



## Enterprise Computing with Java

### MCA-305 UNIT II

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason

UII. 48

---

---

---

---

---

---

---

---



## Learning Objectives

- JSP Basics and Architecture:
- JSP directives, Scripting elements, standard actions, implicit objects, jsp design strategies.
- Struts:
- Introduction to Struts and its architecture,
- Advantages and applications of Struts.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason

UII. 48

---

---

---

---

---

---

---

---



## JSP Basics

- Today, the Java Enterprise APIs have expanded to encompass a number of areas: RMI and CORBA for remote object handling, JDBC for database interaction, JNDI for accessing naming and directory services, Enterprise JavaBeans for creating reusable business components, JMS (Java Messaging Service) for message oriented middleware, JAXP for XML processing, and JTA (Java Transaction API) for performing atomic transactions.
- In addition, J2EE also supports *servlets*, an *extremely popular* Java substitute for CGI scripts. The combination of these technologies allows programmers to create distributed business solutions for a variety of tasks.
- In late 1999, Sun Microsystems added a new element to the collection of Enterprise Java tools: JavaServer Pages (JSP) which is another Java technology for developing web applications.
- JavaServer Pages are built on top of Java servlets and are designed to increase the efficiency in which programmers, and even nonprogrammers, can create web content.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason

UII. 48

---

---

---


---

---

---

---

---



## What is JSP?

- JavaServer Pages is a technology for developing web pages that include dynamic content.
- Unlike a plain HTML page, which contains static content that always remains the same, a JSP page can change its content based on any number of variable items, including the identity of the user, the user's browser type, information provided by the user, and selections made by the user.
- A JSP page contains standard markup language elements, such as HTML tags, just like a regular web page. However, a JSP page also contains special JSP elements that allow the server to insert dynamic content in the page.
- JSP elements can be used for a variety of purposes, such as retrieving information from a database or registering user preferences.
- When a user asks for a JSP page, the server executes the JSP elements, merges the results with the static parts of the page, and sends the dynamically composed page back to the browser.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason

---

---

---


---

---

---

---

---



## Advantages of JSP Technology

- Portability
- Reuse of Components & Tag libraries
- Separation of Static and Dynamic Content
- Suitable for N-Tier Architecture
- Performance

S.No	Servlets	JSP
1.	Servlets are less efficient in generating dynamic contents.	JSP offers a clear separation of static and dynamic contents.
2.	Servlets files must be compiled by the user.	JSP files are automatically compiled.
3.	Developers need to have significant knowledge of Java.	Developers need not have significant knowledge of Java.
4.	Management of beans is done using java code	Management of beans is done using tags

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason

---

---

---


---

---

---

---

---



## Problem with Servlets

- JSP is based on the servlet technology which was introduced by Sun in 1996 as small Java-based applications for adding dynamic content to web applications. While servlets are great in many ways, they are cumbersome to program and thus generally reserved for programmers.
- However for servlets even a small change in code one needs to call the developer.
- In a servlet each HTML tag must be embedded in a String and sent using the println method of the PrintWriter object. Combining fixed or static template data with dynamic content is easier with JSP.
- In any web application, a program on the server processes requests and generates responses. In a simple one-page application, one need not be concerned about the design of this piece of code; all logic can be lumped together in a single program. However for larger web applications this is not desirable. JSP pages offer separation of business logic from view.
- JSP technology can be used as an important part in all kinds of web applications, from the simplest to the most complex.
- According to Sun, "JSP technology is an extension of servlet technology created to support authoring of HTML and XML pages."

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason

---

---

---


---

---

---

---

---



## Problems with Servlets (Contd.)

- In many Java servlet-based applications, processing the request and generating the response are both handled by a single servlet class.
- Thorough Java programming knowledge is needed to develop and maintain all aspects of the application, since the processing code and the HTML elements are lumped together.
- Changing the look and feel of the application, or adding support for a new type of client (such as a WML client), requires the servlet code to be updated and recompiled.
- It's hard to take advantage of web-page development tools when designing the application interface. If such tools are used to develop the web page layout, the generated HTML must then be manually embedded into the servlet code, a process which is time consuming, error prone, and extremely boring.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UIIL

---

---

---


---

---

---

---

---



## Problems with Servlets (Contd.)

- Adding JSP to the puzzle lets you solve these problems by separating the request processing and business logic code from the presentation.
- Instead of embedding HTML in the code, you place all static HTML in a JSP page, just as in a regular web page, and add a few JSP elements to generate the dynamic parts of the page.
- The request processing can remain the domain of the servlet, and the business logic can be handled by JavaBeans and EJB components.
- Separating the request processing and business logic from presentation makes it possible to divide the development tasks among people with different skills. The result is a much more productive development process. It also makes it possible change different aspects of the application independently, such as changing the business rules without touching the user interface.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UIIL

---

---

---


---

---

---

---

---



## Writing a JSP file

- A JSP page consists of interwoven HTML tags and Java code.
- The HTML tags represent the presentation part and the code produces the contents.
- In its most basic form, a JSP page can include only the HTML part.
- **//SimplePage.jsp**
- `<HTML>`
- `<HEAD> </HEAD>`
- `<BODY> JSP is easy. </BODY>`
- `</HTML>`
- To write Java code in your JSP file, you embed the code in `<% ... %>` tags. Ex:
- `<BODY><% out.println("JSP is easy"); %></BODY>`

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UIIL

---

---

---


---

---

---

---

---



## //DisplayServerTime.jsp

- <HTML> <HEAD>
- <TITLE>Displaying the server time</TITLE> </HEAD>
- <BODY> Welcome. The server time is now
- <% java.util.Calendar now = java.util.Calendar.getInstance();
- int hour = now.get(java.util.Calendar.HOUR\_OF\_DAY);
- int minute = now.get(java.util.Calendar.MINUTE);
- if (hour<10) out.println("0" + hour);
- else out.println(hour); out.println(":");
- if (minute<10) out.println("0" + minute);
- else out.println(minute); %>
- </BODY>
- </HTML>

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UIIL

---

---

---


---

---

---

---

---



## How JSP Works?

- Inside the JSP container is a special servlet called the page compiler. The servlet container is configured to forward to this page compiler all HTTP requests with URLs that match the .jsp file extension.
- This page compiler turns a servlet container into a JSP container. When a .jsp page is first called, the page compiler parses and compiles the .jsp page into a servlet class.
- If the compilation is successful, the .jsp servlet class is loaded into memory. On subsequent calls, the servlet class for that .jsp page is already in memory; however, it could have been updated. Therefore, the page compiler servlet will always compare the timestamp of the .jsp servlet with the .jsp page. If the .jsp page is more current, recompilation is necessary.
- With this process, once deployed, JSP pages only go through the time-consuming compilation process once.
- To save the user from the unpleasant situation of a compiling JSP page, a mechanism in JSP allows the .jsp pages to be pre-compiled before any user request for them is received. Alternatively, you can also deploy your JSP application as a web archive file in the form of a compiled servlet.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UIIL

---

---

---


---

---

---

---

---



## JSP Architecture

- JSP pages are high level extension of servlet and it enable the developers to embed java code in html pages. JSP files are finally compiled into a servlet by the JSP engine. Compiled servlet is used by the engine to serve the requests.
- *javax.servlet.jsp* package defines two interfaces:
- *JSPPage*
- *HttpJspPage*
- These interfaces defines the three methods for the compiled JSP page. These methods are:
- *jspInit()*
- *jspDestroy()*
- *\_jspService(HttpServletRequest request, HttpServletResponse response)*

