JSP –Java Server Pages

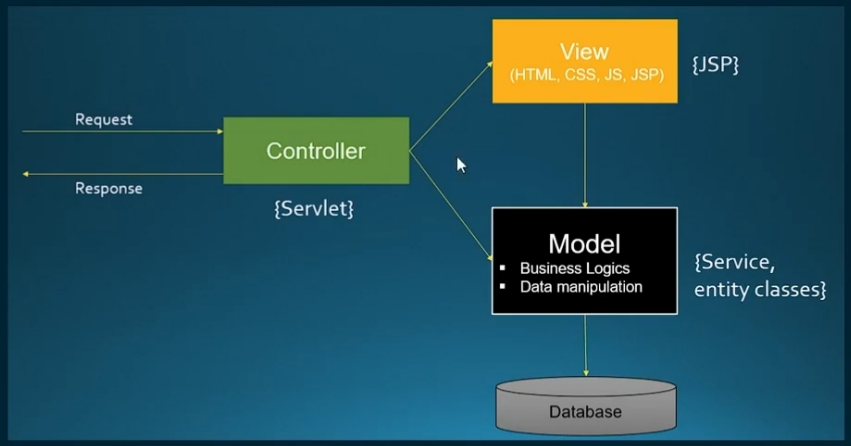
- Java Server Pages (JSP) is a technology which is used to develop web pages by inserting Java code into the HTML pages by making special JSP tags. The JSP tags which allow java code to be included into it are <% —-java code—-%>.
- It can consist of HTML and XML with JSP actions and commands.
- It can be used as HTML page, which can be used in forms and registration pages with the dynamic content.
- Dynamic content includes some fields like dropdown, checkboxes, etc. whose value will be fetched from the database.
- 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.

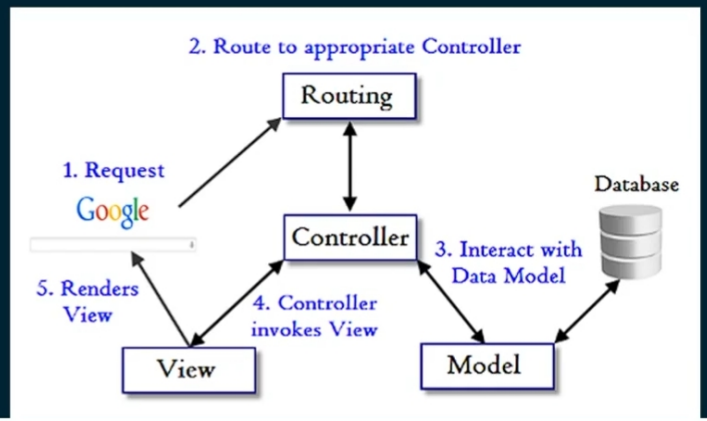
JSP Lifecycle

JSP Life Cycle is defined as translation of JSP Page into servlet as a JSP Page needs to be converted into servlet first in order to process the service requests. The Life Cycle starts with the creation of JSP and ends with the disintegration of that.

Phases of JSP Life Cycle

When the browser asks for a JSP, JSP engine first checks whether it needs to compile the page. If the JSP is last compiled or the recent modification is done in JSP, then the JSP engine compiles the page.

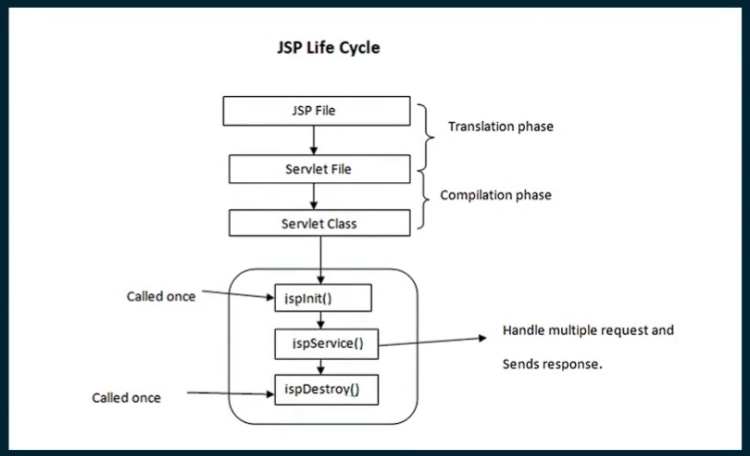

View
(HTML, CSS, JS, JSP)
{JSP}
Request
Controller
Response
{Servlet}
Model
Business Logics
Data manipulation
{Service,
entity classes}
Database


