# JSP

Unit3

Web Technology –N.P. Gopalan

Faculty-Fiona Coutinho

## Advantages

- Develop dynamic websites
- Developers can insert java code in html
- Development and maintenance simple
- JSP pages efficient as they load into web servers memory for the first time

## First JSP

Saved as .jsp

```
<html>
<body>
<%=new java.util.Date() %>
</body>
</html>
```

## Comparison

- ASP is similar technology from Microsoft .JSP simpler compared to VB and is portable to other O.S and non Microsoft web servers
- Servlets: Similar to JSP .Convenient to write java in html then the other way round
- SSI: server-side includes makes static page dynamic but JSP better as includes servlets
- Javscript :has no access to server resources and cannot deal with cookies,form submitted data and HTTP
- Static html: does not contain dynamic data

## Components of JSP

- Directives
- Declarations
- Scriptlets
- Expressions
- Standard Actions
- Custom tags

# Components of JSP:Directives

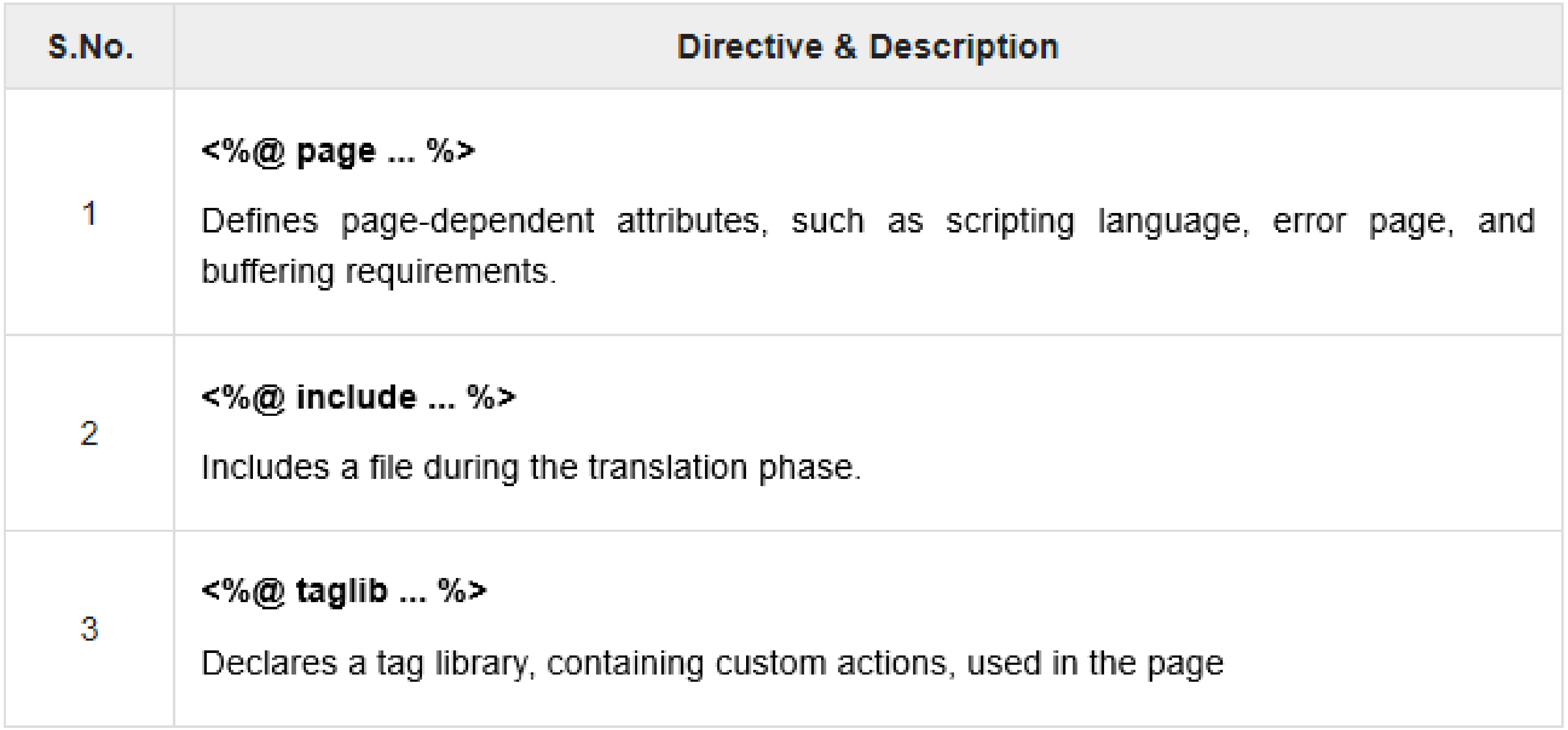
- Directives provide directions and instructions to the container,
- telling it how to handle certain aspects of the JSP processing.
- It usually has the following form -