In the compiled JSP file these methods are present. Programmer can define *jspInit()* and *jspDestroy()* methods, but the *\_jspService(HttpServletRequest request, HttpServletResponse response)* method is generated by the JSP engine.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UIIL

---

---

---

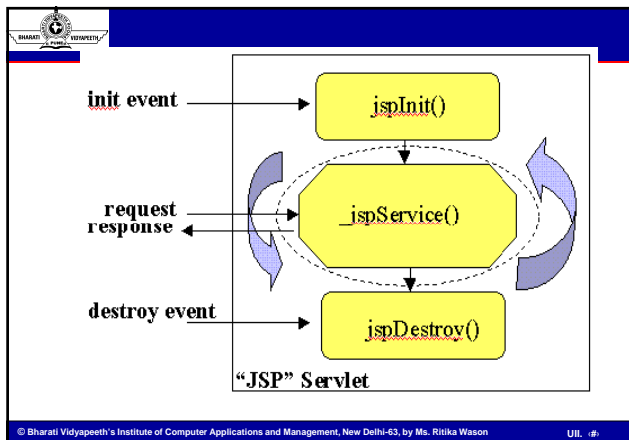
---

---

---

---

---




---

---

---

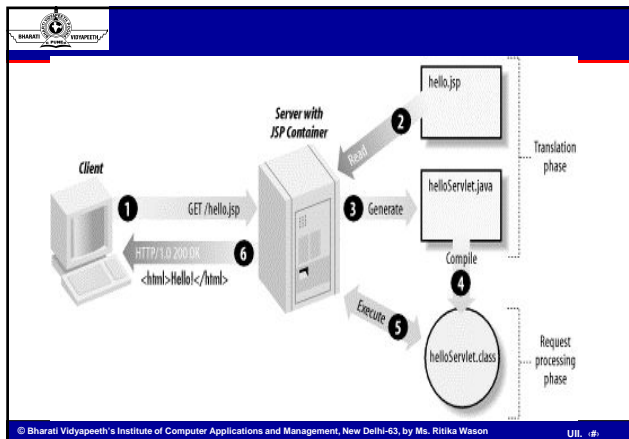
---

---

---

---

---




---

---

---

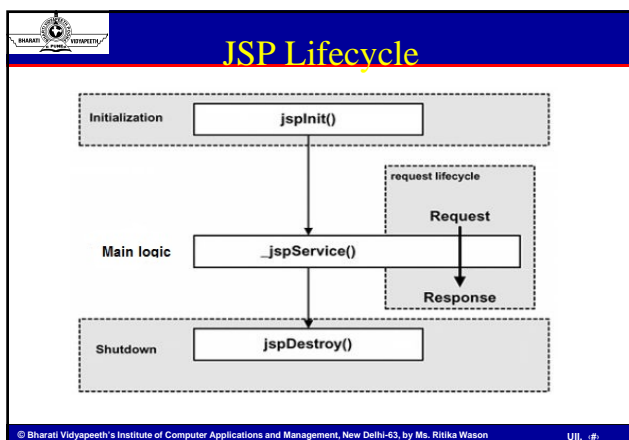
---

---

---

---

---




---

---

---


---

---

---

---

---



## JSP LifeCycle

- The key to understanding the low-level functionality of JSP is to understand the simple life cycle they follow.
- A JSP life cycle can be defined as the entire process from its creation till the destruction which is similar to a servlet life cycle with an additional step which is required to compile a JSP into servlet.
- The following are the paths followed by a JSP
  - Compilation
  - Initialization
  - Execution
  - Cleanup
- JSP Compilation: When a browser asks for a JSP, the JSP engine first checks to see whether it needs to compile the page. If the page has never been compiled, or if the JSP has been modified since it was last compiled, the JSP engine compiles the page.
- The compilation process involves three steps: Parsing the JSP; Turning the JSP into a servlet; Compiling the servlet.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason

UII

---

---

---


---

---

---

---

---



## JSP Lifecycle (Contd.)

- **JSP Initialization:** When a container loads a JSP it invokes the `jspInit()` method before servicing any requests.
- `public void jspInit() { // Initialization code... } Initialization is performed only once.`
- **JSP Execution:** This phase represents all interactions with requests until the JSP is destroyed. Whenever a JSP page is requested, loaded and initialized, the JSP engine invokes the `_jspService()` method in the JSP.
- `void _jspService(HttpServletRequest request, HttpServletResponse response) { // Service handling code... } The _jspService() method of a JSP is invoked once per a request and is responsible for generating the response for that request and this method is also responsible for generating responses to all seven of the HTTP methods ie. GET, POST, DELETE etc.`
- **JSP Cleanup:** The `jspDestroy()` method is the JSP equivalent of the destroy method for servlets. Override `jspDestroy` when you need to perform any cleanup, such as releasing database connections or closing open files.
- `public void jspDestroy() { // Your cleanup code goes here. }`

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason

UII

---

---

---


---

---

---

---

---



## The JSP API

- The JSP technology is based on the JSP API that consists of two packages: `javax.servlet.jsp` and `javax.servlet.jsp.tagext`.
- In addition to these two packages, JSP also needs the two servlet packages—`javax.servlet` and `javax.servlet.http`.
- The `javax.servlet.jsp` package has two interfaces and four classes. The interfaces are as follows:
  - `JspPage`
  - `HttpJspPage`
- The four classes are as follows:
  - `JspEngineInfo`
  - `JspFactory`
  - `JspWriter`
  - `PageContext`
- In addition, there are also two exception classes: `JspException` and `JspError`.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason

UII

---

---

---


---

---

---

---

---



## The JspPage Interface

- The JspPage is the interface that must be implemented by all JSP servlet classes. The JspPage interface does extend the javax.servlet.Servlet interface.
- The JspPage interface has two methods, JspInit() and JspDestroy(), whose signatures are as follows:
  - public void jspInit()
  - public void jspDestroy()
- jspInit, which is similar to the init method in the javax.servlet.Servlet interface, is called when the JspPage object is created and can be used to run some initialization. This method is called only once during the life cycle of the JSP page: the first time the JSP page is invoked.
- The jspDestroy method is analogous with the destroy method of the javax.servlet.Servlet interface. This method is called before the JSP servlet object is destroyed. You can use this method to do some clean-up, if you want.
- Most of the time, however, JSP authors rarely make full use of these two methods.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UII

---

---

---


---

---

---

---

---



## Example

- <%! public void jspInit()
- { System.out.println("Init"); }
- public void jspDestroy() { System.out.println("Destroy"); } %>
- The HttpJspPage Interface**
- This interface directly extends the JspPage interface.
- There is only one method: \_jspService. This method is called by the JSP container to generate the content of the JSP page. The \_jspService has the following signature:
  - public void \_jspService(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UII

---

---

---


---

---

---

---

---



## The JspFactory Class

- The JspFactory class is an abstract class that provides methods for obtaining other objects needed for the JSP page processing. The class has the static method getDefaultFactory() that returns a JspFactory object. From the JspFactory object, a PageContext and a JspEngineInfo object can be obtained that are useful for the JSP page processing. These objects are obtained using the JspFactory class's getEngineInfo() method and the getPageContext method(), whose signatures are given here:
  - public abstract JspEngineInfo getEngineInfo()
  - public abstract PageContext getPageContext ( Servlet requestingServlet, ServletRequest request, ServletResponse response, String errorPageURL, boolean needsSession, int buffer, boolean autoFlush)

