Servlets

Servlet technology was given by "sun microsystem".

Server :-

① web server

① tomcat (http server)
② smtp
③ FTP
④ IIS

② application server
it is having 2 containes

① weblogic server
② Glassfish server
③ JBOSS
④ Jrun
⑤ IBM
:
etc.,

Servlet API

① javax.servlet.*; (basic)
② javax.servlet.http.*; (advanced)

① Servlet(i) → javax.servlet.*;
② ServletRequest(i) → javax.servlet.*;
③ ServletResponse(i) → javax.servlet.*;
④ RequestDispather(obi) → javax.servlet.*;
⑤ GenericServlet(ab) → javax.servlet.*;
⑥ HTTPServlet(ab) → javax.servlet.http.*;
⑦ HTTPServletRequest(i) → javax.servlet.http.*;
⑧ HTTPServletResponse(i) → javax.servlet.http.*;
⑨ Filter(i) → javax.servlet.*;
⑩ ServletConfig(i) → javax.servlet.*;

if want to develop one servlet. we must follow some standards as follows.

steps to develop web application

step ① Develop web application architecture.

step ② Develop the web resources

step ③ Develop the Deployment Discripter file (web.xml)

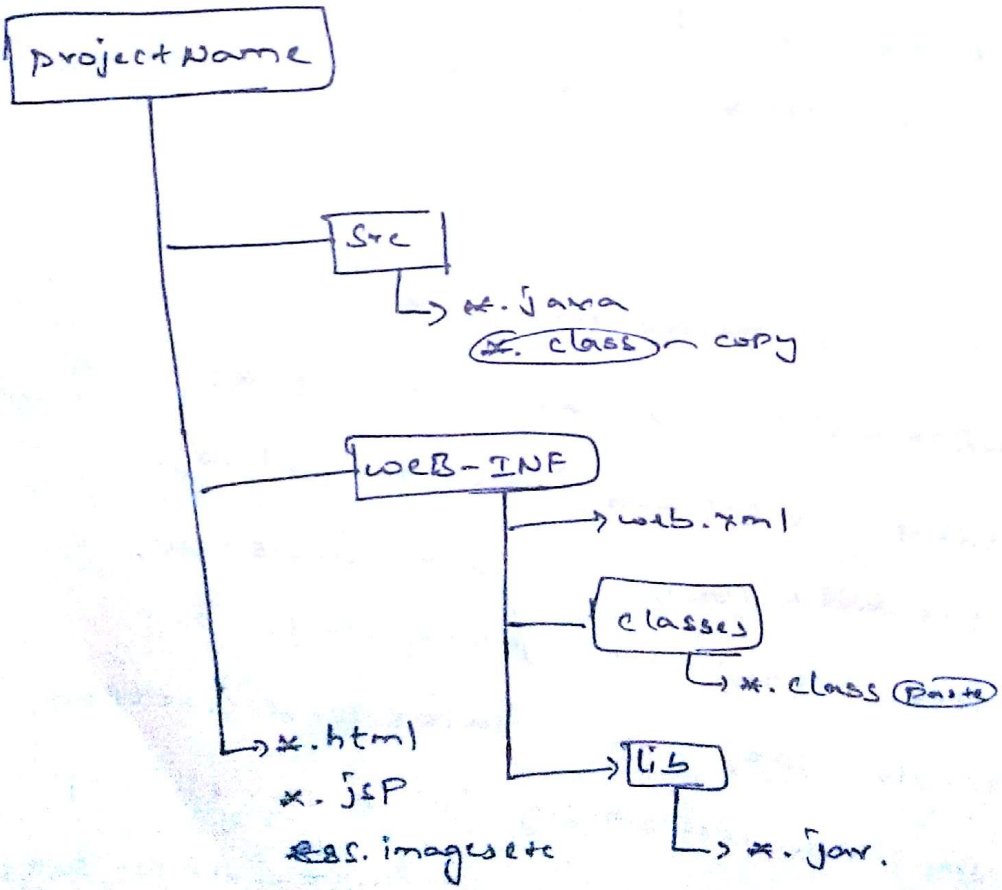
step ④ Compile the servlet java file.

⑤ Deploy the web application into server.

⑥ Start the web server.

⑦ open the browser and we can make a request.

⑧ Enter the input values and we can check. (testing)



Servlet is an interface it contains the collection of methods (it contains life cycle methods).

