Object Oriented Programming

LESSON 11

Java Sever Page (JSP)

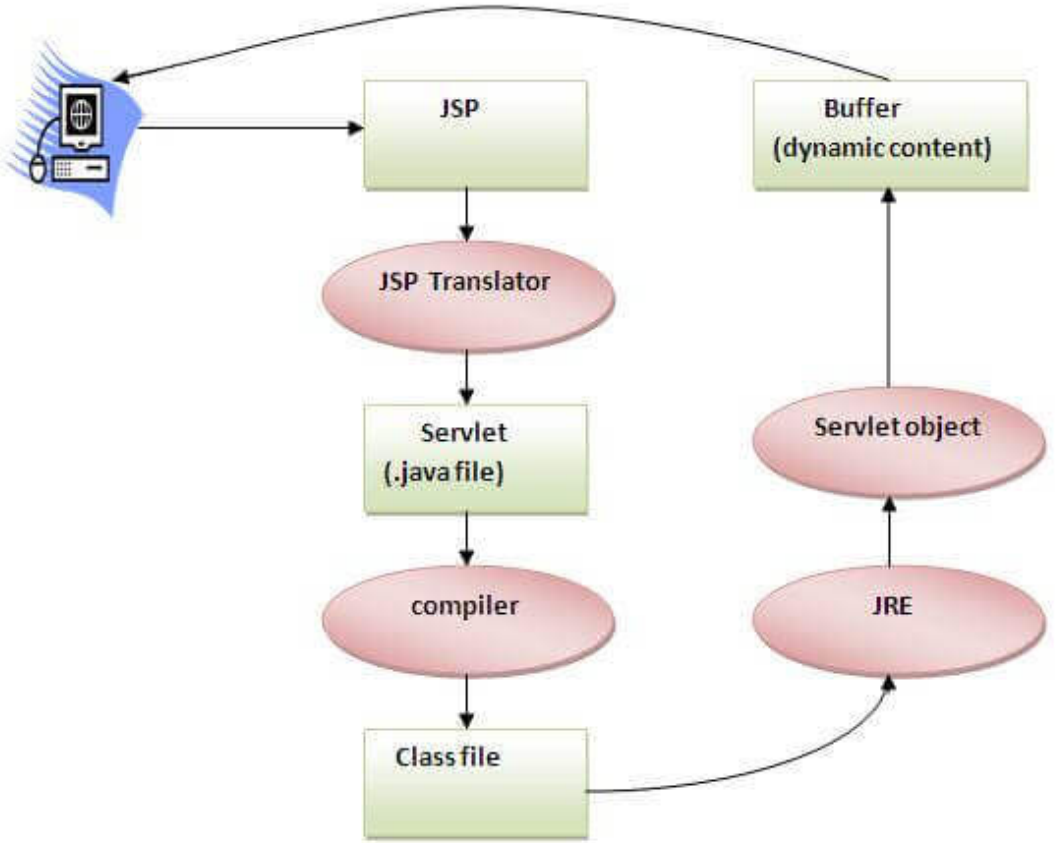
Outline

- Introduction
- JSP Scripting elements
- 9 predefined objects
- JSP Directive elements
- JSP Exception
- Action elements
- Expression language
- JSTL
  - Core Tags
  - Function Tags
  - Formatting Tags
  - XML Tags
  - SQL Tags

Introduction

- **Servlet** technology is used to create a web application (resides at server side and generates a dynamic web page).
- **JSP** technology is used to create web application just like Servlet technology. It can be thought of as 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 designing and development. It provides some additional features such as Expression Language, Custom Tags, etc.

Life Cycle of JSP



Example

- Run Apache Tomcat 10.0.4
- Create web folder: *%tomcathome%*/webapps/jsp1
- Create file index.jsp and add content:

```
<html>
  <body>
    <% out.print(2*5); %>
  </body>
</html>
```

- Save and then browse at address: http://localhost:8080/jsp1/

JSP Scripting elements

- JSP Script let tag (Scripting elements)
  `<% ... %>`
- JSP Scripting elements
  - **scriptlet tag**: execute java source code in JSP
    `<% java code %>`
  - **expression tag**: write to the output stream of the response
    `<%= statement %>`
  - **declaration tag**: declare fields and methods
    `<%! Declarations %>`

9 predefined objects

Object	Type
out	JspWriter
request	HttpServletRequest
response	HttpServletResponse
config	ServletConfig
application	ServletContext
session	HttpSession
pageContext	PageContext
page	Object
exception	Throwable

JSP Directive elements

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

```
<%@ directive attribute="value" %>
```

  - **page directive**: defines attributes that apply to an entire JSP page
  - **include directive**: include the contents of any resource it may be jsp file, html file or text file.
  - **taglib directive**: used to define a tag library that defines many tags.

