Java Server Pages

Web Application

- “A web application is such a website where the pages are generated dynamically, based on the user’s choices and actions”
- There are 3 main components of any web application. There is:
  - The Web Browser
  - The Web Server
  - The Database

JSP

- Java Server Pages (JSP) is a technology which is used to develop web pages by inserting Java code into the HTML pages by making special JSP tags. The JSP tags which allow java code to be included into it are <% ----java code----%>.
- It can consist of either HTML or XML (combination of both is also possible) with JSP actions and commands.
- It can be used as HTML page, which can be used in forms and registration pages with the dynamic content into it.
- Dynamic content includes some fields like dropdown, checkboxes, etc. whose value will be fetched from the database.
- This can also be used to access JavaBeans objects.
- We can share information across pages using request and response objects.
- JSP can be used for separation of the view layer with the business logic in the web application.

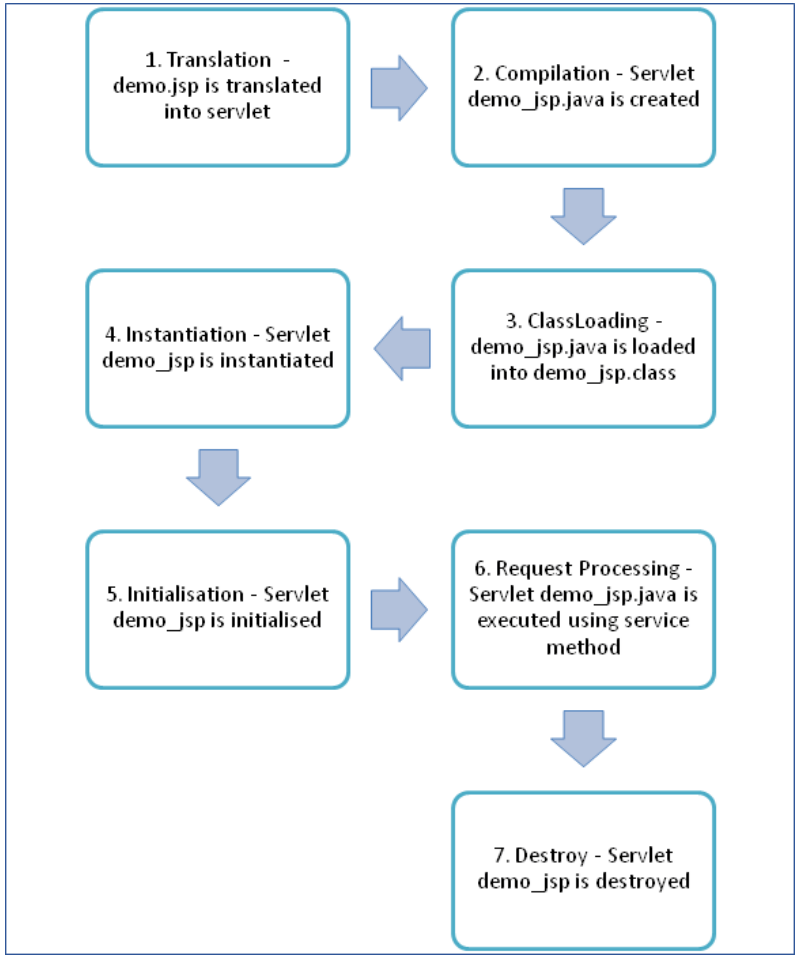
Why use JSP ?

- In Java server pages JSP, the execution is much faster compared to other dynamic languages.
- It is much better than Common Gateway Interface (CGI).
- Java server pages JSP are always compiled before its processed by the server as it reduces the effort of the server to create process.
- Java server pages JSP are built over Java Servlets API. Hence, it has access to all Java APIs, even it has access to JNDI, JDBC EJB and other components of java.
- JSP are used in MVC architecture (which will be covered in MVC architecture topic) as view layer.
- The request is processed by a view layer which is JSP and then to servlet layer which is java servlet and then finally to a model layer class which interacts with the database.
- JSP is an important part of Java EE, which is a platform for enterprise level applications.

