JSP

1)JSP is a technology that help software developer s to create dynamic web pages.

2)with jsp, we can separate the business logic from static element.

3)jsp doesn’t give you anything that we couldn’t do this in servlet.but it is more convenient to write regular html than to have lot of statements that generates the html.also by separating the look from the dynamic content.we can put different people on different task that is web page design expert can build the html, leaving places for developers to insert that dynamic content.

4)architecturally jsp may be viewed as high level abstraction of java servlets. jsp’s are translated into servlet at run time. each jsp’s servlet is cached and reused until the original jsp is modified.

5)jsp compiler is a program that passes jsp and transforms them into executable java servlet.this program is part of servlet container and run automatically when first time jsp is accessed.

6)we can find the generated servlet java file and class file in the below path.

E:\JAVA\javasoftware\apache-tomcat-6.0.18\work\Catalina\localhost\jsp\org\apache\jsp

7)there are two ways,we can access our jsp.

->by typing jsp file name directly in the URL.

->by configuring the jsp in web.xml.

<servlet>

<servlet-name>date</servlet-name>

<jsp-file>/date.jsp</jsp-file>

</servlet>

<servlet-mapping>

<servlet-name>date</servlet-name>

<url-pattern>/date</url-pattern>

</servlet-mapping>

8)we can go with second approach when we want to pass initialization information to jsp.on the other hand context init-params are bound to servlet context object and they are available regardless of how our jsp is invoked.

**DIFFERENCE BETWEEN JSP AND SERVLETS**

|  |  |
| --- | --- |
| **JSP** | **SERVLETS** |
| 1)JSP supports HTTP and HTTPS. | 1)servlets are not bound to specific protocol. |
| 2)whenever the first request comes in the jsp,get converted to servlet by servlet container and generated(.class) file is used to generate dynamic content. | 1)already compiled .class file are used to generate dynamic content. |
| 3)implicit objects are available. | 3)we have to explicitly defined implicit object in servlet. |
| 4)dynamic content is kept separate from static content(java inside the html). | 4)dynamic content is tightly coupled with static content(html) inside java. |
| 5)jsp can not be used only by developers but also web design experts. | 5)only used by developer. |

**JSP LIFE CYCLE**

WEB.XML

If necessary,container

SUCCESS loads the jsp

FAILURE

SUCCESS

SUCCESS

REQ

RES

**COMPILE & INSTANTIATE**, PARSE THE JSP,TRANSLATING THE JSP INTO SERVLET,COMPILE THE SERVLET INSTANTIATING BY INVOKING DEFAULT CONSTRUCTOR

I

SERVLET CONTAINER

**INITIALIZE**  INVOKE jspInit() METHOD

**SERVICE**  INVOKE \_jspService(METHOD)

DoXXX()

**DESTROY** INVOKE jspDestroy() METHOD

1)life cycle of jsp is controlled by servlet container.when a request comes to a jsp the container performs following steps.

->compilation

->initialization

->execution

->cleanup

**COMPILATION**

1)when a request comes to a jsp,servlet container first checks ,whether we need to compile jsp.if the jsp as never been compiled or if the jsp has been modified.since it was last compiled then container compiles the jsp.

2)the compilation fails involved three steps.

->parsing the jsp.

->translating the jsp into servlet.

->compiling the servlet.

**INITIALIZATION PHASE**

Public void jspInit(){}

1)when a container loads the translated servlet.it invokes the jspInit() method before service() any request.

2)jspInit() method use a chance to initialize the translated servlet before handling any client request.for example reading data from property file etc.

3)jspInit() method is called only once during the entire life cycle of a jsp.we may possibly override this method. If we have a initialization code then we may override this method.if we don’t override this method then one from HttpJspBase class is used.

**EXECUTION PHASE**

Public void \_jspService(HttpServletRequest req,HttpServletResponse res)throws ServletException,IoException;

1)whenever a container receives a request it starts a new thread and invokes the \_jspService() method of translated servlet.

2)this method is called,whenever a client request comes in.that is this method is called one or more time in jsp life cycle.

3)this method is responsible for generating responses to all seven of the Http methods.

4)we should not override this method.whatever we write in the jsp,it becomes the part of \_jspService() method when jsp get translated in servlet.

5)if we don’t write anything in jsp(blank jsp).we won’t get any compile time and runtime exeception instead we get the blank page as the response.

**CLEANUP PHASE**

Public void jspDestroy(){}

1)when a container wants to unload the translated servlet of the jsp it calls jspdestroy() method before unloading the translated servlet from the service.

2)that decision of went to invoke this method ,rests on the shoulder of container.

3)this method is used to carry out any care taking task (closing a property file).

4)this method is called only once in the lifecycle of the jsp.if we don’t override this method then one from the HttpJspBase is used.

5)we may possibly override this method.

