- It cannot be used for session tracking if large set of details need to be stored for a user session.

Review Questions

Q. 1 Write a short note on CGI. Describe its advantages and disadvantages.

Q. 2 What are servlets ? Explain the request/response paradigm of servlets.

Q. 3 Write a short note on Web Application Model.

Q. 4 Write a short note on Servlet API.

Q. 5 Write a short note on GenericServlet class.

❑❑❑

UNIT - II

CHAPTER 4

Java Server Pages (JSP)

Syllabus Topics

Introduction, JSP LifeCycle, JSP Implicit Objects and Scopes, JSP Directives, JSP Scripting Elements, JSP Actions : Standard actions and customized actions.

Syllabus Topic : Introduction to JSP

4.1 Introduction to JSP (Java Server Pages)

Q. What is JSP ?
Q. What is a role of JSP? Explain it in detail.
Q. Explain the processing of JSP pages in detail.

- JSP (Java Server Pages) technology can be used to create dynamic web application. Its methodologies are more efficient then servlets.
- Basically it separates the presentation logic from the business logic and thereby it provides more work independency to the designer and developer of the web application.
- Every JSP page includes HTML tags and JSP tags.
- We can separate designing and development very easily in JSP, that's why jsp pages are easier to maintain than servlet.
- JSP also provides some additional features such as Expression Language, Custom Tag etc.

JSP Processing

Process of creating the web page using JSP in a web server :

(i) Browser send HTTP request to the web server for a JSP page, and redirected to JSP engine.

(ii) The JSP engine loads the JSP page and converts it into a servlet if it is not created yet.

(iii) The JSP engine compiles the servlet and creates the class file and forward the request to servlet engine.

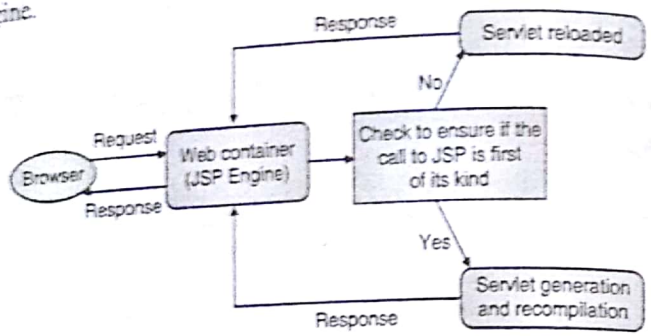

Fig. 4.1.1 : JSP Processing

(iv) Servlet engine load and execute the Servlet class.

(v) The servlet accept the input from the client request, process the result and send the output back to the client.

4.1.1 Advantages and Disadvantages of JSP

Advantages

Q. Discuss the advantages of JSP over Servlet.

(i) **JSP Tags :** Various JSP tags can be used to include the java code embedded with HTML tags.

(ii) **Output of a JSP page :** Output of a JSP page is in standard HTML format. Thus client require a compatible browser to display the JSP page not a JVM.

(iii) **Embedded HTML tags :** The view of a JSP page is embedded in HTML tags and CSS that allows easier web application development.

- Web graphics designer can design the display tags of JSP page. It can use graphics designer tools like FrontPage, Dreamweaver etc.
- The dynamic JSP coding can be done by Java developers.
- JDeveloper tools can be used to test the JSP code.

(iv) **Easier than Servlet :** JSP coding are easier than developing corresponding servlet program.

(v) **JSP containers :** Allows to use and access inbuilt objects and action elements in JSP pages.

(vi) **HTTP based :** JSP program uses the HTTP request/response communication model.

Disadvantages

Q. Discuss the disadvantages of JSP over Servlet.

(i) **Difficult to debug :** During the compilation of JSP pages errors are shown in the corresponding translated servlet program. Thus it is difficult to trace the errors in JSP pages.

(ii) **High disk space :** Storing the JSP pages in web server needs more memory space than its equivalent servlet.

(iii) **Compile Time :** Takes more compilation time because of translation process into servlet.

(iv) **User interface is limited to HTML tags :** The HTML tags and client side script like JavaScript are used to create the interface design of a JSP page. Thus user interface design does not support automatic validation.

4.1.2 Difference between JSP and Servlets

Q. Discuss the difference between JSP and Servlets.

Sr. No.	Parameter	JSP	Servlets
1.	Program	JSP is a scripting language that can generate dynamic content. It uses java programming as scripting with HTML tags with other client side scripting elements like JavaScript.	Servlets are pure Java class file which is used to develop web application. Servlet file need to be compiled and convert into its corresponding class file and loaded into the web server.
2.	Time-to execute	JSP programs run slower when compared to Servlet. During the compilation process JSP pages are converted into Java Servlets, then compiled and converted into executable class file.	Servlets run faster when compared to JSP pages. Servlets can be directly compiled to create an executable class file. There is no conversion process involved.
3.	Codding style	HTML tags and client side scripting elements can be used to design the user interface. It provides higher level of modularity in the development of web application. JSP elements can be used to implement the business logic of the application.	In servlet HTML tags cannot be separated from the java coding. Interface design and business logic of the application need to written in the java class file. Developing web application using servlet require good java programming skills than of using JSP.

Sr. No.	Parameter	JSP	Servlets
4.	MVC	In Model-View-Controller design pattern JSP pages are used to create the view part of the application.	In Model-View-Controller design pattern servlet are used to create the controller part of the application.
5.	Preferred	JSP techniques are used when the data processing required for an application is relatively less.	When the application requires more data processing and manipulation, servlet could be used.
6.	Custom Tags	JSP allows using custom tags. For Example - Java beans can be used directly with JSP pages using custom tags and action elements which can directly call Java beans.	Servlets does not support the facility of using the custom tags.

