

        <br>

        Enter the password: <input type="password" name="pass">

        <br>

        <input type="submit">

    </form>

</body>

</html>

**public/WEB-INF/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>/login</url-pattern>

    </servlet-mapping>

    <filter>

        <filter-name>LoginFilter</filter-name>

        <filter-class>LoginFilter</filter-class>

    </filter>

    <filter-mapping>

        <filter-name>LoginFilter</filter-name>

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

    </filter-mapping>

    <filter>

        <filter-name>LogFilter</filter-name>

        <filter-class>LogFilter</filter-class>

    </filter>

    <filter-mapping>

        <filter-name>LogFilter</filter-name>

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

    </filter-mapping>

</web-app>

**Programe 42: send parameter to filter**

**public/MyFilterServlet.java**

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

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

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

**public** **class** **MyFilterServlet** **extends** **HttpServlet**{

    PrintWriter out;

**public** **void** **service**(HttpServletRequest req,HttpServletResponse res)**throws**

ServletException,IOException{

        res.setContentType("text/html");

        out = res.getWriter();

**ServletContext** ctx = getServletContext();

**String** s = (String)ctx.getAttribute("name");

        out.println("driver = "+s);

    }

}

**public/MyFilterInitParam.java**

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

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

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

**public** **class** **MyFilterInitParam** **implements** **Filter**{

    FilterConfig config;

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

        this.config = config;

    }

**public** **void** **doFilter**(ServletRequest req, ServletResponse res, FilterChain chain)**throws**

ServletException,IOException{

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

        config.getServletContext().setAttribute("name",s);

        res.getWriter().println("MyFilter1 "+s);

        chain.doFilter(req,res);

    }

**public** **void** **destroy**(){

    }

}

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>MyFilterServlet</servlet-name>

        <servlet-class>MyFilterServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>MyFilterServlet</servlet-name>

        <url-pattern>/filter</url-pattern>

    </servlet-mapping>

    <filter>

        <filter-name>MyFilterInitParam</filter-name>

        <filter-class>MyFilterInitParam</filter-class>

        <init-param>

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

            <param-value>oracle.jdbc.driver.OracleDriver</param-value>

        </init-param>

    </filter>

    <filter-mapping>

        <filter-name>MyFilterInitParam</filter-name>

        <url-pattern>/filter</url-pattern>

    </filter-mapping></web-app>

**Programe 43: servlet and filter via annotation**

**public/MyAnnnotationServlet.java**

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

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

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

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

**@WebServlet(name = "MyAnnotationServlet", urlPatterns = {"/hello"})**

**public** **class** **MyAnnotationServlet** **extends** **HttpServlet**{

**public** **void** **service**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

**ServletContext** ctx = getServletContext();

**Integer** count = (Integer)ctx.getAttribute("count");

        out.println("<h2>Hello World Servlet Annotation Example</h2> "+count);

    }

}

**public/MyFilter.java**

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

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

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

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

**@WebFilter(urlPatterns = {"/hello"}, filterName = "MyFilter")**

**public** **class** **MyFilter** **implements** **Filter**{

    FilterConfig config;

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

        this.config = config;

    }

**public** **void** **doFilter**(ServletRequest req, ServletResponse res, FilterChain chain)**throws**

ServletException,IOException{

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

        out.println("MyFilter");

**ServletContext** ctx = config.getServletContext();

**Integer** count = (Integer)ctx.getAttribute("count");

**if**(count == null){

            count = **new** **Integer**(0);

        }

        count = **new** **Integer**(count.intValue()+1);

        ctx.setAttribute("count",count);

        chain.doFilter(req,res);

        out.println("filter output");

    }

**public** **void** **destroy**(){

    }

}

**public/MyFilter1.java**

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

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

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

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

**@WebFilter(urlPatterns = {"/hello"}, filterName = "MyFilter1")**

**public** **class** **MyFilter1** **implements** **Filter**{

    FilterConfig config;

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

        this.config = config;

    }

**public** **void** **doFilter**(ServletRequest req, ServletResponse res, FilterChain chain)**throws**

ServletException, IOException{

        res.getWriter().println("hello filter2");

        chain.doFilter(req,res);

