Web Component Development Using Java

Session: 7

JavaServer Pages

Objectives

- Explain the need of JSP
- List the benefits of JSP
- Identify the situations where to use JSP and servlets
- Explain JSP architecture
- Identify various phases in the life cycle of a JSP page
- Explain the various scriptlet elements in JSP
- List and explain the use of various directives in JSP

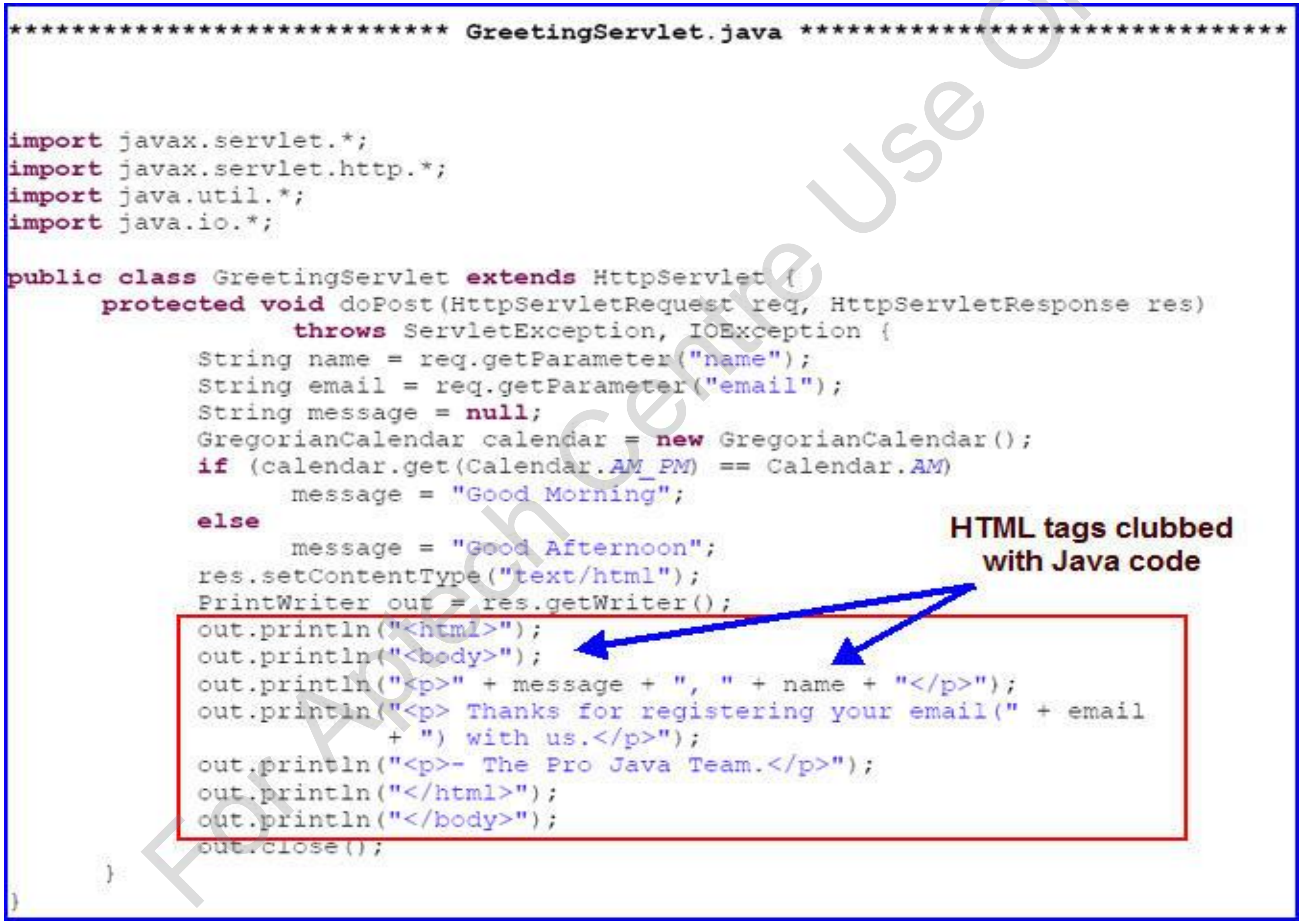
Introduction 1-3

Scenario: A developer has to greet the user on his/her successful registration on the Web page.

- **Solution:**
  - Can be accomplished using Servlet technology.
  - Create a Servlet class named `GreetingServlet`.
  - Processes the request and generates a response to be sent to the client.