Syllabus Topic : JSP LifeCycle

4.2 Lifecycle of JSP

Q. Explain the life cycle of JSP.

- JSP life cycle defines a set of activities performed when a web application deploys a jsp page within the web container for the client to access. JSP life cycle matches with the life cycle of servlet with the additional process of converting a jsp page into its corresponding servlet program.
- The four phases of JSP life cycle are :

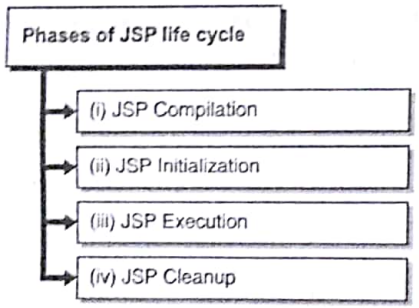

Fig. C4.1 : Phases of JSP life cycle

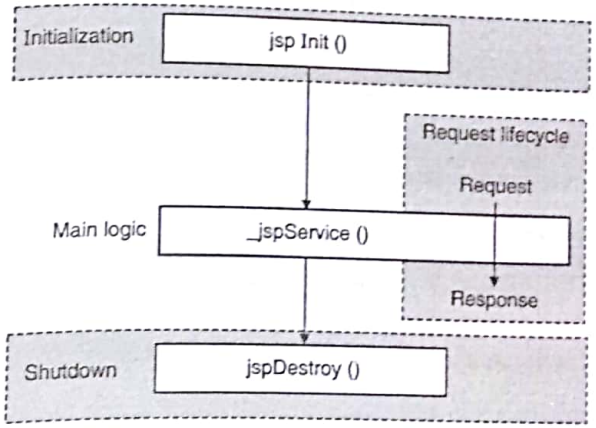

Fig. 4.2.1 : JSP life cycle

(i) JSP Compilation

- JSP engine checks whether it require to compile the jsp page or not, when a client tries to access the page. It compiles the page; if the page is never compiled before or it is modified after the last compilation.
- The compilation process performs the following tasks :
  - Parsing the JSP page to identify the jsp elements that are used.
  - Convert the JSP page into the corresponding servlet program.
  - Compile the auto created servlet program to generate the executable class file.

(ii) JSP Initialization

- When a container loads a JSP page, jspInit() method is invoked for performing the initialization needed before servicing client requests.
- jspInit() method can be overridden to perform resource initialization, that need to be done only once within the entire life cycle.

```
public void jspInit()
{
    // Initialization code
}
```

- jspInit() method can be used to initialize the database connection, open a file, check and gain access to the available resources etc.

(iii) JSP Execution

- JSP execution life cycle method is used to process the client request and send back the response.

- _jspService() method can be used to process the client request and send the response back.
- _jspService() method has two parameters :
  1. HttpServletRequest and
  2. HttpServletResponse

☞ **Format of _jspService() method**

```
void _jspService(HttpServletRequest request,  HttpServletResponse response)
{
    // Service handling code
}
```

- It is called once for each client request.
- It can process response to all the seven HTTP methods ie. GET, POST, DELETE etc.

➜ **(iv) JSP Cleanup**

- JSP cleanup removes the jsp page from the container.
- The jspDestroy() method of JSP is same as servlets destroy() method.
- jspDestroy() can be overridden to perform cleanups like releasing database connections or closing the files etc.

☞ **jspDestroy( ) method format**

```
public void jspDestroy()
{
    // cleanup code required
}
```

4.2.1 JSP Architecture

Q. Explain the JSP architecture/model in detail.

- JSP engine or container is responsible for managing the architecture of a JSP page. It is responsible for making the JSP pages accessible for the client.
- Runtime environment and services required for a JSP page for its execution is provided by the JSP container and Web server.
- It process various elements used within the JSP pages.
- The position of JSP container and JSP files in a Web Application.

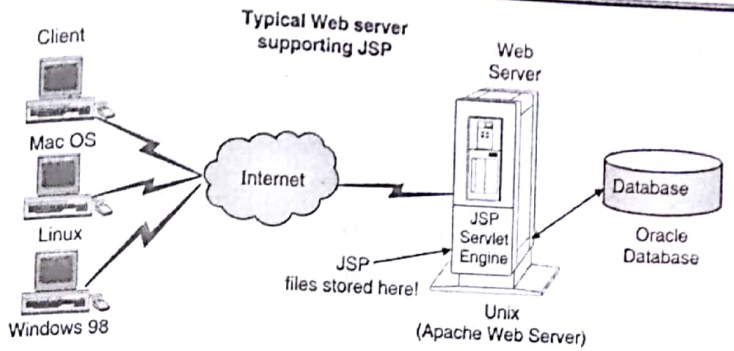

Fig. 4.2.2 : JSP Engine and JSP pages

☞ **Types of JSP Architecture**

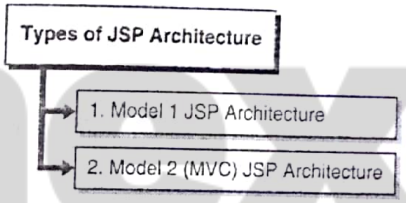

Fig. C4.2 : Types of JSP Architecture

➜ **1. Model 1 JSP Architecture**

Q. Discuss Model 1 JSP architecture.
Q. Explain the advantages and disadvantages of Model 1 architecture.

Scenario of Model-1 architecture processing is

(i) Client can send the request to access the JSP page.

(ii) JSP page can access the Java Bean that implements the business logic of the web application.

(iii) Java Bean can be designed to connect the database and fetch the required data from the database or save data permanently.

(iv) JSP generates the response and sent back to the client browser.

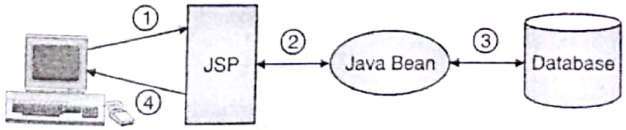

Fig. 4.2.3 : Model 1 JSP Architecture

Advantages of Model 1 JSP Architecture

(i) Easy to develop web application where JSP page provides the facility of view and controller.