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

## Components of JSP:Directives

- Directives can have a number of attributes which you can list down as key-value pairs and separated by commas.
- The blanks between the @ symbol and the directive name, and between the last attribute and the closing %>, are optional.
- The 3 most important directives in JSP are :

## Components of JSP:Directives



| S.No. | Directive & Description |
|---|---|
| 1 | **<%@ page ... %>**<br>Defines page-dependent attributes, such as scripting language, error page, and buffering requirements. |
| 2 | **<%@ include ... %>**<br>Includes a file during the translation phase. |
| 3 | **<%@ taglib ... %>**<br>Declares a tag library, containing custom actions, used in the page |

## Components of JSP:Directives

1.The page Directive

- Provide instructions to the container.
- These instructions pertain to the current JSP page.
- You may code page directives anywhere in your JSP page.
- By convention, page directives are coded at the top of the JSP page.
- Basic syntax of the page directive −

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

## Attributes associated with the page directive –

- **errorPage:**Defines the URL of another JSP that reports on Java unchecked runtime exceptions.
- **Extends:**Specifies a superclass that the generated servlet must extend.
- **Language:**Defines the programming language used in the JSP page.
- **Session:**Specifies whether or not the JSP page participates in HTTP sessions

## Components of JSP:Directives

2.The include directive is used to include a file during the translation phase.

- This directive tells the container to merge the content of other external files with the current JSP during the translation phase.
- You may code the include directives anywhere in your JSP page.
- The general usage form is −

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

- The filename in the include directive is actually a relative URL. If you just specify a filename with no associated path, the JSP compiler assumes that the file is in the same directory as your JSP.

## Components of JSP:Directives

3.The taglib Directive:The JavaServer Pages API allow you to define custom JSP tags that look like HTML or XML tags and a tag library is a set of user-defined tags that implement custom behavior.

- The taglib directive declares that your JSP page uses a set of custom tags, identifies the location of the library, and provides means for identifying the custom tags in your JSP page.
- The taglib directive follows the syntax given below –

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

- Here, the uri attribute value resolves to a location the container understands and the prefix attribute informs a container what bits of markup are custom actions.

# Components of JSP:Declaratives

- **<**%!  field or method declaration %**>**
- The jsp declaration tag can declare variables as well as methods(class level used anywhere in jsp page).
- In this example of JSP declaration tag, we are declaring the field and printing the value of the declared field using the jsp expression tag

```
<html>
<body>
<%! Int cnt=0;
private int getcount()
{cnt++;return cnt;}%>
<%= getcount() %>  //output 1
<%= getcount() %>  // output 2
</body>
</html>
```

## Expressions

- Syntax of expression is <% =“any thing” %> //result is always string

Eg.

**<**%! int data=50; %>

**<**%= "Value of the variable is:"+data %**>**

## scriptlets

- The jsp scriptlet tag can only declare variables not methods.

