Dt : 14/6/2022

***imp**

Listeners in Servlet Programming:

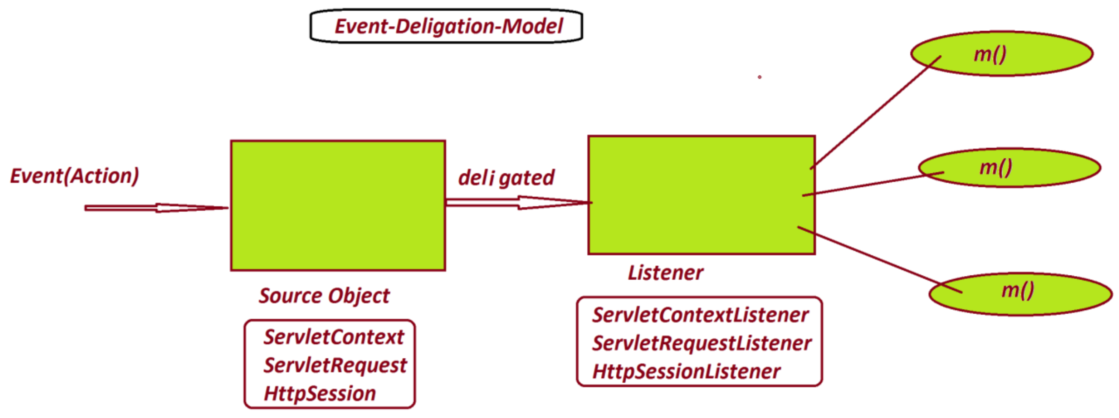
=>Listerners are the components executed background to Servlet Objects and which respond to the events(actions) performed on Servlet Objects.

Note:

=>when event(action) is performed on Servlet object deligated to Listener and the Listener will execute related method to handle the event,is known as Event-Delegation-Model.

Diagram:

Maipathii



=>we can have listeners to the following Servlet objects:

1.ServletContext

2.ServletRequest

3.HttpSession

1.ServletContext:

Event_Class : ServletContextEvent

Event_Listener : ServletContextListener

methods :

public void contextInitialized(javax.servlet.ServletContextEvent)

public void contextDestroyed(javax.servlet.ServletContextEvent)

2.ServletRequest:

Event_Class : ServletRequestEvent

Event_Listener : ServletRequestListener

methods :

public void requestInitialized(javax.servlet.ServletRequestEvent)

public void requestDestroyed(javax.servlet.ServletRequestEvent)

3.HttpSession:

Event_Class : HttpSessionEvent

Event_Listener : HttpSessionListener

methods :

public void sessionCreated(javax.servlet.http.HttpSessionEvent)

Venkatesh Maipathii

```
 public void sessionDestroyed(javax.servlet.http.HttpSessionEvent)
```

===

Ex_Program : Application to demonstrate Listeners in Servlet programming

(Update HttpSession application with following Listeners)

ContextListener.java

```
package test;

import javax.servlet.*;

import javax.servlet.annotation.*;

@WebListener

public class ContextListener implements ServletContextListener

{

        public void contextInitialized(ServletContextEvent sce) {

                System.out.println("Contect Object Initialized...");

        }

        public void contextDestroyed(ServletContextEvent sce) {

               System.out.println("Context Object Destroyed...");

        }

}
```

RequestListener.java

```
package test;

import javax.servlet.*;

import javax.servlet.annotation.*;

@WebListener
```

Venkatesh Maipathii

```
public class RequestListener implements ServletRequestListener{
        public void requestInitialized(ServletRequestEvent sre) {
                System.out.println("Request object initialized...");
        }
        public void requestDestroyed(ServletRequestEvent sre) {
                System.out.println("Request Object destroyed...");
        }
}
```

SessionListener.java

```
package test;
import javax.servlet.http.*;
import javax.servlet.annotation.*;
@WebListener
public class SessionListener implements HttpSessionListener,
HttpSessionAttributeListener{
        public void sessionCreated(HttpSessionEvent hse) {
                System.out.println("Session Created....");
        }
        public void sessionDestroyed(HttpSessionEvent hse) {
                System.out.println("Session Destroyed...");
        }
        public void attributeAdded(HttpSessionBindingEvent hsbe) {
                System.out.println("Attribute Added to Session...");
        }
```

Venkatesh Maipathii

```
        public void attributeRemoved(HttpSessionBindingEvent hsbe) {

                System.out.println("Attribute removed from Session..");

        }

}
```

==

faq:

Types of Listeners in Servlet Programming"

=>Listeners in Servlet programming are categorized into three types:

1.Application Level Listener

2.Request Level Listener

3.Session Level Listener

1.Application Level Listener:

=>The process of adding Listener to the ServletContext object is known as Application Level Listener.

