**Java Server pages (JSP)**

JSP technology gives a web and java developers a simple yet powerful mechanism for creating web applications.

Java Server Pages or JSP for short is Sun's solution for developing dynamic web sites. JSP provide excellent server side scripting support for creating database driven web applications.

JSP is a language for developing JSP pages, which are text based document that describes how to process request and construct response.

It allows creating web contents with static and dynamic web components.

JSP used for generating dynamic web pages at server side.

* Java code embedded in HTMl.
* More HTML code less java code.
* Separate presentation and business logic.
* Template based content generation.
* JSP page translated into a java source file that contains a servlet class definition.

First\_jsp

Byte code

WEB CONTAINER

WEB SERVER First.jsp First\_jsp.java First\_jsp.class

First step, the web container translates the JSP file into java source file that contains the servlet class definition.

Second step, the web container compiles the servlet source code into a java class file.

The third step, the servlet class byte code is loaded into web-container’s JVM using a class loader.

The fourth step, the web container creates an instance of the servlet class.

The fifth step, the container initializes the servlet by calling the jspInit() method.

The sixth step, with each request the container class calls the \_jspService() method for the JSP page.

The seventh step, When the web container removes the JSP servlet instance from the service,it first calls the jspDestroy() method.

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

JSP technology allows you to easily create Web content that has both static and dynamic components.

**Feature:**

* Seperation of static and dynamic contents.
* Write once run anywhere:

JSP pages can be moved easily across platforms, and across web servers, without any changes.

Servers available for executing JSP pages.

* Tomcat from Apache
* Blazix from Desiderata software
* WebLogic from BEA systems.
* WebSphere from IBM

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

# JSP ARCHITECTURE

**J**SP pages are high level extension of servlet and it enable the developers to embed java code in html pages. JSP files are finally compiled into a servlet by the JSP engine. Compiled servlet is used by the engine to serve the requests.

*javax.servlet.jsp* package defines two interfaces:

* *JSPPage*
* *HttpJspPage*

These interfaces defines the three methods for the compiled JSP page. These methods are:

* *jspInit()*
* *jspDestroy()*
* *\_jspService(HttpServletRequest request,HttpServletResponse response)*

In the compiled JSP file these methods are present. Programmer can define *jspInit()* and *jspDestroy()* methods, but the \_*jspService(HttpServletRequest request,HttpServletResponse response)* method is generated by the JSP engine.

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

# INTRODUCTION TO JSP TAGS

**I**n this lesson we will learn about the various tags available in JSP with suitable examples. In JSP tags can be devided into 4 different types. These are:  
**A. Directives**

In the directives we can import packages, define error handling pages or the session information of the JSP page.  
  Syntax of JSP directives is:

<%@directive attribute="value" %>

Where directive may be:

1. **page**:

page is used to provide the information about it.  
Example: <%@page language="java" %>   
 Attributes:

1. language="java"  
   This tells the server that the page is using the java language. Current JSP specification supports only java language.  
   Example: <%@page language="java" %>
2. extends="mypackage.myclass"  
   This attribute is used when we want to extend any class.
3. import

We can use comma(,) to import more than one packages.  
Example: <%@page language="java" import="java.sql.\*,mypackage.myclass" %>

1. session="true"  
   When this value is true session data is available to the JSP page otherwise not. By default this value is true.  
   Example: <%@page language="java" session="true" %>
2. errorPage="error.jsp"  
   errorPage is used to handle the un-handled exceptions in the page.  
   Example: <%@page language="java" session="true" errorPage="error.jsp"  %>
3. isErrorPage="true/false"  
    isE rrorPage is used to handle the un-handled exceptions in the page.  
   Example: <%@page language="java" session="true" isErrorPage="true %>
4. contentType="text/html;charset=ISO-8859-1"  
   Use this attribute to set the mime type and character set of the JSP.  
   Example: <%@page language="java" session="true" contentType="text/html;charset=ISO-8859-1"  %>
   1. **include directory:**

* The *include* directives is used to include the contents of the file at any location within the JSP page.
* This directory allows including static html or JSP files at the time the JSP page is translated into a Servlet.
* The files usually used for navigation, tables, headers and footers that are common to multiple pages.
* The possible attribute is *file*

Example: <%@ include file="/header.jsp" %> 

* 1. **taglib directory:**

taglib is used to use the custom tags in the JSP pages (custom tags allows us to defined our own tags).

