Advance Java

Prof. Honey Parmar, Assistant Professor
IT and Computer Science

Parul® University

DIGITAL LEARNING CONTENT

Unit -3

Java Server Pages

Introduction to JSP

- **JSP** technology is used to create web applications just like Servlet technology.
- It can be considered an extension to servlet because it provides more functionality than servlet such as expression language, JSTL, etc.
- A JSP page consists of HTML tags and JSP tags.
- The JSP pages are easier to maintain than Servlet because we can separate design and development.
- It provides additional features such as Expression Language, Custom Tags etc.

Need for JSP

- Overcomes the limitations of Servlets by separating business logic and presentation.
- Reduces complexity in web development.
- Provides easier management for large-scale web applications.

Advantages of JSP over Servlets

1. Extension to Servlet

✔ JSP technology is the extension to Servlet technology. We can use all the features of the Servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy

2. Easy to maintain

✔ JSP can be easily managed because we can easily separate our business logic with presentation logic. In Servlet technology, we mix our business logic with the presentation logic

Advantages of JSP over Servlets

3. Fast Development: 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.

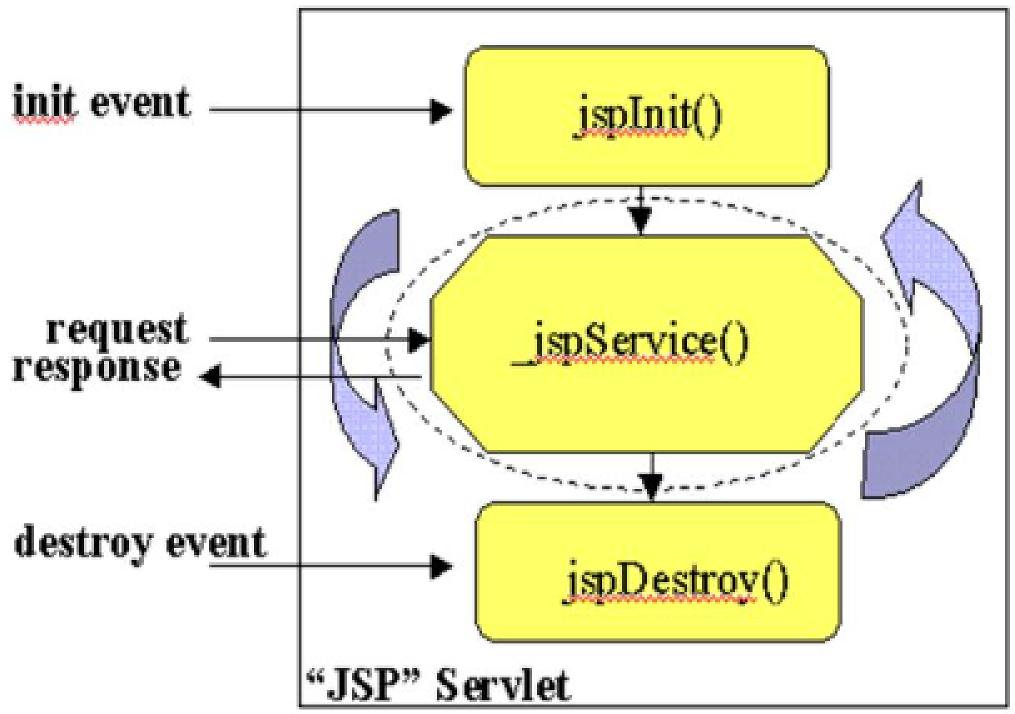
4. Less code than Servlet

✔ In JSP, we can use many tags such as action tags, JSTL, custom tags, etc. that reduces the code. Moreover, we can use EL, implicit objects, etc.

Life cycle of a JSP

- Translation of JSP Page
- Compilation of JSP Page
- Classloading (the classloader loads class file)
- 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).