2.Request Level Listener:

=>The process of adding Listener to the ServletRequest object is known as Request Level Listener.

3.Session Level Listener:

=>The process of adding Listener to the HttpSession object is known as Session Level Listener.

===

Venkatesh Maipathii

structure of web.xml from Servlet Programming:

```
<web-app>
  <context-param></context-param>
  <listener>
      Listener_Class_name
  </listener>
 <servlet>
    <servlet-name></servlet-name>
    <servlet-class></servlet-class>
    <init-param></init-param>
  </servlet>
  <servlet-mapping>
    <servlet-name></servlet-name>
    <url-pattern></url-pattern>
  </servlet-mapping>
  <filter>
       <filter-name></filter-name>
       <filter-class></filter-class>
        <init-param></init-param>
  </filter>
  <filter-mapping>
    <filter-name></filter-name>
    <url-pattern></url-pattern>
```

Venkatesh Maipathii

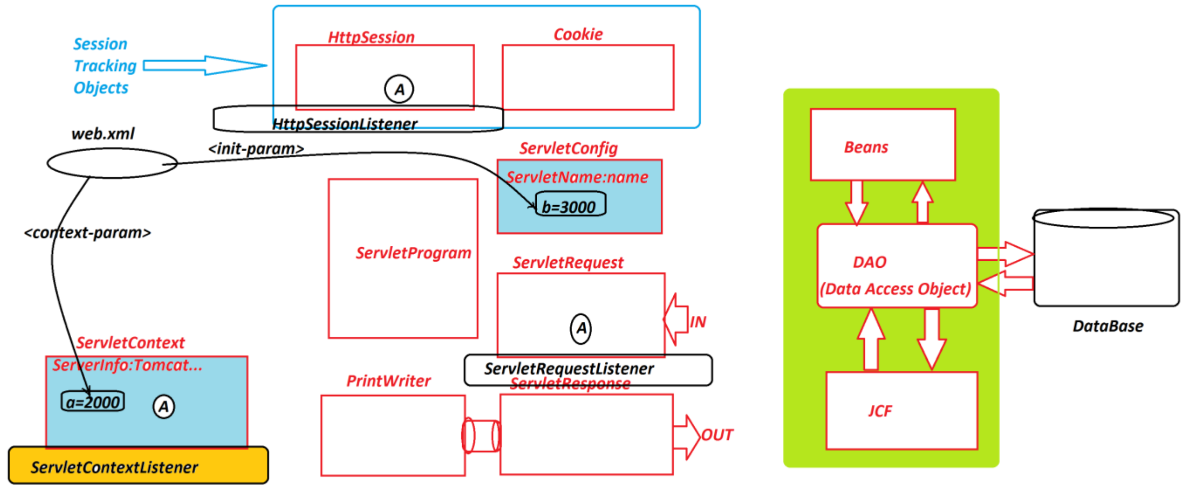
```
 </filter-mapping>

 <welcome-file-list>

  <welcome-file></welcome-file>

 </welcome-file-list>

</web-app>
```



imp

Annotations in Servlet Programming:

define Annotation?

=>The tag based information which is added to the programming component like class,Interface,method and variable is known as annotation.

=>These Annotations are represented using "@" symbol.

=>These Annotations will specify the information to the compiler at compilation stage or Execution controls at execution stage.

Venkatesh

Exp:

@ Override

@ SuppressWarnings("unchecked")

@ SuppressWarnings("rawtypes")

=>The following are some important annotations used in Servlet programming:

(a)@ WebServlet

(b)@ WebFilter

(c)@ WebInitParam

(d)@ WebListener

(a)@ WebServlet:

=>This '@ WebServlet' is declared to the ServletProgram and used for identifying Servlet program in execution process.

(b)@ WebFilter:

=>This '@ WebFilter' is declared to FilterProgram and used for identifying Filter program in execution process.