    }

**public** **void** **destroy**(){

    }

}

**Programe 44: send parameter to filter via annotation**

**public/MyAnnotationServlet.java**

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

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

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

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

**@WebServlet(name = "MyAnnotationServlet", urlPatterns = {"/hello"})**

**public** **class** **MyAnnotationServlet** **extends** **HttpServlet**{

**public** **void** **service**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException, IOException{

        res.setContentType("text/html");

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

        out.println("<h2>Hello World Servlet Annotation Example</h2> ");

    }

}

**public/InitParamFilter.java**

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

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

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

**@WebFilter(urlPatterns = {"/hello"},**

    initParams = { @WebInitParam(name = "simpleParam", value = "paramValue")})

**public** **class** **InitParamFilter** **implements** **Filter**{

**private** **FilterConfig** filterConfig = null;

**@Override**

**public** **void** **doFilter**(ServletRequest req, ServletResponse res, FilterChain chain)**throws** IOException,

ServletException{

**String** s = filterConfig.getInitParameter("simpleParam");

        res.getWriter().println("Myfilter1 "+s);

        chain.doFilter(req,res);

    }

**public** **void** **init**(FilterConfig filterConfig)**throws** ServletException{

        this.filterConfig = filterConfig;

    }

**public** **void** **destroy**(){

    }

}

Async context

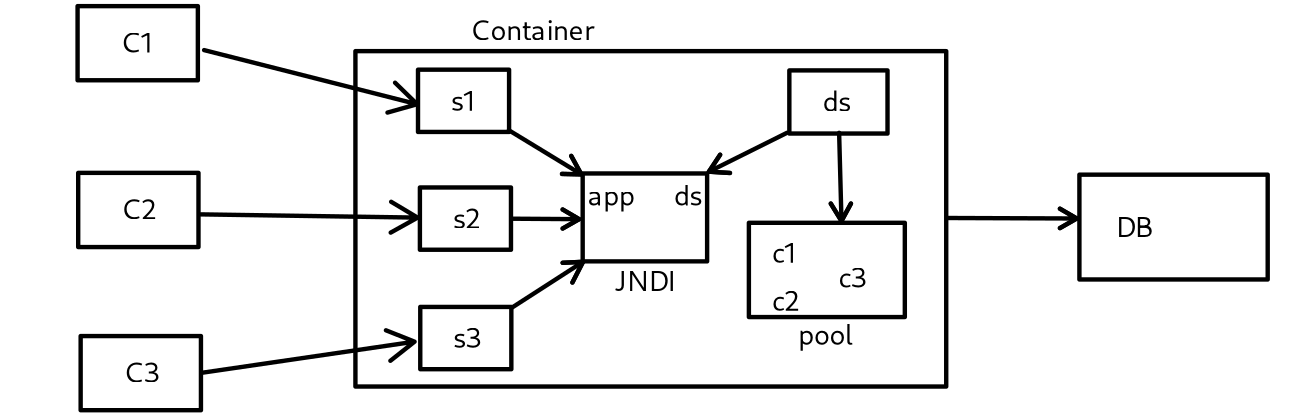
**Connection Polling**

Importance of pooling is reuse the object. Making new object for each time using new operator is heavy weight process use cloning instead of new operator. Likewise pooling of connection object is also maintained. And we did thread pooling and socket pooling is also maintained, we know that a separate is created for each client every time. But we have small chat server, suppose 1 lac clients are connected at same time and making new socket object for each client is heavy weight process. Use socket from connection pool instead of new socket

We have a class DataSource and we just have to call getConnection method of DataSource, it will return the connection object and don’t think that we have to associate DataSource with connection pool because we can create many connection pool and assign its name may be oracle’s pool will be different, MySql , DB2. In first wizard server will ask for database details, obviously driver, url, username and password, name for connection pool, how many initial objects and increment when out of objects all the work will be done at server side and now i have to bring the DataSource object from to my application then i will call getConnection and it will return connection object, But the problem is how will we bring the DataSource class object in our application and the solution is **JNDI(Java Naming and Directory Interface)**