**COMMENTS IN JSP**

1)there is only one type of jsp comments ,available as per jsp specification.

<%-- my comments --%>

Note-1)<!-- -->

Its not a jsp comment .this is an html comment therefor there is nothing called hidden comment or output comment a per jsp specification.

**SCOPE OF JSP OBJECT**

1)every object created in jsp will have a scope.object scope in jsp is segregated into four parts and they are

->page scope

->request scope

->session scope

->application scope

**PAGE SCOPE**

-it means the jsp object can be access only within the page,where it was created(like private variables in java).

**REQUEST SCOPE**

-jsp object created during request scope,can be access from any page that serves that request(more than one page can a serve a single request)

**SESSION SCOPE**

-it means,jsp object is accessible in different pages within the same session.

**APPLICATION SCOPE**

-a jsp object is created using application scope can be accessed from any pages across the application till the application or server goes down.

**IMPLICIT OBJECT**

1)implicit objects are the part of the object that are created by container automatically.so that developers need not to create them explicitly.

2)these objects use standard variables.

3)implicit objects are available only within the \_jspService() method and not in any declaration.hence our own methods can’t access them as they are local to the service() method.

4)there are nine implicit objects available in jsp.

|  |  |  |  |
| --- | --- | --- | --- |
| **VARIABLE** | **OF TYPE** | **SCOPE** | **DESCRIPTION** |
| 1)out | Javax.servlet.jsp.JspWriter | page | Out object is used to write response to the outputStream.a specific out object is available for each jsp page. |
| 2)request | Javax.servlet.http.HttpServletRequest | request | Operation is similar to HttpServletRequest object. |
| 3)response | Javax.servlet.http.HttpServletResponse | page | Operation is similar to HttpServletResponse object. |
| 4)config | Javax.servlet.ServletConfig | page | Operation is similar to ServletConfig object. |
| 5)session | Javax.servlet.http.HttpSession | session | Operation is similar to HttpSession object. |
| 6)application | Javax.servlet.ServletContext | application | Operation is similar to ServletContext object. |
| 7)exception | Java.lang.Throwable | page | This object is only available to pages that has isErrorPage set to true with directive  <%@page isErrorPage=”true” %> |
| 8)page | Java.lang.Object | page | This is an another name for “this” and used to call the methods defined by the translated servlet class.thus translated servlet page object page is created as “final.java.lang.Object page=this” |
| 9)pageContext | Javax.servlet.jsp.PageContext | page | pageContext provides a single API to manage all the implicit objects. |

Application=pageContext.getServletContext();

Config=pageContext.getServletConfig();

Session=pageContext.getSession();

Out=pageContext.getOut();

->this API is extensively use,if we are implementing jsp custom tag handler.

->a typical use of pageContext implicit object is to include another resource or forward the request to another resource.

**JSP TAGS**

1)tags are important syntax element of jsp.jsp tags are some what like html tags and its start with start tag “<”,a tag body and end tag “>”.

2)jsp tag can be divided into five different types.

->declaration tag

->expression tag

->scriplet tags

->action tags

->directive tags

**1)DECLARATION TAG**

->this tag allows the developer to declare variable and method.it start with (<%!) and end with (%>).

->code place within this tag should end with(;) at the end of each code.

**2)EXPRESSION TAG**

->this tag allows the developers to output any data on the generated page.the data automatically get converted to string and printed on to the outputStream.

->expression tag must start with (<%=) and end with (%>);

->code placed within this tag should not end with (;).

**3)SCRIPLET TAG**

1)this tag allows the developers to write any amount of valid java code and this code are placed in \_jspService() method at the time of translation.

2)scriptlet tag must start with (<%) and end with (%>).

3)code placed within this tag should end with (;) at end of each code.

**4)ACTION TAG**

<jsp:action-name action attributes />

->action tags in jsp are used to perform action on particular page.following actions are available in jsp.

|  |  |
| --- | --- |
| **ACTION-NAME** | **PURPOSE** |
| 1)forward | Forward the request to a new page. |
| 2)include | Include any given resources into the jsp. |
| 3)param | Defines the parameter(key-value string pair) |
| 4)useBean | Find or instantiate a java bean. |
| 5)setProperty | Sets the property of a java bean. |
| 6)getProperty | Get the property of a java bean and inserts it into outStream. |

**FORWARD ACTION**

1)forward action forwards the request to another resource such as static page(h tml,text,etc) or dynamic page(jsp or servlet).

2)when we forward the request to a servlet from jsp.it will invoke the proper doXXX() method in the target servlet depending on the http method in the request.if the doXXX() method does not exit in the target servlet then client receives error information as a response.

**INCLUDE ACTION**

1)include action includes either static(html,text,etc) or dynamic(servlet,jsp) content inside the jsp.when we include servlet in jsp,it will include the response of proper doXXX() method into jsp depending on the http method in the request.if doXXX() method doesn’t exist in target servlet then container doesn’t throw any compile time & run time exception. in this case it just ignore the include statement.