(c)@ WebInitParam:

=>This '@ WebInitParam' is used to initialize the parameters with

Venkatesh Maipathii

ServletConfig and FilterConfig objects.

(d)@ WebListener:

=>This '@ WebListener' is declared to the ListenerProgram and used for identifying listener program in execution process.

Note:

=>When we use annotations in Servlet programming,the Servlet programs can be executed without depending on web.xml mapping file.

=>All the annotations related to ServletProgramming are available from "javax.servlet.annotation" package.

==

Dt : 15/6/2022

faq:

define Servlet Collaboration?

=>The process of exchanging the information among Servlet programs is known as Servlet Collaboration.

=>Servlet Collaboration can be done in two ways:

(i)Using 'RequestDispatcher'

=>forward()

=>include()

(ii)Using 'sendRedirect()' method

Venkatesh Maipathii

define sendRedirect()?

=>sendRedirect() method is avialble from 'HttpServletResponse' and which is used

to send information to another servlet program in communication process.

=>In this process another Servlet program may be in same WebApp or diff WebAppl.

Method Signature:

public abstract void sendRedirect(java.lang.String) throws java.io.IOException;

Ex:

WebApp-1 : TestApp1

input.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
   <form action="first" method="post">
   UserName:<input type="text" name="uname"><br>
   MailId:<input type="text" name="mid"><br>
   <input type="submit" value="Display">
   </form>
</body>
</html>
```

FirstServlet.java

```
package test;

import java.io.*;

import javax.servlet.*;
```

Venkatesh Maipathii

```
import javax.servlet.http.*;

import javax.servlet.annotation.*;

@SuppressWarnings("serial")

@WebServlet("/first")

public class FirstServlet extends HttpServlet{

 protected void doPost(HttpServletRequest req,HttpServletResponse res)

 throws ServletException,IOException{

            String uName = req.getParameter("uname");

            String mId = req.getParameter("mid");

            res.sendRedirect("http://localhost:8082/TestApp2/second?uname="+

            uName+"&mid="+mId);

 }

}
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
  <welcome-file-list>
    <welcome-file>input.html</welcome-file>
  </welcome-file-list>
</web-app>
```

WebApp-2 : TestApp2

SecondServlet.java

```
package test;

import java.io.*;

import javax.servlet.*;

import javax.servlet.http.*;

import javax.servlet.annotation.*;
```

Venkatesh Maipathii

```
@SuppressWarnings("serial")
@WebServlet("/second")
public class SecondServlet extends HttpServlet{
 protected void doGet(HttpServletRequest req,HttpServletResponse res)
 throws ServletException,IOException{
		PrintWriter pw = res.getWriter();
		res.setContentType("text/html");
		pw.println("====SecondServlet====");
		pw.println("<br>Name:"+req.getParameter("uname"));
		pw.println("<br>MailId:"+req.getParameter("mid"));