(ii) Less development time is required as the business logic can be designed separately using the bean component.

Disadvantages of Model 1 JSP Architecture

(i) **Navigation control is decentralized :** Every page should consist of the logic to determine the next page. If a JSP page name is modified; it needs to be modified in all the pages which it refers to. Hence it leads to the maintenance problem.

(ii) **Time consuming :** Developing custom tags consume more time in JSP than to use scriptlet tag.

(iii) **No extendibility :** Model 1 JSP architecture is suitable for small applications not for large applications.

2. Model 2 (MVC) JSP Architecture

Q. Discuss Model 2 JSP architecture.

Q. Explain the advantages and disadvantages of Model 2 (MVC) architecture.

Model 2 is based on the MVC (Model View Controller) design pattern. The MVC design pattern consists of three components as model, view and controller.

(i) **Model :** The model is used to handle the data and business logic of the application. Java bean components are used to design the model component.

(ii) **View :** The view represent the format of data display as user interface or presentation. JSP pages with HTML tags are used to design the view element.

(iii) **Controller :** The controller form as an interface between view and model. It accepts the user request and communicates between the model and view elements. Servlet programs are generally used to design the controller element.

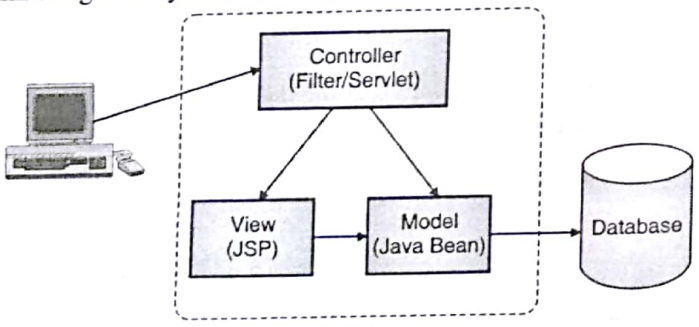

Fig. 4.2.4 : Model 2 (MVC) JSP Architecture

Advantages of Model 2 (MVC) JSP Architecture

(i) **Navigation control is centralized :** In MVC design pattern controller holds the logic used for the entire page navigation. It also acts as an interface for model and view elements.

(ii) **Easier to maintain, test and modify because of the separation of three elements**

Disadvantages of Model 2 (MVC) JSP Architecture

(i) When compared to Model 1 Controller is an added element need to be included within the Model 2.

(ii) Changes made in controller require the compilation of the classes and redeployment of the application.

Syllabus Topic : JSP Implicit Objects and Scopes

4.3 Implicit Objects and Scopes

4.3.1 Implicit Objects in JSP

Q. Enlist the implicit objects of JSP. Explain any 4 of them in detail.

Q. Discuss the implicit objects of JSP along with their scope.

Q. Explain the following : (1) Request (2) pageContext (3) Exception (4) Response (5) Page

- JSP Implicit Objects are predefined Java objects that JSP Container makes available to each JSP page. During the application development, programmer can use the implicit objects directly within the JSP page without explicit declaration.
- JSP Implicit Objects are also known as pre-defined variables of JSP.
- JSP supports NINE Implicit Objects as :

Object	Description	Scope
Request	– It is the HttpServletRequest object that can be used to fetch the information from the client. – Each time a client requests a page, JSP engine creates a new request object to access the client data. – Request object provides methods to retrieve the HTTP information like form data, cookies, HTTP methods etc.	Request
Response	– It is the HttpServletResponse object that can be used to send the response back to the client.	Response

Object	Description	Scope
	– It can be used to create new HTTP headers. – Allows to add new cookies, HTTP status codes or date stamps etc.	
Out	– It is the PrintWriter object used to send output data to the client. – Method include to write boolean, char, int, double, object, String etc. into the client browser as : 1. **out.print(dataType dt) :** Display a specific data type value to the client browser. 2. **out.println(dataType dt) :** Display a specific data type value to the client browser and terminate the line with new line character. 3. **out.flush() :** Used to flush the stream output to the client.	Page
Session	– It is the HttpSession object of the associated client request. – The session object can be used to handle the session tracking among the different client requests.	Session
Application	– It is the ServletContext object associated with the application. – Application object represent the JSP page through its entire lifecycle. – It is created during the initialization phase of the JSP page. It will be removed by jspDestroy() method.	Application
Config	– It is the ServletConfig object associated with the page. – It can be used to access the JSP engine initialization parameters like context path, file locations etc. – It can also get the information about the servlet class file generated by the JSP page. – **For example :** config.getServletName() – returns the servlet name	Page
pageContext	– It is used for server specific features like higher performance JspWriters. – It stores references to the request and response objects for each request. – The application, config, session and out objects are created from the attributes of pageContext object.	Page

Object	Description	Scope
	– It also stores the information about the directives used by the JSP page, like buffering information, error PageURL and page scope. – The PageContext object defines four scope value as PAGE_SCOPE, REQUEST_SCOPE, SESSION_SCOPE and APPLICATION_SCOPE.	
Page	– Page object represent "this" keyword for the current servlet object. – It can be used to call all the methods defined by the translated servlet class. – It acts as an actual reference to the instance of the page.	Page
Exception	– The Exception object can be used to access the exception data by the JSP page. – It contains all the exception thrown from the previous page. – It can be used to display the appropriate responses for different error condition.	Page

Syllabus Topic : JSP Directives

4.4 JSP Directives

Q. What is directive? Explain.

Q. Explain with an example addition of java bean to a JSP page.

- JSP directives are used to provide various instructions to the JSP container, to handle the processing of a JSP page.
- JSP directives affect the creation of servlet class, during the translation of JSP page to a servlet.

Format of JSP Directives

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

- Multiple attributes of a directive can be separated by commas.
- The blank space between @ symbol and the directive name and between the last attribute and the closing %> are optional.