<% Your Java Source Code

String s=null;

S=request.getparameter(“un”);%>

- Jsp engine puts this Code in _jspService() method .Variables available to scriptlets are:

1. Request represents client request ,is subclass of HttpServletRequest-used to get data submitted along with the req.
2. Response is subclass HttpServletResponse
3. Session: represents HttpSession object linked with the request
4. Out :output stream object  which sends output to the client

## Standard Action

- <jsp:include page=“”/>:includes output of included file into this JSP file at runtime.
- <jsp:forward page=“”/>:redirects to different page without notifying the browser

## Custom Tags

Syntax is : <%@ taglib prefix = "ex" uri = "WEB-INF/custom.tld"%>

- Prefix is alias for tag library name

JSP provides these implicit objects:

1. Out
2. Request
3. Response
4. Session
5. Application
6. Config
7. Page
8. PageContext

# Custom Tags

1. Out :is instantiated implicitly from JSP Writer class .used for displaying anything within delimiters.Eg. out.println("Hi Hello!!")
2. Request : is implicit object of HttpServletRequest class. Request parameters accessed as follows: request.getParameter("name of form element")
3. Response: implicit object of HttpServletResponse class.Response parameters can be modified or set. Eg.Response.setBufferSize("50") modifies http headers.
4. Session: belongs to HttpSession class and maintains session info(as name – value pairs). Eg. Session.setValue("name","Eva")
5. Application:Object belongs to ServletContext class and maintains certain info in scope of app. Eg. Appliction.setValue("Servername","www.myserver.com")
6. Config: has Servlet configuration details
7. Page: has page info
8. PageContext: object of PageContext class used to access other implicit objects

## Custom Tags

- A custom tag is a user-defined JSP language element.
- When a JSP page containing a custom tag is translated into a servlet, the tag is converted to operations on an object called a tag handler.
- The Web container then invokes those operations when the JSP page's servlet is executed.
- JSP tag extensions lets you create new tags that you can insert directly into a JavaServer Page. The JSP 2.0 specification introduced the Simple Tag Handlers for writing these custom tags.
- To write a custom tag, you can simply extend **SimpleTagSupport** class and override the **doTag()** method, where you can place your code to generate content for the tag.

# Custom Tags :Example (Additional Info)

Steps to create Custom tag:

Step1 create tag class:

```
package newTag;
import javax.servlet.jsp.tagext.*;
import javax.servlet.jsp.*;
import java.io.*;
public class HelloTag extends SimpleTagSupport {
  public void doTag() throws JspException, IOException {
    JspWriter out = getJspContext().getOut();
    out.println("Hello Custom Tag!");
  }
}
```

# Custom Tags :Example (Additional Info)

This code has doTag() method which takes the current JspContext object using the getJspContext() method and uses it to send "Hello Custom Tag!" to the current JspWriter object.

Step2: Compile the above class and copy it in a directory available in the environment variable CLASSPATH.

Step3: Finally, create the following tag library file:

<Tomcat-Installation-Directory>webapps\ROOT\WEB-INF\custom.tld.

```
<taglib>
  <tlib-version>1.0</tlib-version>
  <jsp-version>2.0</jsp-version>
  <short-name>Example TLD</short-name>
    <tag>
    <name>Hello</name>
    <tag-class>newTag.HelloTag</tag-class>
    <body-content>empty</body-content>
  </tag>
</taglib>
```

## Custom Tags :Example (Additional Info)

Step4: Use the above defined custom tag Hello in our JSP program as follows –

```
<%@ taglib prefix = "ex" uri = "WEB-INF/custom.tld"%>
<html>
  <head>
    <title>A sample custom tag</title>
  </head>
    <body>
    <ex:Hello/>
  </body>
</html>
```

Step5: Call the above JSP and it should produce the following result :

```
Hello Custom Tag!
```

## Reading request information

When http client send request to web server it also sends some http variables like remote address,remote host,content type,etc.

How to fetch Some of these variable using the request object is shown next.

# Reading request information

```
<% @page contentType=“text/html” import=“java.util.*” %>
<html><body>
<%=“Request Method”+request.getMethod() %>  //GET or POST
<%= "Path Info:"+request.getPathInfo() %> //null
<%= “Request URI:"+ request.getRequestURI() %> //      /examples/jsp/hello.jsp
<%= "Path Info:"+request.getPathTranslated() %>  // null
<%= “Protocol:"+request.getProtocol() %>   // HTTP/1.1
<%= "QueryString:"+request.getQueryString() %> //null
<%= "Content Length:"+request.getContentLength() %> //-1
 <%="Content Type:"+request.getContentType() %> //null
<%="ServerName:"+ request.getServerName() %> //localhost
<%= “ServerPort:"+ request.getServerPort() %>  //8080
<%= "Remote User:"+request.getRemoteUser() %> //null
<%= "Remote Address:"+request.getRemoteAddr() %> //127.0.0.1
<%="Remote Host:"+request.getRemoteHost() %> //localhost
<%= "Auth Type:"+request.getAuthType() %> //null</body></html>
```