 }
}
```

http://localhost:8082/TestApp2/second?uname=Alex&mid=a@gmail.com

--

faq:

wt is the diff b/w

(i)forward() method

(ii)sendRedirect() method

=>when we use forward() method,then the same request is forwarded to next servlet program.

=>when we use sendRedirect() method then new requset is generated from the WebBrowser and forwarded to next servlet program.

Venkatesh Maipathii

---

faq:

define Servlet Life-Cycle?

=>Servlet Life-Cycle demonstrates different states of Servlet program from Starting of servlet program execution to ending of Servlet program execution.

1.Loading process

2.Instantiation proccess

3.Initialization process

4.Request handling process

5.Destroying process

1.Loading process:

=>The process of identifying the servlet program based on url-pattern and loading onto WebContatiner is known as Loading prrocess.

Note:

=>we can have url-pattern in

(i)Annotation

(ii)web.xml

2.Instantiation proccess:

=>When Servlet program loaded onto WebContainer it is automatically instantiated known as Instantiation processs.

=>After instantiation process we can find the following Life-Cycle methods:

Genericservlet

Venkatesh Maipathii

(a)init()

(b)service()

(c)destroy()

HttpServlet

(a)init()

(b)doPost()/doGet()/service()

(c)destroy()

Filter

(a)init()

(b)doFilter()

(c)destroy()

faq:

define Life-Cycle methods?

=>The methods which are executed automatically in same order are known as Life-Cycle methods.

3.Initialization process:

=>The process of making the programming components ready for execution is known as Initialization process.

=>we use init() method to perform initialization.

Note:

=>Through Initialization process we can make Bean objects and DAO objects ready for execution.

=>This initialization process is performed only once.

Venkatesh Maipathii

4.Request handling process:

=>The process of taking the request and providing the response is known as Request Handling process.

=>we use service()/doPost()/doGet()/doFilter() methods to perform Request Handling process.

Note:

=>In Multi-User Applications,

=>init() method is executed only once,but doPost()/doGet()/service() methods are executed for all the multiple Users.

5.Destroying process:

=>The process of closing or destroying the resources after execution is known as Destroying process.

=>we use destroy() method to perform destroying process.

==

Summary of Objects from JDBC:

1.Connection Object

2.Statement Object

3.PreparedStatement Object

4.CallableStatement Object

5.NonScrollable ResultSet Object

6.Scrollable ResultSet object

7.DatabaseMetaData object

8.ParameterMetaData object

Venkatesh Maipathii

9.ResultSetMetaData object

10.RowSet Object

  (a)JdbcRowSet

  (b)CachedRowSet

    =>WebRowSet

      (i)FilteredRowSet

      (ii)JoinRowset

*

11.Connection Pooling Object

  =>Vector<E> Object

Summary of Objects from Servlet Programming:

1.ServletContext object

2.ServletConfig object

3.ServletRequest/HttpServletRequest object

4.ServletResponse/HttpServletResponse object

5.PrintWriter object

6.Cookie object

7.HttpSession object

8.Bean class object

9.DAO Layer objects

10.JCF object

Venkatesh Maipathii

===
==

****imp***

JSP Programming:(Unit-3)

=>JSP Stands for 'Java Server Page' and which is response from WebApplication.

=>JSP is tag based programming language and which is more easy when compared to Servlet programming.

=>Programs in JSP are saved with (.jsp) as an extention.

=>JSP programs are combination of both HTML code and Java Code.

=>JSP provides the following tags to write JavaCode part of JSP programs:

1.Scripting tags

=>Scriptlet tag

=>Expression tag

=>Declarative tag

2.Directive tags

=>page

=>include

=>taglib

3.Action tags

=>jsp:include

=>jsp:forward

=>jsp:param

Venkatesh Maipathii

=>jsp:useBean

=>jsp:setProperty

=>jsp:getProperty

Dt : 16/6/2022

1.Scripting tags:

=>Scripting tags are used to write JavaCode part of JSP programs.

=>Scripting tags are categorized into the following:

(a)Scriptlet tag

(b)Expression tag

(c)Declarative tag

(a)Scriptlet tag:

=>Scriptlet tag is used to write normal JavaCode part of JSP programs

syntax:

<% ---JavaCode--- %>

(b)Expression tag:

=>Expression tag is used to assign the value to variable or which is used to display the data to the WebBrowser.

syntax:

Venkatesh Maipathii

<%= expression %>

(c)Declarative tag:

=>Declarative tag is used to declare variables and methods in JSP programs.

syntax:

<%! variables;methods %>

==

2.Directive tags:

=>The tags which are used to specify the directions in translation process are known as Directive Tags.

The following are the types of Directive tags:

(a)page

(b)include

(c)taglib

(a)page:

=>'page' directive tag specifies the translator to add the related attribute to current JSP page.

Venkatesh Maipathii

syntax:

<%@ page attribute="value" %>

exp:

<%@ page import="java.util.*"%>

List of attributes:

1.import

2.contentType

3.extends

4.info

5.buffer

6.language

7.isELIgnored

8.isThreadSafe

9.autoFlush

10.session

11.pageEncoding

12.errorPage

13.isErrorPage

(b)include:

=>'include' directive tag specifies the file to be included to current

Venkatesh Maipathii

JSP page.

syntax:

<%@ include file="file-name"%>

Exp:

<%@ include file="input.html" %>

(c)taglib:

=>'taglib' directive tag specifies to add specified url to current JSP page and which is used part of EL(Expression Lang) and JSTL (JSP Standard Tag Lib).

syntax:

<%@ taglib url="urloftaglib" prefix="prefixoftaglib"%>

==

Exp program1:

JSP Application to calculate factorial of given number.

Note:

=>JSP files are created part of WebContent.

input.html

```
<!DOCTYPE html>
<html>
```

Venkatesh Maipathii

```
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
 <form action="Display.jsp" method="post">
 Enter the Value:<input type="text" name="v"><br>
 <input type="submit" value="Factorial">
 </form>
</body>
</html>
```

Display.jsp