Types of directive tag

There are three types of directive tag

Directive	Description
<%@ page ... %>	Used for page-dependent attributes, like scripting language, error page, and buffering requirements.
<%@ include ... %>	Used to include a file during the translation phase.
<%@ taglib ... %>	Used to declare a tag library, containing custom actions used in the JSP page.

4.4.1 page Directive

Q. What is page directives? Explain page directives and its attributes.

List of attributes of page directive are :

Page directive Attribute	Purpose
Buffer	Specify output stream buffering model. <%@ page buffer="16kb" %>
autoFlush	Used to control the servlet output buffer.
contentType	Specify the character encoding format. <%@ page contentType=application/msword %>
errorPage	Used to mention the URL of another JSP page to be displayed during the occurrence of runtime exceptions. <%@ page errorPage="myerrorpage.jsp" %>
isErrorPage	Used to specify that the current page is used as an errorPage or not. <%@ page isErrorPage="true" %>
Extends	Used to mention the name of a superclass that the generated servlet must extend. It is rarely used.
Import	Used to specify a list of packages or classes to be included with in the JSP page. <%@ page import="java.util.Date" %>
Info	Used to specify a string information details. Info value can be accessed with the servlet's getServletInfo() method. <%@ page info="composed by Shreeji" %>

Page directive Attribute	Purpose
isThreadSafe	If we want to control multithreaded behavior of JSP page, we can use isThreadSafe attribute of page directive. By default, the value is **true**. If we make it **false**, the web container will serialize the multiple requests, i.e. it will wait until the JSP finishes responding to a request before passing another request to it. <%@ page is ThreadSafe="false" %>
Language	Indicate the programming language used in the JSP page. The default value is "java".
Session	Used to specify that the JSP page handles the HTTP sessions or not.
isELIgnored	Used to specify that EL (Expression Language) within a JSP page will be considered or ignored. <%@ page isELIgnored="true" %>//Now EL will be ignored
isScriptingEnabled	Used to specify that scripting elements are allowed to use within a JSP page.

Example : To import java.util package to access the Date.

```
<%@ page import="java.util.*" %>
<HTML>
<BODY>
<%
  out.println( "Evaluating date now" );
  Date date = new Date();
%>
Hello! The time is now <%= date %>
</BODY>
</HTML>
```

The page directive can contain the list of imported packages, separate the package names by commas.

For example

```
<%@ page import="java.util.*,java.sql.*" %>
```

4.4.2 Include Directive

Q. Explain <jsp:include>action in detail.

Include directive can be used to embed one file into the current JSP page during the translation phase.

☞ **Format of include directive**

```
<%@ include file="relative url" >
```

- The filename within the include directive use the relative URL name.
- XML equivalent of the include directive is

```
<jsp:directive.include file="relative url" />
```

☞ **Example :** Includes the content of hello.jsp into a new jsp page.

```
<HTML>
<BODY>

  Content of the current page

  Content of hello.jsp page<BR>
<%@ include file="hello.jsp" %>

  Content of the current page
</BODY>
</HTML>
```

4.4.3 taglib Directive

- The JavaServer Pages allow defining custom JSP tags that look like HTML or XML tags. A tag library is a group of user defined tags that implement the custom behavior.
- The taglib directive can be used to declare a set of custom tags that jsp uses and specify the location of the library

☞ **Format of taglib directive**

```
<%@ taglib uri="uri" prefix="prefixOfTag" >
```

- uri attribute value specify the location detail and the prefix attribute specify a container about the custom actions.
- XML equivalent of the taglib directive is :

```
<jsp:directive.taglib uri="uri" prefix="prefixOfTag" />
```


4.4.3(A) <jsp:forward> Action

- JSP forward action tag can be used to forward a request from the JSP page to another resource like a JSP, Servlet or static html page.
- Request can be forwarded with or without parameter.

☞ **Format**

(1) Forwarding along with parameters

```
<jsp:forward page="Relative_URL_of_Page ">
<jsp:param ... />
<jsp:param ... />
<jsp:param ... />
  ...
<jsp:param ... />
</jsp:forward>
```

(2) Forwarding without parameters

```
<jsp:forward page="Relative_URL_of_Page" />
```

<jsp:param> element is used to pass the value of parameters while forwarding to another resource. Format is:

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

☞ **Attribute of forward action**

Attribute	Description
Page	Relative URL of another resource such as a static page, another JSP page, or a Java Servlet.

☞ **Rules and Semantics**

- The JSP forward action tag effectively terminates the execution of the current JSP page and forward to the new resource. Forwarding to the new page is based on dynamic conditions.

  For example

  1. Forward to an error page if exception occurs.
  2. Forward to a login page if user login fails.

- The forwarding occurs from the server side without notifying the client. Thus the URL in browser's address bar does not change.
- The container throws HTTP 404 error, if the forwarded page could not be found.

☞ **Example :** Consider two JSP files (a) date.jsp and (b) main.jsp as

date.jsp

```
<p>
Today's date: <%= (new java.util.Date()).toString()%>
</p>
```

main.jsp

```
<html>
<head>
<title>Forward Action Example</title>
</head>
<body>
        Forward action Example
        <jsp:forward page="date.jsp" />
</body>
</html>
```

- While accessing main.jsp it display result as

Output

```
Today's date: 27-11-2017 12:30:40
```

- It discarded the content of main page and display only the content from the forwarded page.

4.4.3(B) <jsp:include> Action

- <jsp:include> action tag can be used to insert set of file content into the existing jsp page.

☞ **Syntax**

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

- <jsp:include>tag inserts the file at the time the page is requested, not during the translation time like include directive.

☞ **Attributes of include action**

Attribute	Description
Page	Relative URL of the page to be included.
Flush	boolean attribute determines the included resource has its buffer flushed before it is included.