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UII

---

---

---


---

---

---

---

---



## The JspEngineInfo Class

- The JspEngineInfo class is an abstract class that provides information on the JSP container.
- Only one method, getSpecificationVersion(), returns the JSP container's version number. Because this is the only method currently available, this class does not have much use.
- You can obtain a JspEngineInfo object using the getEngineInfo() method of the JspFactory class.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UIIL

---

---

---


---

---

---

---

---



## The PageContext Class

- PageContext represents a class that provides methods that are implementation-dependent.
- The PageContext class itself is abstract, so in the \_jspService method of a JSP servlet class, a PageContext object is obtained by calling the getPageContext() method of the JspFactory class.
- The PageContext class provides methods that are used to create other objects. For example, its getOut() method returns a JspWriter object that is used to send strings to the web browser. Other methods that return servlet-related objects include the following:
  - getRequest(), returns a ServletRequest object
  - getResponse(), returns a ServletResponse object
  - getServletConfig(), returns a ServletConfig object
  - getServletContext(), returns a ServletContext object
  - getSession(), returns an HttpSession object

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UIIL

---

---

---


---

---

---

---

---



## The JspWriter Class

- The JspWriter class is derived from the java.io.Writer class and represents a Writer that you can use to write to the client browser. Of its many methods, the most important are the print and println methods. Both provide enough overloads that ensure you can write any type of data.
- Additional methods allow you to manipulate the buffer. For instance, the clear method clears the buffer. It throws an exception if some of the buffer's content has already been flushed. Similar to clear is the clearBuffer method, which clears the buffer but never throws any exception if any of the buffer's contents have been flushed.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UIIL

---

---

---

---


---

---

---

---





## JSP Syntax

- A JSP page can have Java code and HTML tags. More formally, a JSP page has elements and template data. The elements, also called JSP tags, make up the syntax and semantics of JSP. Template data is everything else. Template data includes parts that the JSP container does not understand, such as HTML tags.
- There are three types of elements: Directive elements, Scripting elements, Action elements.
- Elements have two forms: the XML form and the `<% ... %>` alternative form.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UJIL

---

---

---


---

---

---

---

---



## JSP Directives

- JSP directives provide directions and instructions to the container, telling it how to handle certain aspects of JSP processing.
- A JSP directive affects the overall structure of the servlet class. It usually has the following form:
  - `<% @ directive attribute="value" %>`
- 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.
- There are three types of directive tag:

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

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UJIL

---

---

---


---

---

---

---

---



## The page Directive

- The **page** directive is used to provide instructions to the container that 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.
- Following is the basic syntax of page directive:
  - `<% @ page attribute="value" %>`
- You can write XML equivalent of the above syntax as follows:
  - `<jsp:directive.page attribute="value" />`
- Following list of attributes associated with the page directive:

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UJIL

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## The include Directive

- 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 *include* directives anywhere in your JSP page.
- The general usage form of this directive is as follows:
- `<%@ 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.
- You can write XML equivalent of the above syntax as follows:
- `<jsp:directive.include file="relative url" />`

---


---

---

---

---

---



# The taglib Directive

- The JavaServer Pages API allows 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 a means for identifying the custom tags in your JSP page.
- The taglib directive follows the following syntax:
  - `<% @ taglib uri="uri" prefix="prefixOfTag" %>`
  - Where 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.
- You can write XML equivalent of the above syntax as follows:
  - `<jsp:directive.taglib uri="uri" prefix="prefixOfTag" />`

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason

UII

---


---

---

---

---

---



## JSP Scripting Elements

- JSP scripting elements let you insert code into the servlet that will be generated from the JSP page. There are three forms:
  1. *Expressions of the form `<%= expression %>`, which are evaluated and inserted into the servlet's output*
  2. *Scriptlets of the form `<% code %>`, which are inserted into the servlet's `_jspService` method (called by service)*
  3. *Declarations of the form `<%! code %>`, which are inserted into the body of the servlet class, outside of any existing methods.*

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason
UII1

---

---

---


---

---

---

---

---



## Scriptlets

- Scriptlets are the code blocks of a JSP page. Scriptlets start with an opening `<%` tag and end with a closing `%>` tag. Ex:
- `<% @ page session="false" %> <% @ page import="java.sql.*" %>`
- `<% try { Class.forName("sun.jdbc.odbc.JdbcOdbcDriver"); System.out.println("JDBC driver loaded"); } catch (ClassNotFoundException e) { System.out.println(e.toString()); } %>`
- `<HTML> <HEAD> <TITLE>Display All Users</TITLE> </HEAD> <BODY> <CENTER> <BR><H2>Displaying All Users</H2> <BR> <TABLE> <TR> <TH>First Name</TH> <TH>Last Name</TH> <TH>User Name</TH> <TH>Password</TH> </TR> <% String sql = "SELECT FirstName, LastName, UserName, Password" + " FROM Users"; try { Connection con = DriverManager.getConnection("jdbc:odbc:JavaWeb"); Statement s = con.createStatement(); ResultSet rs = s.executeQuery(sql); while (rs.next()) { %> <TR> <TD><% out.print(rs.getString(1)); %></TD> <TD><% out.print(rs.getString(2)); %></TD> <TD><% out.print(rs.getString(3)); %></TD> <TD><% out.print(rs.getString(4)); %></TD> </TR> <% } rs.close(); s.close(); con.close(); } catch (SQLException e) { } %> </TABLE> </CENTER> </BODY> </HTML>`

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason
UII1

---

---

---


---

---

---

---

---



## Declarations

- Declarations allow you to declare methods and variables that can be used from any point in the JSP page.
- Declarations also provide a way to create initialization and clean-up code by utilizing the `jspInit()` and `jspDestroy()` methods.
- A declaration starts with a `<%!` and ends with a `%>` and can appear anywhere throughout the page. For example, a method declaration can appear above a page directive that imports a class, even though the class is used in the method.
- `<%! String getSystemTime() { return Calendar.getInstance().getTime().toString(); } %>`
- `<% @ page import="java.util.Calendar" %>`
- `<% @ page session="false" %>`
- `<% out.println("Current Time: " + getSystemTime()); %>`
- `<%! int i; %>`

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason
UII1

---

---

---


---

---

---

---

---



## Expressions

- Expressions are the last type of JSP scripting elements. Expressions get evaluated when the JSP page is requested and their results are converted into a String and fed to the print method of the out implicit object. If the result cannot be converted into a String, an error will be raised at translation time. If this is not detected at translation time, at request-processing time, a ClassCastException will be raised.
- An expression starts with a `<%=` and ends with a `%>`. You don't add a semicolon at the end of an expression.
- Current Time: `<%= java.util.Calendar.getInstance().getTime() %>`
- This expression will be translated into the following statement in the `_jspService` method of the generated servlet:
- `out.print( java.util.Calendar.getInstance().getTime() );`