Page directive

- The page directive defines attributes that apply to an entire JSP page.
  - import
  - contentType
  - extends
  - info
  - buffer
  - language
  - isELIgnored
  - isThreadSafe
  - autoFlush
  - session
  - pageEncoding
  - errorPage
  - isErrorPage

Include directive

- The include directive is used to include the contents of any resource it may be jsp file, html file or text file.
- The include directive includes the original content of the included resource at page translation time (the jsp page is translated only once so it will be better to include static resource).

```
<%@ include file="resourceName" %>
```

Taglib directive

- The JSP taglib directive is used to define a tag library that defines many tags. We use the TLD (Tag Library Descriptor) file to define the tags.

```
<%@ taglib uri="urioftheteglibrary" prefix="prefixoftaglibrary" %>
```

- We will talk more Advanced JSP!

JSP Exception

- 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:
  - By **errorPage** and **isErrorPage** attributes of page directive
    ```
    <%@ page errorPage="error.jsp" %>
    ```
  - By **<error-page>** element in web.xml file

```
<error-page>
  <exception-type>java.lang.Exception</exception-type>
  <location>/error.jsp</location>
</error-page>
```

Action elements

- The action tags are used to control the flow between pages and to use Java Bean. The Jsp action tags are given below.

JSP Action Tags	Description
jsp:forward	forwards the request and response to another resource.
jsp:include	includes another resource.
jsp:useBean	creates or locates bean object.
jsp:setProperty	sets the value of property in bean object.
jsp:getProperty	prints the value of property of the bean.
jsp:plugin	embeds another components such as applet.
jsp:param	sets the parameter value. It is used in forward and include mostly.
jsp:fallback	can be used to print the message if plugin is working. It is used in jsp:plugin.

Expression language

- The **Expression Language** (EL) simplifies the accessibility of data stored in the Java Bean component, and other objects like request, session, application etc.
- There are many implicit objects, operators and reserve words in EL.
- It is the newly added feature in JSP technology version 2.0.

```
${ expression }
```

Expression language

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.

EL – Scopes

- The 4 scopes: pageScope, requestScope, sessionScope and applicationScope.
- Example:

```
<html>
<body>
<% session.setAttribute("favorite", "swimming and reading."); %>

<p>Favorite: ${sessionScope.favorite} </p>
<p>Favorite: ${favorite} </p>
</body>
</html>
```

EL – Scopes

- We prefixed: pageScope, requestScope, sessionScope or applicationScope only and only if the attribute naming conflict.
- Example:

```
<html>
<body>
<% session.setAttribute("favorite", "swimming and reading."); %>
<% request.setAttribute("favorite", "playing games and football.");
%>
<p>Session Favorite: ${sessionScope.favorite} </p>
<p>Request Favorite: ${requestScope.favorite} </p>
<p>Favorite: ${favorite} </p>
</body>
</html>
```

EL – Scopes

- If we don’t prefixed: pageScope, requestScope, sessionScope or applicationScope, EL will check implicit values in a specific order: `pageScope, requestScope, sessionScope,` and `applicationScope, param, paramValues, header, headervalues, initParam, cookie, pageContext.`

```
<html>
<body>
<% session.setAttribute("favorite", "swimming and reading."); %>
<%request.setAttribute("favorite", "playing games and football."); %>
<p>Session Favorite: ${sessionScope.favorite} </p>
<p>Request Favorite: ${requestScope.favorite} </p>
<p>Favorite: ${favorite} </p>
</body>
</html>
```