It is an independent service, it has nothing to do with connection pool or servlet. Java says if you want to remotely look up the object and give it to a client then you can use this service. You have to bind the object with a name in this service. For biding the object it has method bind of class InitialContext in JNDI service, we don’t call the bind we just look up the object

**How connection poll works**



**Notes on Pooling**

The addition of JDBC connection pooling to your application usually involves little or no code modification but can often provide significant benefits in terms of application performance, concurrency and scalability. Improvements such as these can become especially important when your application is tasked with servicing many concurrent users within the requirements of sub second response time

There are many scenarios in software architecture where some type of object pooling is employed as a technique to improve application performance. Object pooling is effective for two simple reasons. First, the run time creation of new software objects is often more expensive in terms of performance and memory than the reuse of previously created objects.Second, garbage collection is an expensive process so when we reduce the number of objects to clean up we generally reduce the garbage collection load.

In a pooling scenario your application first creates an object pool that can both cache pooled objects and issue objects that are not in use back to the application. For example, pooled objects could be database connections, process threads, server sockets or any other kind of objects that may be expensive to create from scratch. As your application first starts asking the pool for objects they will be newly created but when the application has finished with the object it is returned to the pool rather than destroyed. At this point the benefits of object pooling will be realized since, now as the application needs more objects, the pool will be able to issue recycled objects that have previously been returned by the application.

**JDBC Connection Pooling**

JDBC connection pooling is conceptually similar to any other form of object pooling. Database connections are often expensive to create because of the overhead of establishing a network connection and initializing a database connection session in the back end database. In turn, connection session initialization often requires time consuming processing to perform user authentication, establish transactional contexts and establish other aspects of the session that are required for subsequent database usage.

Additionally, the database’s ongoing management of all of its connection sessions can impose a major limiting factor on the scalability of your application. Valuable database resources such as locks, memory, cursors, transaction logs, statements handles and temporary tables all tend to increase based on the number of concurrent connection sessions.

All in all, JDBC database connections are both expensive to initially create and then maintain over time.

Connection Pooling simply means creating, managing, and predetermined number of connection objects. A client application would then perform a JNDI look up to retrieve a reference to a DataSource objects that implements the ConnectionPoolDataSource interface. The client application would not need make any special provisions to use the pooled data source; the code would be no different from code written for a nonpooled DataSource.

**JNDI(Java Naming and Directory Interface)**

A naming service maintains a set of bindings, which relate names to objects. The J2EETM naming service is based on the Java Naming and Directory InterfaceTM (JNDI) API. The JNDI API allows application components and clients to look up distributed resources, service, and EJBTM components.

Sun Java System Application Server provides a naming environment, or context, which is compliant with standard J2EE 1.4 requirements. A Context objects provides the methods for bindings. The InitialContext is the handle to the J2EE naming service that application components and clients use for look ups.

**How to create pool**

login to weblogic console -> services -> data source -> new -> generic data source -> enter name -> JNDI name -> select database type -> next -> provide pool information -> select driver(Oracle’s Driver(thin) for Application Continuity; Version:Any) -> next -> next -> database name -> host name -> database username -> password -> next -> next -> check Admin Server -> finish -> save

**Programe 45: connection pool look up standalone application**

**import** javax.naming.\*;

**import** java.util.\*;

**import** java.sql.\*;

**import** javax.sql.\*;

**class** **MyPool**{

**public** **static** **void** **main**(String args[]){

**try**{

**Properties** parm = **new** **Properties**();

            parm.setProperty("java.naming.factory.initial","weblogic.jndi.WLInitialContextFactory");

            parm.setProperty("java.naming.provider.url","t3://localhost:7001");

            parm.setProperty("java.naming.security.principal","weblogic");

            parm.setProperty("java.naming.security.credentials","Weblogic12c");

**InitialContext** ctx = **new** **InitialContext**(parm);

**DataSource** ds = (DataSource)ctx.lookup("tindi");

**Connection** con = ds.getConnection();

**Statement** s = con.createStatement();

**ResultSet** result = s.executeQuery("select \* from emp101");

**while**(result.next()){

                System.out.println("id: "+result.getString(1));