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UII

---

---

---


---

---

---

---

---



## JSP Standard Action Elements

- Standard action elements basically are tags that can be embedded into a JSP page. At compile time, they also are replaced by Java code that corresponds to the predefined task.
- The JSP standard action elements are:
- `jsp:useBean` ; `jsp:setProperty` ; `jsp:getProperty` ; `jsp:param` ; `jsp:include` ; `jsp:forward` ; `jsp:plugin` ; `jsp:params` ; `jsp:fallback`
- The `jsp:useBean`, `jsp:setProperty`, and `jsp:getProperty` elements are related to Bean.
- The `jsp:param` element is used in the `jsp:include`, `jsp:forward`, and `jsp:plugin` elements to provide information in the name/value format.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UII

---

---

---


---

---

---

---

---



## jsp:include

- The `jsp:include` action element is used to incorporate static or dynamic resources into the current page. This action element is similar to the `include` directive, but `jsp:include` provides greater flexibility because you can pass information to the included resource.
- The syntax for the `jsp:include` action element has two forms. For the `jsp:include` element that does not have a parameter name/value pair, the syntax is as follows: `<jsp:include page="relativeURL" flush="true"/>`
- If you want to pass information to the included resource, use the second syntax:
- `<jsp:include page="relativeURL" flush="true"> ( <jsp:param ... /> ) * </jsp:include>`
- The `page` attribute represents the URL of the included resource in the local server. The `flush` attribute indicates whether the buffer is flushed. In JSP 1.2, the value of the `flush` attribute must be true. In the second form, the `*` indicates that there can be zero or more elements in the brackets. This means that you also can use this form even though you are not passing any information to the included resource.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UII

---

---

---


---

---

---

---

---



## jsp:forward

- The `jsp:forward` action element is used to terminate the execution of the current JSP page and switch control to another resource. You can forward control either to a static resource or a dynamic resource.
- The syntax for the `jsp:forward` action element has two forms. For the `jsp:forward` element that does not have a parameter name/value pair, the syntax is as follows:
- `<jsp:forward page="relativeURL"/>` If you want to pass information to the included resource, use the second syntax:
- `<jsp:forward page="relativeURL"> ( <jsp:param . . . /> )*`  
`</jsp:include>` The page attribute represents the URL of the included resource in the local server.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UII1

---

---

---


---

---

---

---

---



## jsp:plugin, jsp:params and jsp:fallback

- jsp:plugin**
- The `jsp:plugin` action element is used to generate HTML `<OBJECT>` or `<EMBED>` tags containing appropriate construct to instruct the browser to download the Java Plugin software, if required, and initiates the execution of a Java applet or a JavaBeans component specified. This is beyond the scope of this book and won't be discussed further.
- jsp:params**
- The `jsp:params` action element can occur only as part of the `<jsp:plugin>` action and will not be discussed in this book.
- jsp:fallback**
- The `jsp:fallback` action element can occur only as part of the `<jsp:plugin>` action

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UII1

---

---

---


---

---

---

---

---



## Comments

- Two types of comments are allowed inside a JSP page:
- Comments that are to be displayed in the resulting HTML page at the client browser
- Comments used in the JSP page itself
- For comments that are meant to be displayed in the HTML page, you use the comments tags in HTML. This kind of comment must be sent as normal text in a JSP page. For example, the following code sends an HTML comment to the browser:
- `<% out.println("<!-- Here is a comment -->"); %>`
- This is equivalent to the following:
- `<!-- Here is a comment -->` For comments in the JSP page itself, you use the `<% ... --%>` tag pair. For example, here is a JSP comment: `<%-- Here is a comment --%>` A JSP comment can contain anything except the closing tag. For example, this is an illegal comment: `<%-- Here is a comment --%> --%>`

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UII1

---

---

---


---

---

---

---

---



## Converting into XML syntax

- JSP pages can also be represented using XML, and representing a JSP page as an XML document presents the following benefits:
  - The content of the JSP page can be validated against a set of descriptions.
  - The JSP page can be manipulated using an XML tool.
  - The JSP page can be generated from a textual representation by applying an XML transformation.
- Directives**
  - The non-XML syntax for a JSP directive takes the following form:
    - `<% @ directive (attribute="value") * %>`
  - The XML syntax for a directive is as follows:
    - `<jsp:directive:directiveName attribute_list />`

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UII14

---

---

---

---

---


---

---

---

---

---



## Converting into XML syntax (Contd.)

- Scripting Elements**
  - A scripting element can be one of the following: declaration, scriptlet, or expression. The XML syntax for each of the three is:
- Declarations**
  - The alternative syntax for a declaration is as follows:
    - `<% ! declaration code %>` This is equivalent to the following XML syntax:
      - `<jsp:declaration> declaration code </jsp:declaration>`
- Scriptlets**
  - The alternative syntax for a scriptlet is as follows:
    - `<% scriptlet code %>` This is equivalent to the following XML syntax:
      - `<jsp:scriptlet> scriptlet code </jsp:scriptlet>`

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UII14

---

---

---

---

---


---

---

---

---

---



## Converting into XML syntax (Contd.)

- Expressions**
  - The alternative syntax for an expression is as follows:
    - `<%= expression %>` This is equivalent to the following XML syntax:
      - `<jsp:expression> expression </jsp:expression>`
  - Template Data**
    - The `<jsp:text>` XML tag is used to enclose template data in JSP. The syntax is as follows:
      - `<jsp:text> text </jsp:text>`

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UII14

---

---

---

---

---


---

---

---

---

---



## Introduction

- Implicit objects in jsp are the objects that are created by the container automatically and the container makes them available to the developers, the developer do not need to create them explicitly.
- Since these objects are created automatically by the container and are accessed using standard variables; hence, they are called implicit objects.
- The implicit objects are parsed by the container and inserted into the generated servlet code.
- They are available only within the jspService method and not in any declaration.
- Implicit objects are used for different purposes. Our own methods (user defined methods) can't access them as they are local to the service method and are created at the conversion time of a jsp into a servlet.
- But we can pass them to our own method if we wish to use them locally in those functions.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UIIL

---

---

---

---

---


---

---

---

---

---



## Implicit Objects

- There are a total of nine implicit objects, which are listed as follows:

S.No	Object	Class
1	application	javax.servlet.ServletContext
2	config	javax.servlet.ServletCofig
3	exception	java.lang.Throwable
4	out	javax.servlet.jsp.JspWriter
5	page	java.lang.Object
6	PageContext	javax.servlet.jsp.PageContext
7	request	javax.servlet.ServletRequest
8	response	javax.servlet.ServletResponse
9	session	javax.servlet.http.HttpSession

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UIIL

---

---

---

---

---


---

---

---

---

---



## Implicit Objects

- Application:** These objects has an application scope. These objects are available at the widest context level, that allows to share the same information between the JSP page's servlet and any Web components with in the same application.
- Config:** These object has a page scope and is an instance of javax.servlet.ServletConfig class. Config object allows to pass the initialization data to a JSP page's servlet. Parameters of this objects can be set in the deployment descriptor (web.xml) inside the element <jsp-file>. The method getInitParameter() is used to access the initialization parameters.
- Exception:** This object has a page scope and is an instance of java.lang.Throwable class. This object allows the exception data to be accessed only by designated JSP "error pages."