☞ **Example :** Consider two JSP pages (a)date.jpp and (b) main.jsp as :

date.jsp

```
<p>
Today's date: <%= (new java.util.Date()).toLocaleString()%>
</p>
```

main.jsp

```
<html>
<head>
<title>include Action Example</title>
</head>
<body>
        include action Example
<jsp:include page="date.jsp" flush="true" />
</body>
</html>
```

While accessing main.jsp it display result as :

Output

```
include action Example
Today's date: 27-10-2017 12:30:40
```

Syllabus Topic : JSP Scripting Elements

4.5 JSP Scripting Elements

4.5.1 JSP Scriptlets

Q. What are the two ways of writing scriptlets? Explain with suitable example.

- JSP Scriptlets are used to include a block of Java code into the JSP page.
- Java code can be included with <% and %> characters that becomes a block of java code known as a scriptlet.
- Scriptlet codes are executed when the JSP page is invoked.
- It cannot create an HTML response by itself. Implicit out object can be used inside a scriptlet to display a response back to the client.
- Implicit request object can be used inside the scriptlet to fetch the information sent by the client.
- Implicit response object can be used to send the response back to the client.

☞ **Example**

```
<HTML>
<BODY>
<%
  // This is a scriptlet.
  out.println( "Evaluating date" );
  java.util.Date date = new java.util.Date();
%>
Hello! The time is now <%= date %>
<%
  out.println( "<BR>Your machine's address is " );
  out.println( request.getRemoteHost());
%>
</BODY>
</HTML>
```

☞ **Mixing Scriptlets and HTML**

- Scriptlet coding and html tags can be mixed by repeating the scriptlet tags
- One scriptlet can be closed to include the normal HTML tags. Once the usage of HTML tags are over java code can be continued in another scriptlet.
- Control statements like while loop, for loop or if-else statement will control the HTML tags mixed within the scriptlet too.
- **For example :** If HTML tags are kept within a loop-statement, tags will be repeated for each loop iteration.

☞ **Example :** Code to generate a table containing the numbers from 1 to N.

```
<TABLE BORDER=2>
<%
  for ( int i = 0; i < n; i++ )
  {
%>
    <TR>
    <TD>Number</TD>
    <TD><%= i+1 %></TD>
    </TR>
```



```
<%
  }
%>
</TABLE>
```

4.5.2 JSP Expressions

- JSP expression tag can be used to embed the value of some expression directly into the http response of the JSP page.
- Expression should be written with the tag <% = %>.
- It is a single line statement that should not end with semicolon.

☞ **Example :** Code to display the time using expression tag

```
<HTML>
<BODY>
        Hello! The time is now <%= new java.util.Date() %>
</BODY>
</HTML>
```

4.5.3 JSP Comments

Q. Explain how to write comments in JSP.

JSP comments are used to include a set of statements that JSP container should ignore during the compilation.

☞ **Syntax of JSP comments**

```
<%-- This is JSP comment --%>
```

- It can be used to include coding standards and details that are useful during the maintenance of the application.
- A JSP comment can be used to hide a JSP page section during the development and debugging stage to trace the error and logic of the program.

☞ **Example for JSP Comments**

```
<html>
<head><title>A Comment Test</title></head>
<body>
<h2>A Test of Comments</h2>
<%-- This comment will not be visible in the page source --%>
</body>
</html>
```

4.5.4 JSP Declarations

Q. Discuss how to make declaration in JSP.

- JSP declarations can be used to declare variables or class definitions into the JSP page.

☞ **Syntax of JSP declaration**

```
<%!
    Lines of declarations and class definitions
%>
```

- Variables declared inside the declaration cannot be initialized. It can assign the value only at the successor scriptlet tag.
- Class definition can be included within the declaration tag without the creation of an object.
- Function definition can be included without function call.
- Creation of an object or calling a function can be included within the successor scriptlet tag.

☞ **Example :** JSP code to accept a number from the HTML form, and find its factorial using a recursive function.

fact.html

```
<html>
<body>
<form name=frm method=post action=fact.jsp >
<center>
<br> No <input type=text name=txtno>
<br><input type=submit value=submit>
<input type=reset value=clear>
</center>
</form>
</body>
</html>
```

fact.jsp

```
<html>
<body>
<%!
int fact(int n)
  {
      if (n<=0)
          return 1;
      else
          return n*fact(n-1);
  }
%>

<%
  String st=request.getParameter("txtno");
  int n=Integer.parseInt(st);
  int val=fact(n);
%>
<%="n = "+n%>
<br>
<%="n! = "+val%>
</body>
<%-- This program includes jsp scriptlets, expressions, declarations and comments --%>
</html>
```


Syllabus Topic : JSP Actions : Standard Actions and Customized Actions

4.6 JSP Actions : Standard Actions and Customized Actions

Q. What do you mean by JSP standard actions and customized actions ?

- Basically there are so many JSP action tags or elements. Each one is used to perform some specific tasks. The action tags are used to control the flow between pages and it is also used to use Java Bean.
- There are **two** forms of action elements
  1. Standard 2. Custom.
- Standard actions and Java language-based scripting has same usage. Other standard actions also help us to achieve most of the things that we achieve with scripting elements.
- Difference between standard and custom actions are that we can rely on any J2EE-compliant JSP container to provide support for all the standard actions defined in the JSP specification. Custom actions will have to be imported manually.

☞ **Syntax**

- The basic general syntax of standard and custom actions is :

```
<prefix:tagname firstAttribute="value" secondAttribute="value" thirdAttribute="value... >
  ...
  ...
</prefix:tagname>
```

- Each **standard action** element consists of a start tag, <prefix:tagname> and an end tag of the same name, </prefix:tagname>.
- These start tag can be any named attributes, followed by equal sign (=) and then the value which is typically surrounded by double quotes or by single quotes.
- A **standard action** may have a body, but it usually doesn't have body at all.
- The prefix for all standard actions is jsp.
- Standard actions can work only if the java object on which is applied obeys at least some minimal conventions. The java objects that need to be used by standard actions need to follow JavaBeans specification.