* Attributes
  + - 1. Uri-Uniform Resource Identifier that identify the tag library descriptor, which is used to uniquely name the set of custom tags and informs the server what to do the specified tags.
      2. Prefix-It is used to define custom tags. It is prefix,which is used in the action elements names for all actions in the library

Example: <%@ taglib uri="tlds/taglib.tld" prefix="mytag" %>

* 1. **Declarations**This tag is used for defining the functions and variables to be used in the JSP.  
      <%!  
       //java codes  
        %>

JSP Declaratives begins with <%! and ends %> with .We can embed any amount of java code in the JSP Declaratives. Variables and functions defined in the declaratives are class level and can be used anywhere in the JSP page.

* + 1. **Scriplets**  
       In this tag we can insert any amount of valid java code and these codes are placed in \_*jspService* method by the JSP engine.  
        JSP Scriptlets begins with

<%

//java codes

%>

* + 1. **Expressions**We can use this tag to output any data on the generated page. These data are automatically converted to string and printed on the output stream.  
       Ans::  <%=  5\*2   %>

Current Date **:**:- <% =new java.util.Date() %>

**Output:**

Ans::  10

Current Date::- Thu Aug 03 10:10:10 GMT+5.30 2009

**JSP Actions**

What is JSP Actions?

Servlet container provides many built in functionality to ease the development of the applications. Programmers can use these functions in JSP applications. The JSP Actions tags enables the programmer to use these functions. The JSP Actions are XML tags that can be used in the JSP page.

Here is the list of JSP Actions:

* jsp:include   
  The jsp:include action work as a subroutine, the Java servlet temporarily passes the request and response to the specified JSP/Servlet. Control is then returned back to the current JSP page.  
    < jsp:include  > </ jsp:include  >
* jsp:param   
  The jsp:param action is used to add the specific parameter to current request. The jsp:param tag can be used inside a jsp:include, jsp:forward or jsp:params block.
* jsp:forward   
  The jsp:forward tag is used to hand off the request and response to another JSP or servlet. In this case the request never return to the calling JSP page.
* jsp:plugin   
  In older versions of Netscape Navigator and Internet Explorer; different tags is used to embed applet. The jsp:plugin tag actually generates the appropriate HTML code the embed the Applets correctly.
* jsp:useBean   
  The jsp:useBean  tag is used to instantiate an object of Java Bean or it can re-use existing java bean object.
* jsp:getProperty   
  The jsp:getProperty is used to get specified property from the JavaBean object.
* jsp:setProperty   
  The jsp:setProperty tag is used to set a property in the JavaBean object.

**JSP Implicit Objects**

Implicit objects in jsp are the objects that are created by the container automatically and the container makes them available to the developers, the developer do not need to create them explicitly. Since these objects are created automatically by the container and are accessed using standard variables;

There are nine implicit objects. Here is the list of all the implicit objects:

|  |  |
| --- | --- |
| Object | Class |
| application | javax.servlet.ServletContext |
| config | javax.servlet.ServletConfig |
| exception | java.lang.Throwable |
| out | javax.servlet.jsp.JspWriter |
| page | java.lang.Object |
| PageContext | javax.servlet.jsp.PageContext |
| request | javax.servlet.ServletRequest |
| response | javax.servlet.ServletResponse |
| session | javax.servlet.http.HttpSession |

