Handout-4: Introduction to JSP

1.1 JSP OVERVIEW

- Java Server Pages (JSP) is a technology for developing web pages that support dynamic content which helps developers insert java code in HTML pages by making use of special JSP tags, most of which start with <% and end with %>.
- A JavaServer Pages component is a type of Java servlet that is designed to fulfill the role of a user interface for a Java web application. Web developers write JSPs as text files that combine HTML or XHTML code, XML elements, and embedded JSP actions and commands.
- Using JSP, you can collect input from users through web page forms, present records from a database or another source, and create web pages dynamically.
- JSP tags can be used for a variety of purposes, such as retrieving information from a database or registering user preferences, accessing JavaBeans components, passing control between pages and sharing information between requests, pages etc.

1.2 WHY USE JSP?

JavaServer Pages often serve the same purpose as programs implemented using the Common Gateway Interface (CGI). But JSP offer several advantages in comparison with the CGI.

- Performance is significantly better because JSP allows embedding Dynamic Elements in HTML Pages itself instead of having a separate CGI files.
- JSP are always compiled before it's processed by the server unlike CGI/Perl which requires the server to load an interpreter and the target script each time the page is requested.
- JavaServer Pages are built on top of the Java Servlets API, so like Servlets; JSP also has access to all the powerful Enterprise Java APIs, including JDBC, JNDI, EJB, JAXP etc.
- JSP pages can be used in combination with servlets that handle the business logic, the model supported by Java servlet template engines.

Finally, JSP is an integral part of J2EE, a complete platform for enterprise class applications. This means that JSP can play a part in the simplest applications to the most complex and demanding.

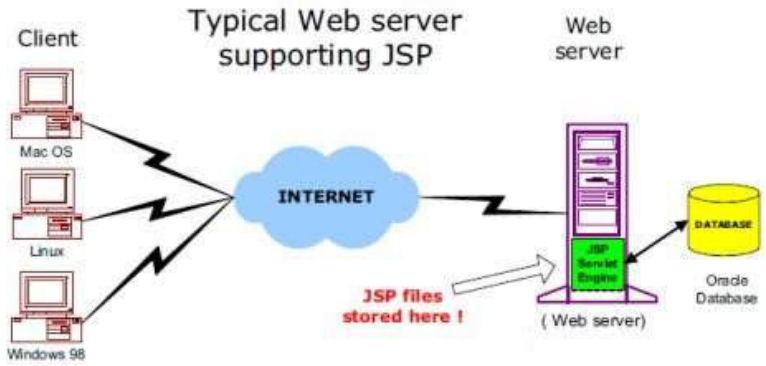
1.3 Advantages of JSP

Following is the list of other advantages of using JSP over other technologies:

- **Vs. Active Server Pages (ASP):** The advantages of JSP are twofold. First, the dynamic part is written in Java, not Visual Basic or other MS specific language, so it is more powerful and easier to use. Second, it is portable to other operating systems and non-Microsoft Web servers.
- **Vs. Pure Servlets:** It is more convenient to write (and to modify!) regular HTML than to have plenty of println statements that generate the HTML.
- **Vs. Server-Side Includes (SSI):** SSI is really only intended for simple inclusions, not for "real" programs that use form data, make database connections, and the like.
- **Vs. JavaScript:** JavaScript can generate HTML dynamically on the client but can hardly interact with the web server to perform complex tasks like database access and image processing etc.
- **Vs. Static HTML:** Regular HTML, of course, cannot contain dynamic information.

1.4 JSP ARCHITECTURE

- The web server needs a JSP engine i.e. container to process JSP pages.
- The JSP container is responsible for intercepting requests for JSP pages.
- We will make use of Apache which has built-in JSP container to support JSP pages development.
- A JSP container works with the Web server to provide the runtime environment and other services a JSP needs. It knows how to understand the special elements that are part of JSPs.
- Following diagram shows the position of JSP container and JSP files in a Web Application.

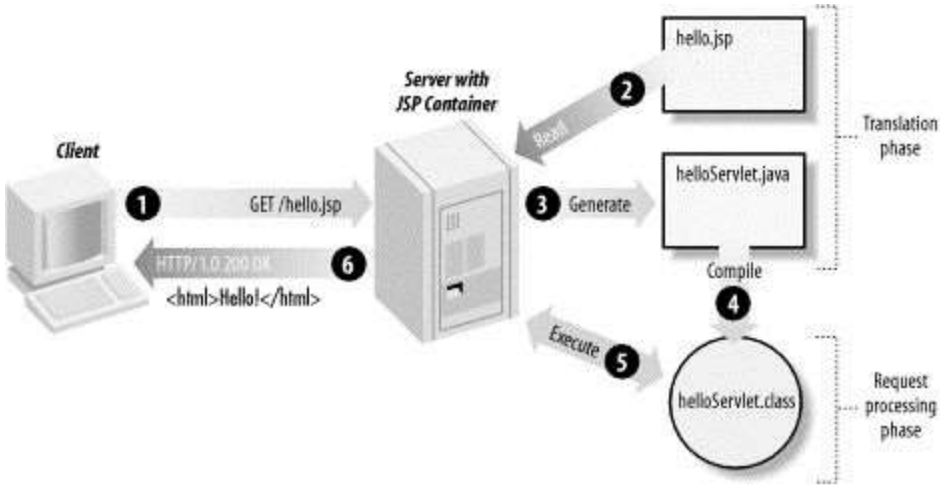


1.5 JSP PROCESSING

The following steps explain how the web server creates the web page using JSP:

- As with a normal page, your browser sends an HTTP request to the web server.
- The web server recognizes that the HTTP request is for a JSP page and forwards it to a JSP engine. This is done by using the URL or JSP page which ends with **.jsp** instead of .html.
- The JSP engine loads the JSP page from disk and converts it into a servlet content. This conversion is very simple in which all template text is converted to println ( ) statements and all JSP elements are converted to Java code that implements the corresponding dynamic behavior of the page.
- The JSP engine compiles the servlet into an executable class and forwards the original request to a servlet engine.
- A part of the web server called the servlet engine loads the Servlet class and executes it. During execution, the servlet produces an output in HTML format, which the servlet engine passes to the web server inside an HTTP response.
- The web server forwards the HTTP response to your browser in terms of static HTML content.
- Finally web browser handles the dynamically generated HTML page inside the HTTP response exactly as if it were a static page.

All the above mentioned steps can be shown below in the following diagram:



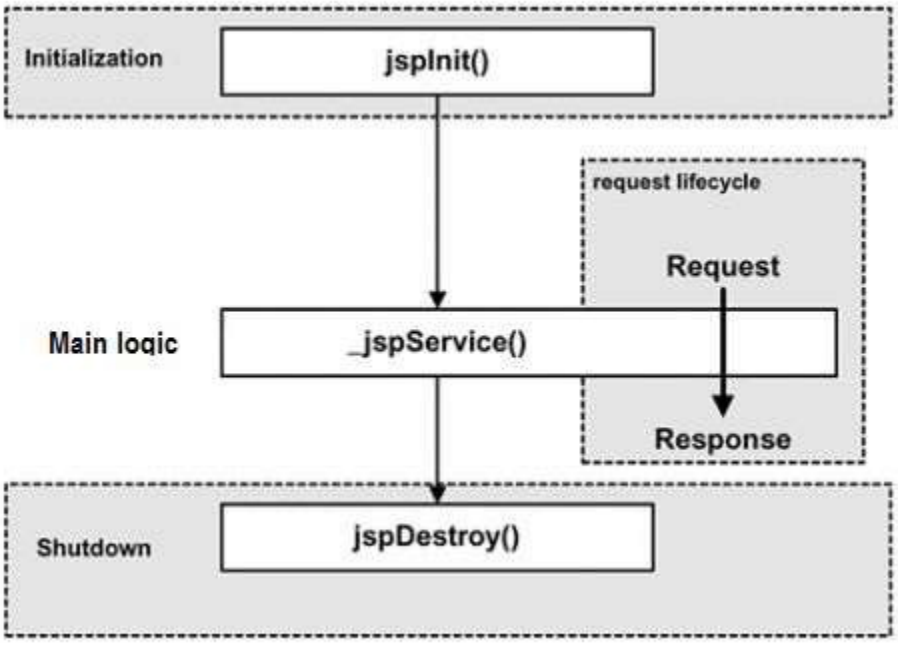
- Typically, the JSP engine checks to see whether a servlet for a JSP file already exists and whether the modification date on the JSP is older than the servlet.

- If the JSP is older than its generated servlet, the JSP container assumes that the JSP hasn't changed and that the generated servlet still matches the JSP's contents. This makes the process more efficient than with other scripting languages (such as PHP) and therefore faster.
- So in a way, a JSP page is really just another way to write a servlet without having to be a Java programming wiz. Except for the translation phase, a JSP page is handled exactly like a regular servlet.

1.6 JSP LIFE CYCLE

- 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

The four major phases of JSP life cycle are very similar to Servlet Life Cycle and they are as follows:



1.6.1 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.

1.6.2 JSP Initialization

When a container loads a JSP it invokes the **jspInit()** method before servicing any requests. If you need to perform JSP-specific initialization, override the **jspInit()** method:

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

Typically initialization is performed only once and as with the servlet init method, you generally initialize database connections, open files, and create lookup tables in the jspInit method.

