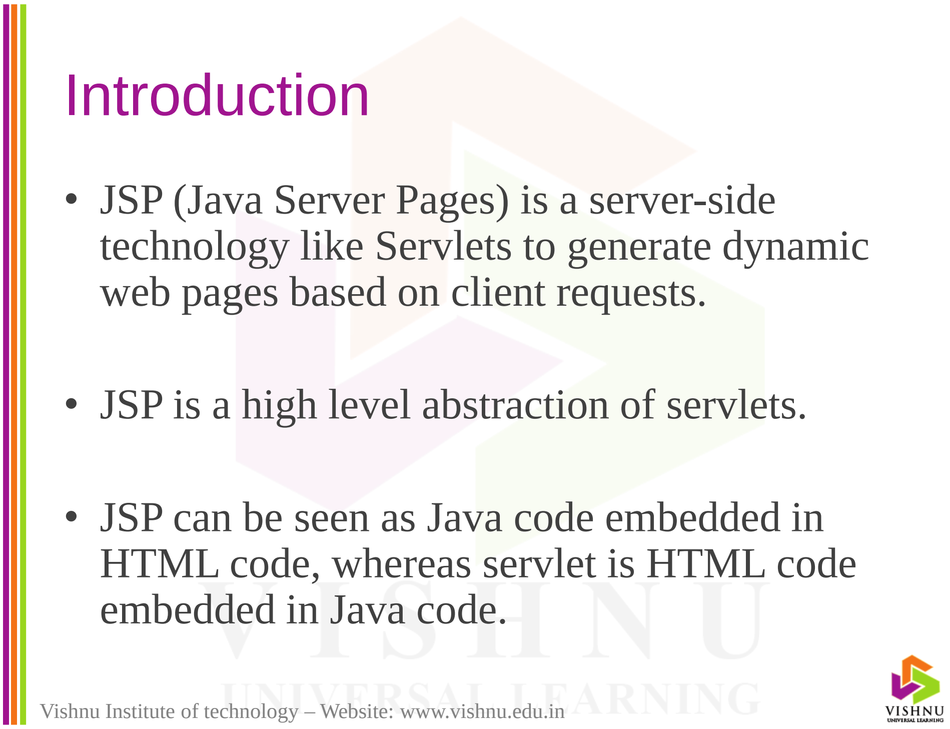
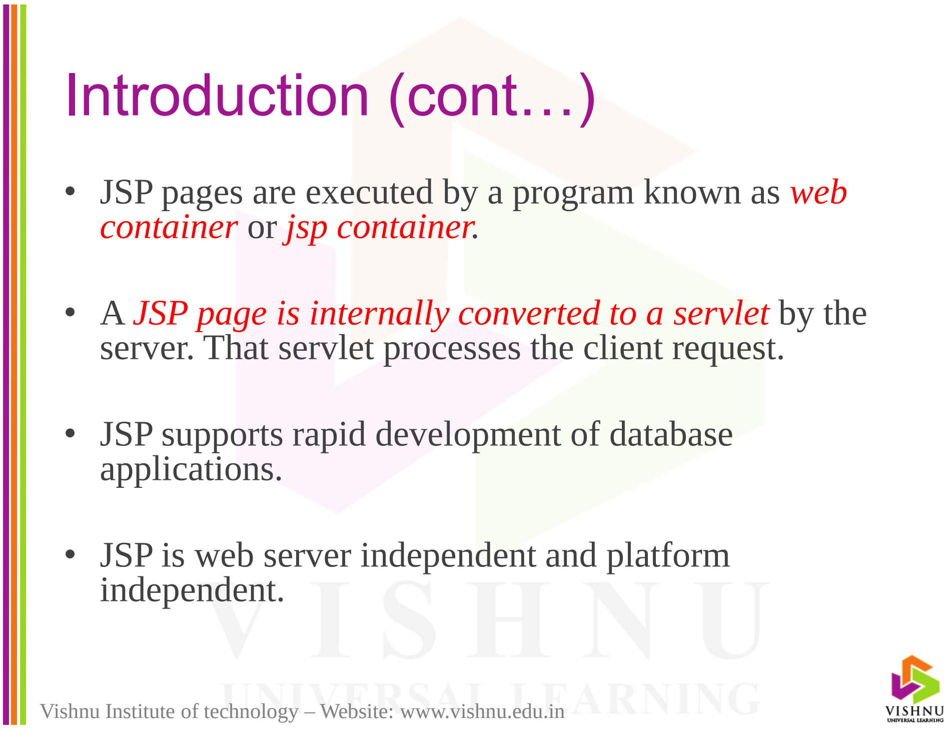
JSP
[Java Server Pages]



