**JSP (JavaServer Pages)**

* **JSP** technology is used to generate a web application just like Servlet technology.
* It can be considered as an extension to servlet because it offers more functionality than servlet such as expression language, jstl etc.
* A JSP page contains HTML tags and JSP tags. The jsp pages are easier to sustain than servlet because we can separate designing and development.

**Advantages of JSP over Servlet**

**1) Extension to Servlet**

It uses all the features of servlet in JSP. In addition, it can use implied objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy.

**2) Easy to maintain**

It can be easily achieved because it can easily separate business reason with presentation logic.

**3) Fast Development**

There is no need to recompile and redeploy

If JSP page is modified, we don't need to recompile and redeploy the project.

The servlet code needs to be updated and recompiled if we have to change the look and feel of the application.

**4) Less code than Servlet**

It uses a lot of tags such as action tags, jstl, custom tags etc. that reduces the code. Moreover, it can use EL, implicit objects etc.

**Life cycle of a JSP Page**

* 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).

# JSP Scriptlet tag (Scripting elements)

In JSP, java code can be written inside the jsp page using the scriptlet tag.

## **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

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

<%  java source code %>

**JSP expression tag**

The code placed within **JSP expression tag** is *written to the output stream of the response*. So there is no need to write out.print() to write data. It is mainly used to print the values of variable or method.

**JSP Declaration Tag**

It is used *to declare fields and methods*.

**Difference between JSP Scriptlet tag and Declaration tag**

|  |  |
| --- | --- |
| **Jsp Scriptlet Tag** | **Jsp Declaration Tag** |
| The jsp scriptlet tag can only declare variables not methods. | The jsp declaration tag can declare variables as well as methods. |

**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 and theses are:

1. **JSP out implicit object**

For writing any data to the buffer, JSP provides an implicit object named out. It is the object of JspWriter.

1. **JSP request implicit object**

The **JSP request** is an implicit object of type HttpServletRequest and it can be used to get request data such as parameter, header information, remote address, server name, server port, content type andcharacter encoding

1. **JSP response implicit object**

In JSP, response is an implicit object of type HttpServletResponse. The instance of HttpServletResponse is created by the web container for each jsp request. It can also be used to add or manipulate response such as redirect response to another resource, send error etc.

1. **JSP config implicit object**

In JSP, config is an implicit object of type *ServletConfig*. This object can be used to get initialization parameter for a particular JSP page. The config object is created by the web container for each jsp page. In general, it is used to get initialization parameter from the web.xml file.

1. **JSP application implicit object**

In JSP, application is an implicit object of type *ServletContext* which is created only once by the web container when application or project is deployed on the server.

This object can be used to get initialization parameter from configuaration file (web.xml). It can also be used to get, set or remove attribute from the application scope.

# session implicit object

|  |  |  |
| --- | --- | --- |
| In JSP, session is an implicit object of type HttpSession.The Java developer can use this object to set, get or remove attribute or to get session information.   1. **pageContext implicit object**  |  | | --- | | It is an implicit object of type PageContext class.The pageContext object can be used to set, get or remove attribute from one of the following scopes:   * page * request * session * application | | page implicit object It is an implicit object of type Object class. This object is assigned to the reference of auto generated servlet class. exception implicit object It is an implicit object of type java.lang.Throwable class. This object can be used to print the exception. But it can only be used in error pages. | |

**JSP directives**

The **jsp directives** are messages that tells the web container how to translate a JSP page into the corresponding servlet.

There are three types of directives:

* page directive
* include directive
* taglib directive

1. **JSP page directive**

The page directive defines attributes that apply to an entire JSP page.

**Attributes of JSP page directive**

* import
* contentType
* extends
* info
* buffer
* language
* isELIgnored
* isThreadSafe
* autoFlush
* session
* pageEncoding
* errorPage
* isErrorPage

1. **import**

The import attribute is used to import class, interface or all the members of a package. It is similar to import keyword in java class or interface.

1. **contentType**

It defines the MIME (Multipurpose Internet Mail Extension) type of the HTTP response. The default value is "text/html;charset=ISO-8859-1".

1. **extends**

The extends attribute defines the parent class that will be inherited by the generated servlet.

1. **info**

This attribute simply sets the information of the JSP page which is retrieved later by using getServletInfo() method of Servlet interface.

1. **buffer**

The buffer attribute sets the buffer size in kilobytes to handle output generated by the JSP page. The default size of the buffer is 8Kb.