EL – params

- Simplified

  ```
  <%=request.getParameter("…") %>
  ```

  as

  ```
  ${param["…"]}
  ```

- Example: http://localhost:8080/UsingEL/params.jsp?name=Sreyphanne&favorite=Playing%20game

```
<html>
<body>
Name: ${param["name"] }
<br/>param["favorite"]: ${param.favorite }
<br/>param.favorite: ${param.favorite }
</body>
</html>
```

EL – params

- Listing all parameters: `paramValues`
- In example below we use JSTL core <c:forEach> tag (JSTL will be discussed later):
  http://localhost:8080/UsingEL/params.jsp?favorites=Playing%20game,Eating%20steak,Shopping

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<body>
<c:forEach items="${paramValues.favorites[0]}" var="n">
<br/>- ${n }
</c:forEach>
</body>
</html>
```

JSTL can be downloaded here:

https://repo1.maven.org/maven2/org/glassfish/web/jakarta.servlet.jsp.jstl/2.0.0/jakarta.servlet.jsp.jstl-2.0.0.jar
https://repo1.maven.org/maven2/jakarta/servlet/jsp/jstl/jakarta.servlet.jsp.jstl-api/2.0.0/jakarta.servlet.jsp.jstl-api-2.0.0.jar

And copy and paste in WEB-INF/lib

EL – params

- Listing all parameters: `paramValues`
- In example below we use JSTL core <c:forEach> tag (JSTL will be discussed later):
  http://localhost:8080/UsingEL/params.jsp?favorites=Playing%20game,Eating%20steak,Shopping

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<body>
<c:forEach items="${param.favorites}" var="n">
<br/>- ${n }
</c:forEach>
</body>
</html>
```

EL – Headers

- Short form for accessing headers: ${header.name}
- Example:

```
<html>
<body>
 <h1>Request Headers</h1>
 Accept: ${header["Accept"] }
 <br/>Connection: ${header["Connection"] }
 <br/>Accept-Language: ${header["Accept-Language"] }
 <br/>Host: ${header["Host"] }
 <br/>User-Agent: ${header["User-Agent"] }
</body>
</html>
```

JSTL – JSP Standard Tag Library

- For faster develop using advantages of custom tags
- JSTL tags are categorized as:

Tag Name	Description
Core tags	The JSTL core tag provide variable support, URL management, flow control, etc. The URL for the core tag is **http://java.sun.com/jsp/jstl/core**. The prefix is **c**.
Function tags	The functions tags provide support for string manipulation and string length. The URL for the functions tags is **http://java.sun.com/jsp/jstl/functions** and prefix is **fn**.
Formatting tags	The Formatting tags provide support for message formatting, number and date formatting, etc. The URL for the Formatting tags is **http://java.sun.com/jsp/jstl/fmt** and prefix is **fmt**.
XML tags	The XML tags provide flow control, transformation, etc. The URL for the XML tags is **http://java.sun.com/jsp/jstl/xml** and prefix is **x**.
SQL tags	The JSTL SQL tags provide SQL support. The URL for the SQL tags is **http://java.sun.com/jsp/jstl/sql** and prefix is **sql**.

JSTL – Core tags

- List of all core tags are here: https://jakarta.ee/specifications/tags/2.0/tagdocs/
- To use the core tags we need to declare a top of page as:

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

- Example:

```
<%@ taglib
uri="http://java.sun.com/jsp/jstl/
core" prefix="c" %>
<html>
<body>
<c:out value="Hello world!"/>
</body></html>
```

JSTL – Core tags

- Tags list:

Tags	Description
c:out	It display the result of an expression, similar to the way <%=...%> tag work.
c:import	It Retrives relative or an absolute URL and display the contents to either a String in 'var',a Reader in 'varReader' or the page.
c:set	It sets the result of an expression under evaluation in a 'scope' variable.
c:remove	It is used for removing the specified scoped variable from a particular scope.
c:catch	It is used for Catches any Throwable exceptions that occurs in the body.
c:if	It is conditional tag used for testing the condition and display the body content only if the expression evaluates is true.
c:choose, c:when, c:otherwise	It is the simple conditional tag that includes its body content if the evaluated condition is true.
c:forEach	It is the basic iteration tag. It repeats the nested body content for fixed number of times or over collection.
c:forTokens	It iterates over tokens which is separated by the supplied delimeters.
c:param	It adds a parameter in a containing 'import' tag's URL.
c:redirect	It redirects the browser to a new URL and supports the context-relative URLs.
c:url	It creates a URL with optional query parameters.