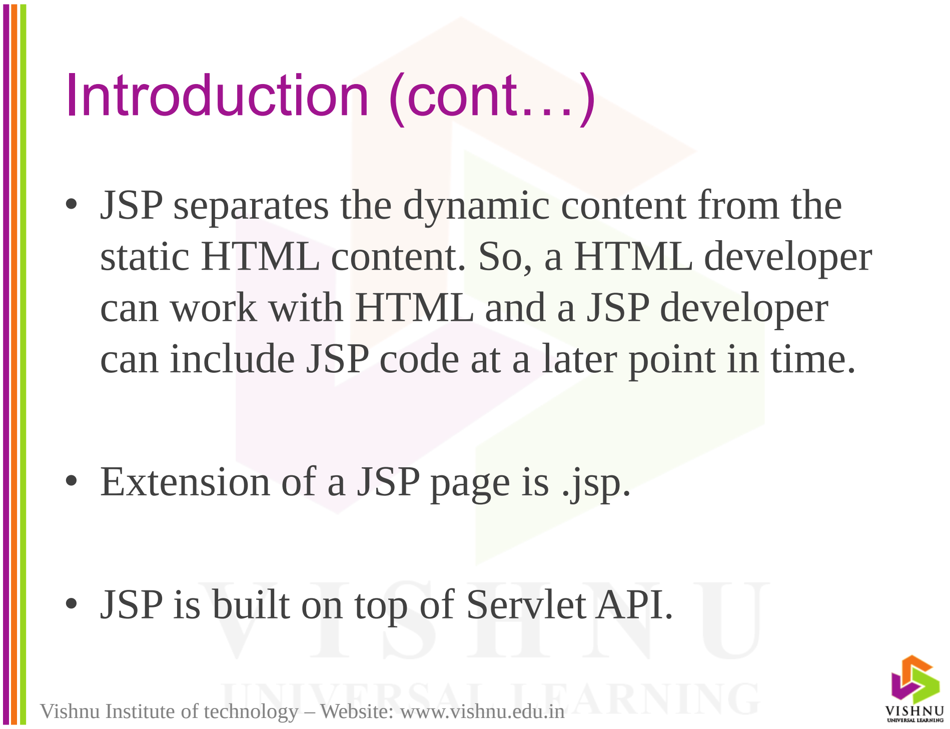
Introduction

- JSP (Java Server Pages) is a server-side technology like Servlets to generate dynamic web pages based on client requests.
- JSP is a high level abstraction of servlets.
- JSP can be seen as Java code embedded in HTML code, whereas servlet is HTML code embedded in Java code.



Introduction (cont…)

- JSP pages are executed by a program known as *web container* or *jsp container*.
- A *JSP page is internally converted to a servlet* by the server. That servlet processes the client request.
- JSP supports rapid development of database applications.
- JSP is web server independent and platform independent.



Introduction (cont…)

- JSP separates the dynamic content from the static HTML content. So, a HTML developer can work with HTML and a JSP developer can include JSP code at a later point in time.
- Extension of a JSP page is .jsp.
- JSP is built on top of Servlet API.

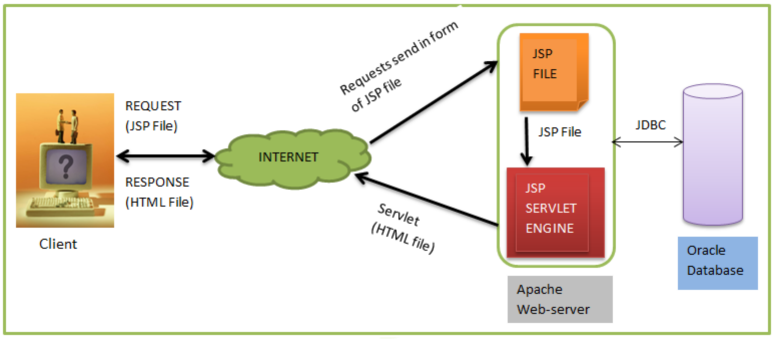
Disadvantages of Servlets

- Servlet contains both business logic (java code) and presentation logic (HTML).
- Difficult for maintaining web applications.
- Takes more time for development.

JSP Container or JSP Engine

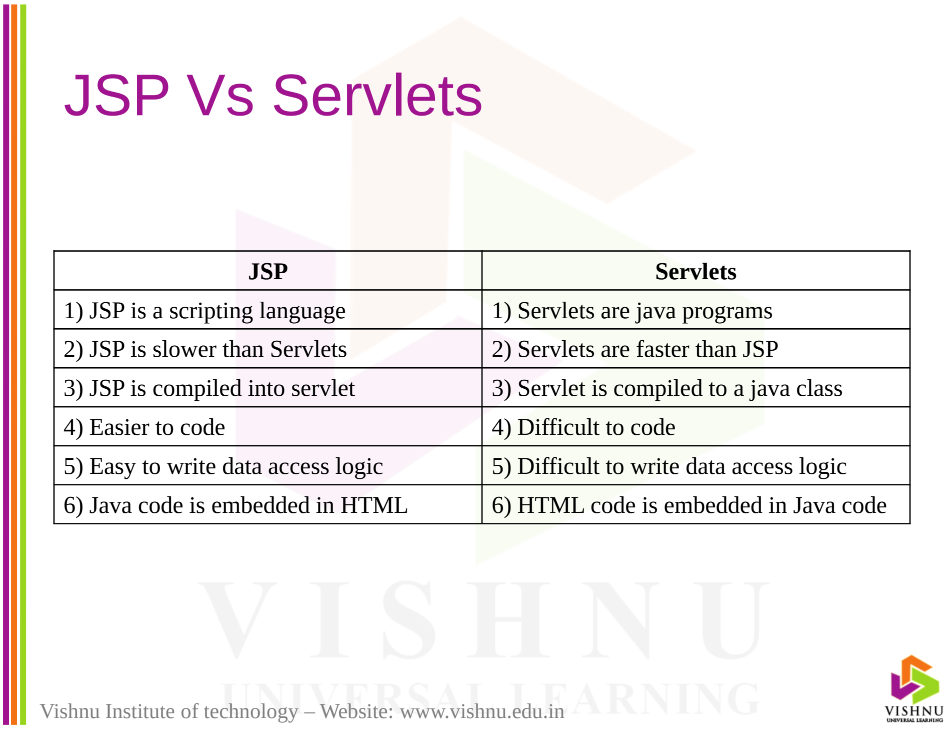
- JSP container is a program (special servlet) which forwards the jsp page request to a servlet container.

How JSP Works?