* **application**: These objects has an application scope. These objects are available at the widest context level, that allows to share the same information between the JSP page's servlet and any Web components with in the same application.
* **config**: These object has a page scope and is an instance of javax.servlet.ServletConfig class. Config object allows to pass the initialization data to a JSP page's servlet. Parameters of this objects can be set in the deployment descriptor (web.xml) inside the element <jsp-file>. The method getInitParameter() is used to access the initialization parameters.
* **exception**: This object has a page scope and is an instance of java.lang.Throwable class. This object allows the exception data to be accessed only by designated JSP "error pages."
* **out**: This object allows us to access the servlet's output stream and has a page scope. Out object is an instance of javax.servlet.jsp.JspWriter class. It provides the output stream that enable access to the servlet's output stream.
* **page**: This object has a page scope and is an instance of the JSP page's servlet class that processes the current request. Page object represents the current page that is used to call the methods defined by the translated servlet class. First type cast the servlet before accessing any method of the servlet through the page.
* **pagecontext**: PageContext has a page scope. Pagecontext is the context for the JSP page itself that provides a single API to manage the various scoped attributes. This API is extensively used if we are implementing JSP custom tag handlers. PageContext also provides access to several page attributes like including some static or dynamic resource.
* **request**: Request object has a request scope that is used to access the HTTP request data, and also provides a context to associate the request-specific data. Request object implements javax.servlet.ServletRequest interface. It uses the getParameter() method to access the request parameter. The container passes this object to the \_jspService() method.
* **response**: This object has a page scope that allows direct access to the HTTPServletResponse class object. Response object is an instance of the classes that implements the javax.servlet.ServletResponse class. Container generates to this object and passes to the \_jspService() method as a parameter.
* **session**: Session object has a session scope that is an instance of javax.servlet.http.HttpSession class. Perhaps it is the most commonly used object to manage the state contexts. This object persist information across multiple user connection.

**JSP Scopes**

**There are four scope in JSP**

One of the most powerful features of JSP is that a JSP page can access, create, and modify data objects on the server. You can then make these objects visible to JSP pages. When an object is created, it defines or defaults to a given scope. The container creates some of these objects, and the JSP designer creates others.

The *scope* of an object describes how widely it's available and who has access to it. For example, if an object is defined to have page scope, then it's available only for the duration of the current request on that page before being destroyed by the container. In this case, only the current page has access to this data, and no one else can read it. At the other end of the scale, if an object has application scope, then any page may use the data because it lasts for the duration of the application, which means until the container is switched off.

**Page Scope**

Objects with *page scope* are accessible only within the page in which they're created. The data is valid only during the processing of the current response; once the response is sent back to the browser, the data is no longer valid. If the request is forwarded to another page or the browser makes another request as a result of a redirect, the data is also lost.

**Request Scope**

Objects with *request scope* are accessible from pages processing the same request in which they were created. Once the container has processed the request, the data is released. Even if the request is forwarded to another page, the data is still available though not if a redirect is required.

* How to set request scope attributes

1. Using request.setAttribute(String name,PageContext.REQUEST\_SCOPE)
2. Using pageContext.setAttribute()

**Session Scope**

Objects with *session scope* are accessible from pages processing requests that are in the same session as the one in which they were created. A *session* is the time users spend using the application, which ends when they close their browser, when they go to another Web site, or when the application designer wants (after a logout, for instance). So, for example, when users log in, their username could be stored in the session and displayed on every page they access. This data lasts until they leave the Web site or log out.

**Application Scope**

Objects with *application scope* are accessible from JSP pages that reside in the same application. This creates a global object that's available to all pages.

Application scope uses a single namespace, which means all your pages should be careful not to duplicate the names of application scope objects or change the values when they're likely to be read by another page (this is called *thread safety*). Application scope variables are typically created and populated when an application starts and then used as read-only for the rest of the application.

**Example of declaration and expression**

<%@page contentType="text/html" %>

<html>

<body>

<%!  
int cnt=0;  
private int getCount()

{  
//increment cnt and return the value  
cnt++;  
return cnt;  
}  
%>

<p>Values of Cnt are:</p>

<p><%=getCount()%></p>

<p><%=getCount()%></p>

</body>

</html>

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

<%@page contentType="text/html" import="java.util.\*" %>

<html>  
<body>  
<p>&nbsp;</p>  
<div align="center">  
<center>  
<table border="0" cellpadding="0" cellspacing  
="0" width="460" bgcolor="#EEFFCA">

<tr>  
<td width="100%"><font size="6" color  
="#008000">&nbsp;Date Example</font></td>

</tr>  
<tr>  
<td width="100%"><b>&nbsp;Current Date   
and time is:&nbsp; <font color="#FF0000">  
<%= new java.util.Date() %>  
</font></b></td>  
</tr>  
</table>  
</center>  
</div>  
</body>  
</html>

**Working with sessions**

the session between different JSP pages. In any web application user moves from one page to another and it becomes necessary to track the user data and objects throughout the application. JSP provide an implicit object "session", which can be use to save the data specific the particular to the user.

In this tutorial we will create an application that takes the user name from the user and then saves into the user session. We will display the saved data to the user in another page.