Advantages of JSP

- The advantage of JSP is that the programming language used is JAVA, which is a dynamic language and easily portable to other operating systems.
- JSP can also include the database connections into it. It can contain all type of java objects & It is very easy to maintain
- Performance and scalability of JSP are very good because JSP allows embedding of dynamic elements in HTML pages.
- As it is built on Java technology, hence it is platform independent and not depending on any operating systems.
- Multithreading & Exception Handling
- It enables to separate presentation layer with the business logic layer in the web application.

Life Cycle


1. Translation - demo.jsp is translated into servlet
2. Compilation - Servlet demo_jsp.java is created
3. ClassLoading - demo_jsp.java is loaded into demo_jsp.class
4. Instantiation - Servlet demo_jsp is instantiated
5. Initialisation - Servlet demo_jsp is initialised
6. Request Processing - Servlet demo_jsp.java is executed using service method
7. Destroy - Servlet demo_jsp is destroyed

JSP

- It is a webpage scripting language which can generate content on the fly.
- Comparatively it is slower than Servlets as it takes a lot of time to compile.
- Coding in JSP is much easier than in Servlets.
- JSP acts as a view when in MVC.
- JSP is preferred when there isn’t much data processing required.
- Another thing about JSP is that you can build custom tags which can call Java beans.
- Last but not the least, with JSP we can achieve a great deal of functionality with the help of JavaScript at the client end.

Servlets

- These are compiled Java programs which can also help create dynamic web applications. Servlets are faster than JSP.
- You have to write a lot of code for Servlets.
- Servlets act as controllers in MVC.
- Servlets are better than JSP when you have to do a lot of data processing and manipulation.
- You can’t create custom tags in servlets and last but not the least you can’t use JavaScript at client side with Servlets.

Sample Code

<html> <body>

<h2>Hello World!</h2>

Server Time is <%= new java.util.Date() %>

</body> </html>

Use of “<” an angle bracket and then a “%”percentage sign.

<% ..... %>

JSP Scripting Elements

- **JSP Expressions**: It is a small java code which you can include into a JSP page. The syntax is “<%= some java code %>”
- **JSP Scriptlet**: The syntax for a scriptlet is “<% some java code %>”. You can add 1 to many lines of Java code in here.
- **JSP Declaration**: The syntax for declaration is “<%! Variable or method declaration %>”, in here you can declare a variable or a method for use later in the code.

JSP Expressions

- Using the JSP Expression you can compute a small expression, always a single line, and get the result included in the HTML which is returned to the browser.
- <%= .... %>
- Converting a string to uppercase
- <%= new String(“Hello World”).toUpperCase() %>

JSP Scriptlets

- Scripting Element allows you to put in a lot of Java code in your HTML code.
- This Java code is processed top to bottom when the page is the processed by the web server.
- Here the result of the code isn't directly combined with the HTML rather you have to use "out.println()" to show what you want to mix with HTML.

```
<h2> Hello World</h2>
<%
for(inti=0; i<= 5; i++) {
out.println("<br/> counting: " + i); }
%>
```

JSP Declarations

- Declarations come in handy when you have a code snippet that you want executed more than once. Using the declaration, you can declare the method in the beginning of the code and then call the same method whenever you need in the same page.
- Sysntax

  <% !

  Declaring Variable

  %>

JSP Declarations

```
<%!
    String makeItLower(String data)
    {
        returndata.toLowerCase();
    }
%>
```

- Here is how you call it
- Hello World:<%= makeItLower(“Hello World”) %>

JSP comments

- This tags are used to comment in JSP and ignored by the JSP container.
- <!—comment -->