2. Route to appropriate Controller
Routing
1. Request
Google
Controller
3. Interact with
Data Model
Database
5. Renders
View
4. Controller
invokes View
View
Model

JSP Lifecycle is depicted in the below diagram.

Compilation process of JSP page involves three steps:

- Parsing of JSP
- Turning JSP into servlet
- Compiling the servlet



JSP pages are more advantageous than Servlet:

- They are easy to maintain.
- No recompilation or redeployment is required.
- JSP has access to entire API of JAVA .
- JSP are extended version of Servlet.

Features of JSP

- Coding in JSP is easy :- As it is just adding JAVA code to HTML/XML.
- Reduction in the length of Code :- In JSP we use action tags, custom tags etc.
- Connection to Database is easier :-It is easier to connect website to database and allows to read or write data easily to the database.
- Make Interactive websites :- In this we can create dynamic web pages which helps user to interact in real time environment.
- Portable, Powerful, flexible and easy to maintain :- as these are browser and server independent.
- No Redeployment and No Re-Compilation :- It is dynamic, secure and platform independent so no need to re-compilation.
- Extension to Servlet :- as it has all features of servlets, implicit objects and custom tags

Difference between Servlet and JSP

Servlet	JSP
Servlet is a java code.	JSP is a html based code.
Writing code for servlet is harder than JSP as it is html in java.	JSP is easy to code as it is java in html.
Servlet plays a controller role in MVC approach.	JSP is the view in MVC approach for showing output.
Servlet is faster than JSP.	JSP is slower than Servlet because the first step in JSP lifecycle is the translation of JSP to java code and then compile.
Servlet can accept all protocol requests.	JSP only accept http requests.
In Servlet, we can override the service() method.	In JSP, we cannot override its service() method.
In Servlet by default session management is not enabled, user have to enable it explicitly.	In JSP session management is automatically enabled.
In Servlet we have to implement everything like business logic and presentation logic in just one servlet file.	In JSP business logic is separated from presentation logic by using javaBeans.
Modification in Servlet is a time consuming task because it includes reloading, recompiling and restarting the server.	JSP modification is fast, just need to click the refresh button.

Advantages of using JSP

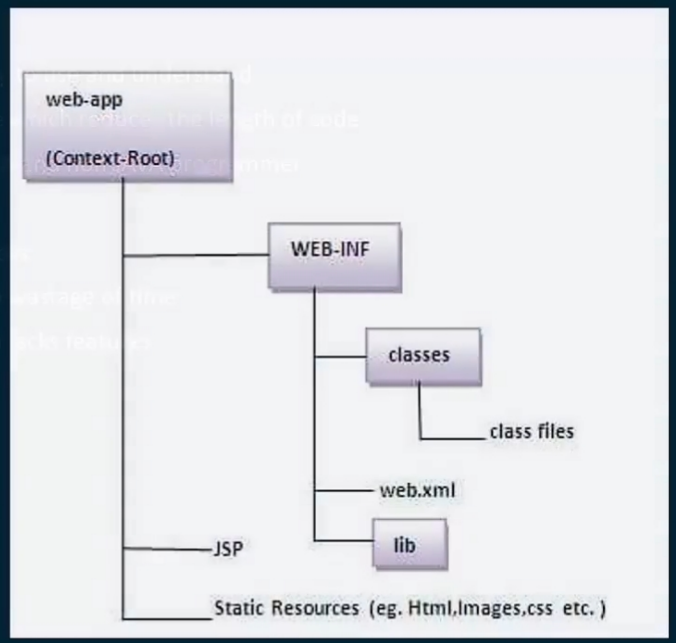
- It does not require advanced knowledge of JAVA
- It is capable of handling exceptions
- Easy to use and learn
- It can tags which are easy to use and understand
- Implicit objects are there which reduces the length of code
- It is suitable for both JAVA and non JAVA programmer

Disadvantages of using JSP

- Difficult to debug for errors.
- First time access leads to wastage of time
- It's output is HTML which lacks features.