                System.out.println("name: "+result.getString(2));

            }

        }

**catch**(Exception e){

            System.out.println(e);

        }

    }

}

**Programe 46: connection pool web application**

**public/PoolServletInitialContext.java**

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

**import** javax.annotation.Resources;

**import** javax.naming.\*;

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

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

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

**import** javax.sql.\*;

**import** java.sql.\*;

**@WebServlet(name = "PoolServlet1", urlPatterns = {"/pool1"})**

**public** **class** **PoolServletInitialContext** **extends** **HttpServlet**{

**@Override**

**protected** **void** **doGet**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

        out.println("<html><body>");

**try**{

**InitialContext** ctx = **new** **InitialContext**();

**DataSource** ds = (DataSource)ctx.lookup("tindi");

**Connection** c = ds.getConnection();

**Statement** s = c.createStatement();

**String** s1 = "select \* from emp101";

**ResultSet** rs = s.executeQuery(s1);

**while**(rs.next()){

                out.println("id: "+rs.getString(1));

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

                out.println("<br>");

            }

        }

**catch**(Exception e){

            out.println(e);

        }

    }

}

**Programe 47 connection pool web application via annotation**

**public/PoolServlet.java**

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

**import** javax.annotation.Resource;

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

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

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

**import** javax.sql.\*;

**import** java.sql.\*;

**@WebServlet(name = "PoolServlet", urlPatterns = {"/pool"})**

**public** **class** **PoolServlet** **extends** **HttpServlet**{

**@Resource(mappedName = "tindi")**

    DataSource ds;

**@Override**

**protected** **void** **doGet**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

        out.println("<html><body>");

**try**{

**Connection** c = ds.getConnection();

**Statement** s = c.createStatement();

**String** s1 = "select \* from emp101";

**ResultSet** rs = s.executeQuery(s1);

**while**(rs.next()){

                out.println("id: "+rs.getString(1));

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

                out.println("<br>");

            }

        }

**catch**(Exception e){

            out.println(e);

        }

    }

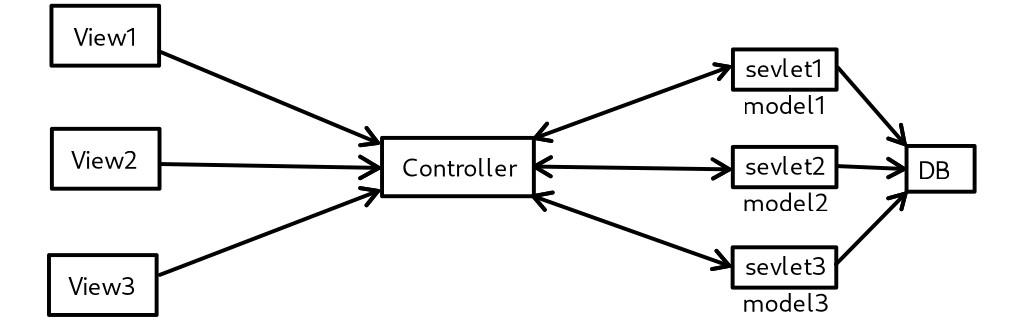
}

**MVC**

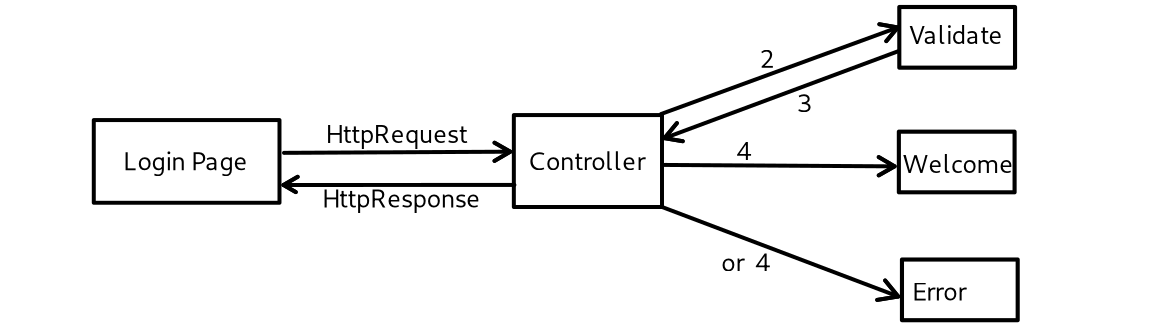
**Model:** Business logic or Database logic