Simple Example

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<!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>Guru JSP Example</title>
</head>
<body>
<%-- This is a JSP example with scriplets, comments , expressions --%>
<% out.println("This is JSP Example"); %>
<% out.println("The number is "); %>
<%! int num12 = 12; int num32 = 12; %>
<%= num12*num32 %>
Today's date: <%= (new java.util.Date()).toLocaleString()%>
</body>
</html>
```

JSP Directives: Page, Include & Taglib

JSP directives

- JSP directives are the messages to JSP container. They provide global information about an entire JSP page.
- JSP directives are used to give special instruction to a container for translation of JSP to servlet code.
- In JSP life cycle phase, JSP has to be converted to a servlet which is the translation phase.
- They give instructions to the container on how to handle certain aspects of JSP processing
- Directives can have many attributes by comma separated as key-value pairs.
- In JSP, directive is described in <%@ %> tags.

JSP directives

- Syntax of Directive:
  - <%@ directive attribute="" %>
- There are three types of directives:
  - Page directive
  - Include directive
  - Taglib directive

JSP Page directive

- **Syntax of Page directive:**
- <%@ page…%>It provides attributes that get applied to entire JSP page.
- It defines page dependent attributes, such as scripting language, error page, and buffering requirements.
- It is used to provide instructions to a container that pertains to current JSP page.

JSP Page directive

- Language
- Extends
- Import
- contentType
- info
- session
- isThreadSafe
- autoflush
- buffer
- IsErrorPage
- pageEncoding
- errorPage
- isELIgonored

JSP Include

- JSP "include directive“ is used to include one file to the another file
- This included file can be HTML, JSP, text files, etc.
- It is also useful in creating templates with the user views and break the pages into header&footer and sidebar actions.
- It includes file during translation phase

JSP Taglib Directive

- JSP taglib directive is used to define the tag library with "taglib" as the prefix, which we can use in JSP.
- More detail will be covered in JSP Custom Tags section
- JSP taglib directive is used in the JSP pages using the JSP standard tag libraries
- It uses a set of custom tags, identifies the location of the library and provides means of identifying custom tags in JSP page.
- **Syntax of taglib directive:**
- <%@ taglib uri="uri" prefix="value"%>Here "uri" attribute is a unique identifier in tag library descriptor and "prefix" attribute is a tag name.

Example

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="tg" uri="http://java.sun.com/jsp/jstl/core" %>
<!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>Directive JSP</title>
<tg:hello/>
</head>
<body>
</body>
</html>
```

Implicit Objects

- JSP implicit objects are created during the translation phase of JSP to the servlet.
- These objects can be directly used in scriplets that goes in the service method.
- They are created by the container automatically, and they can be accessed using objects.
- Now these objects are given to you for free, it’s not like you have to go ahead and pay to use them (like you do for some APIs), so there is no need to create them, you can just use them in your JSP pages. Here are a couple of the most commonly used server objects:

9 types of implicit objects

- out
- request
- response
- config
- application
- session
- pageContext
- page
- exception

Out

- Out is one of the implicit objects to write the data to the buffer and send output to the client in response
- Out object allows us to access the servlet's output stream
- Out is object of javax.servlet.jsp.jspWriter class
- While working with servlet, we need printwriter object

Page

- This object is of the type of pagecontext.
- It is used to get, set and remove the attributes from a particular scope
- Scopes are of 4 types:
  - Page
  - Request
  - Session
  - Application

JSP Action Tags
jsp useBean, jsp include, jsp forward

JSP actions useBean

- JSP actions use the construct in XML syntax to control the behavior of the servlet engine.
- We can dynamically insert a file, reuse the beans components, forward user to another page, etc. through JSP Actions like include and forward.
- Unlike directives, actions are re-evaluated each time the page is accessed.
- **Syntax:**
- <jsp:action_name attribute="value" />

Types of actions

- jsp:useBean
- jsp:include
- jsp:setProperty
- jsp:getProperty
- jsp:forward
- jsp:plugin
- jsp:attribute
- jsp:body
- jsp:text
- jsp:param
- jsp:attribute
- jsp:output

JSTL

- JSTL stands for Java Server page standard tag library, and it is a collection of custom JSP tag libraries that provide common web development functionality.