The Directory structure of JSP

The directory structure of JSP page is same as Servlet. We contain the JSP page outside the WEB-INF folder or in any directory.



Scope of JSP Objects

The availability of a JSP object for use from a particular place of the application is defined as the scope of that JSP object. Every object created in a JSP page will have a scope. Object scope in JSP is segregated into four parts and they are page, request, session and application.

- page

'page' scope means, the JSP object can be accessed only from within the same page where it was created. The default scope for JSP objects created using <jsp:useBean> tag is page. JSP implicit objects out, exception, response, pageContext, config and page have 'page' scope.

- request

A JSP object created using the 'request' scope can be accessed from any pages that serves that request. More than one page can serve a single request. The JSP object will be bound to the request object. Implicit object request has the 'request' scope.

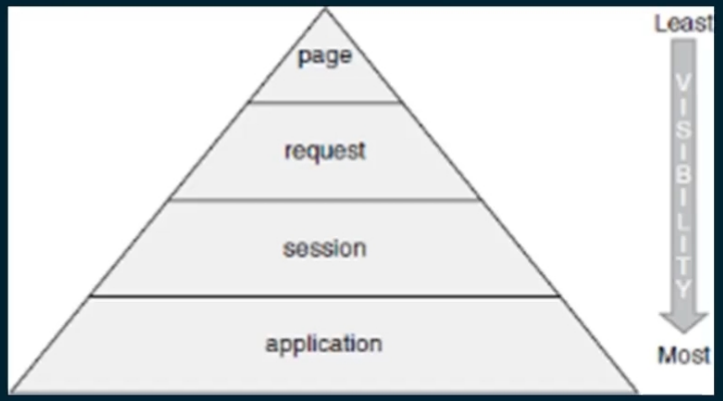
- session

'session' scope means, the JSP object is accessible from pages that belong to the same session from where it was created. The JSP object that is created using the session scope is bound to the session object. Implicit object session has the 'session' scope.

- application

A JSP object created using the 'application' scope can be accessed from any pages across the application. The JSP object is bound to the application object. Implicit object application has the 'application' scope.

Any object's visibility can be controlled by putting them in the appropriate scope in the Following figure