JSP Action Tags	Description
jsp:forward	Forwards the request and response to another resource.
jsp:plugin	Embeds another components such as applet.
jsp:param	Sets the parameter value. It is used in forward and include mostly.
jsp:useBean	Creates or locates bean object.
jsp:include	Includes another resource.
jsp:text	Used to write template text in JSP pages and documents.
jsp:body	Defines dynamically-defined XML element's body.
jsp:setProperty	Sets the value of property in bean object.
jsp:attribute	Allows you to define the value of a tag attribute in the body of an XML element instead of in the value of an XML attribute.
jsp:fallback	Can be used to print the message if plugin is working. It is used in jsp:plugin.
jsp:params	Add additional parameters to the request and include it in the body of a <jsp:include> or <jsp:forward>
jsp:getProperty	Prints the value of property of the bean.

☞ **Common attributes to action elements.**

Basically there are two attributes that are **common** to all Action elements.

1. Id attribute

The id attribute uniquely identifies the Action element. It allows the action to be referenced inside the JSP page. If the Action creates an instance of an object, the id value can be used to reference it through the implicit object PageContext.

2. Scope attribute

- The scope attribute has four possible values:
  (a) Page (b) Request
  (c) Session (d) Application.
- Scope attribute signifies the lifecycle of the Action element. The id attribute and the scope attribute are directly related, as the scope attribute determines the lifespan of the object associated with the id.

(A) jsp:forward action tag

- The jsp: forward action tag is used to forward the request to another resource it may be jsp, html or another resource. There is only one mandatory attribute, which is page="URL."

☞ **Syntax of jsp : forward action tag without parameter**

```
<jsp:forward page="relativeURL | <%= expression %>" />
```

☞ **Syntax of jsp : forward action tag with parameter**

```
<jsp:forward page="relativeURL | <%= expression %>">
<jsp:param name="parametername" value="parametervalue | <%=expression%>" />
</jsp:forward>
```

☞ **Example 1 :** jsp:forward action tag without parameter

One.jsp

```
<html>
<body>
<h2>this is FIRST page</h2>

<jsp:forward page="Two.jsp" />
</body>
</html>
```

Two.jsp

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

Example 2 : jsp:forward action tag with parameter

One.jsp

```
<html>
<body>
<h2>this is index page</h2>
 <jsp:forward page="Two.jsp" >
<jsp:param name="name" value="javatpoint.com" />
</jsp:forward>
 </body>
</html>
```

Two.jsp

```
<html>
<body>
 <% out.print("Today is:"+java.util.Calendar.getInstance().getTime()); %>
<%= request.getParameter("name") %>

</body>
</html>
```

(B) jsp:include action tag

- This action lets you insert files (it may be jsp, html or servlet) into the page being generated.

Syntax : include action tag without parameter

```
<jsp:include page = "relative URL | <%= expression %>" flush = "true" />
```

- The jsp include action tag includes the **resource** at request time so if there is changes in future, it doesn't matter. But the jsp:include tag can be used to include static as well as dynamic pages.

Syntax of jsp : include action tag with parameter

```
<jsp:include page="relativeURL | <%= expression %>">
<jsp:param name="parametername" value="parametervalue | <%=expression%>" />
</jsp:include>
```


Difference between jsp include directive and include action

Sr. No.	JSP include directive	JSP include action
1.	Includes the original content in the generated servlet.	Calls the include method.
2.	Better for static pages.	Better for dynamic pages.
3.	Includes resource at translation time.	Includes resource at request time.

Attributes

There are two attributes associated with the include action:

- **Page :** The relative URL of the page to be included.
- **Flush :** The boolean attribute determines whether the included resource has its buffer flushed before it is included.
- Let's see one example

Random_num.jsp

```
<p>New Random Number is : <%= (new java.util.Random())..nextInt()%></p>
```

Randome_Demo.jsp

```
<html>
<head>
 <title>Example for include Action</title>
</head>
<body>
<center>
<jsp:include page = "Random_num.jsp" flush = "true" />
</center>
 </body>
</html>
```

(C) <jsp:useBean> Action

- It is used to locate/instantiate a bean class.
- If bean object of the Bean class is already created, it doesn't create the bean (by taking scope in to consideration). But if object of bean is not created, it instantiates the bean.
- It is very versatile.

```
<jsp:useBean id = "name" scope= "page | request | session | application"
class = "package.class" type= "packageName.className"
beanName="packageName.className | <%= expression >"
</jsp:useBean>
```

Attributes

- Following are the Attributes of jsp:useBean action tag

1. **id :** It is used to identify the bean in the specified scope.
2. **Class :** It instantiates the specified bean class, while instantiating it must have no argument or no constructor. It must not be abstract also.
3. **beanName :** It instantiates the bean using the java.beans.Beans.instantiate() method.
4. **Type :** If the bean already exists in the scope then this attribute provides data type to bean. It is mainly used with class or beanName attribute. If it is used without class or beanName, then bean will not be instantiated.
5. **scope:** represents the scope of the bean. The default scope is page.
   - (I) **page :** It is a default scope. It specifies that we can use this bean within the JSP page.
   - (II) **request :** It has wider scope than page. It specifies that we can use this bean from any JSP page that processes the same request.
   - (III) **session :** It has wider scope than request. It specifies that we can use this bean from any JSP page in the same session whether processes the same request or not.
   - (IV) **application :** It has wider scope than session. It specifies that we can use this bean from any JSP page in the same application.

Example

MyFun.java

```
package MyPack;
public class UseBeanDemo
  {
      public int square(int n)
      {
          return n*n;
      }
  }
```

index.jsp file

```
<jsp:useBean id="obj" class="MyPack.UseBeanDemo"/>
<%
int m=obj.square(4);
out.print("square of 4 is "+m);
%>
```


(D) <jsp:setProperty> Action

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 %>}"
/>
```

- If you have to set all the values of incoming request in the bean, it can be done like.

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

- If you have to set value of the incoming specific property, it can be done like.