```
<%@ page language="java"
        contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"
        errorPage="Error.jsp"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%!
int fact;
int factorial(int n)
{
     fact=1;
     for(int i=n;i>=1;i--)
     {
          fact=fact*i;
     }
     return fact;
}
%>
<%
int val = Integer.parseInt(request.getParameter("v"));
int result = factorial(val);
out.println("Factorial : "+result+"<br>");
%>
<%@include file="input.html"%>
</body>
</html>
```

Errorr.jsp

Venkatesh Maipathii

```
<%@ page language="java"
         contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"
        isErrorPage="true"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
out.println("Enter only Integer value...<br>");
%>
<%= exception %>
<br>
<%@include file="input.html"%>
</body>
</html>
```

web.xml

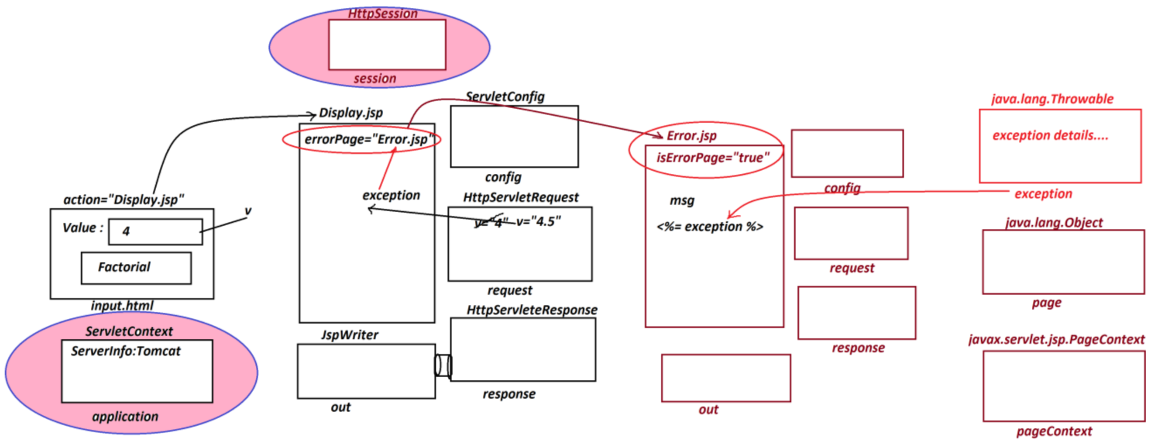
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
  <welcome-file-list>
     <welcome-file>input.html</welcome-file>
  </welcome-file-list>
</web-app>
```

Execute the application as follows:

http://localhost:8082/JSPApp1

Diagram:

Venkatesh Maipathii



===

=>The following are the implicit objects of JSP:

application	-	***javax.servlet.ServletContext***
config	-	***javax.servlet.ServletConfig***
request	-	***javax.servlet.http.HttpServletRequest***
response	-	***javax.servlet.http.HttpServletResponse***
out	-	***javax.servlet.jsp.JspWriter***
session	-	***javax.servlet.http.HttpSession***
exception	-	***java.lang.Throwable***
page	-	***java.lang.Object***
pageContext	-	***javax.servlet.jsp.PageContext***

==

3.Action tags:

=>Action Tags are used to include some basic actions like inserting

Venkatesh Maipati

some other page resources ,forwarding the request to another page, creating or locating the JavaBean instances and,setting and retriving the bean properties in JSP pages.

Note:

=>These action tags are used in Execution process at runtime.

--

=>The following are some Important action tags available in JSP:

1.<jsp:include>

2.<jsp:forward>

3.<jsp:param>

4.<jsp:useBean>

5.<jsp:setProperty>

6.<jsp:getProperty>

1.<jsp:include> :

=>This action tag allows to include a static or dynamic resource such as HTML or JSP specified by a URL to be included in the current JSP while processing request.

=>If the resource is static then its content is included in the JSP page.

Venkatesh Maipathii

=>If the resource is dynamic then its result is included in the JSP page.

syntax:

```
<jsp:include attributes>
  <---Zero or more jsp:param tags--->
</jsp:include>
```

attributes of include tag:

****imp***

page : Takes a relative URL,which locates the resource

to be included in the JSP page.

```
<jsp:include page="/Header.html"/>
<jsp:include page="<%=mypath%>"/>
```

flush : Takes true or false,which indicates whether or not the buffer needs to be flushed before including resource.