## Retrieve Data posted from html file to jsp file

Consider Getname.html: which accepts users name and displays it on showname.jsp page.

```
<html>
  <body>

    <form action = “Showname.jsp" method = "GET">
      First Name: <input type = "text" name = "first_name">
      <br />
      Last Name: <input type = "text" name = "last_name" />
      <input type = "submit" value = "Submit" name=“B1”/>
    </form>

  </body>
</html>
```

# Retrieve Data posted from html file to JSP file

**Showname.jsp : Retrieves the value entered by user using Request object.**

```
<% @page contentType="text/html"  %>
<html>
  <head>
    <title> Read Form Data</title>
  </head>
    <body>
    <center>
    <h1>Form Data</h1>
        <ul>
      <li><p><b>First Name:</b>
        <%= request.getParameter("first_name")%>
      </p></li>
      <li><p><b>Last  Name:</b>
        <%= request.getParameter("last_name")%>
      </p></li>
    </ul>
    </body>
</html>
```

## JSP Sessions

In any web application user moves from one page to other and in order to track user data and objects throughout the application, JSP provides sessions.

## JSP Sessions

```
<html>
  <body>
          <form action = “save.jsp" method = "GET">
        userName: <input type = "text" name = “username">
        <br />
<input type = "submit" value = "Submit"/>
     </form>
        </body>
</html>
```

## JSP Sessions

```
Save.jsp: <%@ page  language="java"  %>
<% String username=request.getParameter("username");
If(username==null) username="";
Session.setAttribute("username",username);
%>
<html><head><title>Saved data</title></head><body>
<a href="displaysessionvalue.jsp">Click to view session value</a>
</body></html>
```

## JSP Sessions

displaysessionvalue.jsp :

```
<%@ page  language="java"  %>
<% String username=(String) session.getAttribute("username");
If(username==null) username="";%>
<html>
<head>
<title>Saved data</title>
</head>
<body>
<p> Welcome : <%=username %><p></body></html>
```

# JSP Cookies

- Cookies are text files stored on the hard disk of  client computer
- They are kept for various information tracking purposes.
- Steps involved in identifying and returning users −

1. Server script sends a set of cookies to the browser. For example, name, age, or identification number, etc.
2. Browser stores this information on the local machine for future use.
3. When the next time the browser sends any request to the web server then it sends those cookies information to the server and server uses that information to identify the user or may be for some other purpose as well.

## JSP Cookies

- Cookies are objects of javax.servlet.http.Cookie.
- Value of a cookie uniquely identifies client so its used for session management .It has a name,single value and optional attributes as comment,path and domain wualifiers,maximum age,version number
- getCookies() of request object returns array of cookie objects
- Cookies constructed using ;

Cookie(java.lang.String name,java.lang.String value)

## Cookie Methods

1. String getComment(): get the purpose described for the cookie
2. setComment(String purpose):Sets a comment that describes cookie's purpose.
3. getMaxAge():Get cookie age in seconds. -1 indicating the cookie will persist until browser shutdown.
4. setMaxAge(int seconds):Sets how much time in seconds should elapse before the cookie expires. By default, the cookie will last for the current session.
5. String getValue():gets the value associated with the cookie.
6. setValue(String newValue):sets the value associated with the cookie.
7. String getName(): returns the name of the cookie, which cannot be changed after creation.
8. String getPath(): returns prefix of all URLs for which this cookie is targeted..
9. setPath(String uri):This cookie should be presented only with requests beginning with this URL.