Introduction 2-3

- Figure shows the `GreetingServlet` class.



```
**************************** GreetingServlet.java ********************************


import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;
import java.io.*;

public class GreetingServlet extends HttpServlet {
      protected void doPost(HttpServletRequest req, HttpServletResponse res)
                  throws ServletException, IOException {
            String name = req.getParameter("name");
            String email = req.getParameter("email");
            String message = null;
            GregorianCalendar calendar = new GregorianCalendar();
            if (calendar.get(Calendar.AM_PM) == Calendar.AM)
                  message = "Good Morning";
            else
                  message = "Good Afternoon";
            res.setContentType("text/html");
            PrintWriter out = res.getWriter();
            out.println("<html>");
            out.println("<body>");
            out.println("<p>" + message + ", " + name + "</p>");
            out.println("<p> Thanks for registering your email(" + email
                  + ") with us.</p>");
            out.println("<p>- The Pro Java Team.</p>");
            out.println("</html>");
            out.println("</body>");
            out.close();
      }
}
```

Introduction 3-3

- In the code:
  - To generate the output, the `GreetingServlet` code uses Java code that will display the string value in the output.
  - HTML elements embedded in the same Servlet that take care of the presentation of the string on the Web page.
  - Embedding HTML elements along with the Java code result in the development of rigid applications, which have a tight coupling between presentation and application layer.
  - Any modifications in the Servlet results in page re-compilation, which means embedded HTML elements are also re-compiled.

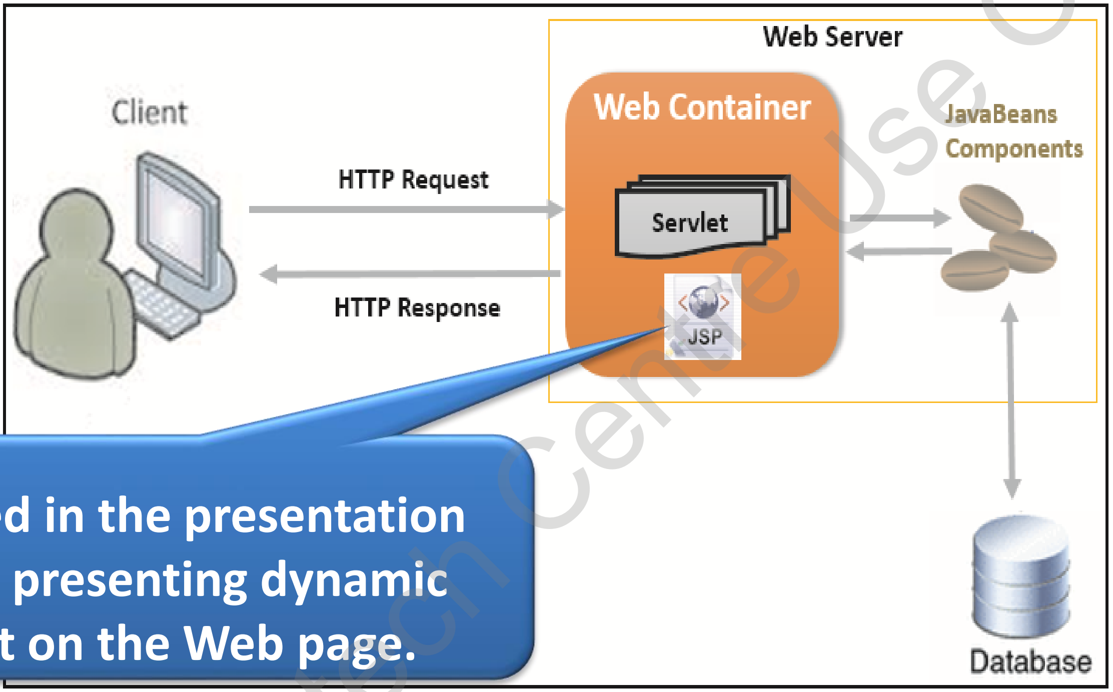
To segregate the presentation text from the application, Sun laid a new specification referred to as **JavaServer Pages (JSP)**.

JSP Page

- The JSP specification is an extension of the existing Servlet API and leverages all the features of Servlets.
- JSP page:
  - A text document containing static content, JSP tags, and Java code which simplifies the generation of dynamic contents.
  - Static contents are expressed as HTML, XML, or XHTML markup tags and are used to generate the user interfaces on the page.
  - Java code and JSP elements are used to generate dynamic contents on the Web page.