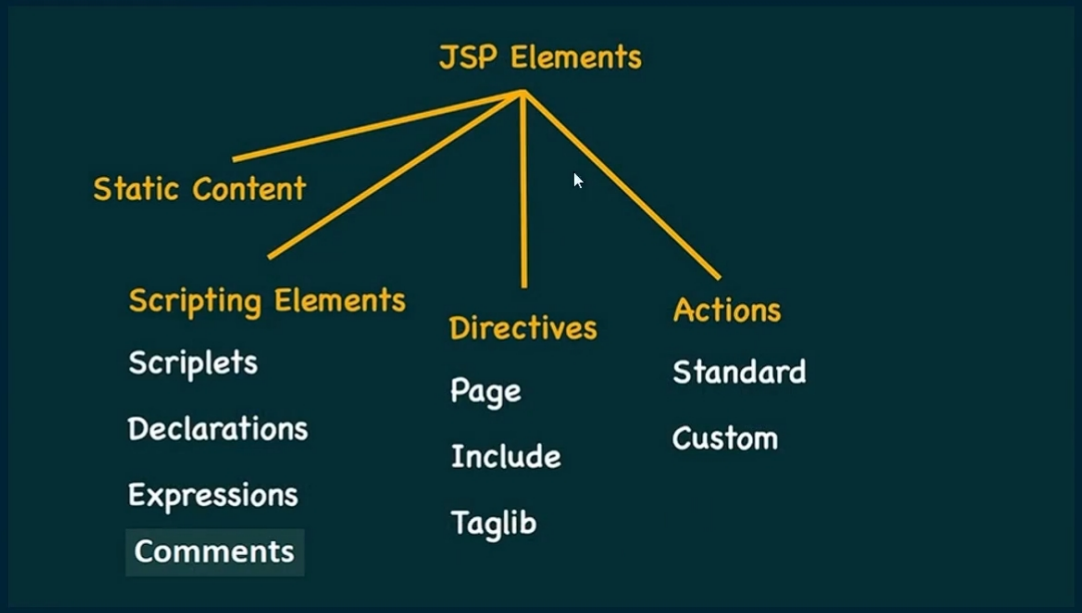


JSP Scope and State Management

- JSPs formalize this with four separate scopes
  1. Page :Within the same program component (web page)
  2. Request :Within the same request
  3. Session :Within all requests from the same session
  4. Application :Within all sessions for one servlet context

- Each can be accessed by different sets of program components
- Some exist for different periods of time

JSP Elements



JSP syntax

i) Java Scriplets :- It allows us to add any number of JAVA code -variables and expressions.

Syntax: <% java code %>

ii) Declaration Tag :-It is used to declare variables.

Syntax: <%! Dec var %>

Example: <%! int var=10; %>

iii) JSP Expression :- It evaluates and convert the expression to a string.

Syntax: <%= expression %>

Example:- <% num1 = num1+num2 %>

iv) JAVA Comments :- It contains the text that is added for information which has to be ignored.

Syntax: <% -- JSP Comments %>

JSP Scripting elements

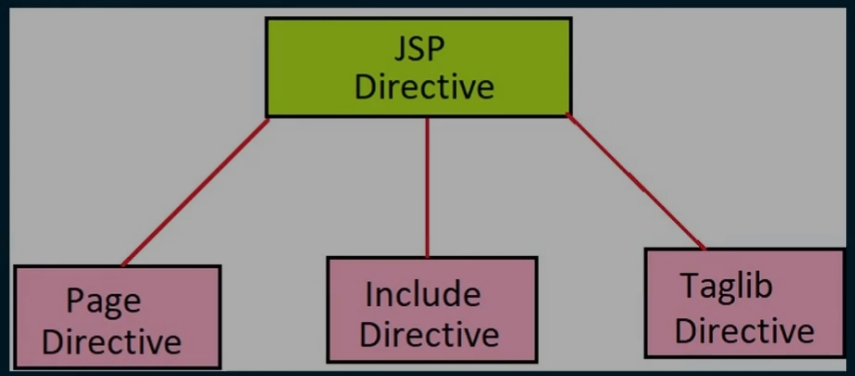
Tags or Elements	Description
<% ----------%>	scriptlet tag ✓
<%! ---------- %>	declaration tag ✓
<%= ----------%>	expression tag ✓
<%-- ---------- --%>	Comment tag ✓

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



JSP Page Directive

Page Directives : JSP page directive is used to define the properties applying the JSP page, such as the size of the allocated buffer, imported packages and classes/interfaces, defining what type of page it is etc.

Syntax

Page directive has following attributes:

<%@ page attribute="value" %>

- language=”scripting language”
- extends=”className”
- import=”importList”
- session=”true|false”
- buffer=”none|sizekb”
- autoFlush=”true|false”
- isThreadSafe=”true|false”
- info=”info_text”
- contentType=”ctinfo”
- errorPage=”error_url”
- isErrorPage=”true|false”
- pageEncoding= “value”
- isELignored= “true|false”

1)language

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

2)extends

The extends attribute defines the parent class that will be inherited by the generated servlet. It is rarely used.

3)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.

4) session

In JSP, the session is the most regularly used implicit object of type HttpSession. It is mainly used to approach all data of the user until the user session is active.

5) 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.

6) autoFlush

The autoFlush attribute specifies whether the buffered output should be flushed automatically when the buffer is filled, or whether an exception should be raised to indicate the buffer overflow.

A value of true (default) indicates automatic buffer flushing and a value of false throws an exception.

7) isThreadSafe

The isThreadSafe option marks a page as being thread-safe. By default, all JSPs are considered thread-safe. If you set the isThreadSafe option to false, the JSP engine makes sure that only one thread at a time is executing your JSP.

value of the attribute is false and expressions are evaluated as specified.