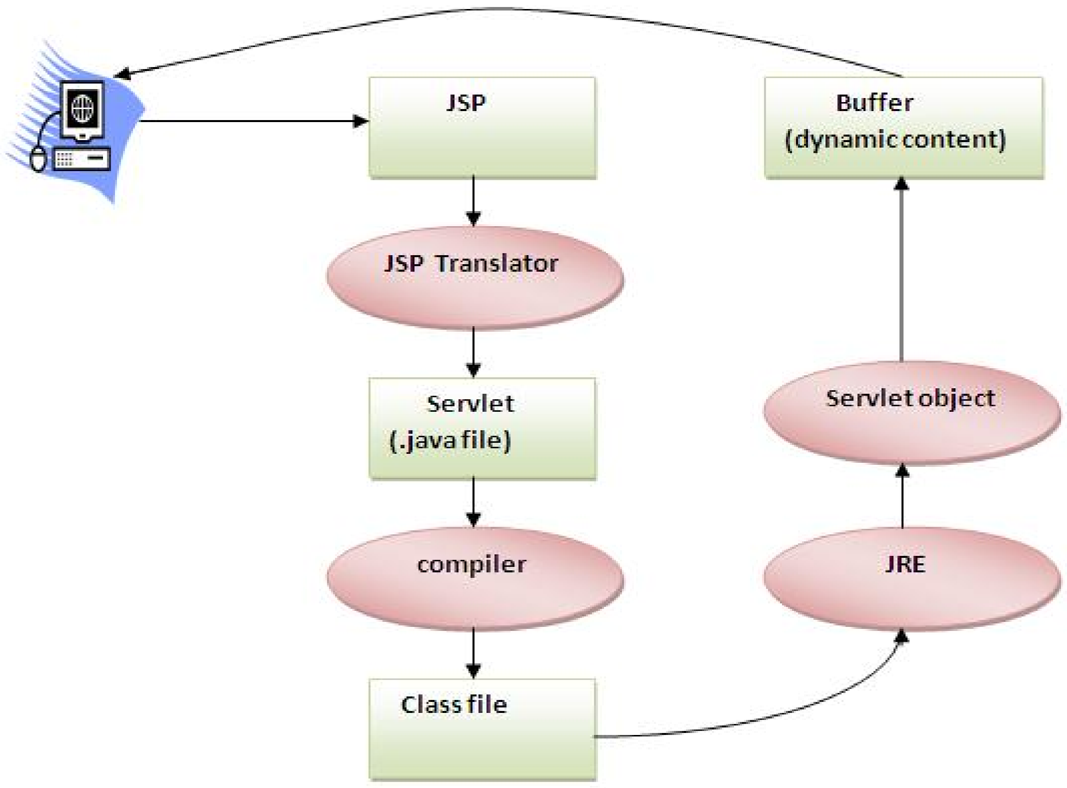
Life cycle of a JSP



Parul® University

DIGITAL LEARNING CONTENT

Life cycle of a JSP



JSP basic tags

- 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:

1. scriptlet tag
2. expression tag
3. declaration tag

scriptlet tag

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

```
<%  java source code %>
```

Example of JSP scriptlet tag

```
<html>
<body>
<% out.print("welcome to jsp"); %>
</body>
</html>
```

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

<%= statement %>

Example of JSP expression tag

In this example of jsp expression tag, we are simply displaying a welcome message.