**View:** end user interaction

**Controller:**



**——————————————————————————————————————————————**



**Programe 48: controller**

**public/ActionServlet.java**

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

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

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

**public** **class** **ActionServlet** **extends** **HttpServlet**{

**public** **void** **service**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

**String** url = req.getParameter("check");

*//out.println(url);*

**RequestDispatcher** rd = req.getRequestDispatcher(url);

        rd.forward(req,res);

    }}

**public/ValidateServlet.java**

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

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

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

**import** java.sql.\*;

**public** **class** **ValidateServlet** **extends** **HttpServlet**{

**public** **void** **service**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

        res.setContentType("text/html");

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

        System.out.println("validate servlet");

**String** name = req.getParameter("name");

**String** pass = req.getParameter("pass");

**try**{

            Class.forName("oracle.jdbc.driver.OracleDriver");

**Connection** c = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","System",

"Oracle10g");

**Statement** s = c.createStatement();

**String** s1 = "select \* from emp101 where name='"+name+"' and pass='"+pass+"'";

**ResultSet** rs = s.executeQuery(s1);

**if**(rs.next()){

**RequestDispatcher** rd = req.getRequestDispatcher("action?check=wel");

                rd.forward(req,res);

            }

**else**{

**RequestDispatcher** rd1 = req.getRequestDispatcher("action?check=err");

                rd1.forward(req,res);

            }

        }

**catch**(Exception e){

            out.println(e);

        }

    }

}

**public/WelcomeServlet.java**

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

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

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

**import** java.sql.\*;

**public** **class** **WelcomeServlet** **extends** **HttpServlet**{

**public** **void** **service**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

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

        res.setContentType("text/html");

        out.println("<html><body>");

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

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

    }

}

**public/ErrServlet.java**

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

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

**import** java.sql.\*;

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

**public** **class** **ErrServlet** **extends** **HttpServlet**{

**public** **void** **service**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

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

        res.setContentType("text/html");

        out.println("<html><body>");

        out.println("Error Servlet invalid username and password");

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

    }

}

**public/login.html**

<html>

<body>

    <form action="action?check=vali" method="post">

        Enter the name: <input type="TEXT" name="name">

        <br>

        Enter the password: <input type="PASSWORD" name="pass">

        <br>

        <input type="submit">

    </form>

</body>

</html>

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>ActionServlet</servlet-name>

        <servlet-class>ActionServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>ActionServlet</servlet-name>

        <url-pattern>/action</url-pattern>

    </servlet-mapping>

    <servlet>

        <servlet-name>ValidateServlet</servlet-name>

        <servlet-class>ValidateServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>ValidateServlet</servlet-name>

        <url-pattern>/vali</url-pattern>

    </servlet-mapping>

    <servlet>

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

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

    </servlet>

    <servlet-mapping>

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

        <url-pattern>/wel</url-pattern>

    </servlet-mapping>

    <servlet>

        <servlet-name>ErrServlet</servlet-name>

        <servlet-class>ErrServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>ErrServlet</servlet-name>

        <url-pattern>/err</url-pattern>

    </servlet-mapping>

</web-app>

**Programe 49: i18n in standalone application**

**import** java.util.\*;

**import** java.text.\*;

**class** **DefaultLocale**{

**public** **static** **void** **main**(String s[]){

        Locale list[] = DateFormat.getAvailableLocales();

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

            System.out.println(list[i].toString());