Servlet (i)

i) public void init (ServletConfig config) throws ServletException

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

iii) public void destroy ()

iv) public String getServletInfo

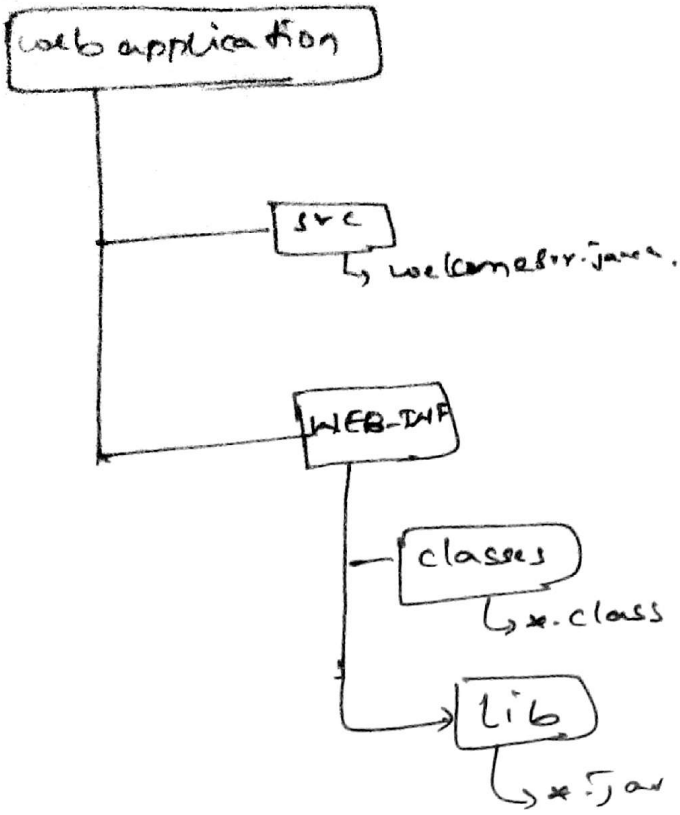
v) public ServletConfig getServletConfig ()

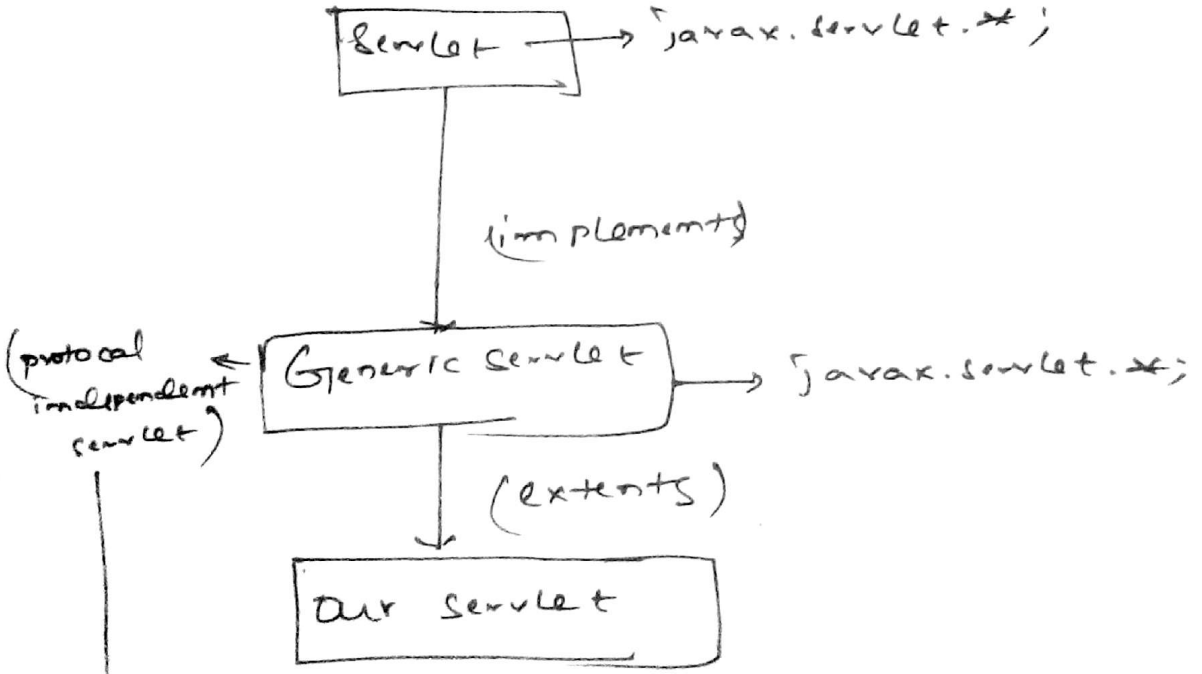
first 2 methods are comes under life cycle