<html>

<body>

<%= "welcome to jsp" %> **</body>** **</html>**

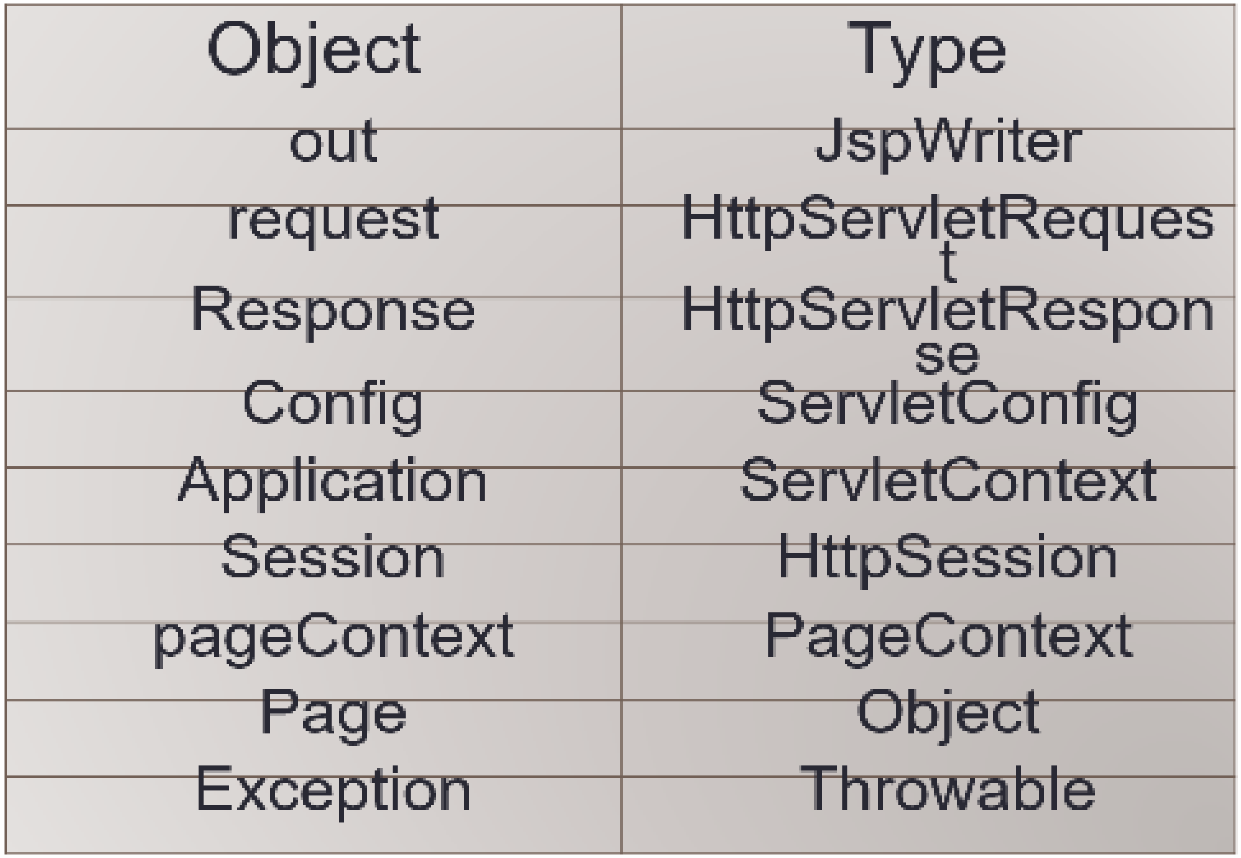
declaration tag

- 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**
- `<%!  field or method declaration %>`

Implicit Objects

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

Implicit Objects



Object	Type
out	JspWriter
request	HttpServletRequest
Response	HttpServletResponse
Config	ServletConfig
Application	ServletContext
Session	HttpSession
pageContext	PageContext
Page	Object
Exception	Throwable

JSP out implicit object

- For writing any data to the buffer, JSP provides an implicit object named out. It is the object of JspWriter. In case of servlet you need to write:

```
PrintWriter out=response.getWriter();
<html>
<body>
<% out.print("Today is:"+java.util.Calendar.getInstance().
  getTime());
%>
</body>
</html>
```

JSP request implicit object

- The **JSP request** is an implicit object of type HttpServletRequest i.e. created for each jsp request by the web container.
- It can be used to get request information such as parameter, header information, remote address, server name, server port, content type, character encoding etc.
- It can also be used to set, get and remove attributes from the jsp request scope.

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 be used to add or manipulate response such as redirecting responses to another resource, send error etc.

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.

JSP application implicit object

- In JSP, application is an implicit object of type *ServletContext*.
- The instance of ServletContext 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 configuration file (web.xml).
- It can also be used to get, set or remove attribute from the application scope.

Parul® University

DIGITAL LEARNING CONTENT

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.

pageContext implicit object

- In JSP, pageContext 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:

1. page
2. request
3. session
4. application

page implicit object:

- In JSP, page is an implicit object of type Object class.
- This object is assigned to the reference of auto generated servlet class. It is written as:

  Object page=this;
- For using this object it must be cast to Servlet type.
- For example:

```
<% (HttpServlet)page.log("message"); %>
```

exception implicit object

- In JSP, exception 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.

error.jsp

```
<%@ page isErrorPage="true" %>
<html>
<body>
```

Introduction to java beans

Java Bean

- A Java Bean is a java class that should follow following conventions:
- It should have a no-arg constructor.
- It should be Serializable.
- It should provide methods to set and get the values of the properties, known as getter and setter methods.

Why use Java Bean?