```
<jsp:setProperty name="Mybean" property="Cust_name" />
```

- If you have to set a specific value in the property, it can be done like.

```
<jsp:setProperty name="Mybean" property="Cust_name" value="Sharma" />
```

(E) <jsp:getProperty> Action

- The jsp:getProperty action tag returns the value of the property.

Syntax of jsp:getProperty action tag

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

For example : Lets see one example of jsp:getProperty action tag

```
<jsp:getProperty name="MyBean" property="Cust_name" />
```

(F) <jsp:plugin> Action

- With the help of this tag we can insert Java components into a JSP page. Basically it determines the type of browser and inserts the **<object>** or **<embed>** tags as needed.
- If the needed plugin is not present, it downloads the plugin and then executes the Java component. The Java component can be either an Applet or a JavaBean.

(G) <jsp:text> Action

- The <jsp:text> action can be used to write the template text in JSP pages and documents.

☞ **Syntax**

```
<jsp:text>Template data</jsp:text>
```

- The body of the template cannot contain other elements; it can only contain text and EL expressions.

(H) <jsp:params>

- With the help of <jsp:param>,we can add additional parameters to the request and we can also include it in the body of a <jsp:include> or <jsp:forward>.
- These additional parameters last only for the duration of the include or forward. Thereafter, the parameters disappear. They don't replace existing parameters of the same name they merely augment the list of values.

(I) <jsp:attribute>

- <jsp:attribute> allows us to define the value of a tag attribute in the body of an XML.

☞ **For example**

```
<jsp:attribute  name="attributeName"  [ trim= "true | false" ]  />
```

- jsp:attribute standard element can be used as a substitute for any of its attributes in all JSP standard actions and custom actions.
- Now, if an action contains any jsp:attribute elements and the action also has a body, it must use the jsp:body tag to represent the body.

4.7 Programs

Program 4.7.1 : Write a JSP program to accept a number from html form, and find the number is even or odd. If even, display in red foreground color else display in blue foreground color.

Ans. :

eo.html

```
<html>
<body>
<form name=frm  method=post action=eo.jsp >
<center>
<br> No <input type=text name=txtno>
<br><input type=submit value=submit>
<input type=reset value=clear>
</center>
</form>
</body>
</html>
```


eo.jsp

```
<html>
<body>
<%
  String st=request.getParameter("txtno");
  int n=Integer.parseInt(st);
  if (n%2 == 0)
  {
        %>
        <font color="red">
        <%=n+" is an even number"%>
        </font>
        <%
  }
else
  {
        %>
        <font color="blue">
        <%=n+" is an odd number"%>
        </font>
        <%
  }
%>
</body>
</html>
```

Program 4.7.2 : Write a JSP program to accept a number from the HTML form, and find its factorial using a recursive function.

Ans. :

fact.html

```
<html>
<body>
<form name=frm  method=post action=fact.jsp >
<center>
<br> No <input type=text name=txtno>
<br><input type=submit value=submit>
```

```
<input type=reset value=clear>
</center>
</form>
</body>
</html>
```

fact.jsp

```
<html>
<body>

  <%!
       int fact(int n)
         {
          if (n<=0)
          return 1;
          else
          return n*fact(n-1);
         }
%>
<%
        String st=request.getParameter("txtno");
       int n=Integer.parseInt(st);
      int val=fact(n);
%>
<%="n = "+n%>
<br>
<%="n! = "+val%>
</body>
</html>
```

Program 4.7.3 : Create a java bean to find the reverse of a number using a read only property. Write a JSP page that uses the bean to find the reverse of the number.

Ans. :

bean2.html

```
<html>
<body>
<form name=frm  method=post action=bean2.jsp >
<center>
```



```
<br> No <input type=text name=txtno>
<br><input type=submit value=submit>
<input type=reset value=clear>
</center>
</form>
</body>
</html>
```

bean2.java

```
package bean;

public class bean2
  {
       int no, rev;
       public void setNo(int no)
       {
            this.no=no;
           reverse();
       }
       void reverse()
       {
           int n=no;
           int r=0,d;
       do
       {
                d=n%10;
                n=n/10;
                r=r*10+d;
       }
  while(n!=0);
  rev=r;
  }
  public int getNo() { return no; }
  public int getRev() { return rev; }
}
```

bean2.jsp

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

```
<%
 String st=request.getParameter("t1");
int no=Integer.parseInt(st);
%>
<jsp:useBean id="b1" scope="session" class="bean.bean2"/>
<jsp:setProperty name="b1" property="no" value="<%=no%>"/>
<br> Number
<jsp:getProperty name="b1" property="no"/>
<br> Reverse
<jsp:getProperty name="b1" property="rev"/>
</body>
</html>
```

Program 4.7.4 : Write a JSP program to find the square root and cube root of a given number [input number through html page].

Ans. :

perform.html

```
<html>
<body>
<form name=frm  method=post action=perform.jsp >
<center>
<br> No <input type=text name=txtno>
<br><input type=submit value=submit>
<input type=reset value=clear>
</center>
</form>
</body>
</html>
```

perform.jsp

```
<html>
<body>
<%
   String st=request.getParameter("txtno");
   int no=Integer.parseInt(st);
   double a=Math.sqrt(no);
   double b=Math.pow(no, 1.0/3);
   out.println("Square root(" +no +") = " +a);
```



```
   out.println("<br>");
   out.println("Cube root(" +no +") = " +b);
%>
</body>
</html>
```

Program 4.7.5 : Write a JSP program that prints the reverse of the given number [user input through html]

Ans. :

reverse.html

```
<html>
<body>
<form name=frm  method=post action=reverse.jsp >
<center>
<br> No <input type=text name=txtno>
<br><input type=submit value=submit>
<input type=reset value=clear>
</center>
</form>
</body>
</html>
```

reverse.jsp