remain 2 methods are comes under non-lifecycle

uses

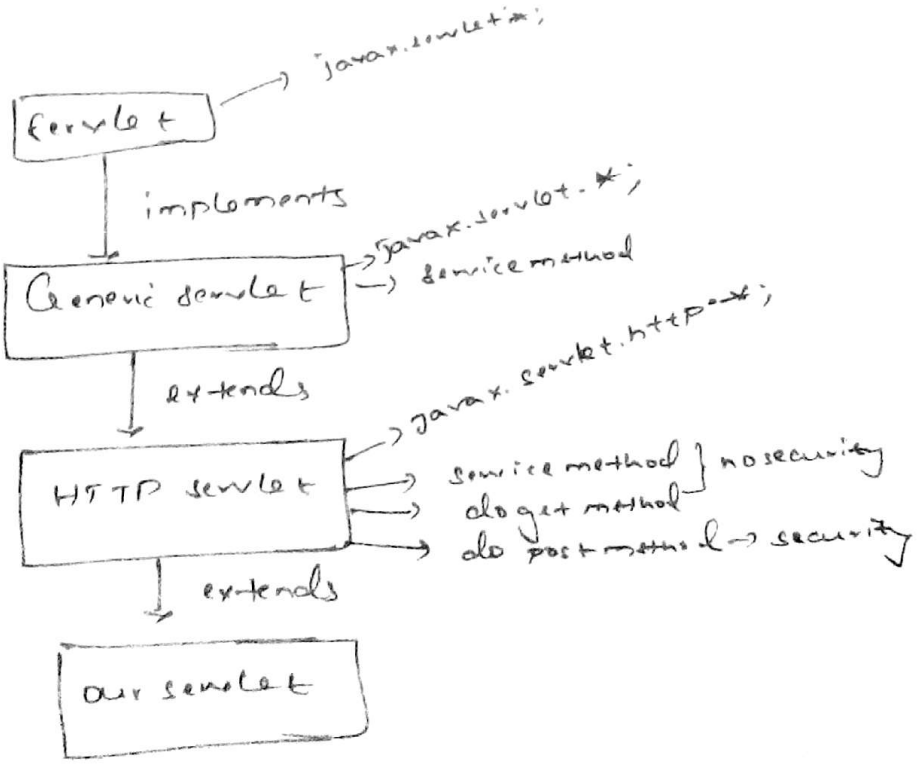
1. if we want to execute any type code once. we place init method
2. client business logic implementing in 2nd method
3. it want to destroying the our server object.
4. who is author, and version get to know 4th method
5. if want to int param, param name, param values we goes to servr get servletconfig (). (finding the information about servlet)

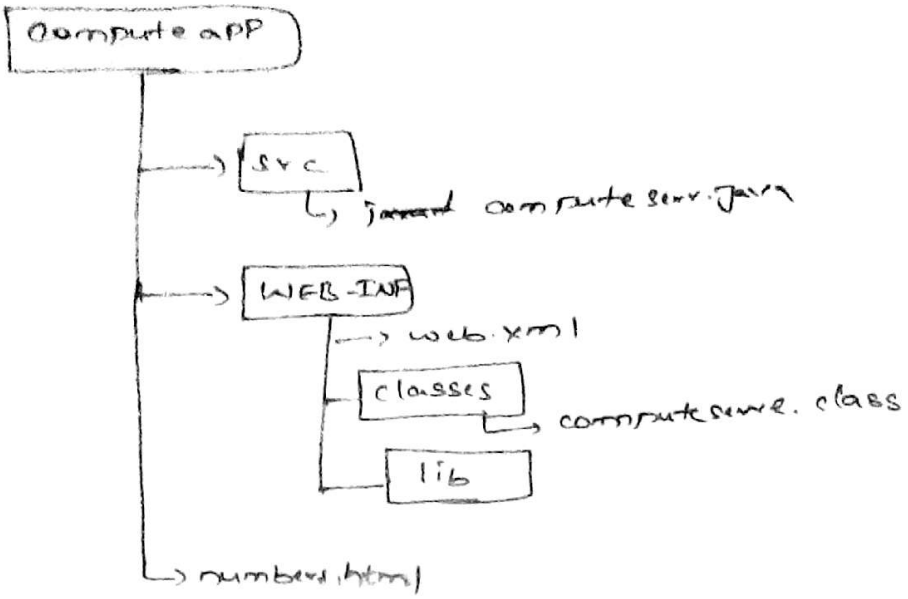




(it is supported by the all the type of protocales like HTTP, SMTP, FTP... etc)