- According to Java white paper, it is a reusable software component.
- A bean encapsulates many objects into one object, so we can access this object from multiple places.
- Moreover, it provides the easy maintenance.

Key Features:
Public default constructor.
Implements Serializable.
Getter and setter methods for properties.
Purpose:
Simplifies data management and interaction in server-side

Simple Example of Java Bean Class

```
import java.io.Serializable;
public class UserBean implements Serializable {
    private String name;
    private int age;
    public UserBean() {} // Default Constructor
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

    public int getAge() { return age; }
    public void setAge(int age) { this.age = age; }
}
```

Instantiating a JavaBean in JSP:

```
<jsp:useBean id="user" class="UserBean" scope="session" />
```

Scopes:

1. page: Current JSP only.
2. request: Current request.
3. session: Entire session.
4. application: Entire application.

Parul® University

DIGITAL LEARNING CONTENT

Writing Properties of JavaBeans

Setting Properties with **<jsp:**
setProperty>:

<jsp:setProperty name="beanName" property="propertyName" value="value" />

Example:

<jsp:setProperty name="user" property="name" value="John Doe" />

<jsp:setProperty name="user" property="age" value="30" />

Automatic Mapping:

<jsp:setProperty name="user" property="*" />

Maps all matching request parameters to JavaBean properties

Parul® University

DIGITAL LEARNING CONTENT

Reading Properties of JavaBeans

getProperty :

```
<jsp:getProperty name="beanName"
property="propertyName" />

<p>Name: <jsp:getProperty name="user" property="name" /
></p>
<p>Age: <jsp:getProperty name="user" property="age" /></
p>
```

jsp:useBean action tag

- The jsp:useBean action tag 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.

Syntax of jsp:useBean action tag:

<jsp:
useBean id= "instanceName" scope= "page | request | sessio
n | application"
class= "packageName.className" type= "packageName.
className"

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.
3. **page:** specifies that you can use this bean within the JSP page. The default scope is page.
4. **request:** specifies that you can use this bean from any JSP page that processes the same request. It has wider scope than page.
5. **session:** specifies that you can use this bean from any JSP page in the same session whether processes the same request or not. It has wider scope than request.
6. **application:** specifies that you can use this bean from any JSP page in the same application. It has wider scope

7. **class:** instantiates the specified bean class (i.e. creates an object of the bean class) but it must have no-arg or no constructor and must not be abstract.
8. **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.
9. **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. In web devlopment, bean class is mostly used because it is a reusable software component that represents data.
- The jsp:setProperty action tag sets a property value or values in a bean using the setter method.

Syntax of jsp:setProperty action tag

```
<jsp:setProperty name="instanceOfBean" property= "*"   |
property="propertyName" param="parameterName"  |
property="propertyName" value="{ string | <%= expression
%>}"
/>
```

Example of jsp:setProperty action tag if you have to set all the values of incoming request in the bean

```
<jsp:setProperty name="bean" property="*" />
```

