UNIT-VII: JSP APPLICATION DEVELOPMENT

7.1 Generating dynamic content :-

1. Jsp generate content that differ based on user input,time of day,state of an external system or any run time conditions.
2. Jsp provide tools for generating dynamic content.
3. Jsp web application development undergoes three phases:
   Development phase
   Deployment phase
   Running phase.

Example program 7.1 for understanding phases

Step1: Create a jsp page

Myjsp/Datetime.jsp

```
<%@ page import="java.util.Calendar"%>
<%@ page import="java.util.Date"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<meta http-equiv="refresh" content="60"/>
<title>Date and Time demonstration</title>
</head>
<body>
<br/><hr/> <h3>Today's date and time is :
<hr/>
<br/>
<jsp:useBean id="date" class="java.util.Date"/>
<c:out value="${date}"/>
</body>
</html>
```

Step2. Create web.xml

```
 <?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd" version="2.4">
</web-app>
```

Step3.copy Myjsp into webapps directory.

Step4. Create a WAR file .

```
C:> jar -cvf Myjsp.war .
```

Step5. Start tomcat and type in browser http://localhost:8080/MyJsp/Datetime.jsp

7.1.1 Using Directive elements:-

Directive elements specify attributes

- type of content produced
- page buffering requirements
- Declaration of other resources
- How to handle run time errors

Example :<%@ page contentType="text/html" %>
<%@ page import="java.util.*,com.ora.jsp.util.*" %>

Using Jsp comments

\It describes what's going on page or to temporarily comment out pieces of page to test different alternatives.

Example :

<%------Calculate sum of 1+2+3 dynamically-------%>

Using Jsp action elements:-

1. It is executed during request processing phase.
2. Access parameters sent with request.
3. Create objects for use in jsp scriplets.
4. It represents dynamic action.
5. It syntax is similar to HTML like elements

General form :-

```
<prefix:action_name attr1="value1" attr2="value2">
 action_body
</prefix:action_name>
```

6. Action s can be grouped into three categories:(i) standard (ii) custom (iii)Jsp standard tag library
7. Standard action: It is defined in JSP specification.
8. Custom action:It is action element which is created for customized tags.
9. JSTL:It is set of libraries that group related actions.
   (i) core
   (ii) XML processing
   (iii) Internationalization (118N) and formatting
   (iv) Relational database Access (SQL)
   (v) Functions

Example program 7.1.3

Easy.jsp

```
<%@ page contentType="text/html" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
<title>Jsp is easy</title>
</head>
<body bgcolor="white">
<h1>Jsp is as easy as -----</h1>
<%----Calcualte the sum of 1+2+3 dynamically ----%>
1+2+3=<c:out value="${1+2+3}"/>
```

```
</body>
</html>
```

Bold underline code represent action element .

7.1.4 Jsp Expression language:-

1. Jsp expressions are java expressions inside “<%=” and “%>” characters.
2. It can be placed in any Jsp page.
3. Access any methods ,variables defined in jsp declarations
4. It can access jsp implicit objects.
5. Create and manipulate java objects.

Example <%=1+1%>
<%=new java.util.Date()%>

Using implicit jsp scripting objects:

Grede fined variables with reference to access request ,application data.

Implicit scripting objects:-

1.	pageContext object	:	provide methods for accessing references to all other objects and Attributes for holding data shared between components in same page.
2.	request object	:	provide methods for accessing all the information that’s available about current request, such as request parameters ,attributes, headers and cookies.
3.	.response object	:	provide methods for setting headers ,status code , adding cookies session tracking.
4.	session object	:	provide access to session data as well as information about session.
5.	Application object	:	It holds references to other objects that more than one user may require access to, such as a database connection pool shared by all application users.
6.	Out object	:	It uses print(),println( ) methods to add text to response message body.
7.	Exception object	:	Available only in error page and contain information about runtime errors.

Using scripting elements:-

1. Scripting elements allow to embed java code in jsp page.
2. Executed during request processing time.
3. Manipulates objects.
4. Perform calculation on run time values,object.
5. Declare methods or variables.

6. Evaluate complex expression,involving java objects.
7. Finally control execution of jsp page

7.3.1 Displaying values:-

Out implicit object is used for displaying values as response to client.

Example program 7.3.1

Fragment.jsp

```
<%@ page language=”java” contentType=”text/html” %>
<html>
<head>
<title>Browser check</title>
</head>
<body bgcolor=”white”>
<% String userAgent=request.getHeader(“User-Agent”);
if(userAgent.indexOf(“MSIE”)!=-1){
%>
you're using Internet Explorer.
<%} else if (userAgent.indexOf(“Mozilla”)!=-1){%>
you're probably using netscape.
<%}else{%>
you're using a browser  I don't know about.
<%}%>
</body>
</html>
```

Condition processing using scriplets:-

Control structures with in scriplets process data conditionally.

Example program 7.3.2

counter.jsp

```
<%@ page language=”java” contentType=”text/html” %>
<%! int globalCounter =0 ; %>
<html>
<head>
<title>A page with a counter </title>
</head>
<body bgcolor=”white”>
This page has been visited”<%=++globalCounter %>times.
<p>
<% int localcounter=0; %>
This counter never increases its value:<%=++localCounter %>
</body>
</html>
```

Declaring variables and methods :-

1. Declarations are enclosed <%! --- %> characters.
2. Declaration code for static ,instance variables , methods.
3. Declaration code   placed outside _jspService() method

4. Declarations can be specified in any part of web page.

Example program 7.4

color.jsp

```
<%@ page language="java" contentType="text/html" %>
<%!
String randomColor( ){
java.util.Random r=new java.util.Random();
int red=(int)(random.nextFloat( ) * 255);
int green=(int)(random.nextFloat( ) * 255);
int blue=(int)(random.nextFloat( ) * 255);
return "#"+Integer.toString(red,16)+Integer.toString(green,16)+Integer.toString(blue,16);
} %>
<html>
<head>
<title>Random color</title>
</head>
<body bgcolor="white">
<h1>Random color</h1>
<table bgcolor="<%= randomColor() %>" >
<tr><td width="100" height="100">&nbsp;</td></tr></table>
</body>
</html>
```

Error handling and debugging:-

Deals syntax errors .