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UIIL

---

---

---

---

---


---

---

---

---

---



## Implicit Objects

- **Out:** This object allows us to access the servlet's output stream and has a page scope. Out object is an instance of `javax.servlet.jsp.JspWriter` class. It provides the output stream that enable access to the servlet's output stream.
- **Page:** This object has a page scope and is an instance of the JSP page's servlet class that processes the current request. Page object represents the current page that is used to call the methods defined by the translated servlet class. First type cast the servlet before accessing any method of the servlet through the page.
- **Pagecontext:** PageContext has a page scope. Pagecontext is the context for the JSP page itself that provides a single API to manage the various scoped attributes. This API is extensively used if we are implementing JSP custom tag handlers. PageContext also provides access to several page attributes like including some static or dynamic resource.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UIIL

---

---

---


---

---

---

---

---



## Implicit Objects

- **Request:** Request object has a request scope that is used to access the HTTP request data, and also provides a context to associate the request-specific data. Request object implements `javax.servlet.ServletRequest` interface. It uses the `getParameter()` method to access the request parameter. The container passes this object to the `_jspService()` method.
- **Response:** This object has a page scope that allows direct access to the **HTTPServletResponse** class object. Response object is an instance of the classes that implements the `javax.servlet.ServletResponse` class. Container generates to this object and passes to the `_jspService()` method as a parameter. **Session:** Session object has a session scope that is an instance of `javax.servlet.http.HttpSession` class. Perhaps it is the most commonly used object to manage the state contexts. This object persist information across multiple user connection.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UIIL

---

---

---


---

---

---

---

---



## Design Strategies

- Many dozens of design strategies have been developed for web applications
- We introduce two here :
  1. Page-centric : Requests are made to JSPs, and the JSPs respond to clients
  2. Dispatcher : Requests are sent to JSPs or servlets that then forward the requests to another JSP or servlet
- In both cases, the goal is to separate logic from presentation and to separate as many concerns in the logic as possible

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UIIL

---

---

---

---

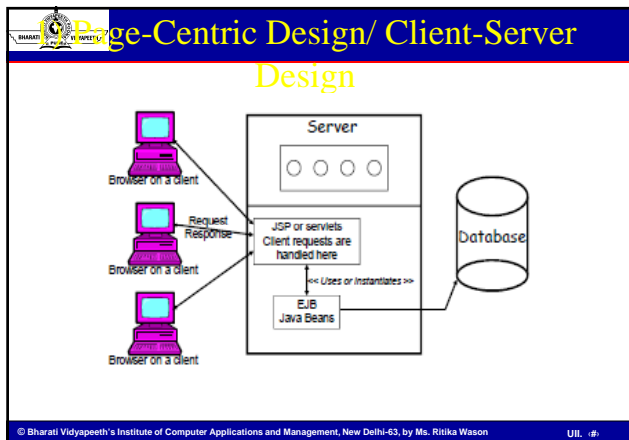
---

---

---

---






---

---

---

---

---

---

---

---

### Page-Centric Design (Contd.)

- This is a simple design to implement
- The JSP author can generate pages easily
- Two variants :
  - *Page-View*
  - *Page-View with a Bean*
- Does not scale up very well to large web sites
- Often results in a lot of Java code in the – JSP authors must be Java programmers
- Design is hard to see
- Hard to maintain

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason

---

---

---

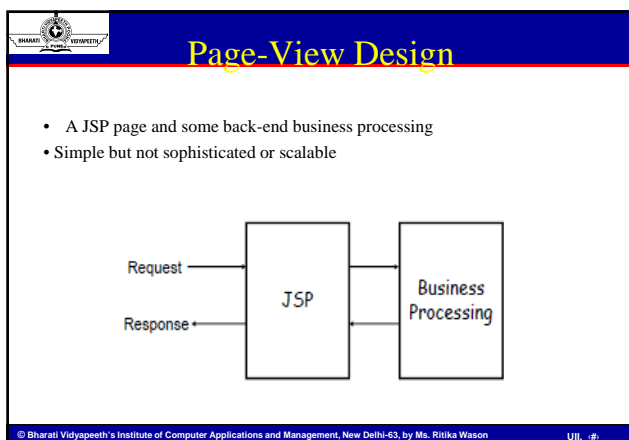
---

---

---

---

---




---

---

---

---

---

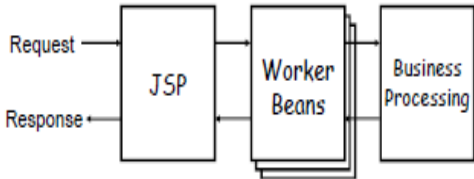
---

---

---

### Page-View with Bean Design

- When more business processing and data access is needed
- Let Java Beans handle the data
- Separates programming (Bean) from presentation (JSP)



The diagram illustrates the Page-View with Bean Design. A 'Request' arrow enters a box labeled 'JSP'. From the 'JSP' box, an arrow points to a box labeled 'Worker Beans'. From the 'Worker Beans' box, an arrow points to a box labeled 'Business Processing'. A 'Response' arrow exits the 'JSP' box.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UII18

---

---

---

---

---

---

---

---

### Dispatcher Design

- A “dispatcher” accepts requests and routes them to the correct place
- Front-end JSP (or servlet) looks at a portion of the request, and then chooses the correct place to forward it
- This is more sophisticated than the page-centric :
  - More flexible and scalable
  - More overhead that is wasted with small applications
- Three versions
  - Mediator-View
  - Mediator-Composite View
  - Service to Workers

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UII18

---

---

---

---

---

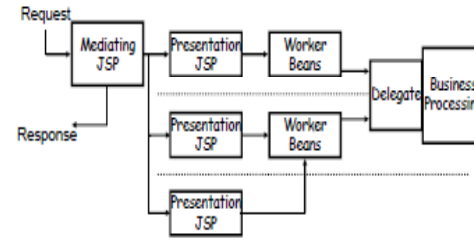
---

---

---

### Mediator-View Design

- The Mediating JSP sends requests to a JSP
- The JSP sets and gets beans and creates a response page



The diagram illustrates the Mediator-View Design. A 'Request' arrow enters a box labeled 'Mediating JSP'. From the 'Mediating JSP' box, an arrow points to a box labeled 'Presentation JSP'. From the 'Presentation JSP' box, an arrow points to a box labeled 'Worker Beans'. From the 'Worker Beans' box, an arrow points to a box labeled 'Business Processing'. A 'Response' arrow exits the 'Mediating JSP' box. Below the 'Presentation JSP' box, there are two more boxes labeled 'Presentation JSP' and 'Worker Beans', connected by a dashed line, indicating a parallel structure.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UII18

---

---

---

---

---

---

---

---

**Mediator-Composite Design**

- The mediator-view helps with dynamically changing data, but sometimes the structure around the data changes
- Consider a web site that sells books, CDs, and videos
  - The title, author and price form the data
  - The structure will be different with each type of item
- With a mediator-view, the JSP has to be changed to introduce new types of products
- The mediator-composite architecture allows nesting of mediator JSPs and HTML ...