Here is the code of the JSP file (savenameform.jsp) that takes the input from user:

<%@ page language="java" %>  
<html>  
<head>  
<title>Name Input Form</title>  
</head>-0  
<body>  
<form method="post" action="savenametosession.jsp">  
<p><b>Enter Your Name: </b><input type="text" name="username"><br>  
<input type="submit" value="Submit">  
  
</form>  
  
</body>

Here is the code of savenametosession.jsp:

<%@ page language="java" %>  
<%  
String username=request.getParameter("username");  
if(username==null) username="";  
  
session.setAttribute("username",username);  
%>  
  
<html>  
<head>  
<title>Name Saved</title>  
</head>  
<body>  
<p><a href="showsessionvalue.jsp">Next Page to view the session value</a><p>  
  
</body>

Here is the code of showsessionvalue.jsp:

<%@ page language="java" %>  
<%  
String username=(String) session.getAttribute("username");  
if(username==null) username="";  
%>  
<html>  
<head>  
<title>Show Saved Name</title>  
</head>  
<body>  
<p>Welcome: <%=username%><p>  
  
</body>

Cookie Class

In JSP cookie are the object of the class javax.servlet.http.Cookie. This class is used to creates a cookie, a small amount of information sent by a servlet to a Web browser, saved by the browser, and later sent back to the server. A cookie's value can uniquely identify a client, so cookies are commonly used for session management. A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.

The getCookies() method of the request object returns an array of Cookie objects. Cookies can be constructed using the following code:

Cookie objects have the following methods.

|  |  |
| --- | --- |
| Method | Description |
| sgetComment() | Returns the comment describing the purpose of this cookie, or null if no such comment has been defined. |
| getMaxAge() | Returns the maximum specified age of the cookie. |
| getName() | Returns the name of the cookie. |
| getPath() | Returns the prefix of all URLs for which this cookie is targeted. |
| getValue() | Returns the value of the cookie. |
| setComment(String) | If a web browser presents this cookie to a user, the cookie's purpose will be described using this comment. |
| setMaxAge(int) | Sets the maximum age of the cookie. The cookie will expire after that many seconds have passed. Negative values indicate the default behavior: the cookie is not stored persistently, and will be deleted when the user web browser exits. A zero value causes the cookie to be deleted |
| setPath(String) | This cookie should be presented only with requests beginning with this URL. |
| setValue(String) | Sets the value of the cookie. Values with various special characters (white space, brackets and parentheses, the equals sign, comma, double quote, slashes, question marks, the "at" sign, colon, and semicolon) should be avoided. Empty values may not behave the same way on all browsers. |

Example Using Cookies

Here is the code of the form (cookieform.jsp) which prompts the user to enter his/her name

<%@ page language="java" %>  
<html>  
<head>  
<title>Cookie Input Form</title>  
</head>  
<body>  
<form method="post" action="setcookie.jsp">  
<p><b>Enter Your Name: </b><input type="text" name="username"><br>  
<input type="submit" value="Submit">  
</form>  
</body>

Here is the code of **setcookie.jsp** file:

<%@ page language="java" import="java.util.\*"%>  
<%  
String username=request.getParameter("username");  
if(username==null)

username="";  
  
  
Date now = new Date();  
String timestamp = now.toString();  
Cookie cookie = new Cookie ("username",username);  
cookie.setMaxAge(365 \* 24 \* 60 \* 60);  
response.addCookie(cookie);  
  
%>  
  
<html>  
<head>  
<title>Cookie Saved</title>  
</head>  
<body>  
<p><a href="showcookievalue.jsp">Next Page to view the cookie value</a><p>  
  
</body>

Here is the code of display cookie page (showcookievalue.jsp):

<%@ page language="java" %>  
<%  
String cookieName = "username";  
Cookie cookies [] = request.getCookies ();  
Cookie myCookie = null;  
if (cookies != null)  
{  
for (int i = 0; i < cookies.length; i++)   
{  
if (cookies [i].getName().equals (cookieName))  
{  
myCookie = cookies[i];  
break;  
}  
}  
}  
%>

**showcookievalue.jsp**  
<html>  
<head>  
<title>Show Saved Cookie</title>  
</head>  
<body>  
  
  
<%  
if (myCookie == null) {  
%>  
No Cookie found with the name <%=cookieName%>  
<%  
} else {  
%>   
<p>Welcome: <%=myCookie.getValue()%>.  
<%  
}  
%>  
</body>