Application Layers

❖ Figure shows the application layers in the Web application.



JSP is used in the presentation layer for presenting dynamic content on the Web page.

❖ Most software applications consist of following three layers:

- ❑ Presentation layer contains user interface and code to build the interface.
- ❑ Application or Business layer contains code to validate data accepted in presentation logic and other task specific code.
- ❑ Data access layer contains code to operate on the databases.

Benefits of JSP

Separation of Content Generation from Presentation

- JSP separates content generation from presentation by implementing presentation logic in a JSP page and content logic on server in a JavaBean component.

Emphasizing Reusable Components

- JSP pages use reusable components, such as JavaBeans which can be used by multiple programs.

Simplified Page Development

- JSP allows a Web developer with limited knowledge in scripting language to design a page.

Access and Instantiate JavaBean Components

- JSP supports the use of JavaBean components with JSP language elements. The user can easily create and initialize beans and get and set their values.

Use of Servlets and JSP

JSP

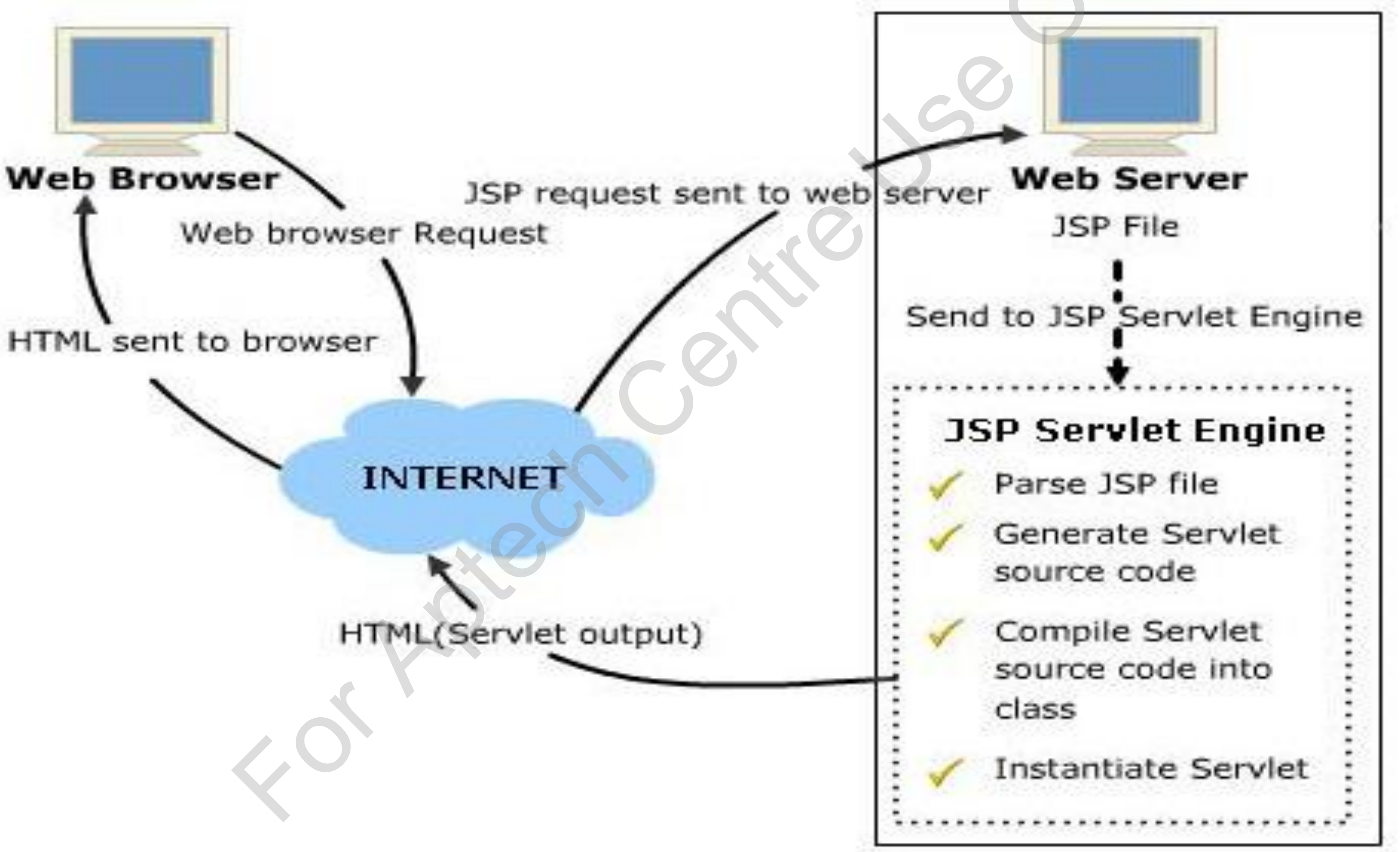
- JSP technology simplifies the process of creating pages by separating the Web presentation from Web content.