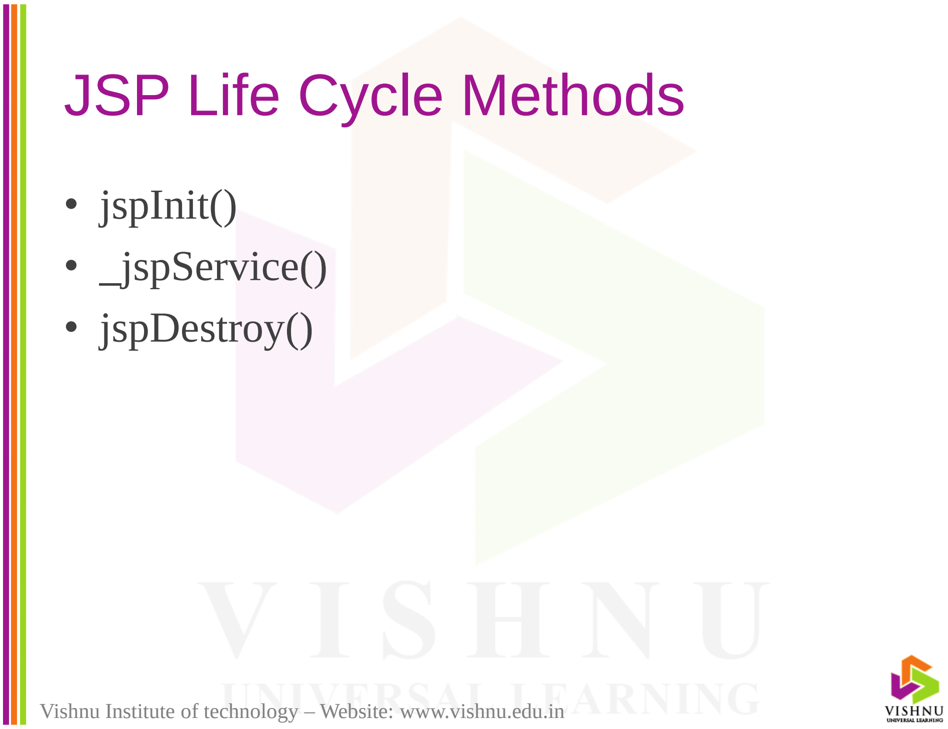
Steps in processing client requests in JSP

1. Webserver identifies .jsp extension and forwards the request to JSP container.
2. JSP container forwards the request to servlet container.
3. Servlet container translates JSP source code to servlet source code.
4. Servlet source code is compiled to create a class file.
5. Class file is loaded and instantiated.
6. Servlet is initialized by invoking *jspInit()* method.
7. Invokes, the *_jspService()* method, by passing request and response objects.



JSP Vs Servlets

JSP	Servlets
1) JSP is a scripting language	1) Servlets are java programs
2) JSP is slower than Servlets	2) Servlets are faster than JSP
3) JSP is compiled into servlet	3) Servlet is compiled to a java class
4) Easier to code	4) Difficult to code
5) Easy to write data access logic	5) Difficult to write data access logic
6) Java code is embedded in HTML	6) HTML code is embedded in Java code



JSP Life Cycle Methods

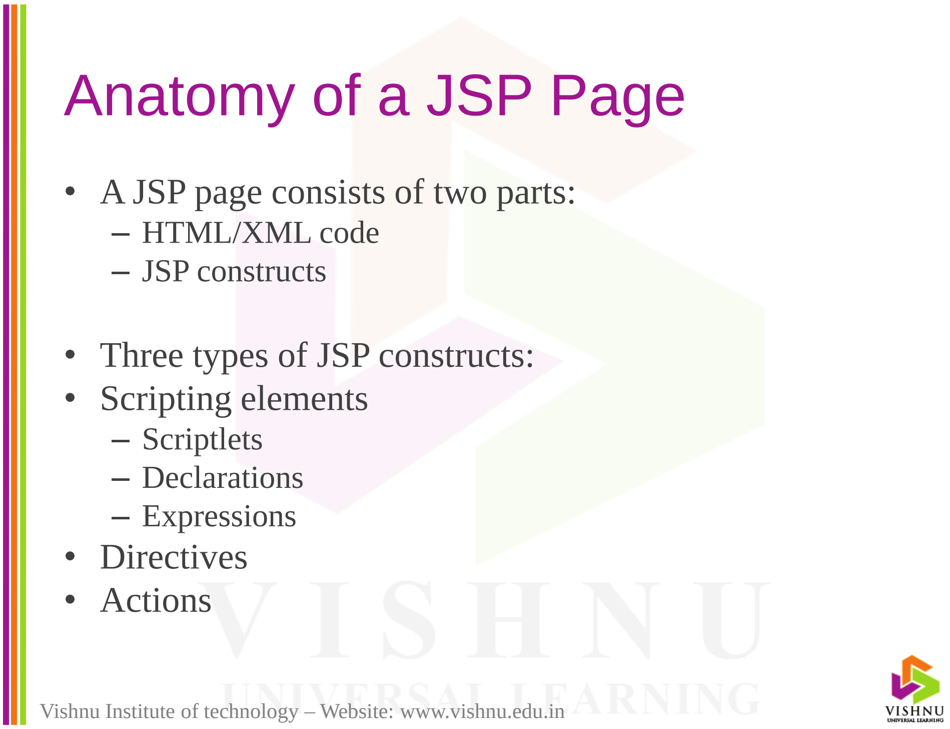
- jspInit()
- _jspService()
- jspDestroy()