1. **language**

The language attribute specifies the scripting language used in the JSP page. The default value is "java".

1. **isELIgnored**

We can ignore the Expression Language (EL) in jsp by the isELIgnored attribute. By default, its value is false i.e. Expression Language is enabled by default.

1. **isThreadSafe**

Servlet and JSP both are multithreaded. If you want to control this behaviour of JSP page, you can use isThreadSafe attribute of page directive.The value of isThreadSafe value is true.If you make it false, the web container will serialize the multiple requests.

1. **errorPage**

The errorPage attribute is used to define the error page, if exception occurs in the current page, it will be redirected to the error page.

1. **isErrorPage**

The isErrorPage attribute is used to declare that the current page is the error page.

1. **Jsp Include Directive**

It is used to include the contents of any resource it may be jsp file, html file or text file. It includes the original content of the included resource at page translation time

1. **JSP Taglib directive**

It is used to define a tag library that defines many tags. We use the TLD (Tag Library Descriptor) file to define the tags. In the custom tag section, we will use this tag so it will be better to learn it in custom tag.

**Exception Handling in JSP**

The exception is normally an object that is thrown at runtime. Exception Handling is the process to handle the runtime errors. There may occur exception any time in your web application. So handling exceptions is a safer side for the web developer. In JSP, there are two ways to perform exception handling:

1. By **errorPage** and **isErrorPage** attributes of page directive
2. By **<error-page>** element in web.xml file

**JSP Action Tags**

There are many JSP action tags or elements. Each JSP action tag is used to perform some specific tasks.

The action tags are used to control the flow between pages and to use Java Bean. The Jsp action tags are the following:

* 1. **Jsp:forward action tag**

It is used to forward the request to another resource it may be jsp, html or another resource.

* 1. **jsp:include action tag**

It is used to include the content of another resource it may be jsp, html or servlet.

It includes the resource at request time so it is better for dynamic pages because there might be changes in future.

**Difference between jsp include directive and include action**

|  |  |
| --- | --- |
| **JSP include directive** | **JSP include action** |
| includes resource at translation time. | includes resource at request time. |
| better for static pages. | better for dynamic pages. |
| includes the original content in the generated servlet. | calls the include method. |

# jsp:useBean action tag

It is used to locate or instantiate a bean class. If bean object of the Bean class is already created, it doesn't create the bean depending on the scope. But if object of bean is not created, it instantiates the bean.

### Attributes and Usage of jsp:useBean action tag

1. **id:**is used to identify the bean in the specified scope.
2. **scope:**represents the scope of the bean. It may be page, request, session or application. The default scope is page.
   1. **page:**specifies that you can use this bean within the JSP page. The default scope is page.
   2. **request:**specifies that you can use this bean from any JSP page that processes the same request. **session:**specifies that you can use this bean from any JSP page in the same session whether processes the same request or not.
   3. **application:**specifies that you can use this bean from any JSP page in the same application.
3. **class:**instantiates the specified bean class but it must have no-argument or no constructor and must not be abstract.
4. **type:**provides the bean a data type if the bean already exists in the scope. It is mainly used with class or beanName attribute. If you use it without class or beanName, no bean is instantiated.
5. **beanName:**instantiates the bean using the java.beans.Beans.instantiate() method.

# jsp:setProperty and jsp:getProperty action tags

The setProperty and getProperty action tags are used for developing web application with Java Bean. Bean class is mostly used because it is a reusable software component that represents data and it sets a property value or values in a bean using the setter method.

# jsp:plugin action tag

It is used to embed applet in the jsp file. The jsp:plugin action tag downloads plugin at client side to execute an applet or bean.

# Custom Tags in JSP

**Custom tags** are user-defined tags which eliminate the possibility of scriptlet tag and separates the business logic from the JSP page.

### Advantages of Custom Tags

1. **Eliminates the need of scriptlet tag**

The custom tags eliminate the need of scriptlet tag which is considered bad programming approach in JSP.

1. **Separation of business logic from JSP**

The custom tags separate the business logic from the JSP page so that it may be easy to maintain.

1. **Re-usability**

The custom tags make the possibility to reuse the same business logic again and again.

# Attributes in JSP Custom Tag

To define the attribute, you need to perform two tasks:

* Define the property in the TagHandler class with the attribute name and define the setter method
* define the attribute element inside the tag element in the TLD file

# Iteration using JSP Custom Tag

You can iterate the body content of any tag using the **doAfterBody()** method of **IterationTag interface**.