**File Handling in JSP**

<%@ page language="java" import="java.io.\*" %>

<html>

<body>

<form method="post">

Enter File Name To Read:: &nbsp; &nbsp;

<input type="text" name="t1">

<input type="submit" name="s1" value="Read File">

</form>

<%! String filename;

int n;

%>

<%

if((request.getParameter("s1"))!=null)

{

filename=request.getParameter("t1");

File f=new File(filename);

if(f.exists())

{

FileInputStream fis=new FileInputStream(f);

//out.println("hhhhhhhhhhhhhhhhh");

while((n=fis.read()) != -1)

{

out.println((char)n);

}

fis.close();

}

else

{

System.out.println("File do not exists");

}

}

%>

</body>

</html>

**Error Handling in JSP**

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1" errorPage="Demo3.jsp" %>

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

<html>

<head>

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

<title>Insert title here</title>

</head>

<body>

<%

int a=5;

int b=0;

int c=a/b;

out.write("Divisiion: "+c);

%>

</body>

</html>

Demo3.jsp

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1" isErrorPage="true"%>

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

<html>

<head>

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

<title>Insert title here</title>

</head>

<body>

Error::Divide by zero not posible<br></br>

<%

out.print( exception);

%>

</body>

</html>

**[4080]-504**

**3. a) Write a jsp code which adds the student education details in a database through Java bean. Also display the student details who have secured first class in their graduation.**

P4080\_504Q3A.jsp

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1" import="java.sql.\*"%>

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

<html>

<head>

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

<title>Que Paper [4080]-504 Question 3a.</title>

</head>

<body>

<form>

First Name:: <input type="text" name="firstName"/></br>

Last Name:: <input type="text" name="lastName"/></br>

Age:: <input type="text" name="age"/></br>

Percentage:: <input type="text" name="percentage"/></br>

<input type="submit" name="submit" value="Submit"/></br>

</form>

<%

if((request.getParameter("submit"))!=null)

{

%>

<jsp:useBean id="student1" class="pack1.pack2.StudentsBean1">

<jsp:setProperty name="student1" property="firstName"/>

<jsp:setProperty name="student1" property="lastName"/>

<jsp:setProperty name="student1" property="age" />

<jsp:setProperty name="student1" property="percentage" />

</jsp:useBean>

<%!

Connection con;

PreparedStatement pst;

Statement st;

ResultSet rs;

%>

<%

String fname=student1.getFirstName();

String lname=student1.getLastName();

String sage=student1.getAge();

float per=student1.getPercentage();

try

{

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

con=DriverManager.getConnection("jdbc:odbc:DSNN");

pst=con.prepareStatement("insert into student values(?,?,?,?)");

pst.setString(1,fname);

pst.setString(2,lname);

pst.setString(3,sage);

pst.setFloat(4,per);

int n=pst.executeUpdate();

if(n>0)

{

st=con.createStatement();

rs=st.executeQuery("select \* from student where per> 60");

out.print("<table border='2' align='center'>");

out.print("<tr>");

out.print("<th colspan=3>");

out.print("Student");

out.print("</th>");

out.print("</tr>");

out.print("<tr>");

out.print("<th > FNAME </th>");

out.print("<th> LNAME </th>");

out.print("<th> AGE </th>");

out.print("<th> Percentage </th>");

out.print("</tr>");

while(rs.next())

{

out.print("<tr>");

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

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

out.print("<td >"+rs.getString(3)+ "</td>");

out.print("<td>"+rs.getFloat(4)+ "</td>");

out.print("</tr>");

}

out.print("</table>");

}

}

catch(ClassNotFoundException cnfe)

{

cnfe.printStackTrace();

}

catch(SQLException se)

{

se.printStackTrace();

}

}

%>

</body>

</html>

StudentsBean1.java

package pack1.pack2;

import java.io.Serializable;

public class StudentsBean1 implements Serializable

{

private String firstName = null;

private String lastName = null;

private String age = "0";

private float percentage=0.0f;

public String getFirstName(){

return firstName;

}

public String getLastName(){

return lastName;

}

public String getAge(){

return age;

}

public float getPercentage()

{

return percentage;

}

public void setFirstName(String firstName){

this.firstName = firstName;

}

public void setLastName(String lastName){

this.lastName = lastName;

}

public void setAge(String age)

{

this.age = age;

}

public void setPercentage(float percentage)

{

this.percentage = percentage;

}

}

**JSP Database**

**INSERT**

<% @ page Language="java" import="java.sql.\*" %>

<html>

<body>

<form method="post">

Enter Employee no:: &nbsp; &nbsp;

<input type="text" name="t1">

Enter Employee name:: &nbsp; &nbsp;

<input type="text" name="t2">

Enter Employee salary:: &nbsp; &nbsp;

<input type="text" name="t3">

<input type="submit" name="s1" value="Insert">

</form>

<% !