2.<jsp:forward>:

=> This action tag forwards a JSP request to another resource and which can be either static or dynamic.

=>If the resource is dynamic then we can use a jsp:param tag to pass

Venkatesh Maipathii

name and value of the parameter to the resource.

sntax:

```
     <jsp:forward attributes>
      <-- Zero or more jsp:param tags-->
     </jsp:forward>
```

Exp:

```
   <jsp:forward page="/Header.html"/>
   <jsp:forward page="<%=mypath%>"/>
```

3.<jsp:param>:

This action tag is used to hold the parameter with value and which is to be forwarded to the next resource.

syntax:

```
 <jsp:param name="paramName"  value="paramValue"/>
```

  ===

4.<jsp:useBean>:

=>This tag is used to instantiate a JavaBean,or locate an existing bean instance and assign it to a variable name(id).

syntax:

Venkatesh Maipathii

<jsp:useBean attributes>

<!-optional body content--->

</jsp:useBean>

Attributes of <jsp:useBean> tag:

(a)id

(b)scope

(c)class

(d)beanName

(e)type

(a)id:

=>which represents the variable name assigned to id attribute of <jsp:useBean> tag and which holds the reference of JavaBean instance.

(b)scope:

=>which specifies the scope in which the bean instance has to be created or located.

scope can be the following:

(i)page scope : within the JSP page,until the page sends response.

(ii)request scope : JSP page processing the same request until a JSP sends response.

(iii)session scope - Used with in the Session.

(iv)application scope - Used within entire web application.

Venkatesh Maipathii

imp

(c)class :

=>The class attribute takes the qualified class name to create a bean instance.

(d)beanName :

=>The beanName attribute takes a qualified class name.

(e)type :

=>The "type" attribute takes a qualified className or interfaceName, which can be the classname given in the class or beanName attribute or its super type.

5.<jsp:setProperty>:

=>This action tag sets the value of a property in a bean, using the bean's setter methods.

Types of attributes:

(a)name

(b)property

(c)value

(d)param

Venkatesh Maipathii

(a)name:

=>The name attribute takes the name of already existing bean as a reference variable to invoke the setter method.

(b)property:

=>which specifies the property name that has to be set,and specifies the setter method that has to be invoked.

(c)value:

=>The value attribute takes the value that has to be set to the specified bean property.

(d)param:

=>which specify the name of the request parameter whose value to be assigned to bean property.

6.<jsp:getProperty>:

=>This action tag gets the value of a property in a bean by using the bean's getter method and writes the value to the current JspWriter.

Types of attributes:

(a)name

(b)property

Venkatesh Maipathii

(a)name:

=>The name attribute takes the reference variable name on which we want to invoke the getter method.

(b)property:

=>which gets the value of a bean property and invokes the getter method of the bean property.

The following are some rare used Actions tags:

7.< jsp:plugIn >:

The <jsp:plugin> action tag provide easy support for including a java applet in the client Web browser, using a built-in or downloaded java plug-in.

Syntax:

```
<jsp:plugin attributes>
  <!-optionally one jsp:params  or jsp:fallback tag-
</jsp:plugin>
```

8. < jsp:fallBack >

The <jsp:fallback> action tag allows us to specify a text message to be displayed if the required plug-in cannot run and this action tag

Venkatesh Maipathii

must be used as a child tag with the <jsp:plugin> action tag.

Syntax:

<jsp:fallback>

Test message that has to be displayed if the plugin cannot be started

</jsp:fallback>

9. < jsp:params >

The <jsp:params> action tag sends the parameters that we want to pass to an applet.

Syntax:

<jsp:params>

<!-one or more jsp:param tags---

</jsp:params>

==

Exp application_2:

JSP Application to demonstrate Login process?