Parul® University

DIGITAL LEARNING CONTENT

Expression Language (EL) in JSP

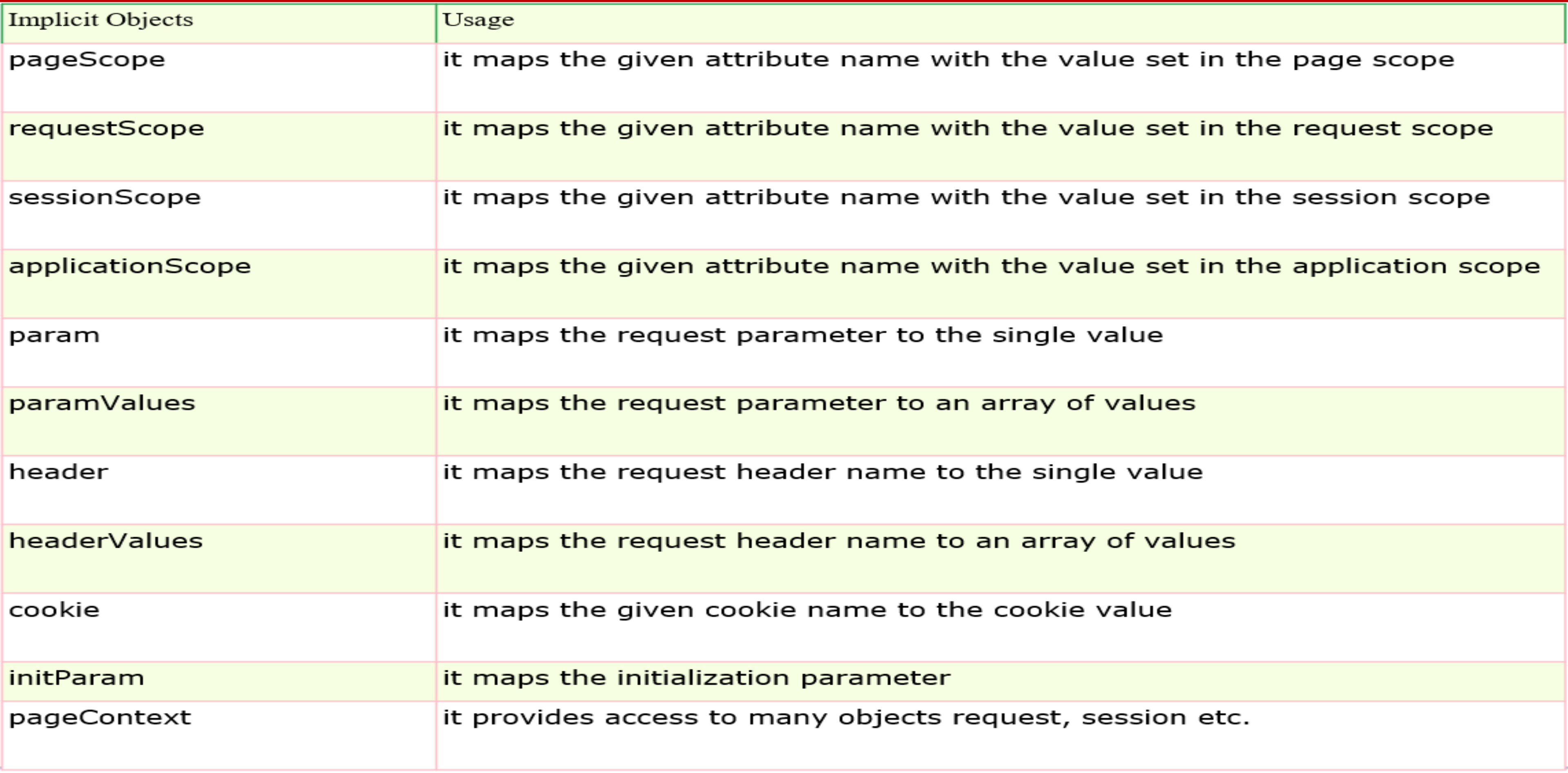
- The **Expression Language** (EL) simplifies the accessibility of data stored in the Java Bean component, and other objects like request, session, application etc.
- **Syntax for Expression Language**