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason ULL

---

---

---

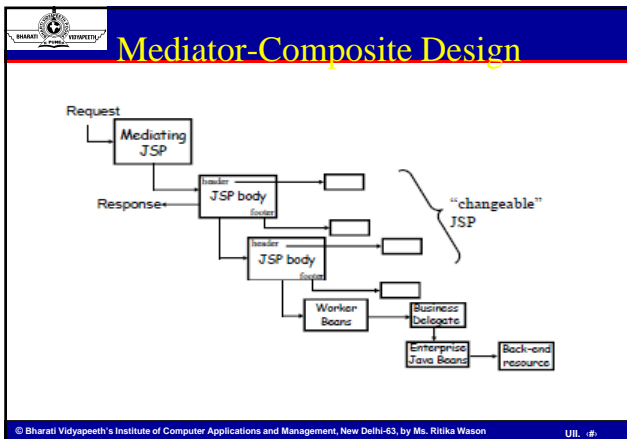
---

---

---

---

---




---

---

---

---

---

---

---

---

**Mediator-Composite Design**

- In this architecture, the JSP makes direct calls to the business logic modules
- This creates a tight coupling between the JSP and the execution or business logic
- In the next architecture, the is separated further from the business logic modules

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason ULL

---

---

---

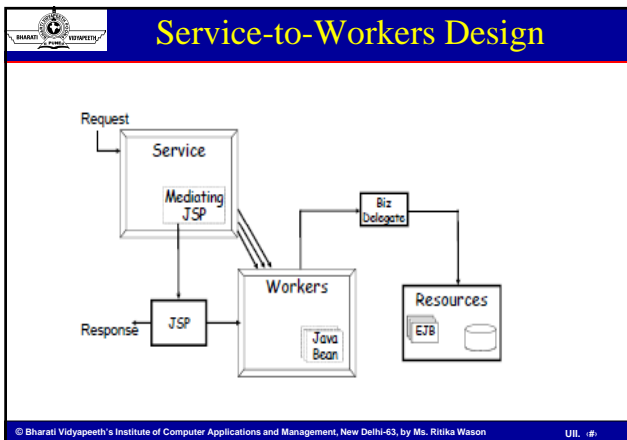
---

---

---

---

---




---

---

---

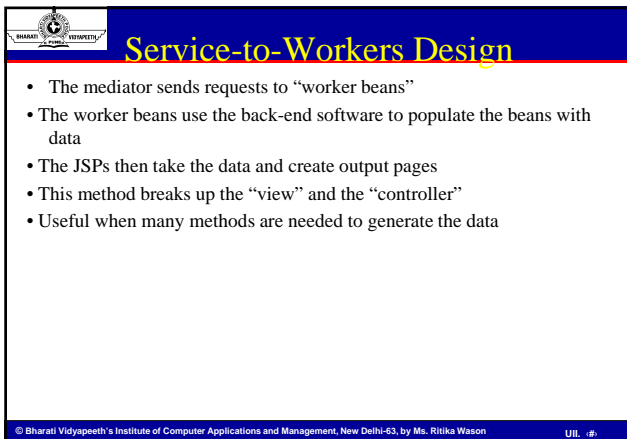
---

---

---

---

---




---

---

---

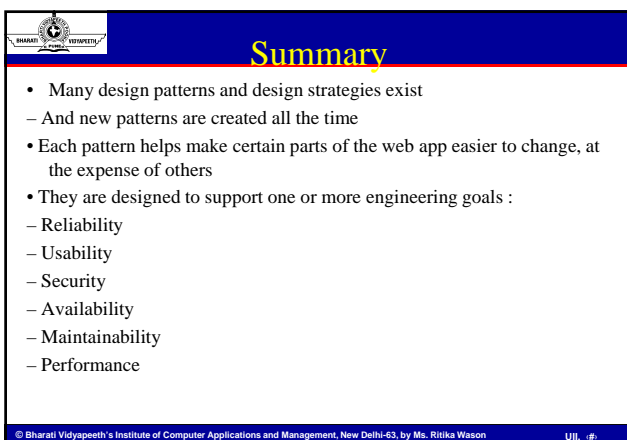
---

---

---

---

---




---

---

---


---

---

---

---

---



## Framework

- A framework is simply defined as a structure that you can build upon
- When you start a new application, a framework allows you to not futz with the under-workings every application needs. It gives you a head start.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UII

---

---

---


---

---

---

---

---



## Struts: Introduction

- Struts is a framework that promotes the use of the Model-View-Controller architecture for designing large scale applications.
- The framework includes a set of custom tag libraries and their associated Java classes, along with various utility classes.
- The most powerful aspect of the Struts framework is its support for creating and processing web-based forms.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UII

---

---

---


---

---

---

---

---



## History

- When Java servlets were first invented, many programmers quickly realized that they were a Good Thing. They were faster and more powerful than standard CGI, portable, and infinitely extensible.
- But writing HTML to send to the browser in endless `println()` statements was tiresome and problematic. The answer to that was Java Server Pages, which turned Servlet writing inside-out. Now developers could easily mix HTML with Java code, and have all the advantages of servlets. The sky was the limit!
- Java web applications quickly became "JSP-centric". This in-and-of itself was not a Bad Thing, but it did little to resolve flow control issues and other problems endemic to web applications.
- Clearly, another paradigm was needed ...

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UII

---

---

---


---

---

---

---

---



## History

- Many clever developers realized that JavaServer Pages AND servlets could be used **together** to deploy web applications. The servlets could help with the control-flow, and the JSPs could focus on the nasty business of writing HTML. In due course, using JSPs and servlets together became known as Model 2 (meaning, presumably, that using JSPs alone was Model 1).
- Of course, there is nothing new under the Sun ... and many have been quick to point out that JSP's Model 2 follows the classic Model-View-Controller design pattern abstracted from the Smalltalk MVC framework. Java Web developers now tend to use the terms Model 2 and MVC interchangeably. In this guide, we use the MVC paradigm to describe the framework architecture, which might be best termed a Model 2/MVC design.
- The Apache Struts Project was launched in May 2000 by Craig R. McClanahan to provide a standard MVC framework to the Java community. In July 2001, version 1.0 was released, and IOHO, Java Model 2 development has never been quite the same.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UII

---

---

---


---

---

---

---

---



## What are Struts?

- According to the Struts main page: Apache Struts is a free open-source framework for creating Java web applications.
- According to Wikipedia: **Apache Struts is an** open-source framework for developing Java EE web applications. It uses and extends the Java Servlet API to encourage developers to adopt a model-view-controller (MVC) architecture.
- Struts is a framework that builds upon the framework provided by J2EE and gives you a starting point to build a web application.
- It allows you to worry less about the mundane mechanics (how requests get handled, how do requests get routed)
- It makes it a little more difficult to deviate from the MVC pattern

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UII

---

---

---


---

---

---

---

---



## What is the Struts Framework?

- The Struts Framework is a standard for developing well-architected Web applications. It has the following features:
- Open source
- Based on the Model-View-Controller (MVC) design paradigm, distinctly separating all three levels:
  - Model:** application state
  - View:** presentation of data (JSP, HTML)
  - Controller:** routing of the application flow
- Implements the JSP Model 2 Architecture
- Stores application routing information and request mapping in a single core file, struts-config.xml
- The Struts Framework, itself, only fills in the View and Controller layers. The Model layer is left to the developer.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UII