Advantages of JSTL

- **Standard Tag**: It provides a rich layer of the portable functionality of JSP pages. It's easy for a developer to understand the code.
- **Code Neat and Clean**: As scriplets confuse developer, the usage of JSTL makes the code neat and clean.
- **Automatic JavabeansInterospection Support**: It has an advantage of JSTL over JSP scriptlets. JSTL Expression language handles JavaBean code very easily. We don't need to downcast the objects, which has been retrieved as scoped attributes. Using JSP scriptlets code will be complicated, and JSTL has simplified that purpose.
- **Easier for humans to read**: JSTL is based on XML, which is very similar to HTML. Hence, it is easy for the developers to understand.
- **Easier for computers to understand**: Tools such as Dreamweaver and front page are generating more and more HTML code. HTML tools do a great job of formatting HTML code. The HTML code is mixed with the scriplet code. As JSTL is expressed as XML compliant tags, it is easy for HTML generation to parse the JSTL code within the document.

JSTL Core

- The core tags are most frequently used tags in JSP. They provide support for
  - Iteration
  - Conditional logic
  - Catch exception
  - url forward
  - Redirect, etc.
- <%@ taglib prefix="c" uri=http://java.sun.com/jsp/jstl/core%>
- prefix can be used to define all the core tags and
- uri is the library of taglib from which it is imported

Example

- <c:redirect url="/"/>
- <c:set var="count" value="100"/>
- <c:if test="${count == 100}">
- <c:out value="The count is 100"/>
- </c:if>

JSP Custom Tags

- It is a user-defined JSP language element.
- When JSP is translated into a servlet, custom tag is converted into a class which takes action on an object and is called as a tag handler.
- Those actions when the servlet is executed are invoked by the web container.
- To create the user-defined custom tag, we need to create the tag handler which will be extending the SimpleTagSupport and have to override doTag() method.
- We need to create TLD where we need to map the class file in TLD.

Advantages of custom tags in JSP

- **Portable** - An action described in a tag library must be usable in any JSP container.
- **Simple** - Unsophisticated users must be able to understand and use this mechanism.Vendors of JSP functionality must find it easy to make it available tousers as actions.
- **Expressive** - The mechanism must support a wide range of actions, includingnested actions, scripting elements inside action bodies, creation, use andupdating of scripting variables.
- **Usable from different scripting languages** - Although the JSP specificationcurrently only defines the semantics for scripts in the Java programming language, we want to leave open the possibility of other scripting languages.
- **Built upon existing concepts and machinery**- We do not want to reinvent whatexists elsewhere. Also, we want to avoid future conflicts whenever we canpredict them

JSP Client Request

Accept	It specifies MIME types that browser or other clients can handle	Image/png or image/jpeg
Accept-charset	It uses the character set used by the browser to display the information	ISO-8859-1
Accept- Encoding	It specifies type of encoding handled by the browser	Gzip or compress
Accept-language	It specifies clients specified language	En,en_us
Authorization	Header used by clients when trying to access password protected web pages	

Following methods are used to read the HTTP

- **Cookie[] getCookies() –** returns an array containing cookie objects that the client has sent
- **Enumeration getAttributeNames() –** contains enumeration of names of attributes for request
- **Enumeration getHeaderNames() -** contains enumeration of names of header .
- **Enumeration getParameterNames() –** contains enumeration of getting parameter names in the request.
- **HttpSessiongetSession() –** returns the current session associated with the request or if does not have a session then it will create a new one.
- **Locale getLocale() –** returns the preferred locale that client will accept content in.It has been assigned to the response. By default, the value will be default locale of the server.
- **Object getAttribute(String name) –** returns the value of named attribute as an object.
- **ServletInputStreamgetInputStream() –** retrievesbody of request as binary data.
- **String getAuthType() –** returns the name of authentication scheme to protect servlet
- **String getCharacterEncoding() –** returns name of the character encoding used in the body of the request.
- **String getContentType() –** returns the MIME type of body of the request.
- **String getContextPath() –** returns the part of request URI indicates context path of URI
- **String getHeader(String name) –** returns the request header as a string
- **String getMethod() –** returns the name of the HTTP method like GET, POST
- **String getParameter(String name) –** returns the parameter of the request as a string.
- **String getPathInfo() –** returns the path information associated with the URL
- **String getQueryString() –** returns the query string that is associated with the request URL
- **String getServletPath() –** returns the part of URLs of the request that calls the JSP
- **String[] getParameterValues(String name) –** returns the array of string objects containing the values that request parameter has