8) info

The info attribute lets you provide a description of the JSP.

9) contentType

The contentType attribute sets the character encoding for the JSP page and for the generated response page. The default content type is text/html, which is the standard content type for HTML pages.

10) errorPage

The errorPage attribute tells the JSP engine which page to display if there is an error while the current page runs. The value of the errorPage attribute is a relative URL.

11) isErrorPage

The isErrorPage attribute indicates that the current JSP can be used as the error page for another JSP.

The value of isErrorPage is either true or false. The default value of the isErrorPage attribute is false.

12) pageEncoding

It is a character encoding in which JSP files are encoded. The default value for page encoding is ISO-8859-1.

It is a character8-bit (single-byte) coded graphic character set. Page encoding is determined as follows:

- Through the value of page encoding in JSP Property group.
- Through the value present in page encoding attribute. A translation error is generated if the page encoding of a JSP page is different from its property groups.
- Through the contentType attribute that has charset value.

The pageEncoding and contentType attribute tells about page encoding of that JSP file in which page directive is physically present.

13) isELignored

The attribute stands for Expression Language ignored. This attribute specifies the inability of an expression to be evaluated. Simply saying that expression will be considered static and will not be evaluated. However the default value of the attribute is false and expressions are evaluated as specified.

TagLib Directive

Taglib Directive : The taglib directive is used to mention the library whose custom-defined tags are being used in the JSP page. It's major application is JSTL(JSP standard tag library).

Syntax:

<%@ taglib uri="taglib URI" prefix="value"%>

The attributes of this directive are:

- tagLibrary URI
- tagPrefix

a. tagLibraryURI

This attribute contains the URL of a Tag Library Descriptor as a value.

b. tagPrefix

It contains unique prefixes for the identification of custom tags.

Install JSTL Library

To begin working with JSP tages you need to first install the JSTL library. If you are using the Apache Tomcat container, then follow these two steps –

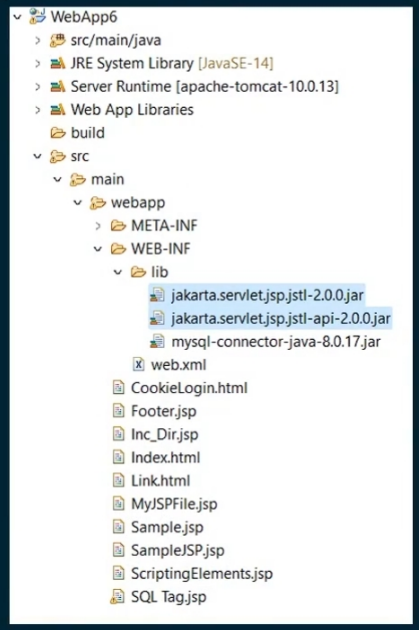
Step 1 – Download the binary distribution from Apache Standard Taglib and unpack the compressed file.

Step 2 – To use the Standard Taglib from its Jakarta Taglibs distribution, simply copy the JAR files in the distribution's 'lib' directory to your application's webapps\ROOT\WEB-INF\lib directory.

To use any of the libraries, you must include a <taglib> directive at the top of each JSP that uses the library.

Installing JSTL on Tomcat 10+

- jakarta.servlet.jsp.jstl-2.0.0.jar (this is the JSTL 2.0 impl of EE4J)
- jakarta.servlet.jsp.jstl-api-2.0.0.jar (this is the JSTL 2.0 API)



Installing JSTL on Tomcat 9

In case you're not on Tomcat 10 yet, but still on Tomcat 9 or older, use JSTL 1.2 via this sole dependency (this is compatible with Tomcat 9 / 8 / 7 / 6 / 5):

- jakarta.servlet.jsp.jstl-1.2.6.jar (this is the JSTL 1.2 impl of EE4J)
- jakarta.servlet.jsp.jstl-api-1.2.7.jar (this is the JSTL 1.2 API)

Classification of The JSTL Tags

The JSTL tags can be classified, according to their functions, into the following JSTL tag library groups that can be used when creating a JSP page –