## Use of Cookies

```
<html>
  <body>
          <form action = “setcookie.jsp" method = “post">
        userName: <input type = "text" name = “username">
        <br />
<input type = "submit" value = "Submit"/>
      </form>
        </body>
</html>
```

## Use of Cookies

**setcookie.jsp :**

```
<%@ page  language="java"  import="java.util.*" %>
<% String username=request.getParameter("username");
If(username==null) username="";
Date now =new Date();
String timestamp=now.tostring();
Cookie cookie=new Cookie("username",username);
Cookie.setMaxage(365*24*60*60);
Response.addCookie(cookie);%>
<html><head><title>Saved cookie data</title></head><body>
<a href="displaycookievalue.jsp">Click to view cookie value</a>
</body></html>
```

## Use of Cookies

**displaycookievalue.jsp:**

```
<%@ page  language=“java”  %>
<% String cookiename=“username”;
Cookie cookies[]=request.getcookies();
Cookie mycookie=null;
If(cookies !=null) {
for(int i=0;i<cookies.length;i++)
{if(cookies[i].getName().equals(cookiename)){mycookie=cookies[i];break;}
}}%>
<html>
<head>
<title>show Saved cookie data</title>
</head>
<body><% if(mycookie==null){%>No cookie with the name <%=cookiename%>
<%else { %><p> Welcome : <%=mycookie.getValue()%><p></body></html>
```

## Disabling Sessions

- Disabling Sessions increases performance of JSP container.
- Sessions are enabled by default so Session object created by Engine everytime a JSP is requested which uses server resources as well as system resources as its stored on server side which also increases traffic as session ID is send from server to client and client too send it along with Request.
- If JSP is accessed frequently and there's no need to identify the user its better to disable sessions.
- Disabled using the following code:

```
<% @page language=“java” session=“false” %>
<html><body>Session disabled</body></html>
```

## General Questions

- Write script to accept no. from user and check if its even
- From 1-10 display all even numbers except for the no. 5

## Some more topics:

# PHP string methods:

- Strcmp,strstr,trim()
- Go through the other methods on page 205

## PHP Date methods:208

- Localtime(),getdate(),time()
- Go through the other methods on page 208

## PHP Math methods:

- Cos(),sin(),tan(),exp(),pi(),pow()
- Go through the other methods on page 210

# PHP-The ArrayIterator

Php arrays tracks the current element using the pointer Iterator.Php has tool to loop over an array called arrayIterator having the following methods:

1. Current()-returns current element
2. Reset()-moves iterator to first element in array and returns its value
3. Next()- moves iterator to next element in array and returns its value
4. Prev()- moves iterator to previous element in array and returns its value
5. End()- moves iterator to last element in array and returns its value
6. Each- returns index and value of current element as an array and moves iterator to next element in array
7. Key()-returns index of the current element

# PHP-The ArrayIterator

```
<?php
$fruits = array(
   "apple" => "yummy",
   "orange" => "ah ya, nice",
   "grape" => "wow, I love it!",
   "plum" => “tasty"
);
$obj = new ArrayIterator( $fruits );
 $obj->rewind();

while( $obj->valid() )
{
   echo $obj->key() . "=" . $obj->current() . "\n";
   $obj->next();
}
// The good thing here is that it can be iterated with foreach loop
foreach ($obj as $key=>$val)
echo $key.":".$val."\n";

?>
```

# PHP-The ArrayIterator

**/* Outputs something like */**

```
Iterating over: 4 values
apple=yummy
orange=ah ya, nice
grape=wow, I love it!
plum=tasty
```

## Built-in array functions:

-Count()

-array_search()

-sort()

- Go through the other methods on page 218 and 219

## Storing Arrays in cookies in Php

```
<?
$address=$_SERVER[‘REMOTE_ADDR’];
$os=$_SERVER[‘os’];
Setcookie(“cookie[0]”,”$address”,time()+200);
Setcookie(“cookie[1]”,”$os”,time()+200);?>
<html><body><? $list=$_COOKIE[“cookie”];
Echo “IP address $list[0]”;
Echo “operating system $list[1]”?></body></html>
```

## Working with Sessions in Php

```
<?
Session_start();
If(isset($_SESSION[‘counter’]))
{$_SESSION[‘counter’]+=1;}
else{$_SESSION[‘counter’]=1;}?>
<html><body><? Echo “You have visited this
page”.$_SESSION[‘counter’].” in this session”;
?></body></html>
```