Whenever we are using the servlet interface we need to providing the implementations of all the life cycle methods but providing the all implementating of all life cycles of methods. but providing the all implementations of life cycle there is no use of client side and also developer side. unnessarily increasing the burden on developer. In order to overcome that problem now we have to go for "generic servlet". generic servlet was implemented by servlet interface.





video 11 session missed

Servlet Communication:-

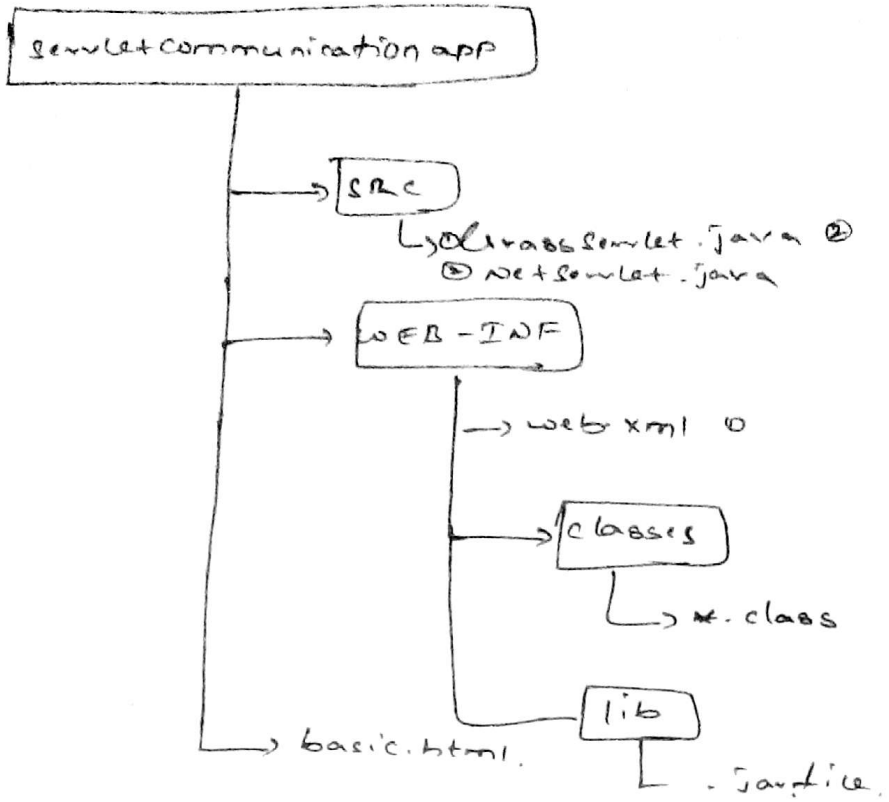
In nowadays implementing one the business logic in one servlet is not good. for the security we are using servlets different. To provide the communication blw the servlet we go for servlet communication

RequestDispatcher → javax.servlet.*;

(i) forward(req, res) } provided by these 2 methods
(ii) include(req, res) }

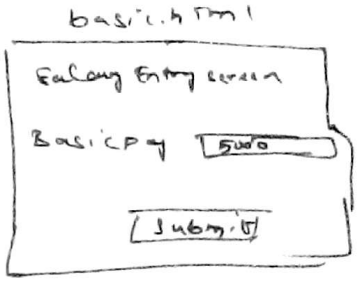
RequestDispatcher rd = request.getRequestDispatcher()

web application architecture



Now we going to make html page.

basic.html




```
<html>
<body bgcolor="yellow">
<center>
<h1> Salary Entry Screen </h1>
<form action="./grass" method="post">
BasicPay <input type="text" name="basic"><br>
<input type="submit" value="send">
</form>
</center>
</body>
</html>
```