7.5.1 Element syntax errors.

- Improper termination of directive element.
- Improper use of action element.
- Mistyped attribute name
- Missing end quote in attribute value.
- Expression language syntax errors.

Debugging a jsp file:-

Debugging is a process get rid of errors.
Hints for debugging jsp 's

- use long variable names ,it makes searching in file easier ,reduce scope of mistakes.
- Validate input data.
- Look at java source files .
- Look in the log files after execution.
- Write output statements to track the value of variable
- Finally use debug tool to debug jsp application.
  Example program 7.5.2

Sharing data between jsp pages ,requests and users:-

Web application consist more than one web page to access information and server side resources.

7.6.1 passing control and data between pages:-

7.6.1.1 passing control between pages:-

Jsp provides wonderful mechanism to transfer control from one page to other using <jsp:forward> action

Syntax:

```
<jsp:forward  page=”relative URL/absolute URL”/>
<jsp:param name=”name1” value=”value1”>
</jsp:forward>
```

1. Execution of <jsp:forward> action terminates processing of current jsp page,so that processing of target page can be stored.
2. On execution of <jsp:forward> from jsp page control never come back again.
3. Buffer cleared before forwarding occurs.
4. Full access to request and response objects.
5. Can be added by using <jsp:param> action in request scope.

Example program 7.6.1.1

Conterpage.jsp

```
<% taglib prefix=”c” uri=”http://java.sun.com/jsp/core”%>
<html.
<head> <title>Counter page</title>
</head>
<body bgcolor=”white'>
<c:set var=”sessionCounter” scope=”session” value=”${sessionCounter + 1 }”/>
<c:set var=”applCounter” scope=”application” value=”${applCounter + 1 }”/>
<h1>Counter page</h1>
this page has been visited <b> ${sesssionCounter}</b>times
within the current session,and <b.${applCounter}</b> times
by all users since the application was startted.,/body>
</html>
```

7.6.1.2 passing data between jsp pages:-

In context of jsp/servlet data can be accessed in four different scopes page ,request,session and application.
scope defines life time ,availability of an object or data.

Page scope :objects can be accessed only with in current jsp page,its life time is of _jspService().
Request scope:objects can be accessed by any page involved in the processing of request.
Session scope:objects can be accessed for the entire browsing period by the pages originating form same browser.
Application:objects can be accessed by any user of the web application and persist for the life time of server.

Example program 7.6.1.2

product.jsp

```
<%@ page language=”java” contentType=”text/html” %>
<%@ taglib prefix=”c” uri=”http://java.sun.com/jsp/jstl/core” %>
<%@ taglib prefix=”c” uri=”http://java.sun.com/jsp/jstl/functions” %>
```

```
<html>
<head>
<title>Product Description</title>
</head>
<body bgcolor="white">
<jsp:useBean id="catalog" scope="application" class="com.ora.jsp.beans.shopping.CatalogBean"
/>
<c:set var='product" value="${catalog.productsbyId[param.id]}"/>
<h1><${fn:escapeXml(product.name)}</h1>
${fn:escapeXml(product.descr)}
<p>
<c:url var="addtocartURL" value="addtocart.jsp">
<c:param name="id" value="{product.id}"/>
</c:url>
<a href="${addtocartURL}">
Add this book to the shooping cart </a>
</body>
</html>
```

addtocart.jsp

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<jsp:usebean id="catalog" scope="application" class="com.ora.jsp.beans.shooping.CatalogBean"/>
<c:set var="product" value="${catalog.productsById[param.id]}"/>
<jsp:usebean id="cart" scope="session" class ="com.ora.jsp.beans.shooping.CartBean" />
<c:set target="${cart}" property="product" value="${product}"/>
<c:redirect url="catalog.jsp"/>
```

Sharing session and application data:-

- Request scope makes data available to multiple pages processing same request. But in many cases,
  Data must be shared over multiple requests .
- Session scope makes information available only to requests form same user.
  application scope makes information available to all requests form any user.
- Session tracking is performed in jsp by three methods:
  (i) Using cookie
  (ii) Using embedded as hidden fields in an HTML form
  (iii) Encoded in URL of response body, typically as links to other application pages.

Example programs 7.7

Conter.jsp

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

```
<body bgcolor =”white”>
<c:set var=”sessionCounter” scope=”session” value=”${sessionCounter” scope=”session” value=”$
{sessioncounter + 1}”/>
<h1>counter page</h1>
This page has been visited <b>${sessionCounter}</b> times
with in the current session.
<p> Click here to load the page through a
<a href=”counter.jsp”>regular link</a>
<p>
click here to load the page through an
<a href=”<C:url value=”counter2.jsp”?>”>encoded link</a>
</body>
</html>
```

1. Memory usage considerations:-

Application and session scope objects consume memory at server side.

1. Memory requirements for session scoped objects should consider :
2. Size of each session scoped objects.
3. Number of concurrent sessions
4. How long session spans