HelloWorld - Servlet

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloServlet extends HttpServlet
{
    public void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException
    {
            response.getWriter().write(“<html><head><title>”);
            response.getWriter().write(“Hello Servlet</title></head>”);
            response.getWriter().write(“<body>”);
            response.getWriter().write("Hello World!");
            response.getWriter().write(“</body></html>”);
    }
}
```

Hello World - JSP

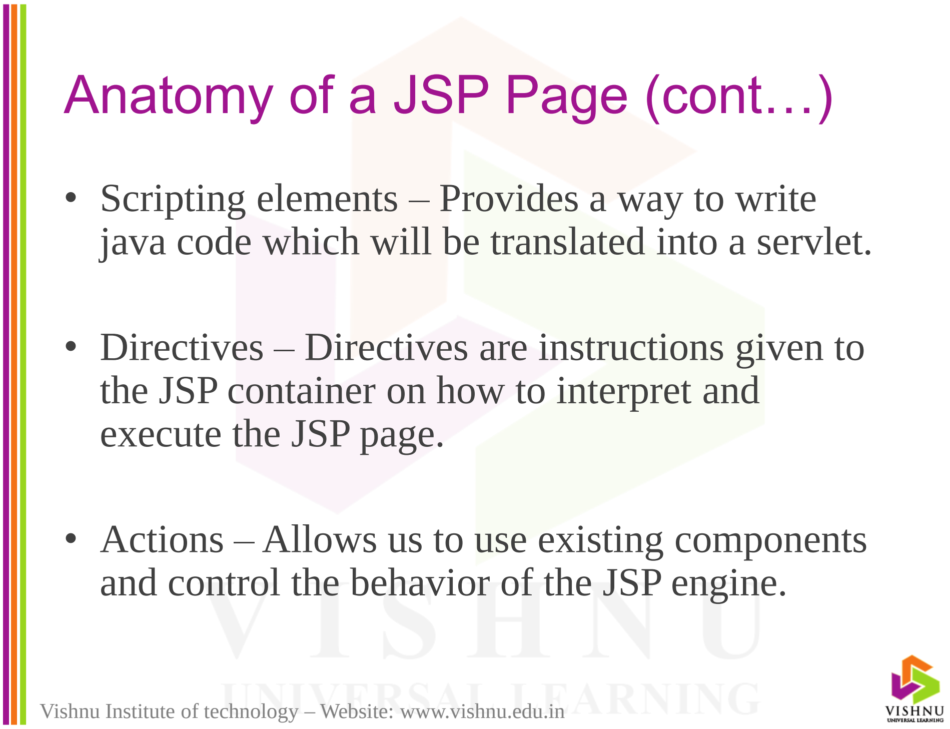
```
<html>
   <head>
        <title>Hello JSP</title>
   </head>
   <body>
        <%  out.print(“Hello World!”);  %>
   </body>
</html>
```



Anatomy of a JSP Page

- A JSP page consists of two parts:
  - HTML/XML code
  - JSP constructs

- Three types of JSP constructs:
- Scripting elements
  - Scriptlets
  - Declarations
  - Expressions
- Directives
- Actions



Anatomy of a JSP Page (cont…)

- Scripting elements – Provides a way to write java code which will be translated into a servlet.

- Directives – Directives are instructions given to the JSP container on how to interpret and execute the JSP page.

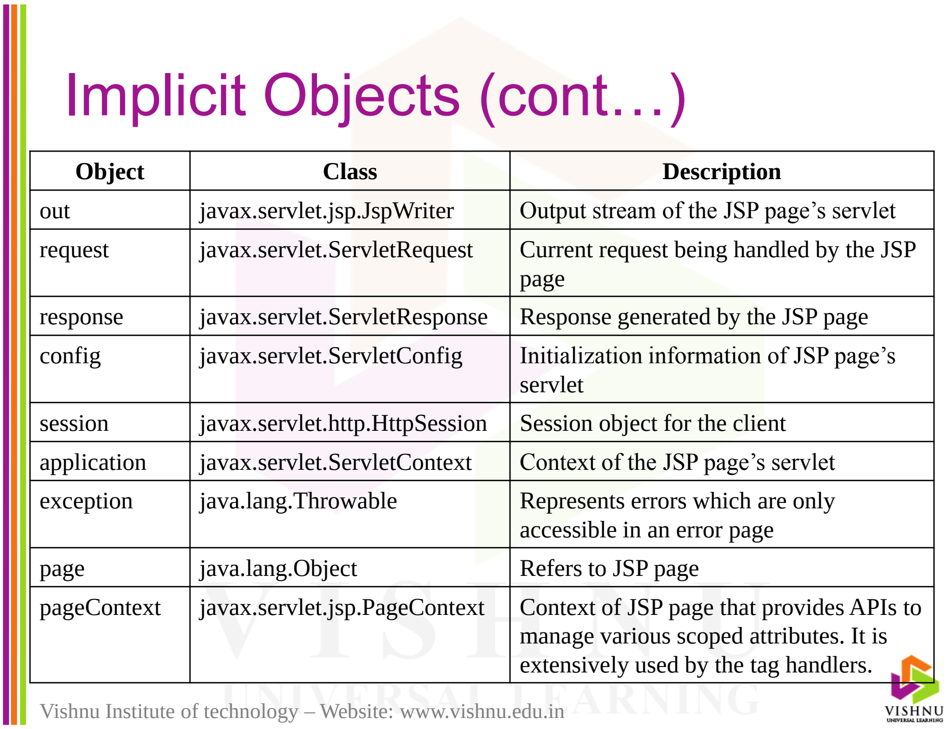
- Actions – Allows us to use existing components and control the behavior of the JSP engine.

Anatomy of a JSP Page (cont…)

- Scriptlets – Provides a way to insert java code in the HTML or XML page.
- Declarations – Provides a way to declare variables and methods.
- Expressions – Used to print the value of a java expression.

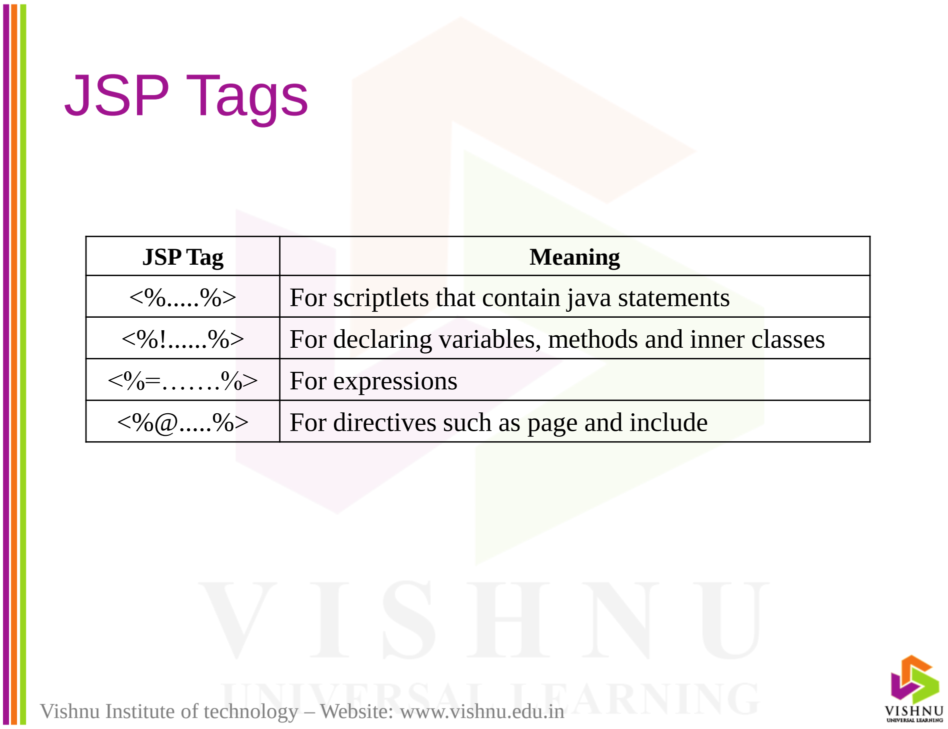
Implicit Objects