JSTL – Core tags <c:import>

Provides all functionalities of the **<include>** action but also allows for the inclusion of absolute URLs.

Attribute	Description	Requ-ired	Default
url	URL to retrieve	Yes	None
context	/ followed by the name of a local web application	No	Current application
charEncoding	Character set	No	ISO-8859-1
var	Name of the variable to store imported text	No	Print to page
scope	Scope of the variable used to store imported text	No	Page
varReader	Name of an alternate variable to expose java.io.Reader	No	None

JSTL – Core tags <c:import>

Include content from files, stream, url, JSP files or other websites. Example:

```
<%@ taglib
  uri="http://java.sun.com/jsp/jstl/core"
  prefix="c" %>
<html>
 <head>
  <title>Tag Example</title>
 </head>
 <body>
 <c:import url="https://www.google.com/" var="data"/>
 <p>${data}</p>
</body>
</html>
```

```
<!- Hello.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>Insert title here</title>
</head><body>
  <h2>Hello everyone!</h2>
</body>
</html>
```

```
<%@ taglib
  uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html><head>
  <title>Tag Example</title>
</head><body>
 <c:import url="hello.jsp" var="data"/>
 <p>${data}</p>
</body></html>
```

JSTL – Core tags <c:url>, <c:redirect>

Make URL, Example:

```
<%@ taglib
  uri="http://java.sun.com/jsp/jstl/core"
  prefix="c" %>
<html>
 <head>
  <title>Tag Example</title>
 </head>
 <body>
  <c:url value="/hello.jsp"/>
  <c:url value="/hello.jsp">
    <c:param name="cat_name">MoMo</c:param>
  </c:url>
 </body>
</html>
```

JSTL – Core tags for condition and switch

<c:if> conditional if:

```
<%@ taglib
  uri="http://java.sun.com/jsp/jstl/core"
  prefix="c" %>
<html>
 <head>
  <title>Tag Example</title>
 </head>
 <body>
 <c:set scope="session" var="greeting" value="1"/>
 <c:if test="${greeting==0}">
   <c:redirect url="https://www.google.com"/>
 </c:if>
 <c:if test="${greeting>0}">
   <c:import url="/hello.jsp"/>
 </c:if>
</body>
</html>
```

Switch case: <c:choose>, <c:when>, <c:otherwise>

```
<%@ taglib
  uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html><head>
  <title>Tag Example</title>
</head><body>
  <c:set scope="request" var="income" value="500" />
  <c:choose>
    <c:when test="${income <= 200}">
       Income is not good.
    </c:when>
    <c:when test="${income > 450}">
        Income is fair.
    </c:when>
    <c:otherwise>
       Income is undetermined...
    </c:otherwise>
  </c:choose>
</body></html>
```

JSTL – Core tags <c:forEach>

Example:
Loop each element in collection:

```
<%@ taglib
  uri="http://java.sun.com/jsp/jstl/core"
  prefix="c" %>
<html>
 <head>
  <title>Tag Example</title>
 </head>
 <body>
 <c:forEach items="${param.favorites}" var="n">
   <br/>- ${n }
 </c:forEach>
</body>
</html>
```

Loop from 1 to 20.

```
<%@ taglib
  uri="http://java.sun.com/jsp/jstl/core"
  prefix="c" %>
<html>
 <head>
  <title>Tag Example</title>
 </head>
 <body>
 <ul>
 <c:forEach var="i" begin="1" end="20">
   <li>Item ${i}</li>
 </c:forEach>
 </ul>
 </body>
</html>
```

JSTL – Core tags <c:set>, <c:remove>

Set/Add new value into a scope: Example, add **cat_name="MoMo"** to scope session:

```
<%@ taglib
  uri="http://java.sun.com/jsp/jstl/core"
  prefix="c" %>
<html>
 <head>
  <title>Tag Example</title>
 </head>
 <body>
 <c:set scope="session" var="cat_name" value="MoMo"/>
My cat's name is: <b>${cat_name}</b>
</body>
</html>
```

Remove **cat_name** variable from session scope:

```
<%@ taglib
  uri="http://java.sun.com/jsp/jstl/core"
  prefix="c" %>
<html>
 <head>
  <title>Tag Example</title>
 </head>
 <body>
   <c:remove var="cat_name"/>
   My cat's name is: <b>${cat_name}</b>
 </body>
</html>
```

JSTL – Function tags

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

Provides a number of common string manipulation functions and others.