---

---

---

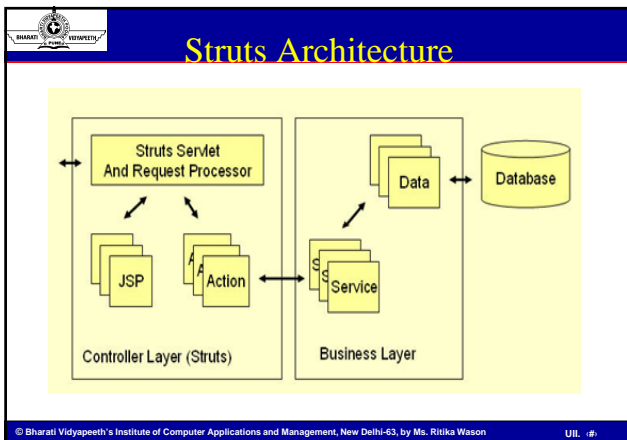
---

---

---

---

---




---

---

---

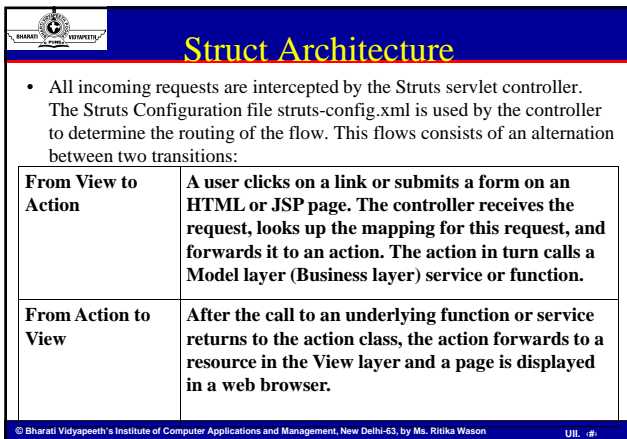
---

---

---

---

---




---

---

---

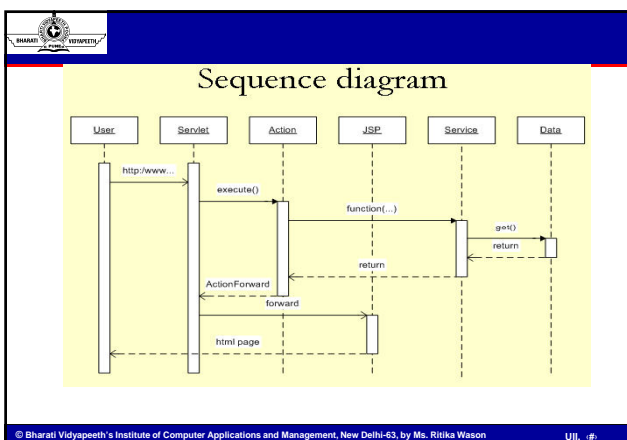
---

---

---

---

---




---

---

---


---

---

---

---

---



## Components of Struts

**Struts consist of four main components:**

- 1. The Struts Framework:** Java classes that provide the *ActionServlet* and *Action* classes as part of the MVC Controller, the *ActionForm* class to facilitate moving data to and from the View, and the *ActionForward* class to aid the controller in forwarding requests.
- 2. JSP Tag Libraries:** Tags expand on HTML forms and fields, help you work with beans and provide other useful features.
- 3. Tiles Plugin:** Allows you to create HTML in re-usable pieces (tiles) that can be put together to make a whole page
- 4. Validator Plugin:** Allows you specify validation information in an xml file and then perform those validations in the browser and/or the on the server.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UIIL

---

---

---


---

---

---

---

---



## Struts Components

- **The Controller**
- This receives all incoming requests. Its primary function is the mapping of a request URI to an action class selecting the proper application module. It's provided by the framework.
- **The struts-config.xml File**
- This file contains all of the routing and configuration information for the Struts application. This XML file needs to be in the WEB-INF directory of the application.
- **Action Classes**
- It's the developer's responsibility to create these classes. They act as bridges between user-invoked URIs and business services. Actions process a request and return an *ActionForward* object that identifies the next component to invoke. They're part of the Controller layer, not the Model layer.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UIIL

---

---

---


---

---

---

---

---



## Struts Components

- **View Resources**
- View resources consist of Java Server Pages, HTML pages, JavaScript and Stylesheet files, Resource bundles, JavaBeans, and Struts JSP tags.
- **ActionForms**
- These greatly simplify user form validation by capturing user data from the HTTP request. They act as a "firewall" between forms (Web pages) and the application (actions). These components allow the validation of user input before proceeding to an Action. If the input is invalid, a page with an error can be displayed.
- **Model Components**
- The Struts Framework has no built-in support for the Model layer. Struts supports any model components:
- JavaBeans; EJB
- CORBA; JDO any other

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UIIL

---

---

---

---


---

---

---

---





## Struts Advantages

- Struts has fundamentally reshaped the way that Web programmers think about and structure a Web application. It is a technology that no Web programmer can afford to ignore.
- Struts reduce the development maintenance time of enterprise web applications. Struts is open-source.
- **Centralized file-based configuration**
  - Rather than hard-coding information into Java programs, many Struts values are represented in XML or property files. This loose coupling means that many changes can be made without modifying or recompiling Java code, and that wholesale changes can be made by editing a single file. This approach also lets Java and Web developers focus on their specific tasks (implementing business logic, presenting certain values to clients, etc.) without needing to know about the overall system layout.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UII

---

---

---


---

---

---

---

---



## Struts Advantages

### Form beans

- In JSP, you can use `property="*" with jsp:setProperty to automatically populate a JavaBean component based on incoming request parameters. Apache Struts extends this capability to Java code and adds in several useful utilities, all of which serve to greatly simplify the processing of request parameters.`

### • Bean tags

- Apache Struts provides a set of custom JSP tags (bean:write, in particular) that let you easily output the properties of JavaBeans components. Basically, these are concise and powerful variations of 7 the standard `jsp:useBean` and `jsp:getProperty` tags.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UII

---

---

---


---

---

---

---

---



## Struts Advantages

- **HTML tags**
  - Apache Struts provides a set of custom JSP tags to create HTML forms that are associated with JavaBeans components. This bean/form association serves two useful purposes:
    - It lets you get initial form-field values from Java objects.
    - It lets you redisplay forms with some or all previously entered values intact.
- **Form field validation**
  - Apache Struts has builtin capabilities for checking that form values are in the required format. If values are missing or in an improper format, the form can be automatically redisplayed with error messages and with the previously entered values maintained.
- This validation can be performed on the server (in Java), or both on the server and on the client (in JavaScript).
- **Consistent approach**
  - Struts encourages consistent use of MVC throughout your app.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UII

---

---

---


---

---

---

---

---



## Struts Disadvantages

- **Bigger learning curve**
  - To use MVC with the standard RequestDispatcher, you need to be comfortable with the standard JSP and servlet APIs. To use MVC with Struts, you have to be comfortable with the standard JSP and servlet APIs and a large and elaborate framework that is almost equal in size to the core system. This drawback is especially significant with smaller projects, near-term deadlines, and less experienced developers; you could spend as much time learning Struts as building your actual system.
- **Worse documentation**
  - Compared to the standard servlet and JSP APIs, Struts has fewer online resources, and many first-time users find the online Apache documentation confusing and poorly organized. There are also fewer books on Apache Struts than on standard servlets and JSP.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UII

---

---

---


---

---

---

---

---



## Struts Disadvantages

- **Less transparent**
  - With Struts applications, there is a lot more going on behind the scenes than with normal Java-based Web applications. As a result, Struts applications are:
    - Harder to understand
    - Harder to benchmark and optimize
- **Rigid approach**
  - The flip side of the benefit that Struts encourages a consistent approach to MVC is that Struts makes it difficult (but by no means impossible) to use other approaches.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UII

---

---

---


---

---

---

---

---



## Introduction

- **JSP Tag Libraries** is the collection of standard tags. JSP tags are the Java components that can be used in a JSP file. JSP tag libraries allow us to make Java APIs or frameworks available to Web designers, move Java code outside of existing JSP pages, reduce Web development and maintenance costs, develop Web applications easier and faster, and reuse your Java code in multiple Web applications. JavaServer Pages (JSP) tag libraries declares modular functionality so that any JSP page can reuse it. Tag libraries shorten the necessity to embed large amounts of Java code in JSP pages by moving the functionality of the tags into tag implementation classes. Tag libraries includes a specific type of library known as Java Server Pages Tag Library(JSTL).