1.6.3 JSP Execution

- This phase of the JSP life cycle represents all interactions with requests until the JSP is destroyed.
- Whenever a browser requests a JSP and the page has been loaded and initialized, the JSP engine invokes the **_jspService()** method in the JSP.
- The _jspService() method takes an **HttpServletRequest** and an **HttpServletResponse** as its parameters as follows:

```
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 i.e. GET, POST, DELETE etc.

1.6.4 JSP Cleanup

- The destruction phase of the JSP life cycle represents when a JSP is being removed from use by a container.
- 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.
- The jspDestroy() method has the following form:

```
public void jspDestroy()
{
      // Your cleanup code goes here.
}
```

1.7 JSP IMPLICIT OBJECTS

- JSP Implicit Objects are the Java objects that the JSP Container makes available to developers in each page and developer can call them directly without being explicitly declared.
- JSP Implicit Objects are also called pre-defined variables.
- JSP supports nine Implicit Objects which are listed below:

Object	Description
request	This is the **HttpServletRequest** object associated with the request.
response	This is the **HttpServletResponse** object associated with the response to the client.
out	This is the **PrintWriter** object used to send output to the client.
session	This is the **HttpSession** object associated with the request.
application	This is the **ServletContext** object associated with application context.
config	This is the **ServletConfig** object associated with the page.
pageContext	This encapsulates use of server-specific features like higher performance **JspWriters**.
page	This is simply a synonym for **this**, and is used to call the methods defined by the translated servlet class.

Exception	The **Exception** object allows the exception data to be accessed by designated JSP.

1.7.1 The request Object

- The request object is an instance of a javax.servlet.http.HttpServletRequest object.
- Each time a client requests a page the JSP engine creates a new object to represent that request.
- The request object provides methods to get HTTP header information including form data, cookies, HTTP methods etc.

1.7.2 The response Object

- The response object is an instance of a javax.servlet.http.HttpServletResponse object.
- Just as the server creates the request object, it also creates an object to represent the response to the client.
- The response object also defines the interfaces that deal with creating new HTTP headers.
- Through this object the JSP programmer can add new cookies or date stamps, HTTP status codes etc.

1.7.3The out Object

- The out implicit object is an instance of a javax.servlet.jsp.JspWriter object and is used to send content in a response.
- The initial JspWriter object is instantiated differently depending on whether the page is buffered or not. Buffering can be easily turned off by using the buffered='false' attribute of the page directive.
- The JspWriter object contains most of the same methods as the java.io.PrintWriter class. However, JspWriter has some additional methods designed to deal with buffering.
- Unlike the PrintWriter object, JspWriter throws IOExceptions.
- Following are the important methods which we would use to write boolean char, int, double, object, String etc.

Method	Description
out.print(dataType dt)	Print a data type value
out.println(dataType dt)	Print a data type value then terminate the line with new line character.
out.flush()	Flush the stream.

1.7.4 The session Object

- The session object is an instance of **javax.servlet.http.HttpSession** and behaves exactly the same way that session objects behave under Java Servlets.
- The session object is used to track client session between client requests.

1.7.5 The application Object

- The application object is direct wrapper around the ServletContext object for the generated Servlet and in reality an instance of a javax.servlet.ServletContext object.
- This object is a representation of the JSP page through its entire lifecycle.
- This object is created when the JSP page is initialized and will be removed when the JSP page is removed by the jspDestroy() method.
- By adding an attribute to application, you can ensure that all JSP files that make up your web application have access to it.

1.7.6 The config Object

- The config object is an instantiation of javax.servlet.ServletConfig and is a direct wrapper around the ServletConfig object for the generated servlet.
- This object allows the JSP programmer access to the Servlet or JSP engine initialization parameters such as the paths or file locations etc.
- The following config method is the only one you might ever use, and its usage is trivial:

```
config.getServletName();
```

- This returns the servlet name, which is the string contained in the <servlet-name> element defined in the WEB-INF\web.xml file

1.7.7 The pageContext Object

- The pageContext object is an instance of a javax.servlet.jsp.PageContext object. The pageContext object is used to represent the entire JSP page.
- This object is intended as a means to access information about the page while avoiding most of the implementation details.

- This object stores references to the request and response objects for each request. The application, config, session, and out objects are derived by accessing attributes of this object.
- The pageContext object also contains information about the directives issued to the JSP page, including the buffering information, the errorPageURL, and page scope.
- The PageContext class defines several fields, including PAGE_SCOPE, REQUEST_SCOPE, SESSION_SCOPE, and APPLICATION_SCOPE, which identify the four scopes. It also supports more than 40 methods, about half of which are inherited from the javax.servlet.jsp. JspContext class.
- One of the important methods is **removeAttribute**, which accepts either one or two arguments. For example, pageContext.removeAttribute ("attrName") removes the attribute from all scopes, while the following code only removes it from the page scope:

```
pageContext.removeAttribute("attrName", PAGE_SCOPE);
```

1.7.8 The page Object

- This object is an actual reference to the instance of the page.
- It can be thought of as an object that represents the entire JSP page.
- The page object is really a direct synonym for the **this** object.

1.7.9 The exception Object

- The exception object is a wrapper containing the exception thrown from the previous page.
- It is typically used to generate an appropriate response to the error condition.