        }

    }

}

**———————————————————————**

**import** java.util.\*;

**class** **MyLocale**{

**public** **static** **void** **main**(String args[]){

*//Locale l = new Locale("fr","FR");*

**Locale** l = Locale.getDefault();

**ResourceBundle** rb = ResourceBundle.getBundle("ResourceBundle",l);

        System.out.println(Locale.getDefault().toString());

**Enumeration** e = rb.getKeys();

**while**(e.hasMoreElements()){

**String** k = (String)e.nextElement();

**String** v = rb.getString(k);

            System.out.println(k+" = "+v);

        }

    }

}

**ResourceBundle\_de.properties**

app.title=Struts 2 Internationalisierung

app.username=Benutzer Name

app.password=Kennwort

app.success\_login=Willkommen! Du hast erfolgrech angemeldet.

**ResourceBundle\_en.properties**

app.title=Struts 2 Internationlization

app.username=User Name

app.password=Password

app.success\_login=Welcome! You have logged in Successfully

**ResourceBundle\_fr.properties**

app.title=Struts 2 Internationalisation

app.username=Nomd'utilisateur

app.password=Mot de passe

app.success\_login=Bienvenue ! Vous avez entre avec succes.

**Programe 50: internationalization(i18n) or multiple language support**

**public/I18nServlet.java**

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

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

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

**import** java.util.\*;

**import** java.sql.\*;

**public** **class** **I18nServlet** **extends** **HttpServlet**{

**public** **void** **service**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

**String** t = null;

**String** name = null;

**String** pass = null;

        res.setContentType("text/html");

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

**Locale** l = req.getLocale();

**ResourceBundle** rb = ResourceBundle.getBundle("ResourceBundle",l);

**Enumeration** e = rb.getKeys();

**while**(e.hasMoreElements()){

**String** k = (String)e.nextElement();

**if**(k.equals("app.title")){

                t=rb.getString(k);

            }

**if**(k.equals("app.username")){

                name=rb.getString(k);

            }

**if**(k.equals("app.password")){

                pass=rb.getString(k);

            }

        }

        out.println("<html>");

        out.println("<body>");

        out.println("<h1 style='color:blue'>"+t+"</h1>");

        out.println("<br>");

        out.println("<form action='login'>");

        out.println(name+" ");

        out.println("<input type='text' name='uname'>");

        out.println("<br>");

        out.println(pass+" ");

        out.println("<input type='text' name='upass'>");

        out.println("<br>");

        out.println("<input type='submit'>");

        out.println("</form>");

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

    }

}

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>I18nServlet</servlet-name>

        <servlet-class>I18nServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>I18nServlet</servlet-name>

        <url-pattern>/i18n</url-pattern>

    </servlet-mapping>

</web-app>

**public/WEB-INF/classes/ResourceBundle\_en.properties**

app.title=Struts 2 Internationlization

app.username=User Name

app.password=Password

app.success\_login=Welcome! You have logged in Successfully

**public/WEB-INF/classes/ResourceBundle\_fr.properties**

app.title=Struts 2 Internationalisation

app.username=Nomd'utilisateur

app.password=Mot de passe

app.success\_login=Bienvenue ! Vous avez entre avec succes.

**public/WEB-INF/classes/ResourceBundle\_de.properties**

app.title=Struts 2 Internationalisierung

app.username=Benutzer Name

app.password=Kennwort

app.success\_login=Willkommen! Du hast erfolgrech angemeldet.

**Programe 51: image upload download**

**public/StoreImageServlet.java**

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

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

**import** com.oreilly.servlet.\*;

**import** java.sql.\*;

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

**public** **class** **StoreImageServlet** **extends** **HttpServlet**{

**public** **void** **service**(HttpServletRequest req, HttpServletResponse res)**throws** ServletException,

IOException{

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

**String** path = getServletContext().getRealPath("image");

**MultipartRequest** mpr = **new** **MultipartRequest**(req,path,500\*1024\*1024);

**String** path1 = mpr.getOriginalFileName("file");*//file is the req parameter name*

**String** path2 = path+"/"+path1;

**FileInputStream** fin = **new** **FileInputStream**(path2);

**try**{

            Class.forName("oracle.jdbc.driver.OracleDriver");

**Connection** c = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","System",

"Oracle10g");