**PARAM ACTION**

1)use to send parameter to any jsp or servlet.

**USEBEAN ACTION**

->useBean action first search for existing object by using id and scope tag attributes.if any object is not found then it creates the new object with the specified id and scope .if the object is found ,it just makes use of that object.

**SETPROPERTY ACTION**

->through that we can set the property in name-value pair.

**GETPROPERTY ACTION**

->through that we can get the properth in name-value pair.

**DIRECTIVE**

->SYNTAX-<%@directive-name directive-attributes %>

1)the directive tag gives special information about the page to the jsp engine(servlet container).this changes the way, jsp engine process the page.

2)there are three types of directive tags.

->include

->page

->taglib

**INCLUDE DIRECTIVE**

->SYNTAX-<%@include file=”resource\_name” %>

1)include directive is used to include the contents of any resources(jsp files,html file & even text file but not servlets) into jsp.the include directive includes the original contents of the included resources at page translation time.

**DIFF BETWEEN INCLUDE DIRECTIVE AND INCLUDE ACTION**

|  |  |
| --- | --- |
| **INCLUDE DIRECTIVE** | **INCLUDE ACTION** |
| 1)using this directive we can include any resources(jsp,html ,text file) but not a servlet. | 1)using this directive we can include any resources along with servlet. |
| 2)contents of the resources get included in jsp only at jsp compilation phase. | 2)contents of the resources get included into the jsp every time user access the jsp. |
| 3)in include directive we spicify the physical file name. | 3)we spicify the url of that resource. |
| 4)we can’t use jsp-param and query parameter. | 4)we can use both. |

**PAGE DIRECTIVE**

SYNTAX-<%@ page directive %>

->page directive has eleven optional attributes that provide special information to servlet container.

**1)Attribute-language**

Discription-language denotes the scripting language used in scriplets,declaration and expression in the jsp page and any included files.default value is java.

SYNTAX-<@%page language=”java” %>

**2)Attribute-info**

Discription-used to provide information for a jsp.details such as cersion,copyright and date are placed in the attribute.this information can be later retrieve by using getServletIbfo() method.

SYNTAX-<%@page info=”MyFirst.jsp” %>

**3)Attribute-contentType**

Discription-this attribute specifies the MIME type and character encoding for the jsp response.the default MIME type is text or html and default character set is ISO-8859-1.

SYNTAX-<%@page contentType=”text/html”%>

Or <%@page contentType=”text/html;charset=ISO-8859-1” %>

**4)Attribute-import**

Discription-functionally is similar to java import.

SYNTAX-<[%@page import=”pkg1.class\_name,pkg2.class\_name” %](mailto:%25@page%20import=)>

<[%@page import=”java.util.Date”%](mailto:%25@page%20import=)>

**GWT-google web toolkit.use to develop UI in same page.like ok behave differently in different page.**

**5)Attribute-extends**

Discription-similar to extends keyword in java.

SYNTAX-<%@page extends-“org.apache.jasper.runtime.HttpJspBase” %>

**6)Attribute-buffer(temporary storage)**

Discription-it sets the buffer size in kb to handle output generated by the jsp page,which is send to client in packets of data.

-to tern of buffer output the value should be null.

-default size of a buffer is 8kb.

SYNTAX-<%@page buffer=”none | size in kb” %>

**7)Attribute-autoFlush**

Discription-it controls the behavior of output buffer.the value is either true or false.if true then buffer will be flush automatically when it is full.if false then exception is throw when then buffer get full.

SYNTAX-<%@page autoFlush=”true | false” %>

**8)Attribute-session**

Discription-behavior is same as HttpSession and the value of the session attribute either true or false.if true then session object refers to the current or new session.if false then we cann’t use the session object or useBean action with scope=session in the jsp ,default value is true.

SYNTAX-<%@page session=”true | false” %>

**9)Attribute-isThreadSafe**

Discription-value of this attribute is either true or false.if true then multiple threads will act on the \_jspService() method.if false at any point of time,only one thread will act on \_jspService() method.

SYNTAX-<%@page isThreadSafe=”true | false” %>

**10)Attribute-isErrorPage**

Discription-the value is either true or false.if true ,declare the current page as the error page and bydefault its set to false.

SYNTAX-<%@page isErrorPage=”true | false” %>

**11)Attribute-errorPage**

Discription-this attribute is used to define the error page .if exception occurs in the current page then it will be redirected to a specified error page.

SYNTAX-<%@page errorPage=”redirectedURL” %>

**TAGLIB DIRECTIVE**

Discription- taglib directive is the collection of custom tag.in jsp we use TLD(tag library descriptor) file to define the tags.

SYNTAX-<%@taglib url=”url of the tag library” prefix=”prefix of tag library” %>

Custom tag-<# #>