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UII

---

---

---


---

---

---

---

---



## Introduction

- The **JavaServer Pages Standard Tag Library (JSTL)** is the collection of the components of the Java EE Web application development platform. JSTL adds a tag library of JSP by extending the JSP specification to perform common tasks, such as XML data processing, conditional execution, loops and internationalization. JSTL provides an better way to embed logic without using embedded Java code directly within a JSP page. Use of a standardized tag set, rather than breaking in and out of Java code leads to more maintainable code and enables separation of concerns between the development of the application code and user interface.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason

---

---

---


---

---

---

---

---



## JSTL

- The JavaServer Pages Standard Tag Library (JSTL) is a collection of useful JSP tags which encapsulates core functionality common to many JSP applications.
- JSTL has support for common, structural tasks such as iteration and conditionals, tags for manipulating XML documents, internationalization tags, and SQL tags. It also provides a framework for integrating existing custom tags with JSTL tags.
- The JSTL tags can be classified, according to their functions, into following JSTL tag library groups that can be used when creating a JSP page: **Core Tags**, **Formatting tags**, **SQL tags**, **XML tags**
- JSTL Functions**

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason

---

---

---


---

---

---

---

---



## Core Tags

- To use any of the libraries, you must include a `<taglib>` directive at the top of each JSP that uses the library.
- Core Tags:**
- The core group of tags are the most frequently used JSTL tags. Following is the syntax to include JSTL Core library in your JSP:
- `<% @ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>`  
There are following Core JSTL Tags:

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason

---

---

---

---

---

---

---

---

Core Tags	
Tag Summary	
<a href="#">catch</a>	Catches any Throwable that occurs in its body and optionally exposes it.
<a href="#">choose</a>	Simple conditional tag that establishes a context for mutually exclusive conditional operations, marked by <when> and <otherwise>
<a href="#">if</a>	Simple conditional tag, which evaluates its body if the supplied condition is true and optionally exposes a Boolean scripting variable representing the evaluation of this condition
<a href="#">import</a>	Retrieves an absolute or relative URL and exposes its contents to either the page, a String in 'var', or a Reader in 'varReader'.
<a href="#">forEach</a>	The basic iteration tag, accepting many different collection types and supporting subsetting and other functionality
<a href="#">forEachTokens</a>	Iterates over tokens, separated by the supplied <code>Google</code> <code>http://www.google.co.in/</code>
<a href="#">out</a>	Like <%= ... >, but for expressions.
<a href="#">otherwise</a>	Subtag of <choose> that follows <when> tags and runs only if all of the prior conditions evaluated to 'false'
<a href="#">param</a>	Adds a parameter to a containing 'import' tag's URL.
<a href="#">redirect</a>	Redirects to a new URL.
<a href="#">remove</a>	Removes a scoped variable (from a particular scope, if specified).
<a href="#">set</a>	Sets the result of an expression evaluation in a 'scope'
<a href="#">url</a>	Creates a URL with optional query parameters
<a href="#">when</a>	Subtag of <choose> that includes its body if its condition evaluates to 'true'

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason

UII

Formatting Tags	
<ul style="list-style-type: none"> <li>Also called Internationalization Tag Library or I18N Tag library, formatting tags list the tags that confirm the client locale settings for formatting and parsing client sensitive data like currency, date, etc in a localized or customized manner. It is used with a prefix "fmt".</li> <li>&lt;% @ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %&gt;</li> </ul>	

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason

UII

Formatting Tags	
Tag Summary	
<a href="#">requestEncoding</a>	Sets the request character encoding
<a href="#">setLocale</a>	Stores the given locale in the locale configuration variable
<a href="#">timeZone</a>	Specifies the time zone for any time formatting or parsing actions nested in its body
<a href="#">setTimeZone</a>	Stores the given time zone in the time zone configuration variable
<a href="#">bundle</a>	Loads a resource bundle to be used by its tag body
<a href="#">setBundle</a>	Loads a resource bundle and stores it in the named scoped variable or the bundle configuration variable
<a href="#">message</a>	Maps key to localized message and performs parametric replacement
<a href="#">param</a>	Supplies an argument for parametric replacement to a containing <message> tag
<a href="#">formatNumber</a>	Formats a numeric value as a number, currency, or percentage
<a href="#">parseNumber</a>	Parses the string representation of a number, currency, or percentage
<a href="#">formatDate</a>	Formats a date and/or time using the supplied styles and pattern
<a href="#">parseDate</a>	Parses the string representation of a date and/or time

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason

UII

**Database Tags**

- <% @ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql" %>

Tag Summary	
<a href="#">transaction</a>	Provides nested database action elements with a shared Connection, set up to execute all statements as one transaction.
<a href="#">query</a>	Executes the SQL query defined in its body or through the sql attribute.
<a href="#">update</a>	Executes the SQL update defined in its body or through the sql attribute.
<a href="#">param</a>	Sets a parameter in an SQL statement to the specified value.
<a href="#">dateParam</a>	Sets a parameter in an SQL statement to the specified java.util.Date value.
<a href="#">setDataSource</a>	Creates a simple DataSource suitable only for prototyping.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UII

---

---

---

---

---

---

---

---

**XML Tags**

- <% @ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>

<a href="#">choose</a>	Simple conditional tag that establishes a context for mutually exclusive conditional operations, marked by <when> and <otherwise>
<a href="#">out</a>	Like <%= ... >, but for XPath expressions.
<a href="#">if</a>	XML conditional tag, which evaluates its body if the supplied XPath expression evaluates to 'true' as a boolean
<a href="#">for Each</a>	XML iteration tag.
<a href="#">otherwise</a>	Subtag of <choose> that follows <when> tags and runs only if all of the prior conditions evaluated to 'false'
<a href="#">param</a>	Adds a parameter to a containing 'transform' tag's Transformer
<a href="#">parse</a>	Parses XML content from 'source' attribute or 'body'
<a href="#">set</a>	Saves the result of an XPath expression evaluation in a 'scope'
<a href="#">transform</a>	Conducts a transformation given a source XML document and an XSLT stylesheet
<a href="#">when</a>	Subtag of <choose> that includes its body if its expression evaluates to 'true'

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Ms. Ritika Wason UII

---

---

---

---

---

---

---

---

**JSTL FUNCTIONS**

<% @ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

**Functions**

- [fn:contains\(\)](#)
- [fn:containsIgnoreCase\(\)](#)
- [fn:endsWith\(\)](#)
- [fn:escapeXml\(\)](#)
- [fn:indexOf\(\)](#)
- [fn:join\(\)](#)
- [fn:length\(\)](#)
- [fn:replace\(\)](#)
- [fn:split\(\)](#)
- [fn:startsWith\(\)](#)
- [fn:substring\(\)](#)
- [fn:substringAfter\(\)](#)
- [fn:substringBefore\(\)](#)
- [fn:toLowerCase\(\)](#)
- [fn:toUpperCase\(\)](#)
- [fn:trim\(\)](#)

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Anukiran Jain UII

---

---

---

---

---

---

---

---