Servlet

- Servlets are well suited for handling binary data dynamically, for example, for uploading files or for creating dynamic images.

Architecture of JSP

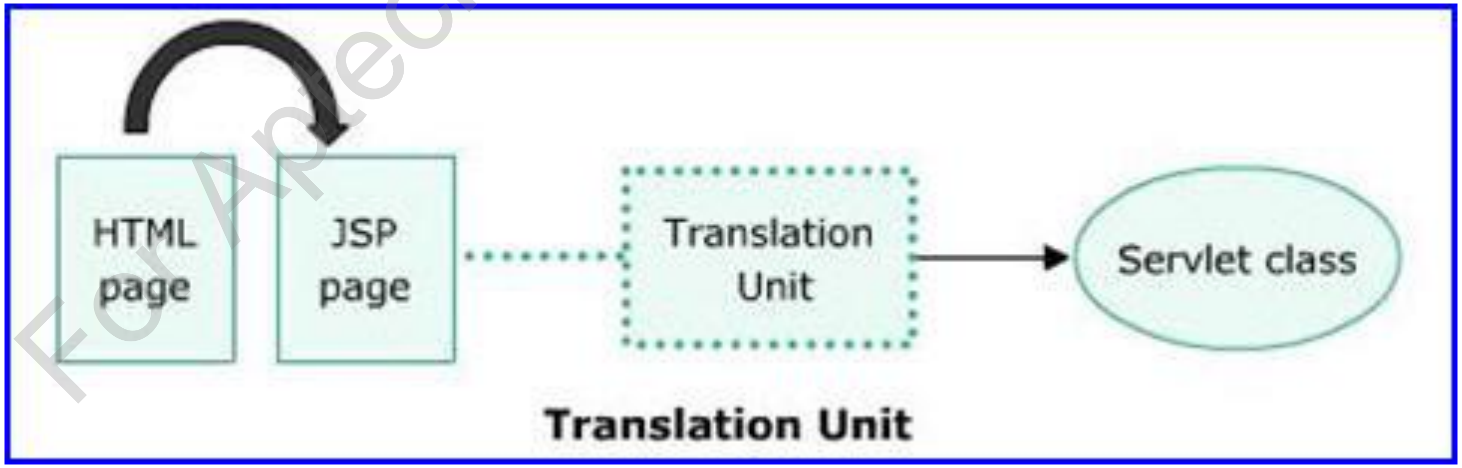
- Figure depicts JSP architecture.



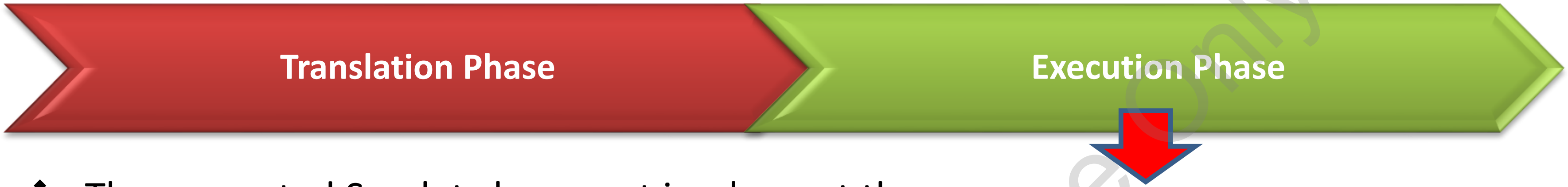
Life Cycle Phases of JSP 1-2

Translation Phase
Execution Phase

- In this phase, a servlet code to implement JSP tags is automatically generated, compiled, and loaded into the Servlet container.
- The JSP engine writes the entire static contents such as markup tags to the response stream in the Servlet.
- The JSP elements are converted into the equivalent Java code in the generated Servlet code.
- After translation, the Servlet source code is compiled to `.class` file which is a compiled Servlet class.
- The generated file is handed to the Servlet container for managing the Servlet life cycle methods.