- Core Tags
- Formatting tags
- SQL tags
- XML tags
- JSTL Functions

Core Tags

The core group of tags are the most commonly used JSTL tags. Following is the syntax to include the JSTL Core library in your JSP –

```
<%@ taglib prefix = "c" uri = "http://java.sun.com/jsp/jstl/core" %>
```

Following table lists out the core JSTL Tags –

S.No.	Tag & Description
1	<c:out> Like <%= ... >, but for expressions.
2	<c:set > Sets the result of an expression evaluation in a 'scope'
3	<c:remove > Removes a scoped variable (from a particular scope, if specified).
4	<c:catch> Catches any Throwable that occurs in its body and optionally exposes it.
5	<c:if> Simple conditional tag which evalutes its body if the supplied condition is true.
6	<c:choose> Simple conditional tag that establishes a context for mutually exclusive conditional operations, marked by <when> and <otherwise>.
7	<c:when> Subtag of <choose> that includes its body if its condition evalutes to 'true'.
8	<c:otherwise > Subtag of <choose> that follows the <when> tags and runs only if all of the prior conditions evaluated to 'false'.
9	<c:import> Retrieves an absolute or relative URL and exposes its contents to either the page, a String in 'var', or a Reader in 'varReader'.
10	<c:forEach > The basic iteration tag, accepting many different collection types and supporting subsetting and other functionality .
11	<c:forTokens> Iterates over tokens, separated by the supplied delimeters.
12	<c:param> Adds a parameter to a containing 'import' tag's URL.
13	<c:redirect > Redirects to a new URL.
14	<c:url> Creates a URL with optional query parameters

Formatting Tags

The JSTL formatting tags are used to format and display text, the date, the time, and numbers for internationalized Websites. Following is the syntax to include Formatting library in your JSP –

<%@ taglib prefix = "fmt" uri = "http://java.sun.com/jsp/jstl/fmt" %>

Following table lists out the Formatting JSTL Tags –

S.No.	Tag & Description
1	<fmt:formatNumber> To render numerical value with specific precision or format.
2	<fmt:parseNumber> Parses the string representation of a number, currency, or percentage.
3	<fmt:formatDate> Formats a date and/or time using the supplied styles and pattern.
4	<fmt:parseDate> Parses the string representation of a date and/or time
5	<fmt:bundle> Loads a resource bundle to be used by its tag body.
6	<fmt:setLocale> Stores the given locale in the locale configuration variable.
7	<fmt:setBundle> Loads a resource bundle and stores it in the named scoped variable or the bundle configuration variable.
8	<fmt:timeZone> Specifies the time zone for any time formatting or parsing actions nested in its body.
9	<fmt:setTimeZone> Stores the given time zone in the time zone configuration variable
10	<fmt:message> Displays an internationalized message.
11	<fmt:requestEncoding> Sets the request character encoding

SQL Tags

The JSTL SQL tag library provides tags for interacting with relational databases (RDBMSs) such as Oracle, mySQL, or Microsoft SQL Server.

Following is the syntax to include JSTL SQL library in your JSP –

<%@ taglib prefix = "sql" uri = "http://java.sun.com/jsp/jstl/sql" %>

Following table lists out the SQL JSTL Tags –

S.No.	Tag & Description
1	<sql:setDataSource> Creates a simple DataSource suitable only for prototyping
2	<sql:query> Executes the SQL query defined in its body or through the sql attribute.
3	<sql:update> Executes the SQL update defined in its body or through the sql attribute.
4	<sql:param> Sets a parameter in an SQL statement to the specified value.
5	<sql:dateParam> Sets a parameter in an SQL statement to the specified java.util.Date value.
6	<sql:transaction > Provides nested database action elements with a shared Connection, set up to execute all statements as one transaction.

XML tags

The JSTL XML tags provide a JSP-centric way of creating and manipulating the XML documents. Following is the syntax to include the JSTL XML library in your JSP.

The JSTL XML tag library has custom tags for interacting with the XML data. This includes parsing the XML, transforming the XML data, and the flow control based on the XPath expressions.