(Using <jsp:forward> <jsp:include> <jsp:param>

DBConnection.java

```
package test;
import java.sql.*;
public class DBConnection {
    private static Connection con=null;//reference variable
```

Venkatesh Maipathii

```
    private DBConnection() {}
    static
    {
     try {
          Class.forName("oracle.jdbc.driver.OracleDriver");
          con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1521:xe","system","manager");
     }catch(Exception e) {e.printStackTrace();}
    }
    public static Connection getCon()
    {
     return con;
    }
}
```

LoginDAO.java

```
package test;

import java.sql.*;

import javax.servlet.http.*;

public class LoginDAO {

    public String fName=null;

    public String login(HttpServletRequest req) {

        try {

                Connection con = DBConnection.getCon();

                PreparedStatement ps = con.prepareStatement

                ("select * from UserReg45 where uname=? and pword=?");

                ps.setString(1,req.getParameter("uname"));

                ps.setString(2,req.getParameter("pword"));

                ResultSet rs = ps.executeQuery();

                if(rs.next()) {
```

Venkatesh Maipathii

```
                        fName = rs.getString(3);
                   }
              }catch(Exception e) {e.printStackTrace();}
              return fName;
       }
}
```

login.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
  <form action="LoginJSP.jsp" method="post">
  UserName:<input type="text" name="uname"><br>
  PassWord:<input type="password" name="pword"><br>
  <input type="submit" value="Login">
  <a href="register.html">NewUser?</a>
  </form>
</body>
</html>
```

LoginJSP.jsp

```
<%@ page language="java"
         contentType="text/html; charset=ISO-8859-1"
         pageEncoding="ISO-8859-1"
         import="test.*"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
String fName = new LoginDAO().login(request);
if(fName==null){
     out.println("Invalid Login process...<br>");
```

Venkatesh Maipathii

```
      %>
      <jsp:include page="login.html"/>
      <%
}else{
      %>
      <jsp:forward page="WelcomeJSP.jsp">
        <jsp:param value="<%=fName %>" name="fname"/>
      </jsp:forward>
      <%
}
%>
</body>
</html>
```

WelcomeJSP.jsp

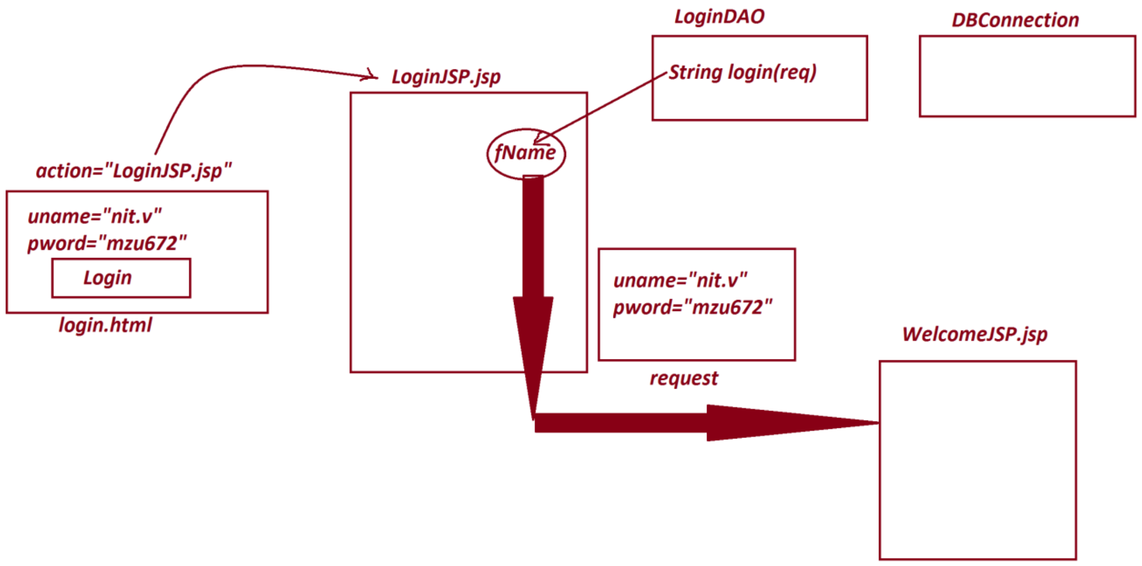
```
<%@ page language="java"
         contentType="text/html; charset=ISO-8859-1"
         pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
String fName = request.getParameter("fname");
out.println("Wlcome User : "+fName+"<br>");
%>
</body>
</html>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
  <welcome-file-list>
    <welcome-file>login.html</welcome-file>
  </welcome-file-list>
</web-app>
```

diagram:

Venkatesh Maipathii



Assignment:

Update above application by displaying the complete user details

----

Venkatesh Ma