**PreparedStatement** ps = c.prepareStatement("insert into image24 values(?,?)");

            ps.setBinaryStream(2,fin,fin.available());

            ps.setString(1,"26");

            ps.executeUpdate();

            c.close();

        }

**catch**(Exception e){

            out.println(e);

        }

        out.println("<html><body>");

        out.println("<h1>Image Saved successfully</h1>");

        out.println("<a href='show'>ShowImage</a>");

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

    }

}

**public/ShowImageServlet.java**

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

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

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

**import** java.sql.\*;

**import** com.oreilly.servlet.\*;

**public** **class** **ShowImageServlet** **extends** **GenericServlet**{

**InputStream** f = null;

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

        res.setContentType("text/html");

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

**try**{

            Class.forName("oracle.jdbc.driver.OracleDriver");

**Connection** c = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","System",

"Oracle10g");

**PreparedStatement** ps = c.prepareStatement("select img from image24 where userid=?");

            ps.setString(1,"26");

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

**String** path = getServletContext().getRealPath("/");

            rs.next();

            f = rs.getBinaryStream("img");

**FileOutputStream** f1 = **new** **FileOutputStream**(path+"/"+"abc12"+".jpg");

**int** i=0;

**while**((i = f.read())!=-1){

                f1.write(i);

            }

            out.println("<img src='abc12.jpg' width='160' height='170'>");

        }

**catch**(Exception e){

            out.println(e);

        }

    }

}

**public/upload.html**

<html>

<body>

    <form action="upload" enctype="multipart/form-data" method="post">

        <input type="file" name = "file">

        <input type="submit">

    </form>

</body>

</html>

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>StoreImageServlet</servlet-name>

        <servlet-class>StoreImageServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>StoreImageServlet</servlet-name>

        <url-pattern>/upload</url-pattern>

    </servlet-mapping>

    <servlet>

        <servlet-name>ShowImageServlet</servlet-name>

        <servlet-class>ShowImageServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>ShowImageServlet</servlet-name>

        <url-pattern>/show</url-pattern>

    </servlet-mapping>

</web-app>

**Programe 52: load on startup**

**public/ApplicationInitializer.java**

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

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

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

**import** java.util.\*;

**import** java.sql.\*;

**public** **class** **ApplicationInitializer** **extends** **HttpServlet**{

**public** **void** **init**(javax.servlet.ServletConfig sc)**throws** ServletException{

**ServletContext** ctx = sc.getServletContext();

**String** val = ctx.getInitParameter("oracletab");

**String** valsql = ctx.getInitParameter("sqltable");

**String** prop = ctx.getRealPath("WEB-INF/db/dp.properties");

**String** oracle = ctx.getRealPath("WEB-INF/dbtable/oracletable.sql");

**String** sql = ctx.getRealPath("WEB-INF/dbtable/sqltable.sql");

        LoadProperties.propLoad(prop);

**if**(val.equals("yes")){

            TableCreate.createTab(oracle);

            System.out.println("Table Successfully Created");

        }

**if**(valsql.equals("yes")){

            TableCreate.createTab(sql);

            System.out.println("Table Successfull Created");

        }

    }

}

**public/LoadProperties.java**

**import** java.io.FileInputStream;

**import** java.util.Enumeration;

**import** java.util.Properties;

**public** **class** **LoadProperties**{

**public** **static** **void** **propLoad**(String path){

**try**{

**FileInputStream** fn = **new** **FileInputStream**(path);

**Properties** pr = **new** **Properties**();

            pr.load(fn);

**Enumeration** en = pr.propertyNames();

**while**(en.hasMoreElements()){

**String** key = (String)en.nextElement();

**String** value = pr.getProperty(key);

                System.setProperty(key,value);

                System.out.println(key+" : "+value);

            }

        }

**catch**(Exception e){

        }

    }

}

**public/TableCreate.java**

**import** java.sql.\*;

**import** java.util.StringTokenizer;

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

**public** **class** **TableCreate**{

**public** **static** **void** **createTab**(String path){

**try**{

**FileInputStream** fn = **new** **FileInputStream**(path);

**byte** br[] = **new** **byte**[fn.available()];

            fn.read(br);

            fn.close();