Life Cycle Phases of JSP 2-2



- The generated Servlet class must implement the `javax.servlet.jsp.HttpJspPage` interface defined in the JSP API.
- The `HttpJspPage` interface defines the life cycle methods.
- Following life cycle methods executed by the Servlet container on the loaded instance of the converted Servlet are as follows:

`jspInit()`	`_jspService()`	`jspDestroy()`
• Similar to `init()` method of Servlet and is invoked on the instantiated Servlet to initialize it with the parameters. • A JSP page can override this method by including a definition for it in a declaration element.	• Similar to `service()` method of Servlet and is invoked by the Servlet container at each request. • Is defined automatically by the JSP container and corresponds to the body of the JSP page.	• Executed by the Servlet container before destroying the JSP page from the memory.

Elements of a JSP Page

❖ Contains tags that are categorized as **Standard Tags** and **Custom Tags**.

Standard Tags

- They are used to process request, include Java objects and directives, and perform actions on the JSP page.
- These tags include scripting tags, directive tags, and action tags.
- All the tags are written within angular brackets <>.

Custom Tags

- These tags perform operations, such as processing forms, accessing databases, and other enterprise services, such as e-mail and directories.
- These are created and loaded in separate libraries.

Scripting Tags 1-5

- The JSP tags are used to embed Java code snippets in the JSP page for generating dynamic contents on the Web page. These tags are as follows:

JSP Expressions

- Can be used to display individual variables or the result of some calculation on a Web page.
- Contains a Java statement whose value will be evaluated and inserted into the generated Web page.
- Is evaluated and the result is converted into a string. This evaluated string will be displayed on the Web page.
- **Syntax:**

```
<%= Java Expression %>
```

where,

- `<%` = is the opening delimiter for the scriptlet.
- `Java Expression` indicates a valid Java statement.
- `%>` is the closing delimiter.

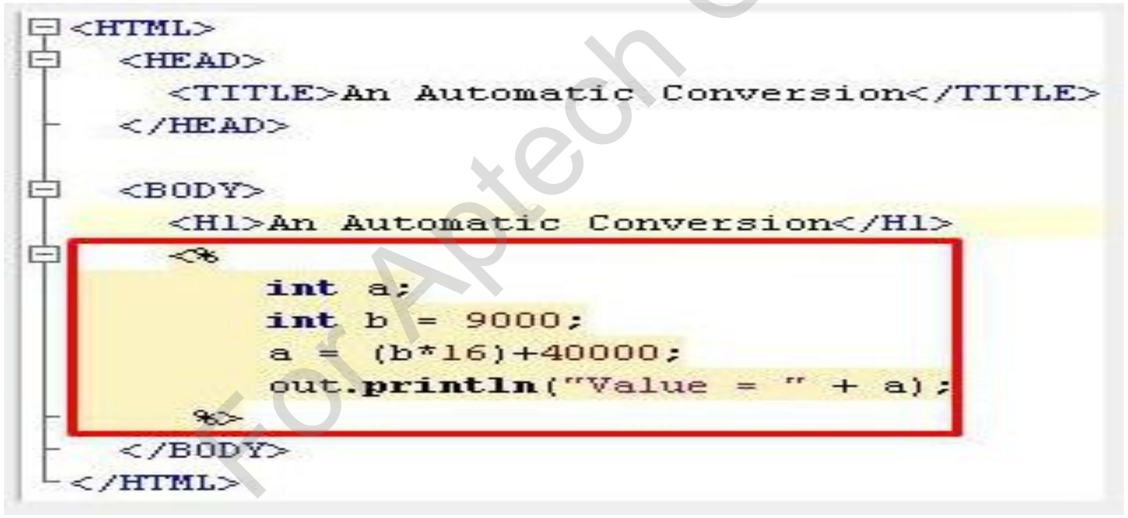
Scripting Tags 2-5

JSP Scriptlets

- Is used to embed Java code within an HTML code.
- Is inserted into the `_jspService()` method of the servlet.
- Are executed when the request of the client is being processed.
- Are used to add complex data to an HTML form.
- **Syntax:**

```
<% Java code fragment %>
```

- Figure depicts the use of scriptlet in a JSP page.



```
<HTML>
    <HEAD>
        <TITLE>An Automatic Conversion</TITLE>
    </HEAD>

    <BODY>
        <H1>An Automatic Conversion</H1>
        <%
            int a;
            int b = 9000;
            a = (b*16)+40000;
            out.println("Value = " + a);
        %>
    </BODY>
</HTML>
```