Connection con;

Statement st;

ResultSet rs;

PreparedStatement pt;

%>

<%

try

{

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

}

catch(ClassNotFoundException e)

{

e.getMessage();

}

if((request.getParameter("s1"))!=null)

{

try

{

con=DriverManager.getConnection("jdbc:odbc:");

String empno=req.getParameter("t1"):

String empname=req.getParameter("t2");

int empsal=Integer.parseInt(req.getParameter("t3"));

// here prepared statement is used, you can also use normal sql statements

pt=con.prepareStatement("Insert into employee values(?,?,?)");

pt.setString(1,empno);

pt.setString(2,empname);

pt.setInt(3,empsal);

int no=pt.executeUpdate();

if(no>0)

{

st=con.createStatement();

rs=st.executeQuery("select \* from Employee");

out.println("<b>ENO ENAME ESAL<br>");

while(rs.next())

{

out.println(rs.getString(1)+" "+rs.getString(2)+" "+rs.getInt(3)+" <br>");

}

}

else

{

out.println("Record is not inserted ");

}

}

catch(SQLException e)

{

e.getMessage();

}

}

%>

</body>

</html>

**DELETE**

<% @ page Language="java" import="java.sql.\*" %>

<html>

<body>

<form method="post">

Enter Employee no:: &nbsp; &nbsp;

<input type="text" name="t1">

<input type="submit" name="s1" value="Delete">

</form>

<% !

Connection con;

Statement st;

ResultSet rs;

PreparedStatement pt;

%>

<%

try

{

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

}

catch(ClassNotFoundException e)

{

e.getMessage();

}

if((request.getParameter("s1"))!=null)

{

try

{

con=DriverManager.getConnection("jdbc:odbc:");

String empno=req.getParameter("t1");

pt=con.prepareStatement("delete from employee where eno=?");

pt.setString(1,empno);

int no=pt.executeUpdate();

if(no>0)

{

st=con.createStatement();

rs=st.executeQuery("select \* from Employee");

out.println("<b>ENO ENAME ESAL<br>");

while(rs.next())

{

out.println(rs.getString(1)+" "+rs.getString(2)+" "+rs.getInt(3)+" <br>");

}

}

else

{

out.println("Record is Deleted ");

}

}

catch(SQLException e)

{

e.getMessage();

}

}

%>

</body>

</html>

**UPDATE**

<% @ page Language="java" import="java.sql.\*" %>

<html>

<body>

<form method="post">

Enter Employee no:: &nbsp; &nbsp;

<input type="text" name="t1">

Enter Employee salary:: &nbsp; &nbsp;

<input type="text" name="t2">

<input type="submit" name="s1" value="Update">

</form>

<% !

Connection con;

Statement st;

ResultSet rs;

PreparedStatement pt;

%>

<%

try

{

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

}

catch(ClassNotFoundException e)

{

e.getMessage();

}

if((request.getParameter("s1"))!=null)

{

try

{

con=DriverManager.getConnection("jdbc:odbc:");

String empno=req.getParameter("t1");

int empsal=Integer.parseInt(req.getParameter("t2"));

pt=con.prepareStatement("update employee set sal=? where eno=?");

pt.setInt(1,empsal);

pt.setString(2,empno);

int no=pt.executeUpdate();

if(no>0)

{

st=con.createStatement();

rs=st.executeQuery("select \* from Employee");

out.println("<b>ENO ENAME ESAL<br>");

while(rs.next())

{

out.println(rs.getString(1)+" "+rs.getString(2)+" "+rs.getInt(3)+" <br>");

}

}

else

{

out.println("Record is not Updated ");

}

}

catch(SQLException e)

{

e.getMessage();

}

}

%>

</body>

</html>