Grass Servlet.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class Grassservlet extends HttpServlet
{
public void doPost(-,-) throws SE, IOE
{
float basic = float.parseFloat(req.getparameter("basic"));
float grass = basic*(basic*0.8f);
req.setAttribute("gr", grass);
RequestDispatcher rd = req.getRequestDispatcher("/net");
rd.forward(req, res);
}
}
```

Net servlet.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class Net servlet extends HttpServlet
{
public void doPost(- req, - res) throws SE, IOE
{
res.setContentType("text/html");
PrintWriter pw = res.getWriter();
float grass = (Float.req.getAttribute("gr"));
float net = grass-(grass*0.2f);
pw.println("<h> Net salary" + net + "</h>");
pw.close();
}
}
```

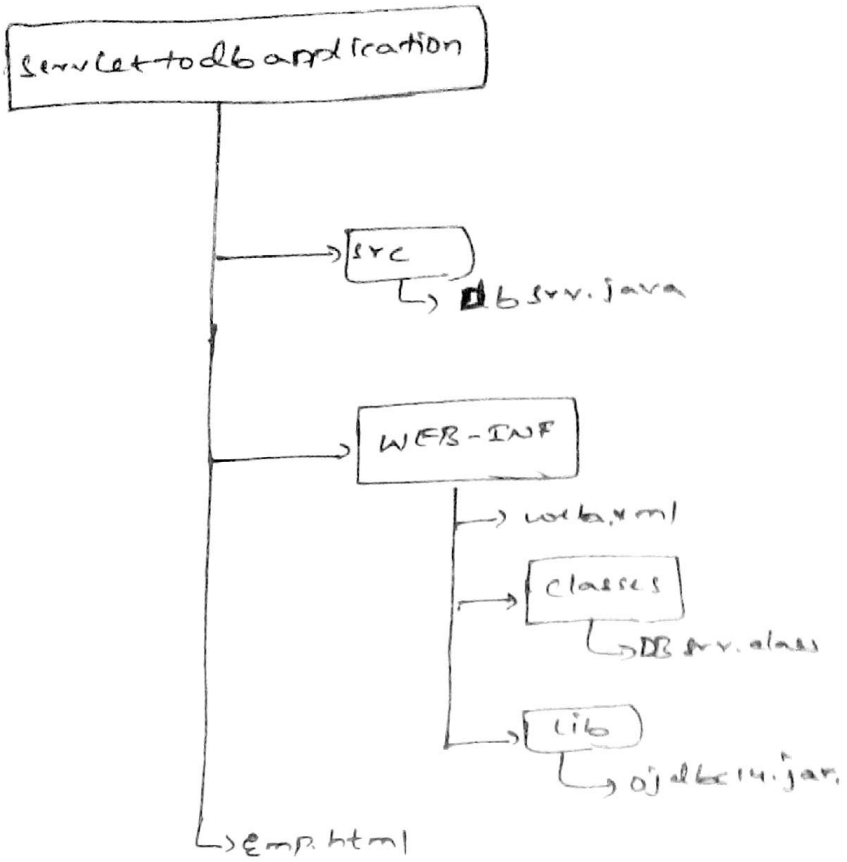
web.xml

```
<web-app>
<servlet>
<servlet-name> one </servlet-name>
<servlet-class> GrassServlet </servlet-class>
</servlet>
<servlet-mapping>
<servlet-name> one </servlet-name>
<url-pattern> /grass </url-pattern>
</servlet-mapping>

<servlet>
<servlet-name> two </servlet-name>
<servlet-class> NetServlet </servlet-class>
</servlet>
<servlet mapping>
<servlet-name> two </servlet-name>
<URL-pattern> /net </url-pattern>
</servlet-mapping>

<welcome-file-list>
<welcome-file> basic.html </welcome-file>
</welcome-file-list>
</web-app>
```