Scripting Tags 3-5

JSP Declarations

- Allow the user to define variables and methods that are inserted into the Servlet, outside the `service()` method.
- Can declare variables within the scriptlet tag also, but that variable will be local to that scriptlet.
- Variable declared using the declaration tag will be visible in the whole JSP page.
- Cannot be declared within the scriptlet tag.
- Can be used to define multiple variables of same data type.
- **Syntax:**

```
<%! Java declaration code %>
```

Scripting Tags 4-5

- The code snippet demonstrates the use of declaration tag in the JSP page along with its output.

```
<html>
...
<body>
<h3>Area of a Rectangle</h3>
<%! int length = 20, breadth = 30, area = 0; %>
<%! Private String units = "cm"; %>

<% area = length * breadth; %>
<p>  The area of the Rectangle is:
<%= area %><%= units %>
</p>
</body>
</html>
```

Scripting Tags 5-5

Comments

- JSP comments are used for documenting JSP code which should not be visible to the client when viewing the source of the Web page.
- These comments are called server-side comments and are encoded within `<%--` and `--%>` symbol.
- **Syntax:**

```
<%-- a JSP comment --%>
```

- **Example:** `<%-- This is the JSP comment in a file --%>`

Directive Tags 1-2

- JSP directives control the processing of the entire page.
- These directives identify the packages to be imported and the interfaces to be implemented.
- These directives do not produce any output. They inform the JSP engine about the actions to be performed on the JSP page.
- These directives affect the overall structure of the Servlet class.
- **Syntax:**

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

where,

- `directiveName` is the name of the directive.
- `attribute` is attribute of directive.
- `value` is the value of attribute specified.

User can declare these directives anywhere on the page. However, mostly they are declared at the top of the JSP page.

Directive Tags 2-2

❖ In JSP, there are three directives:

`page`

- The `page` directive provides instructions that control the structure of the page

`include`

- The `include` directive includes static contents on the current JSP page

`taglib`

- The `taglib` directive includes the custom library tags on the JSP page

page Directive 1-4

- Can be used to import classes from the packages, set the content type for the page, enable or disable page as error page, set the language on the page, and so on.
- Defines and manipulates a number of important attributes that affect the entire JSP page.
- Is written at the beginning of a JSP page.
- Can contain any number of page directives.
- All directives in the page are processed together during translation and result is applied together to the JSP page.

Example:

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

- Used to import more than one package in the JSP page. If more than one package is imported, then separate the package names by commas.

page Directive 2-4

❖ Some of the attributes that can be defined for a page are as follows:

`Language:` Used to specify the language of the script. The default value of language in a JSP page is **`Java.`**

`errorPage:` A path name to this JSP file sends exceptions. If the pathname begins with a **`'/',`** the path is relative to the root directory and is resolved by the Web server. If not, the path name is relative to the current JSP file.

`autoFlush:` The buffered output is flushed automatically when the buffer is full. If the **`autoFlush`** is set to **`true`**, the buffer will be flushed. Otherwise, an exception will be thrown when the buffer overflows.

`Session:` The client must join an HTTP session in order to use the JSP page. If the value is **`true`**, the session object refers to the current or new session. The default value for session is **`true`**.

page Directive 3-4

- **Syntax:**

```
<%@ page attribute %>
```

where,

- `page` indicates page directive.
- `attribute` specifies attribute-value pair of page directive.

- The valid parameters are as follows:

```
  <%@ page language="java"
           extends="className"
           import="className{,+}"
           session="true|false"
           buffer="none|sizeInKB"
           autoFlush="true|false"
           isThreadSafe="true|false"
           info="text"
           errorPage="jspUrl"
           isErrorPage="true|false"
           contentType="mimeType{;charset=charset}"
  %>
```

page Directive 4-4

- The code snippet demonstrates how to set the language on the JSP page by using the `page` directive.

```
<html>
<%@ page language="Java" %>
<head>
     <title>Testing Page Directive</title>
</head>
<body>
     <h1>Testing Page Directive</h1>
     This page is testing Page Directive.
</body>
</html>
```

include Directive 1-2

- Is used to insert content of another resource in a JSP page at run time.
- Content are included physically during page translation:
  - The resource or content can be a file in a text-based format, such as text, HTML, or JSP.
  - The remote resource should be given with the proper relative file path in the `include` directive.
- **Syntax:**