JSTL Functions	Description
fn:contains()	Test if an input string containing the specified substring in a program.
fn:containsIgnoreCase()	Test if an input string contains the specified substring as a case insensitive way.
fn:endsWith()	It is used to test if an input string ends with the specified suffix.
fn:escapeXml()	It escapes the characters that would be interpreted as XML markup.
fn:indexOf()	It returns an index within a string of first occurrence of a specified substring.
fn:trim()	It removes the blank spaces from both the ends of a string.
fn:startsWith()	Checks whether the given string is started with a particular string value.
fn:split()	It splits the string into an array of substrings.
fn:toLowerCase()	It converts all the characters of a string to lower case.
fn:toUpperCase()	It converts all the characters of a string to upper case.
fn:substring()	It returns the subset of a string according to the given start and end position.
fn:substringAfter()	It returns the subset of string after a specific substring.
fn:substringBefore()	It returns the subset of string before a specific substring.
fn:length()	Returns the number of chars inside a string, or the number of items in a collection.
fn:replace()	It replaces all the occurrence of a string with another string sequence.

JSTL – Formatting Tags

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

Provide support for message formatting, number and date formatting etc.

Formatting Tags	Descriptions
fmt:parseNumber	Parses the string representation of a currency, percentage or number.
fmt:timeZone	Specifies a parsing action nested in its body or the time zone for any time formatting.
fmt:formatNumber	Format the numerical value with specific format or precision.
fmt:parseDate	Parses the string representation of a time and date.
fmt:bundle	Creates the ResourceBundle objects which will be used by their tag body.
fmt:setTimeZone	Stores the time zone inside a time zone configuration variable.
fmt:setBundle	Loads the resource bundle and stores it in a bundle configuration variable or the named scoped variable.
fmt:message	Displays an internationalized message.
fmt:formatDate	Formats the time and/or date using the supplied pattern and styles.

JSTL – XML Tags

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

Used for providing a JSP-centric way of manipulating and creating XML documents

We need 2 more libraries to work with XML:

- http://xml.apache.org/xalan-j/index.html
- http://www.apache.org/dist/xerces/j/

XML Tags	Descriptions
x:out	Similar to <%= ... > tag, but for XPath expressions.
x:parse	It is used for parse the XML data specified either in the tag body or an attribute.
x:set	It is used to sets a variable to the value of an XPath expression.
x:choose	It is a conditional tag that establish a context for mutually exclusive conditional operations.
x:when	It is a subtag of that will include its body if the condition evaluated be 'true'.
x:otherwise	It is subtag of that follows tags and runs only if all the prior conditions evaluated be 'false'.
x:if	Evaluates the test XPath expression and if it is true, it will processes its body content.
x:transform	Provides the XSL(Extensible Stylesheet Language) transformation.
x:param	It is used along with the transform tag for setting the parameter in the XSLT style sheet.

JSTL – SQL Tags

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

SQL Tags	Descriptions
sql:setDataSource	It is used for creating a simple data source suitable only for prototyping.
sql:query	It is used for executing the SQL query defined in its sql attribute or the body.
sql:update	It is used for executing the SQL update defined in its sql attribute or in the tag body.
sql:param	It is used for sets the parameter in an SQL statement to the specified value.
sql:dateParam	It is used for sets the parameter in an SQL statement to a specified java.util.Date value.
sql:transaction	It is used to provide the nested database action with a common connection.

References

- https://www.javatpoint.com/steps-to-create-a-servlet-using-tomcat-server
- https://www.javatpoint.com/jsp-tutorial
- https://www.javatpoint.com/servlet-tutorial
- https://www.javatpoint.com/Servlet-interface
- https://www.javatpoint.com/jstl-function-tags
- https://jakarta.ee/specifications/tags/2.0/tagdocs/