<%@ taglib prefix = "x" uri = "http://java.sun.com/jsp/jstl/xml" %>

Before you proceed with the examples, you will need to copy the following two XML and XPath related libraries into your <Tomcat Installation Directory>\lib −

- XercesImpl.jar − Download it from https://www.apache.org/dist/xerces/j/
- xalan.jar − Download it from https://xml.apache.org/xalan-j/index.html

Following is the list of XML JSTL Tags –

S.No.	Tag & Description
1	<x:out> Like <%= ... >, but for XPath expressions.
2	<x:parse> Used to parse the XML data specified either via an attribute or in the tag body.
3	<x:set > Sets a variable to the value of an XPath expression.
4	<x:if > Evaluates a test XPath expression and if it is true, it processes its body. If the test condition is false, the body is ignored.
5	<x:forEach> To loop over nodes in an XML document.
6	<x:choose> Simple conditional tag that establishes a context for mutually exclusive conditional operations, marked by <when> and <otherwise> tags.
7	<x:when > Subtag of <choose> that includes its body if its expression evalutes to 'true'.
8	<x:otherwise > Subtag of <choose> that follows the <when> tags and runs only if all of the prior conditions evaluates to 'false'.
9	<x:transform > Applies an XSL transformation on a XML document
10	<x:param > Used along with the transform tag to set a parameter in the XSLT stylesheet

JSTL Functions

JSTL includes a number of standard functions, most of which are common string manipulation functions. Following is the syntax to include JSTL Functions library in your JSP –

```
<%@ taglib prefix = "fn"
   uri = "http://java.sun.com/jsp/jstl/functions" %>
```

Following table lists out the various JSTL Functions –

S.No.	Function & Description
1	fn:contains() Tests if an input string contains the specified substring.
2	fn:containsIgnoreCase() Tests if an input string contains the specified substring in a case insensitive way.
3	fn:endsWith() Tests if an input string ends with the specified suffix.
4	fn:escapeXml() Escapes characters that can be interpreted as XML markup.
5	fn:indexOf() Returns the index within a string of the first occurrence of a specified substring.
6	fn:join() Joins all elements of an array into a string.
7	fn:length() Returns the number of items in a collection, or the number of characters in a string.
8	fn:replace() Returns a string resulting from replacing in an input string all occurrences with a given string.
9	fn:split() Splits a string into an array of substrings.
10	fn:startsWith() Tests if an input string starts with the specified prefix.
11	fn:substring() Returns a subset of a string.
12	fn:substringAfter() Returns a subset of a string following a specific substring.
13	fn:substringBefore() Returns a subset of a string before a specific substring.
14	fn:toLowerCase() Converts all of the characters of a string to lower case.
15	fn:toUpperCase() Converts all of the characters of a string to upper case.
16	fn:trim() Removes white spaces from both ends of a string.

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

Standard Action Tags available in JSP

There are 12 types of Standard Action Tags in JSP. Here is the list of Standard Action tags in JSP:

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

1) jsp:include

This action will include the required resources like html, servlets and JSP.
This jsp:include action is different from jsp directive. Include directive includes resources at the ***time of translation***, whereas include action includes resources dynamically at ***request time***. Action directives work well for static pages, whereas later works better for dynamic pages. There are two attributes under include:

- **Page:** its value is the url of the required resource.
- **Flush:** it checks that the buffer of the resource is flushed before it is included.

Syntax

```
<jsp:include page="page URL" flush="true/false">
```

Program
(include.jsp)

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Include Example</title>
</head>
<body>
 <jsp:include page="date.jsp" flush="true"></jsp:include>
</body>
</html>
```

2) jsp:useBean

<jsp:useBean> standard action tag is use to establish a connection between a jsp page and java bean.

In web applications of java, jsp and java bean communication is required in the following two cases:

- In a real-time MVC project, a model class (business class) will set the data to a java bean and a jsp (view) will read the data from a bean and finally displays it on the browser. In this class jsp to a java bean communication is required.
- If multiple jsp pages need common java logic then it separates that java code into a bean and then we call the bean from jsp. In this case also jsp to java bean communication is required.

Every java class is not a java bean automatically. A class should have the following qualities to make it as a java bean.

- Class must be public.
- Class must contain default constructor.
- Each private property of the class must have setter or getter or both methods.
- A class can at most implement Serializable interface.

Syntax