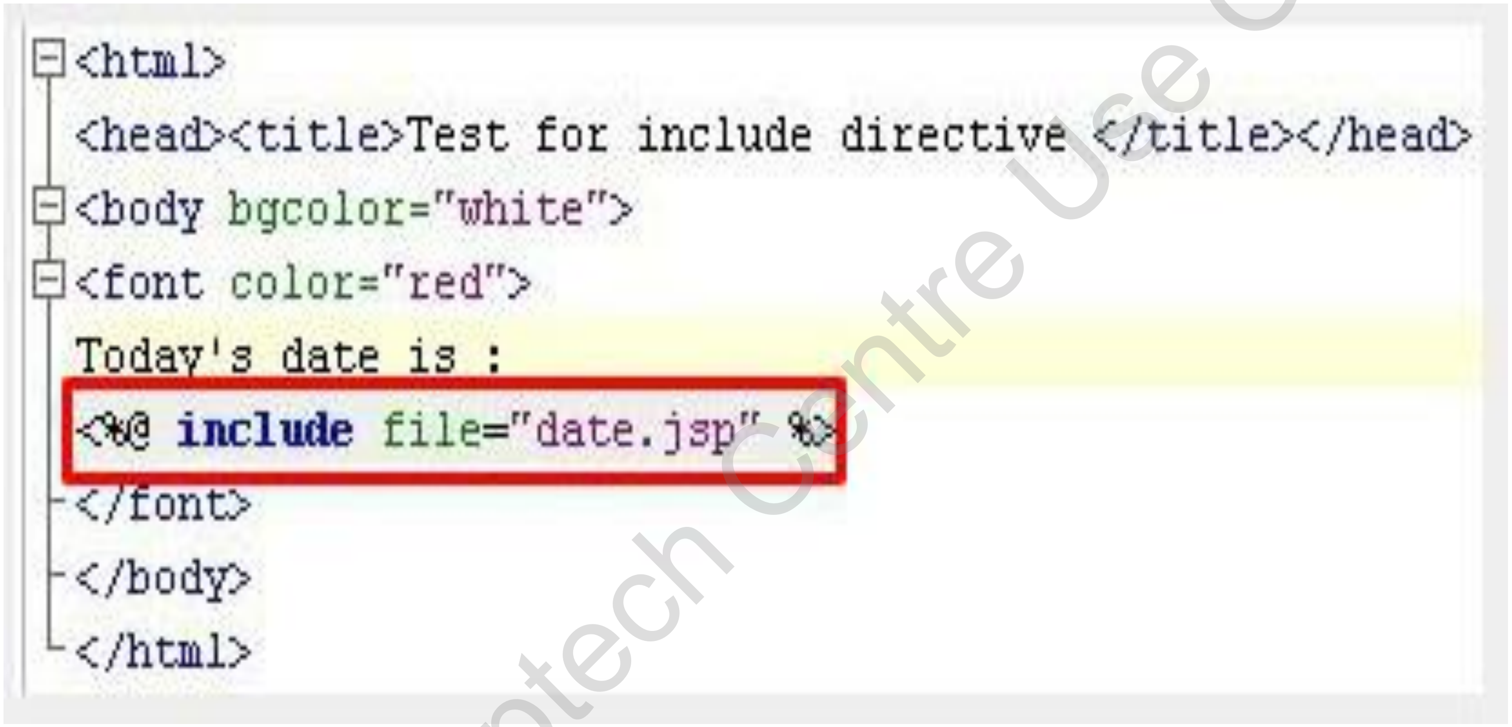
```
<%@ include file = "Filename" %>
```

where,

- `include` indicates the directive and `Filename` specifies the name of a file to include along with relative path.

include Directive 2-2

- Figure depicts the `include` directive.



```
<html>
<head><title>Test for include directive </title></head>
<body bgcolor="white">
<font color="red">
Today's date is :
<%@ include file="date.jsp" %>
</font>
</body>
</html>
```

taglib Directive 1-2

- It allows the JSP page to create custom tags, which are defined by the user.
- A tag library is a group of custom tags that extend the functionality of a JSP page one after the other.
- The `taglib` directive specifies name of the tag library, which contains compiled Java code for a tag to be used.
- Using this tag library, the JSP engine determines what action is to be taken when a particular tag appears in a JSP page.
- **Syntax:**