=> servlet-context is true global for application

=> servlet-config is true local

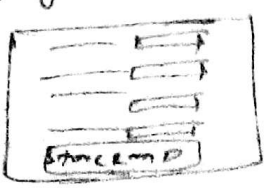
servlet to db applications



client side

Emp.html [inside browser] p.d

```
<html>
<Body bgcolor="yellow">
<center>
<form action="./dbserve" method="post">
Enter Employee Id :< input type="text" name="em.id"><br><br>
Enter the employee name :< input type="text" name="ename"><br><br>
Enter the employee salary :< input type="text" name="esal"><br><br>
Enter the employee deptno:<input type="text" name="deptno"><br><br>
<input type="submit" store Employee>
</form>
</center>
</body>
</html>
```



servlet code

in Tomcat business layer

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.sql.*;

public class DBSrv extends HTTPServlet
{
    Connection con = null;
    PreparedStatement pstmt = null;

public void doPost(-req,-res) throws ServletException, IOException
  {
    res.setContentType("Text/html");
PrintWriter pw = res.getWritter();
int empid = Integer.parseInt(req.getParameter("eid"));
String ename = Integer.parseInt(req.getparameter("ename"))
                no need here because string
float esal = Float.parseFloat(req.getparameter("esal"));
int deptno = Integer.parseInt(req.getParameter("deptno");

    try {
     class.forName("oracle.jdbc.driver.OracleDriver");
     con = DriverManager.getConnection("jdbc:oracle:thin
          :@localhost:1521:xe","sa","sa");

pstmt = con.prepareStatement("insert into employee values
                              (?,?,?,?)");

pstmt.setInt(1,empId);  ①
pstmt.setString(2,ename);  ②
pstmt.setFloat(3,esal);  ③
pstmt.setInt(4,deptno);  ④
int k=pstmt.executeUpdate();
    } catch(Exption e){
      e.printStackTrace()
    }
}
```

```
public void destroy()
 {
  try
   if ( pstmt != null)
      pstmt.close();
     if (con != null)
       con.close();
     } catch (Exception e,)
      {
      e,.printStackTrace();
      }
   }
  }
```

web.xml

```
<web-app>
<servlet>
<servlet-name> employee </servlet-name>
<servlet-class> DBServer </servlet-class>
</servlet>
<servlet-mapping>
<servlet-name> employee </servlet-name>
  <url-pattern> /dbserver </url-pattern>
  </servlet-mapping>
  <welcome-file-list>
  <welcome-file> emp.html </welcome-file>
  </welcome-file-list>
  </web-app>
```

D.I

everytime, anytime client is going to hitting the application that number of times we are going to hit the DBMS layer. so number of times hitting the Database layer will decrease our application performance. that why to overcome that problem we are using init method.

connection code inside init method

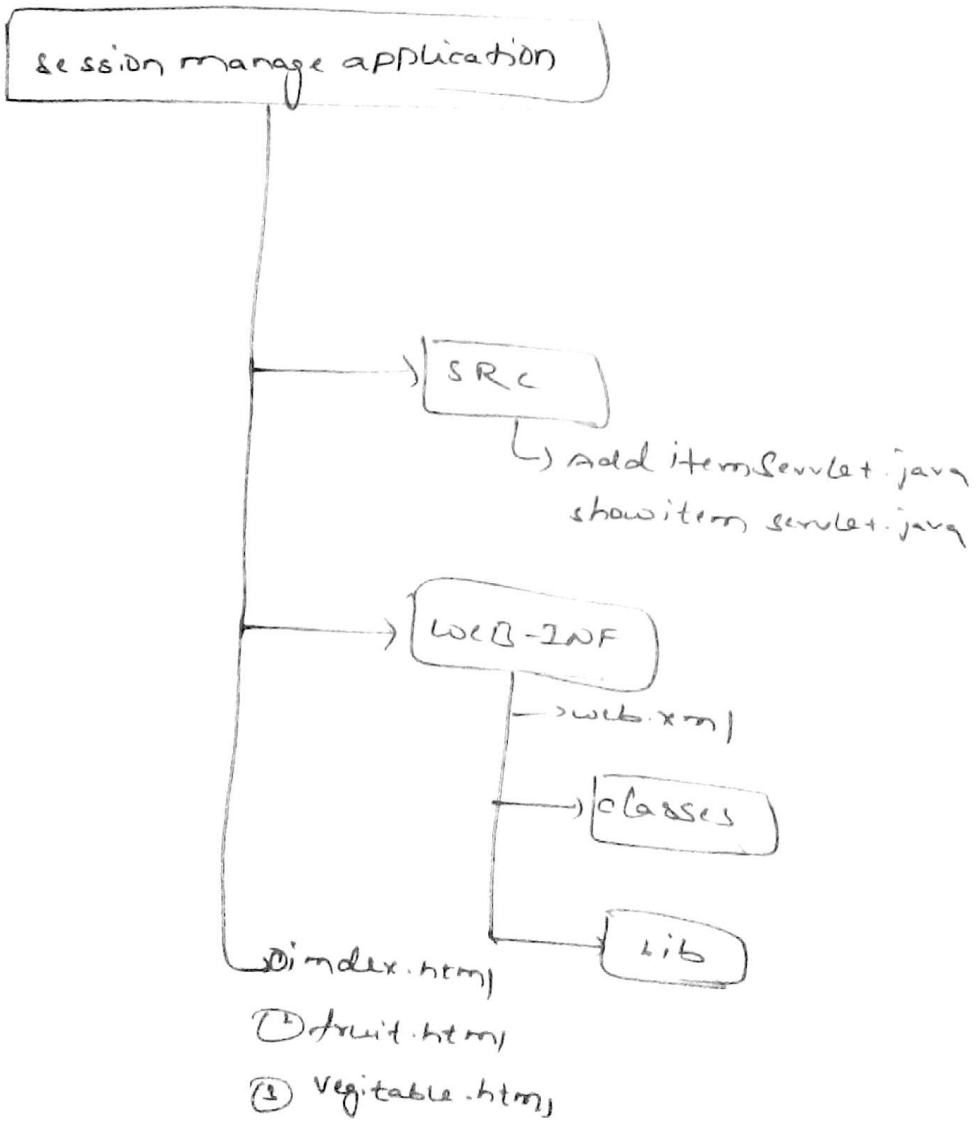
```
public void init (ServletConfig config) throws ServletException
{
try {
class.forName ("oracle.jdbc.driver.OracleDriver");
con: DriverManager.getConnection ("jdbc:oracle:thin:
        localhost:1521:xe", "u", "p");
pstmt: con.prepareStatement ("insert into employee
        values (?,?,?,?)");
} catch (Exception e) {
    e.printStackTrace();
    }
}
```

session tracking mechanism

① permenantly → oracle, mysql, db2 . . . etc

② temporarily → session tracking mechanism

- → HTTP session tracking mechanism (at server side)
- → COOKIES (at client side)
- → Hidden form fields (at client side)
- → URL Rewriting (at client side)



index.html