**String** data = **new** **String**(br);

**StringTokenizer** str = **new** **StringTokenizer**(data,"/");

**Connection** con = ConnectionProvider.getConn();

**Statement** stm = con.createStatement();

**while**(str.hasMoreTokens()){

**String** query = str.nextToken();

**if**(query.trim().equals("stop")){

**break**;

                }

                stm.executeUpdate(query);

                System.out.println(query);

            }

        }

**catch**(Exception e){

            System.out.println(e);

        }

    }

}

**public/ConnectionProvider.java**

**import** java.sql.\*;

**public** **class** **ConnectionProvider**{

**public** **static** **Connection** con = null;

**public** **static** Connection **getConn**(){

**if**(con == null){

**try**{

**String** driver = System.getProperty("driver");

**String** url = System.getProperty("url");

**String** user = System.getProperty("user");

**String** password = System.getProperty("pass");

                Class.forName(driver);

                con = DriverManager.getConnection(url,user,password);

            }

**catch**(Exception e){

            }

        }

**return** con;

    }

}

**public/WEB-INF/db/dp.properties**

driver=oracle.jdbc.driver.OracleDriver

url=jdbc:oracle:thin:**@localhost**:1521:xe

user=system

pass=Oracle10g

**public/WEB-INF/dbtable/oracletable.sql**

create table **login**(name varchar2(20),pass **varchar2**(20))

/

create table **register**(name varchar2(20), pass **varchar2**(20),addr **varchar2**(20),id **number**(20), mail **varchar2**(20))

/

stop

**public/WEB-INF/web.xml**

<web-app>

    <context-param>

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

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

    </context-param>

    <context-param>

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

        <param-value>no</param-value>

    </context-param>

    <servlet>

        <servlet-name>ApplicationInitializer</servlet-name>

        <servlet-class>ApplicationInitializer</servlet-class>

        <load-on-startup>1</load-on-startup>

    </servlet>

    <servlet-mapping>

        <servlet-name>ApplicationInitializer</servlet-name>

        <url-pattern>/table</url-pattern>

    </servlet-mapping>

</ web-app>

**Programe 53: check cookies enabled javascript**

**public/LoginServlet.java**

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

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

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

**import** java.sql.\*;

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

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

        res.setContentType("text/html");

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

        out.println("<html><body>");

**String** name = req.getParameter("name");

**String** pass = req.getParameter("pass");

        out.println(name+" : ");

**try**{

            Class.forName("oracle.jdbc.driver.OracleDriver");

**Connection** c = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system",

"Oracle10g");

**Statement** s = c.createStatement();

**String** s1 = "select \* from emp101 where name='"+name+"' and pass='"+pass+"'";

**ResultSet** rs = s.executeQuery(s1);

**if**(rs.next()){

                out.println("User is Valid");

            }

**else**{

                out.println("User is Invalid");

            }

        }

**catch**(Exception e){

            e.printStackTrace();

        }

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

    }

}

**public/login.html**

<html>

<body>

    <script type="text/javascript">

        function **myFunction**(){

**var** form = document.form1;

**var** name = form.name.value;

**var** pass = form.pass.value;

**if**(navigator.cookieEnabled==false){

                document.write("<h2>Please enable cookies</h2>");

                document.close();

            }

**else**{

**if**( name == "" || pass == "" ){

                    document.write("<h4>Enter both Username and Password</h4>");

                    document.close();

                }

**else**{

**return**;

                }

            }

        }

    </script>

    <form action="login" name="form1" method="post">

        Enter the name: <input type="text" name="name">

        <br>

        Enter the password: <input type="password" name="pass">

        <br>

        <input type="submit" value="submit" onClick="myFunction()">

    </form>

</body></html>

**public/WEB-INF/web.xml**

<web-app>

    <servlet>

        <servlet-name>LoginServlet</servlet-name>

        <servlet-class>LoginServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>LoginServlet</servlet-name>

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

    </servlet-mapping>

</web-app>