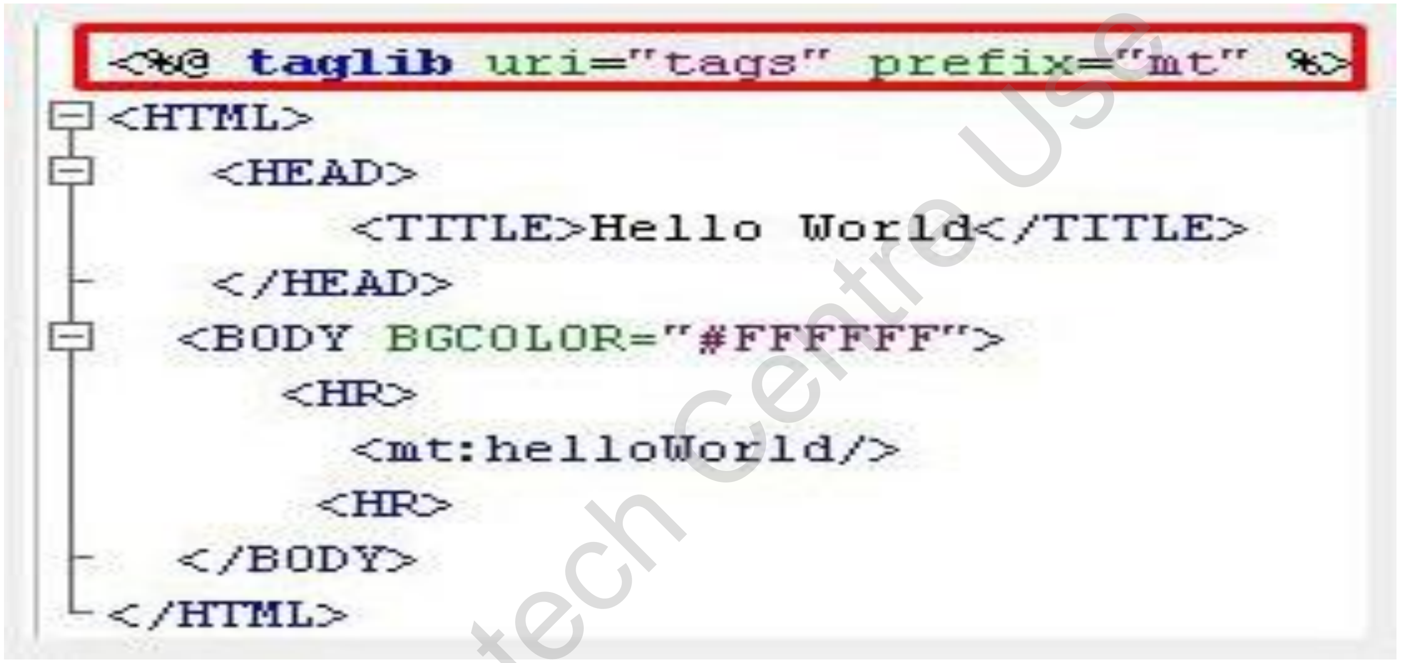
```
<%@ taglib uri="tagLibraryURI" prefix="tagPrefix" %>
```

where,

- `tagLibraryURI` specifies Uniform Resource Identifier (URI) that identifies the tag library descriptor.
- `prefix` specifies tag name that is used to define a custom tag.
- `tagPrefix` defines the prefix string in `prefix:tagname`.

taglib Directive 2-2

- Figure depicts the `taglib` directive.



```
<%@ taglib uri="tags" prefix="mt" %>
<HTML>
    <HEAD>
        <TITLE>Hello World</TITLE>
    </HEAD>
  <BODY BGCOLOR="#FFFFFF">
      <HR>
        <mt:helloWorld/>
      <HR>
  </BODY>
</HTML>
```

Summary

- JSP technology is used for developing dynamic Web sites and provides server-side scripting support for creating Web applications.
- A JSP page is a mixture of standard HTML tags, Web page content, and dynamic content that is specified using JSP elements.
- The JSP elements are expressions, scriptlets, comments, declarations, directives, and actions.
- Directives are used to specify the structure of the resulting servlet and actions are JSP tags that transfer control to other server objects or perform operations on other objects.
- The page directive is used to set the page attributes, such as the language to be used and encoding format.
- The include directive is used to insert a file referenced in the directive into the JSP page.
- The taglib directive links the JSP page to an XML document that describes a set of custom JSP tags and determines which tag handler class can implement the action of each tag.