ing encodeRedirectURL(String URL) – encodes the URL in redirectURL method.
ing encodeURL(String URL) – encodes the URL by including session ID.
olean containsHeader(String name) – it contains a header in the JSP or not.
olean isCommited() – response has been committed or not.
d addCookie(Cookie cookie) – adds cookie to the response
d addDateHeader(String name, String value) – adds response header date name and
ue
d addHeader(String name, String value) – adds response header with name and valu
d addIntHeader(String name,int value) – adds response header with name and intege
ue
d flushBuffer() – forces content in the buffer to the output to the client.
d reset() – clears data in the buffer.
d resetBuffer – clears the content buffer in the response without clearing status codes.
d sendError(intsc,Stringmsg) – sends an error response to the client using status cod
d sendRedirect(String location) – sends a temporary redirect response to the client.
d setBufferSize(int size) – sets buffer size of the body
d setCharacterEncoding(String charset) – sets character encoding
d setContentType(String type) – sets the content type of the response
d setContentLength(intlen) – sets the content length of the response
d setLocale(Locale lcl) – sets the locale type of the response
d setStatus(intsc) – sets the status code of the response

Methods using response object

JSP HTTP Status Codes

- When the request is processed, the response is generated. The response status line consists of HTTP version, a status code and an associated message.
- The message is directly associated with the status code and HTTP version, and it is determined by the server.
- By default 200 is set as a status code in JSP, so we don't need to set explicitly.
- We can set as response.setStatus() method

Five categories

- 100-199 - Here client indicates that it should respond with some action
- 200-299 - It signifies that request is successful
- 300-399 - They are used for files that have been moved and usually include a location header indicating new address
- 400-499 - Indicates error by the client
- 500-599 - Indicates error by the server

Some of the common status codes

- 200 – Indicates everything is fine
- 301 – It has moved permanently
- 304 – Not modified since last change
- 400 – Bad request
- 404 – Not found
- 405 - Method not found
- 500 – Internal Server Error
- 503 - Service unavailable
- 505 – HTTP version not supported

Some of its methods are listed below

- Public void setStatus(intstatusCode)It sets the status code whichever we want to set in that JSP Page.This will give us the message of status code which has been set
- Public void sendRedirect(String URL)It generates 302 response along with the location header giving URL of the new document
- Public void sendError(intcode,Stringmsg)It sends the status code along with the short message and it is formatted inside HTML document.

Form Processing

- Forms are the common method in web processing. We need to send information to the web server and that information.
- There are two commonly used methods to send and get back information to the web server.
- GET & POST

GET

- This is the default method to pass information from browser to web server.
- It sends the encoded information separated by ?character appended to URL page.
- It also has a size limitation, and we can only send 1024 characters in the request.
- We should avoid sending password and sensitive information through GET method.

POST

- Post method is a most reliable method of sending information to the server.
- It sends information as separate message.
- It sends as text string after ?in the URL.
- It is commonly used to send information which are sensitive.

Data processing methods

- getParameter():It is used to get the value of the form parameter.
- getParameterValues():It is used to return the multiple values of the parameters.
- getParameterNames()It is used to get the names of parameters.
- getInputStream()It is used to read the binary data sent by the client.

JSP Filter Mapping in Web.xml

- Filters are used for filtering functionality of the Java web application.
- They intercept the requests from client before they try to access the resource
- They manipulate the responses from the server and sent to the client.
- There are different filters:
  - Authentication filters
  - Data compression filters
  - Encryption filters
  - MIME chain filters
  - Logging Filters
  - Tokenizing filters
- Filters are defined in web.xml, and they are a map to servlet or JSP.When JSP container starts with the web application, it creates the instance of each filter that have been declared in the deployment descriptor.

Following are the filter methods:

- **Public void doFilter(ServletRequest,ServletResponse, FilterChain)**This is called everytime when a request/response is passed from every client when it is requested from a resource.
- **Public void init(FilterConfig)**This is to indicate that filter is placed into service
- **Public void destroy()**This to indicate the filter has been taken out from service.