- Implicit objects are objects that are predefined in the *_jspService()* method of the JSP page's underlying servlet.
- Implicit objects are created automatically.
- Implicit objects contain information about request, response, session, configuration etc…



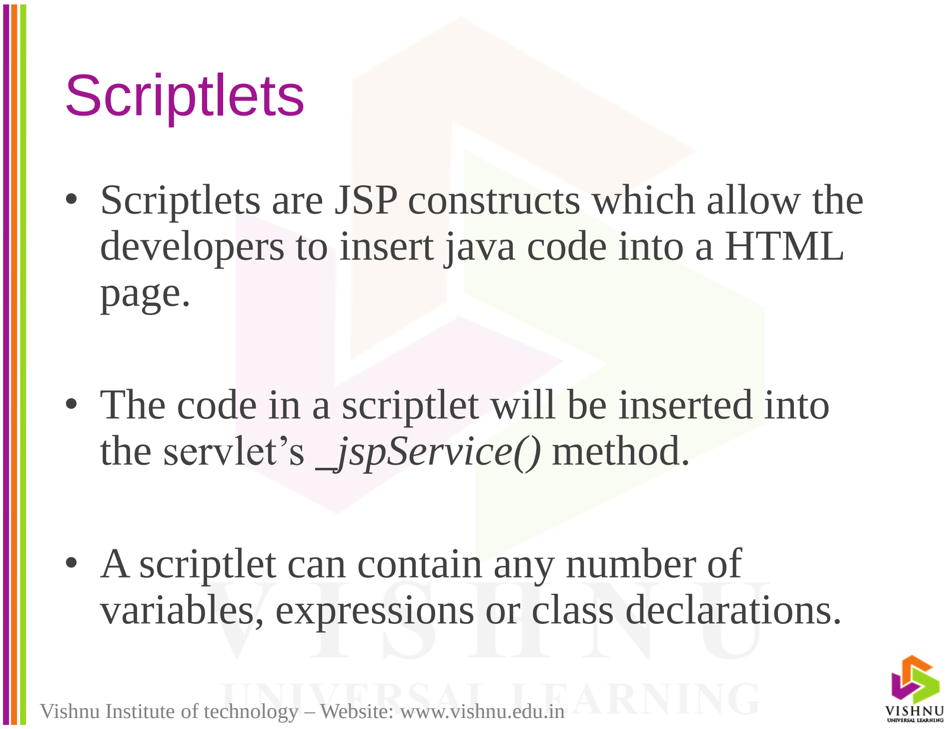
Implicit Objects (cont…)

Object	Class	Description
out	javax.servlet.jsp.JspWriter	Output stream of the JSP page's servlet
request	javax.servlet.ServletRequest	Current request being handled by the JSP page
response	javax.servlet.ServletResponse	Response generated by the JSP page
config	javax.servlet.ServletConfig	Initialization information of JSP page's servlet
session	javax.servlet.http.HttpSession	Session object for the client
application	javax.servlet.ServletContext	Context of the JSP page's servlet
exception	java.lang.Throwable	Represents errors which are only accessible in an error page
page	java.lang.Object	Refers to JSP page
pageContext	javax.servlet.jsp.PageContext	Context of JSP page that provides APIs to manage various scoped attributes. It is extensively used by the tag handlers.



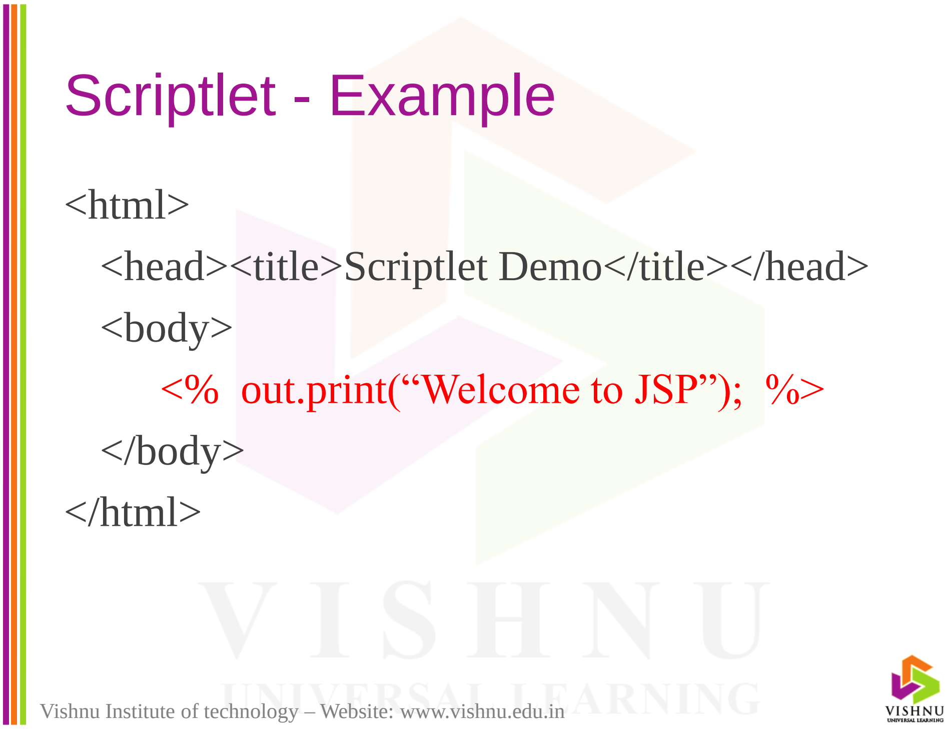
JSP Tags

JSP Tag	Meaning
<%.....%>	For scriptlets that contain java statements
<%!......%>	For declaring variables, methods and inner classes
<%=…….%>	For expressions
<%@.....%>	For directives such as page and include



Scriptlets

- Scriptlets are JSP constructs which allow the developers to insert java code into a HTML page.
- The code in a scriptlet will be inserted into the servlet’s *_jspService()* method.
- A scriptlet can contain any number of variables, expressions or class declarations.



Scriptlet - Example