  ${ expression }

Parul® University

DIGITAL LEARNING CONTENT

Implicit Objects

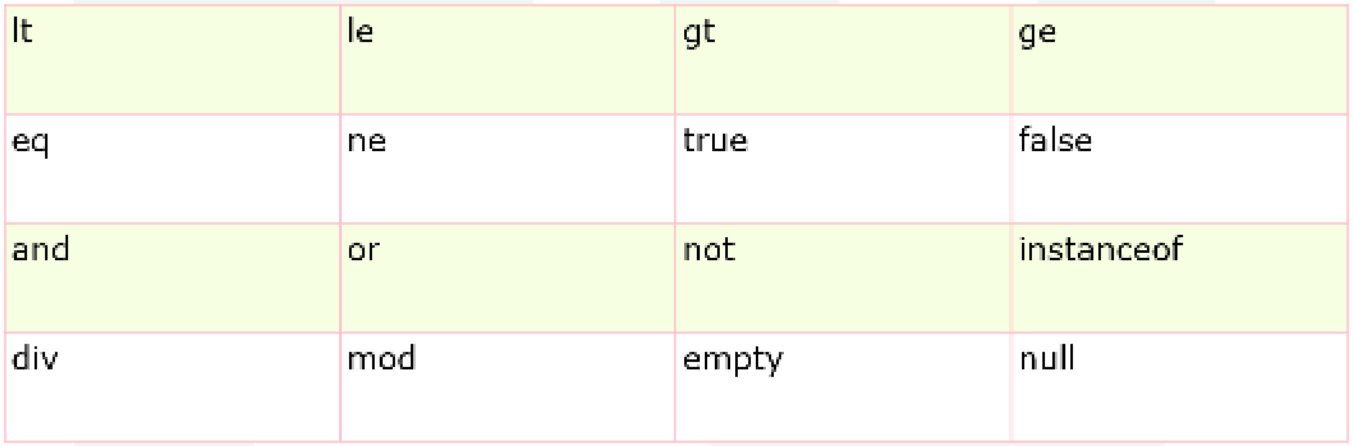


Implicit Objects	Usage
pageScope	it maps the given attribute name with the value set in the page scope
requestScope	it maps the given attribute name with the value set in the request scope
sessionScope	it maps the given attribute name with the value set in the session scope
applicationScope	it maps the given attribute name with the value set in the application scope
param	it maps the request parameter to the single value
paramValues	it maps the request parameter to an array of values
header	it maps the request header name to the single value
headerValues	it maps the request header name to an array of values
cookie	it maps the given cookie name to the cookie value
initParam	it maps the initialization parameter
pageContext	it provides access to many objects request, session etc.

Operators in EL

[] .
()
-(unary) not ! empty
* / div % mod
+ - (binary)
< <= > >= less than less or equal greater than greater or equal
== != eq ne
&& and
|| or
?:

Reserve words



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

Invoking Expression Language

- EL can be used directly in JSP pages to access data and objects.

```
<jsp:useBean id="user" class="UserBean" scope="session" />
Name: ${user.name}
Age: ${user.age}
```

Preventing EL Evaluation

By Default: EL expressions are evaluated in JSP 2.0.

To Disable EL Evaluation:

```
<%@ page isELIgnored="true" %>
```

Disable EL in the deployment descriptor (web.xml):

```
<jsp-config>
   <el-ignored>true</el-ignored>
</jsp-config>
```

Escape EL expressions using \ :

```
\${user.name}
```

Parul® University

DIGITAL LEARNING CONTENT

Accessing Scoped Variables

EL allows you to access variables from any scope (page, request, session, application).

Precedence of Scopes:

pageScope
requestScope
sessionScope
applicationScope

Syntax:

${scopeName.
variableName}

Example:

${pageScope.message}

Parul® University

JSTL -JSP - Standard Tag Library

- The JavaServer Pages Standard Tag Library (JSTL) is a collection of useful JSP tags which encapsulates core functionality common to many JSP applications.
- JSTL has support for common, structural tasks such as iteration and conditionals, tags for manipulating XML documents, internationalization tags, and SQL tags.
- It also provides a framework for integrating existing custom tags with JSTL tags.

- The JSTL tags can be classified, according to their functions

1. **Core Tags**
2. **Formatting tags**
3. **SQL tags**
4. **XML tags**
5. **JSTL Functions**

Install JSTL Library

- If you are using Apache Tomcat container then follow the following two simple steps:
- Download the binary distribution from Apache Standard Taglib and unpack the compressed file.
- 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.

Core Tags

- The core group of tags are the most frequently used JSTL tags.

Syntax:

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

Tag	Description
<c:out >	Like <%= ... >, but for expressions.
<c:set >	Sets the result of an expression evaluation in a 'scope'
<c:remove >	Removes a scoped variable (from a particular scope, if specified).
<c:catch>	Catches any Throwable that occurs in its body and optionally exposes it.
<c:if>	Simple conditional tag which evalutes its body if the supplied condition is true.
<c:choose>	Simple conditional tag that establishes a context for mutually exclusive conditional operations, marked by <when> and <otherwise>
<c:when>	Subtag of <choose> that includes its body if its condition evalutes to 'true'.
<c:otherwise >	Subtag of <choose> that follows <when> tags and runs only if all of the prior conditions evaluated to 'false'.
<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'.
<c:forEach >	The basic iteration tag, accepting many different collection types and supporting subsetting and other functionality .
<c:forTokens>	Iterates over tokens, separated by the supplied delimeters.
<c:param>	Adds a parameter to a containing 'import' tag's URL.
<c:redirect >	Redirects to a new URL.
<c:url>	Creates a URL with optional query parameters

DIGITAL LEARNING CONTENT