JSP(Java Server Pages)

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

Java Server Pages is a technology to create dynamic web pages as servlet technology.

JSP is called extension of Servlet because it has all features of servlet and it is also support implicit object,action tag, Expression language and JSTL.

Advantage of JSP Over Servlet

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

1) Less Code

2) extension of servlet

3) In JSP we can write java code inside the HTML page and in Servlet we can write HTML code inside the Java Code.

4) In JSP we can seperate presentation logic and business logic.

5) No need redeploy if JSP is Modified.

Ex.1

====

<form action="Register.jsp" method="post">

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

pwd<input type="password" name="p1"><br>

-

-

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

</form>

</body>

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

Register.jsp

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

<%

String a=request.getParameter("t1");

String b=request.getParameter("p1");

//database connectivity

out.print("You are Register");

%>

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

ex2

=====

(A.jsp)

==========

<!doctype>

<html>

<head>

<script>

</script>

<style>

</style>

</head>

<body>

<h1>Dps</h1>

<p>jiruiur</p>

<%

double r=Math.sqrt(16);

out.print(r);

%>

</body>

Ex.3

=======

Register.jsp

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

<%

if(request.getParameter("b1")==null)

{

%>

<form action="Register.jsp" method="post">

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

pwd<input type="password" name="p1"><br>

-

-

<input type="submit" name="b1" value="RegisterNow">

</form>

</body>

<%

}

else

{

String a=request.getParameter("t1");

String b=request.getParameter("p1");

//database connectivity

out.print("You are Register");

}

%>

Ex.4

=======

A.html

======

<body>

<form action="P.jsp" method="post">

No<input type="text" name="t1">

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

</form>

</body>

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

P.jsp

=====

<%

int a=Integer.parseInt(request.getParameter("t1"));

if(a==1)

{

%>

<h1>DPS</h1>

<h2>Features</h2>

<%

}

else if(a==2)

{

%>

<p>This is s sss s</p>

<imgsrc="a.jpg" height="100" width="100">

<%

}

else

{

%>

<h3>hhhh</h3>

<h4>KBC</h4>

<%

}

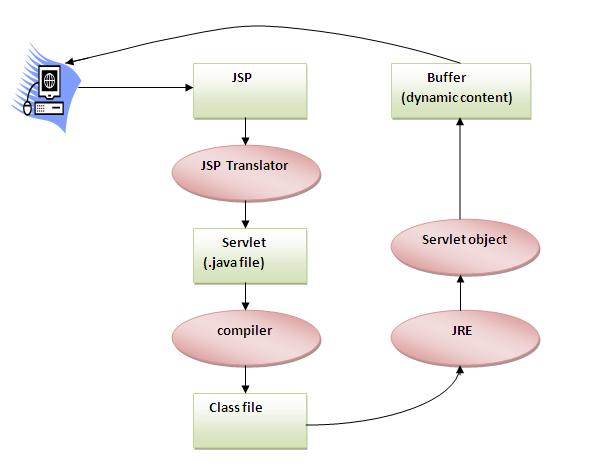
%>

**Life cycle of a JSP Page**

The JSP pages follows these phases:

* Translation of JSP Page
* Compilation of JSP Page
* Classloading (class file is loaded by the classloader)
* Instantiation (Object of the Generated Servlet is created).
* Initialization ( jspInit() method is invoked by the container).
* Reqeust processing ( \_jspService() method is invoked by the container).
* Destroy ( jspDestroy() method is invoked by the container).

**Note: jspInit(), \_jspService() and jspDestroy() are the life cycle methods of JSP.**



The JSP API

The JSP API consists of two packages:

1. javax.servlet.jsp
2. javax.servlet.jsp.tagext

javax.servlet.jsp package

The javax.servlet.jsp package has two interfaces and classes.The two interfaces are as follows:

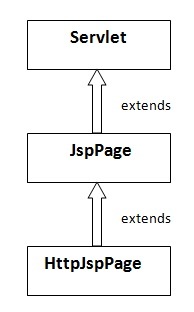
1. JspPage
2. HttpJspPage

The classes are as follows:

* JspWriter
* PageContext
* JspFactory
* JspEngineInfo
* JspException
* JspError

The JspPage interface

According to the JSP specification, all the generated servlet classes must implement the JspPage interface. It extends the Servlet interface. It provides two life cycle methods.



Methods of JspPage interface

1. **public void jspInit():** It is invoked only once during the life cycle of the JSP when JSP page is requested firstly. It is used to perform initialization. It is same as the init() method of Servlet interface.
2. **public void jspDestroy():** It is invoked only once during the life cycle of the JSP before the JSP page is destroyed. It can be used to perform some clean up operation.

The HttpJspPage interface

The HttpJspPage interface provides the one life cycle method of JSP. It extends the JspPage interface.

Method of HttpJspPage interface:

1. **public void \_jspService():** It is invoked each time when request for the JSP page comes to the container. It is used to process the request. The underscore \_ signifies that you cannot override this method.

JSP (Scripting elements)

In JSP, java code can be written inside the jsp page using the scriptlet tag. Let's see what are the scripting elements first.

JSP Scripting elements

The scripting elements provides the ability to insert java code inside the jsp. There are three types of scripting elements:

* scriptlet tag
* expression tag
* declaration tag

JSP scriptlet tag

JSP scriptlet tag

A scriptlet tag is used to execute java source code in JSP. Syntax is as follows:

1. <%  java source code %>

Ex.

<%

int a=10;

double r =Math.sqrt(16);

out.print("Hello");

%>

Ex2

====

<%

voiddisp() //c.t error

{

}

%>

JSP expression tag

The code placed within **JSP expression tag** is *written to the output stream of the response*. So you need not write out.print() to write data. It is mainly used to print the values of variable or method.

Syntax of JSP expression tag

1. **<**%=  statement %**>**

# <%= "Hello" %>

# JSP Declaration Tag

The **JSP declaration tag** is used to declare fields and methods.

The code written inside the jsp declaration tag is placed outside the service() method of auto generated servlet.

So it doesn't get memory at each request.

#### **Syntax of JSP declaration tag**

The syntax of the declaration tag is as follows:

1. **<**%!  field or method declaration %**>**

Ex

===

<%!

int p=10;

voiddisp()

{

}

%>

### Difference between JSP Scriptlet tag and Declaration tag

|  |  |
| --- | --- |
| **JspScriptlet Tag** | **Jsp Declaration Tag** |
| The jspscriptlet tag can only declare variables not methods. | The jsp declaration tag can declare variables as well as methods. |
| The declaration of scriptlet tag is placed inside the \_jspService() method. | The declaration of jsp declaration tag is placed outside the \_jspService() method. |

**JSP Implicit Objects**

There are **9 jsp implicit objects**. These objects are *created by the web container* that are available to all the jsp pages.

The available implicit objects are out, request, config, session, application etc.

A list of the 9 implicit objects is given below:

|  |  |
| --- | --- |
| **Object** | **Type** |
| out | JspWriter |
| request | HttpServletRequest |
| response | HttpServletResponse |
| config | ServletConfig |
| application | ServletContext |
| session | HttpSession |
| pageContext | PageContext |
| page | Object |
| exception | Throwable |