```
<html>
<center>
<a href "fruit.html">fruit</a>
<a href: "vegitable.html">vegitable</a>
</center>
</html>
```

fruit.html

```
<html>
<body bgcolor = "yellow">
<center>
<form action = "./Srv1" method- "Get">
EnterFruitName: <select name = "s1" >
<option value = "apple">Apple</option>
<option value - "banana" > Banana </option>
<option value - "Mango" > Mango </option>
</select>
~~Enterquantity: <input type = "submit" value = "AddItem" />~~
Ent-quantity: <input type = "text" name = "qty" />
<input type = "submit" value = "Addition" />
</form>
<form action = ./Srv2 "method" = "get" >
<input type = "submit" value = "showitem" />
</form>
</center>
</html>
</body>
```

AddItemServlet.java

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
public class AddItemServlet Extends HttpServlet
 {
public void doGet(--,--) throws SE, IOE
   {
String itemname = req.getParameter("n");
String itemqty = req.getParameter("qty");
  HttpSession hs = req.getSession(true);
          hs.setAttribute(itemname, itemqty);
       }
     }
```

ShowItemServlet.java

```
import javax.servlet.http.*;
Import javax.servlet.*;
import java.io.*;
public class ShowItemServlet extends HttpServlet
 { public void doget(--,--) throws SE, IOE
 PrintWriter pw: res.getWriter();
    HttpSession hs= req.getSession(true);
  Enumeration e = hs.getAttributeName();
     while(e.hasMoreElements())
        {
   String key = (String)e.nextElement();
   String value = (String) hs.getAttribute(key);
        }
   pw.println(key+" "+value);
      }
```

while performing the HTTPSession mechanisms, number of users are using that server, that number of sessions are going to be increased. So that performance will be decreased overally. that why we are going to see Cookies, Hiddenforms, URL Rewriting. These are work at client side.

```
String sname = req.getParameter("sname");
String address = req.getParameter("address");
Cookie c1 = new Cookie("studentname", sname);
Cookie c2 = new Cookie("studentaddress", address);
res.addCookie(c1);
res.addCookie(c2);
```

① Hiddenformfields → The data appending in the form of text box

② URL rewriting → The data is appending in the form of URL

filter