```
<html>
   <head><title>Scriptlet Demo</title></head>
   <body>
        <%  out.print(“Welcome to JSP”);  %>
   </body>
</html>
```

Scriptlet – Conditional Processing

- We can execute a set of statements based on the result of a condition in a control statement.

```
<%
        int   number =
10;
        if(number > 10)
        {
%>
<p>Valid input</p>
<%
        }
        else
        {
%>
<p>Invalid input</p>
<%
        }
%>
```

Declarations

- JSP declaration construct is used to declare variables, methods or inner classes.

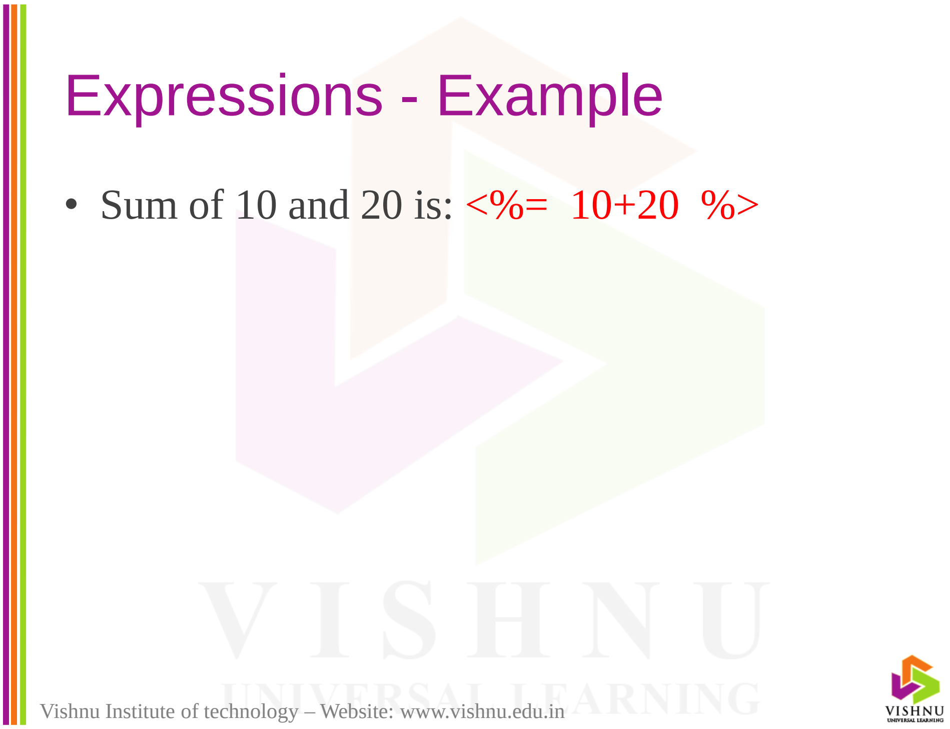
- Variables declared using *declarations* are treated as instance variables.

Declarations - Example

```
<%!  int a = 10;  %>
<%!  int b = 20;  %>
<%!  int sum;   %>
<%   sum = a+b;
     out.print(“Sum = “+sum);
%>
```

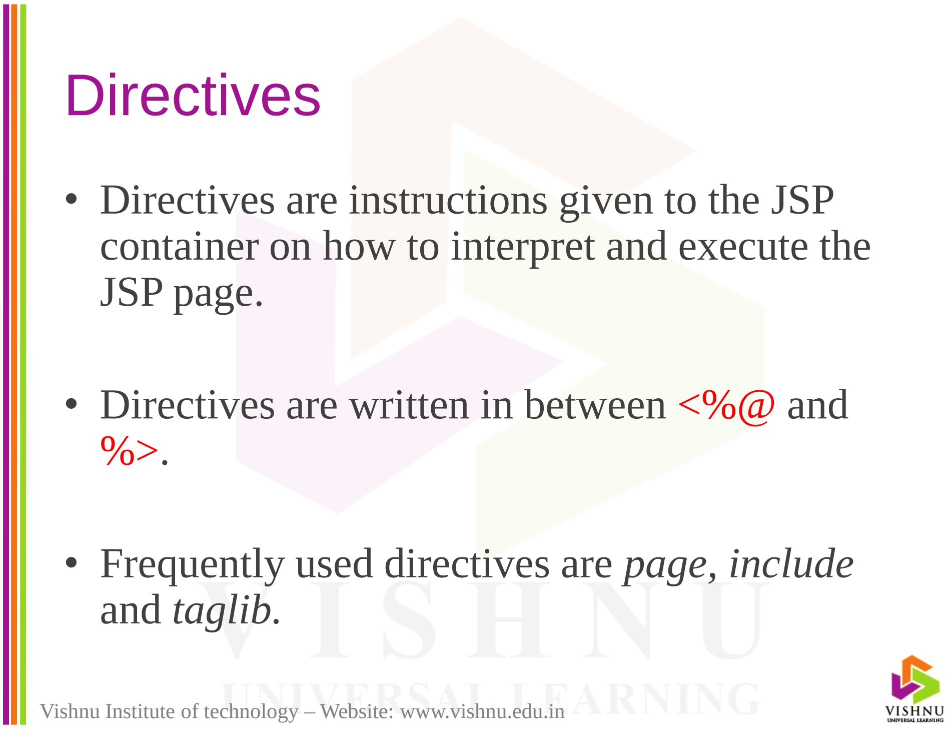
Expressions

- JSP expression construct allows the developers to display tiny amount of data in a JSP page.
- It is an alternative to using out.write() and out.print().



Expressions - Example

- Sum of 10 and 20 is: <%= 10+20 %>



Directives

- Directives are instructions given to the JSP container on how to interpret and execute the JSP page.
- Directives are written in between <%@ and %>.
- Frequently used directives are *page, include* and *taglib.*

Page Directive

Syntax of page directive:

```
<%@      page
         [ language = "java"]
         [ extends = "package.class" ]
         [ import = " package.class | package.*, …" ]
         [session = "true | false" ]
         [ buffer = "none | 8kb | size kb" ]
         [autoFlush = "true | false" ]
         isThreadSafe = "true | false" ]
         [ info = "text" ]
         [ errorPage = "relativeURL" ]
         [contentType = "MIMEType [ ;charset=characterSet ]" | "text/html
;charset=ISO-8859-1" ]
         [ isErrorPage = "true | false" ]
%>
```

Page Directive (cont…)

Attribute	Description	Example
import	Used to import the classes available in java packages. This attribute can be used multiple times	<%@ page import="java.sql.*" %>
session	Specifies whether a session object should be created for the JSP page	<%@ page session=true %>
buffer	Used to specify the size of the buffer in kilo bytes	<%@ page buffer ="10kb" %>
autoFlush	Specifies whether the buffer should be flushed automatically or not	<%@ page autoFlush="true" %>
isThreadSafe	Specifies whether the JSP page can handle concurrent requests or not	<%@ page isThreadSafe="true" %>

Page Directive (cont…)

Attribute	Description	Example
info	Used to specify description for the JSP page	<%@ page info="description…" %>
contentType	Specifies the MIME type and character set being used	<%@ page contentType="text/html;en-US" %>
errorPage	Specifies the URL for the error page to be diplayed in case of any errors	<%@ page errorPage="err.jsp" %>
isErrorPage	The error page should contain this attribute to specify that it is an error page	<%@ page isErrorPage="true" %>

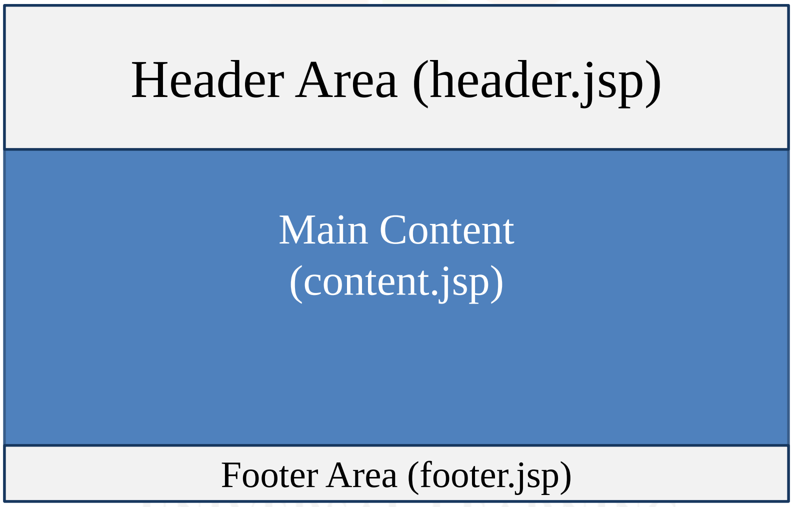
include Directive

- The *include* directive is used to insert the content of another file into the JSP page.

Example:

<%@ include file="header.html" %>

include Directive - Example



Variables, Methods and Classes

Example:

```
<%!  int a = 10;  %>
<%!
        int sum(int a, int b)
        {  return a+b; }
%>
<%  out.write( sum(10,20) );  %>
```

```
<%!
        class  Sample
        {
                int x;
        }
%>
```

Actions

- JSP actions are XML tags that can be used to implement the predefined functionality provided by JSP API.

Action Tag	Description
<jsp:include ……>	Used to include another page into the current JSP page
<jsp:param ……>	Allows to add additional parameters to the current request
<jsp:forward ….>	Transfers the control to the specified page from the current requested page
<jsp:plugin …..>	To embed applets or java beans into a JSP page
<jsp:fallback ….>	Used along with the *plugin* action tag to display error message when the plugin type is not supported

Actions (cont…)

Examples

```
<jsp:include  page="header.jsp" />

<jsp:include  page="books.jsp">
        <jsp:param  name="catid"  value="3" />
</jsp:include>                         <a  href="books.jsp?catid=3>Books</a>

<jsp:forward  page="validate.jsp" />

<jsp:plugin  type="applet"  code="Hello">
        <jsp:fallback>
                <p>Enable Java plug-in in the browser</p>
        </jsp:fallback>
</jsp:plugin>
```

Session Tracking

- Session tracking mechanisms:
  - URL rewriting
  - Hidden form fields
  - Cookies
  - Session objects

Passing Control Between JSP Pages

- To pass the control from one JSP page to another, use <*jsp:forward*> action tag.

Example