```
<html>
<body>
<%
   String st=request.getParameter("txtno");
   int no=Integer.parseInt(st);
   out.println("<br>Number : "+no);
   int rev=0,d;
   do{
        d=no%10;
        no=no/10;
        rev=rev*10+d;
     }
while(no!=0);
```

```
out.println("<br>Reverse : "+rev);
%>
</body>
</html>
```

Program 4.7.6 : Write a JSP program that swaps two numbers. Input the number through HTML.

Ans. :

swap.html

```
<html>
<body>
<form name=frm  method=post action=swap.jsp >
<center>
<br> A <input type=text name=t1>
<br> B <input type=text name=t2>
<br><input type=submit value=submit>
<input type=reset value=clear>
</center>
</form>
</body>
</html>
```

swap.jsp

```
<html>
<body>
<%
   int a=Integer.parseInt(request.getParameter("t1"));
   int b=Integer.parseInt(request.getParameter("t2"));
   out.println("<br>Before Swaping");
   out.println("<br>A = "+a);
   out.println("<br>B = "+b);
   int t=a;
   a=b;
   b=t;
   out.println("<br>After Swaping");
```



```
   out.println("<br>A = "+a);
   out.println("<br>B = "+b);
%>
</body>
</html>
```

Program 4.7.7 : Write a JSP code to print the details entered in the form on the next page. The details are entered in the html page as username in textbox, date of birth in textbox, date of joining in text box, gender in radio button.

Ans. :

accept.html

```
<html>
<body>
<form name=frm  method=post action=display.jsp>
<center> User Details </center>
<br><br>User Name <input type=text name=t1>
<br> DOB <input type=text name=t2>
<br> DOJ <input type=text name=t3>
<br> Gender
<input type=radio name=r1 value="Male"> Male
<input type=radio name=r1 value="Female"> Female
<br><input type=submit value=submit>
<input type=reset value=clear>
</form>
</body>
</html>
```

display.jsp

```
<html>
<body>
<%
   String username=request.getParameter("t1");
   String dob=request.getParameter("t2");
   String doj=request.getParameter("t3");
   String gender=request.getParameter("r1");
   out.println("User Details area..");
   out.println("<br>User Name : " +username);
   out.println("<br>DOB : " +dob);
```

```
    out.println("<br>DOJ : " +doj);
    out.println("<br>Gender : " +gender);
%>
</body>
</html>
```

Program 4.7.8 : Write a JSP application that computes the cubes of the number from 1 to 10 and display in table format.

Ans. :

Cube.jsp

```
<html>
<body>
<table border="1" align="center">
<tr><th> i </th>
<th> i^3  </th></tr>
<%
double r;
for(int i=1; i<=10; i++)
   {
         r=Math.pow(i, 3);
         out.println("<tr><td>" +i +"</td>");
         out.println("<td>" +r +"</td></tr>");
   }
%>
</table>
</body>
</html>
```

Program 4.7.9 : Write a JSP based application that serves the purpose of simple calculator.

Ans. :

calc.html

```
<html>
<head>
<title>Calculator</title>
</head>
<body>
<form method=post action=calc.jsp>
NO-1 <input type=text name=t1>
```



```
NO-2 <input type=text name=t2>
<br><br>
<input type=submit value="+" name="btn">
<input type=submit value="-" name="btn">
<input type=submit value="*" name="btn">
<input type=submit value="/" name="btn">
</form>
</body>
</html>
```

calc.jsp

```
<html>
<body>
<%
   int a=Integer.parseInt(request.getParameter("t1"));
   int b=Integer.parseInt(request.getParameter("t2"));
   int c=0;
   String op=request.getParameter("btn");
   if (op.equals("+")) c=a+b;
   if (op.equals("-")) c=a-b;
   if (op.equals("*")) c=a*b;
   if (op.equals("/")) c=a/b;
   out.println(a+op+b+" = "+c);
%>
</body>
</html>
```

Review Questions

Q. 1 Write the advantages of JSP over servlet.

Q. 2 Differentiate between jsp include directive and jsp include action.

Q. 3 What is page directives ? Explain page directives and its attributes.

Q. 4 Enlist the implicit objects of JSP. Explain any 4 of them in detail.

Q. 5 Write a JSP that accepts user login details and forward the result either 'Access Granted' or 'Access Denied' to result.jsp.

Q. 6 Write a JSP based application that serves the purpose of simple calculator.

Q. 7 What is JSP ? What are the advantages and disadvantages of JSP.

Q. 8 Write a JSP code to print the details entered in the form on the next page. The details are entered in the html page as username in textbox, date of birth in textbox, date of joining in text box, gender in radio button.

Q. 9 What is page directives ? Explain page directives and its attributes.

Q. 10 Write a JSP that swaps two numbers. Input the number through HTML.

Q. 11 Write a JSP application that computes the cubes of the number from 1 to 10, and display in table format.

Q. 12 Explain with an example addition of java bean to a JSP page.

❑❑❑

UNIT - III

CHAPTER 5

Java Beans

Syllabus Topics

Java Beans : Introduction, JavaBeans Properties, Examples.

Syllabus Topic : Introduction

5.1 Introduction

Q. What is Java Bean? Discuss the advantages and disadvantages of it.
Q. Explain the concept of Java Beans in detail.

☞ **Java Bean :** A Java Bean is a java class. JavaBeans are classes that capture and store many objects into a single object (called bean).

- Let's learn some unique characteristics that distinguish a JavaBean from other Java classes :
  - Java Bean class may have a number of properties which can be used in various ways.
  - Java Bean class may have a number of "**getter**" and "**setter**" methods for the properties which increases user friendliness.
  - Java Bean should be serializable.
  - Java Bean provides a default i.e. no-argument constructor.

☞ **Why use Java Bean ?**

- Java Bean is a reusable software component. It is easy to maintain.
- It encapsulates many objects into one object, so we can access this object from multiple places.