```
<jsp:useBean id = "beanName" class = "className" scope = "page | request |  session | application">
</jsp:useBean>
```

- id: id uniquely identifies bean in a specified scope.
- Class: Creates object of bean class.
- Scope

1. Page: It is the default scope that states that we can use bean within this JSP page.
2. Request: It is broader than a page as bean can be used from any page that processes similar requests.
3. Session: It has a wider range than above as bean can be used from any page in the same session.
4. Application: It has the maximum range as bean can be used from any page present in the same application.

3) jsp:setProperty

jsp:setProperty and getProperty are used with usebeans. It is used to modify the properties of beans. It gets executed when new objects are created.

Syntax

```
<jsp:setProperty property="" name="" param="" value=""/>
```

4) jsp:getProperty

- This property is used to get the property of the bean.
- It converts into a string and finally inserts into the output.

Syntax

```
<jsp:getProperty name = "myName" property = "someProperty" .../>
```

5) jsp:forward

It is used to forward the request to another jsp or any static page or Java Servlet.

Here the request can be forwarded with no parameters or with parameters.

Syntax

```
<jsp:forward page = "Relative URL" />
```

6) jsp:plugin

Plugin inserts java objects or embed tags into a JSP page. It has three attributes:

- type: bean
- code: It is the name of the class file.
- codebase: It is the directory of the name of the class file.

Syntax

```
<jsp:plugin type= "applet | bean"  code= "nameOfClassFile"   codebase="directoryNameOfClassFile"
  </jsp:plugin>
```

7) jsp:param

It sends parameters to a bean or applet. It can also be said as the child object of a plugin.

Syntax

```
<jsp:params>
        <jsp:param name=“__” value="__"/ >
</jsp:params>
```

8) jsp:body

The body of an action is normally defined as the body of the tag. Then you can also define the body of an action using the <jsp:body> standard action. This is required if one or more <jsp:attribute>, <jsp:element> appear in the body of the tag.

i) jsp:element

The <jsp:element> dynamically creates an XML element and adds it to the response. It's useful primarily in JSP Documents (JSP pages in XML syntax), where other approaches can't be used because of the well-formedness requirement.

Syntax 1: Without a body

```
<jsp:element name="elementName" />
```

Syntax 2: With a body

```
<jsp:element name="elementName">
  <jsp:attribute> and/or <jsp:body> actions
</jsp:element>
```

ii) jsp:attribute

Once XML is defined, we can also define its attribute dynamically.

Syntax

```
<jsp:attribute name=" Put your XMLattribute here.."></jsp:attribute>
```

If one or more of these actions are used, the body of the tag can be specified only with a <jsp:body> action. If this is omitted, then the body of the tag is empty.

JSP Custom Tags

User-defined tags are known as custom tags. These tags are mainly used for code re-usability.

How to use custom tag?

<prefix:tagname attributeName=attributeValue />

A custom tag may have zero or n attributes.

To create a custom tag we need three things:

1) Tag handler class: is used to write the action for the custom tag. Our tag handler class inherits the TagSupport class. Custom tag action will be written in the doStartTag() method of TagSupport class.

2) TLD file: Tag Library Descriptor file where we will specify our tag name, tag handler class and tag attributes.

Commonly used attributes in TLD file:

1. <name>: Name of the custom tag.
2. <tag-class>: Fully qualified name of the tag handler class.

3) JSP page: A JSP page where we will be using our custom tag.

Expression Language in JSP

We saw in earlier posts, how we can use scriptlets and JSP expressions to retrieve attributes and parameters in JSP with java code and use it for view purpose. But for web designers, java code is hard to understand and that's why JSP Specs 2.0 introduced Expression Language (EL) through which we can get attributes and parameters easily using HTML like tags.

Expression language syntax is ${name} and we will see how we can use them in JSP code.

JSP Expression Language – JSP EL Reserve Words

and	or	not	eq	ne
lt	gt	le	ge	true
false	null	instanceof	empty	div,mod