```
String  un = request.getParameter(“txtuser”);
String pw = request.getParameter(“txtpass”);
//If  valid user
<jsp:forward   page=“uhome.jsp” />
//else
<jsp:forward   page=“error.jsp” />
```

Passing Data Between JSP Pages

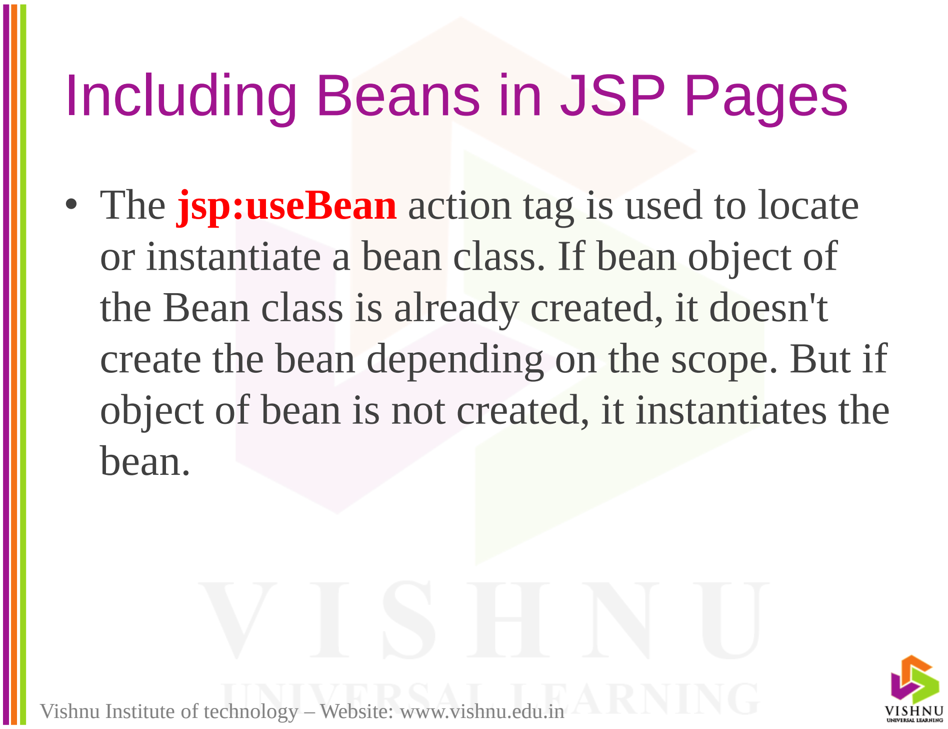
- Data can be passed or shared by using any of the four session tracking mechanisms or by using the *getParameter() or getParameterNames()* methods available on the *request* object.

Sharing Session and Application Data

- The *session* data is available to only for a single user and valid for a single session only. Usernames and passwords are an example for session data.

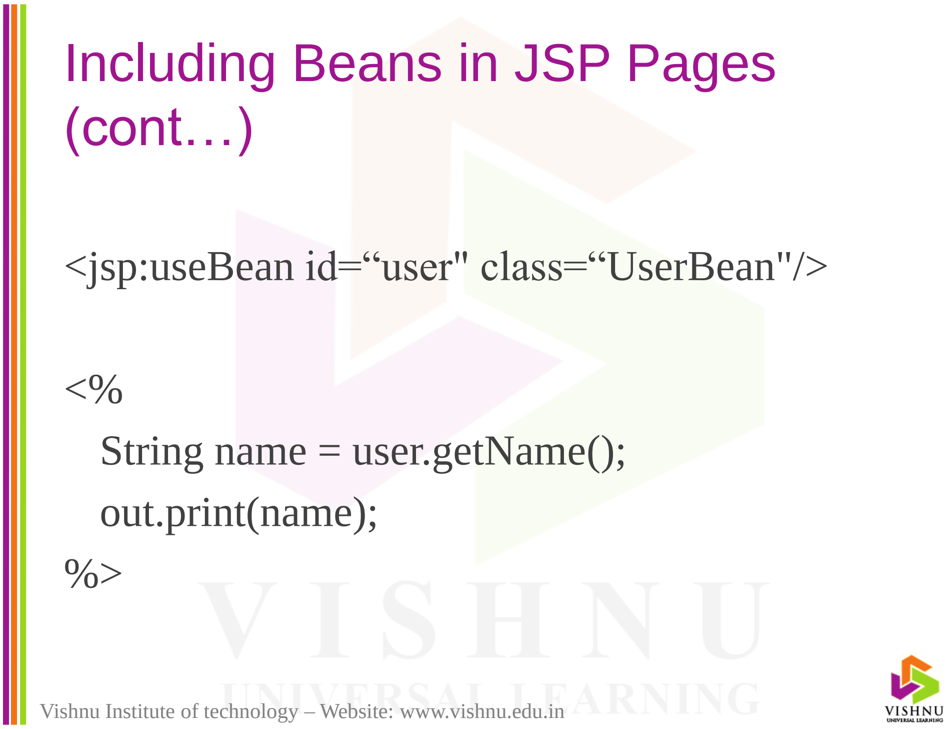
- The *application* is available for all users and all pages. Configuration information like database connection string is an example for application data.

- Use *setAttribute(name, value)* and *getAttribute(name)* for storing and accessing data from session and application objects.



Including Beans in JSP Pages

- The **jsp:useBean** action tag is used to locate or instantiate a bean class. If bean object of the Bean class is already created, it doesn't create the bean depending on the scope. But if object of bean is not created, it instantiates the bean.

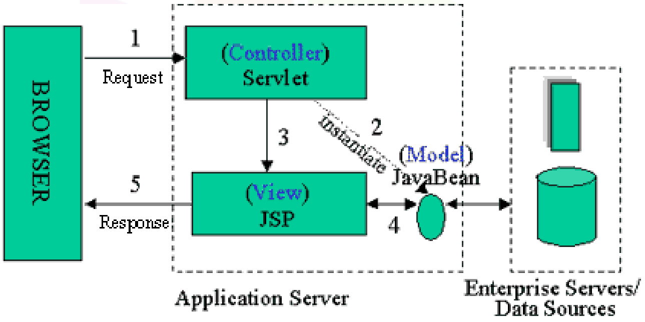


Including Beans in JSP Pages (cont…)

```
<jsp:useBean id=“user" class=“UserBean"/>

<%
   String name = user.getName();
   out.print(name);
%>
```

JSP MVC


BROWSER
1
Request
(Controller)
Servlet
2
Instantiate
3
(Model)
JavaBean
5
Response
(View)
JSP
4
Application Server
Enterprise Servers/
Data Sources