JSP(Java ServerPages)

—----------

JavaServer Pages (JSP) is a *complimentary* technology to *Java Servlet* which facilitates the *mixing* of dynamic and static web contents. JSP is Java's answer to the popular Microsoft's *Active Server Pages* (ASP).

JSP, like ASP, provides a elegant way to mix static and dynamic contents. The main page is written in regular HTML, while special tags are provided to insert pieces of Java programming codes. The *business programming logic* and the *presentation* are cleanly separated. This allows the programmers to focus on the business logic, while the web designer to concentrate on the presentation.

JSP is based on Servlet. In fact, we shall see later that a JSP page is internally translated into a Java servlet. We shall also explain later that "***Servlet is HTML inside Java***", while "***JSP is Java inside HTML***". Whatever you can't do in servlet, you can't do in JSP. JSP makes the creation and maintenance of dynamic HTML pages much easier than servlet. JSP is more convenience than servlet for dealing with the presentation, not more powerful.

JSP is meant to *compliment* Servlet, not a replacement. In a *Model-View-Control* (MVC) design, servlets are used for the controller, which involves complex programming logic. JSPs are used for the view, which deals with presentation. The model could be implemented using JavaBeans or Enterprise JavaBeans (EJB) which may interface with a database.

Advantages of JSP

- *Separation of static and dynamic contents*: The dynamic contents are generated via programming logic and inserted into the *static template*. This greatly simplifies the creation and maintenance of web contents.
- *Reuse of components and tag libraries*: The dynamic contents can be provided by re-usable components such as JavaBean, Enterprise JavaBean (EJB) and tag libraries - you do not have to re-inventing the wheels.
- Java's power and portability.

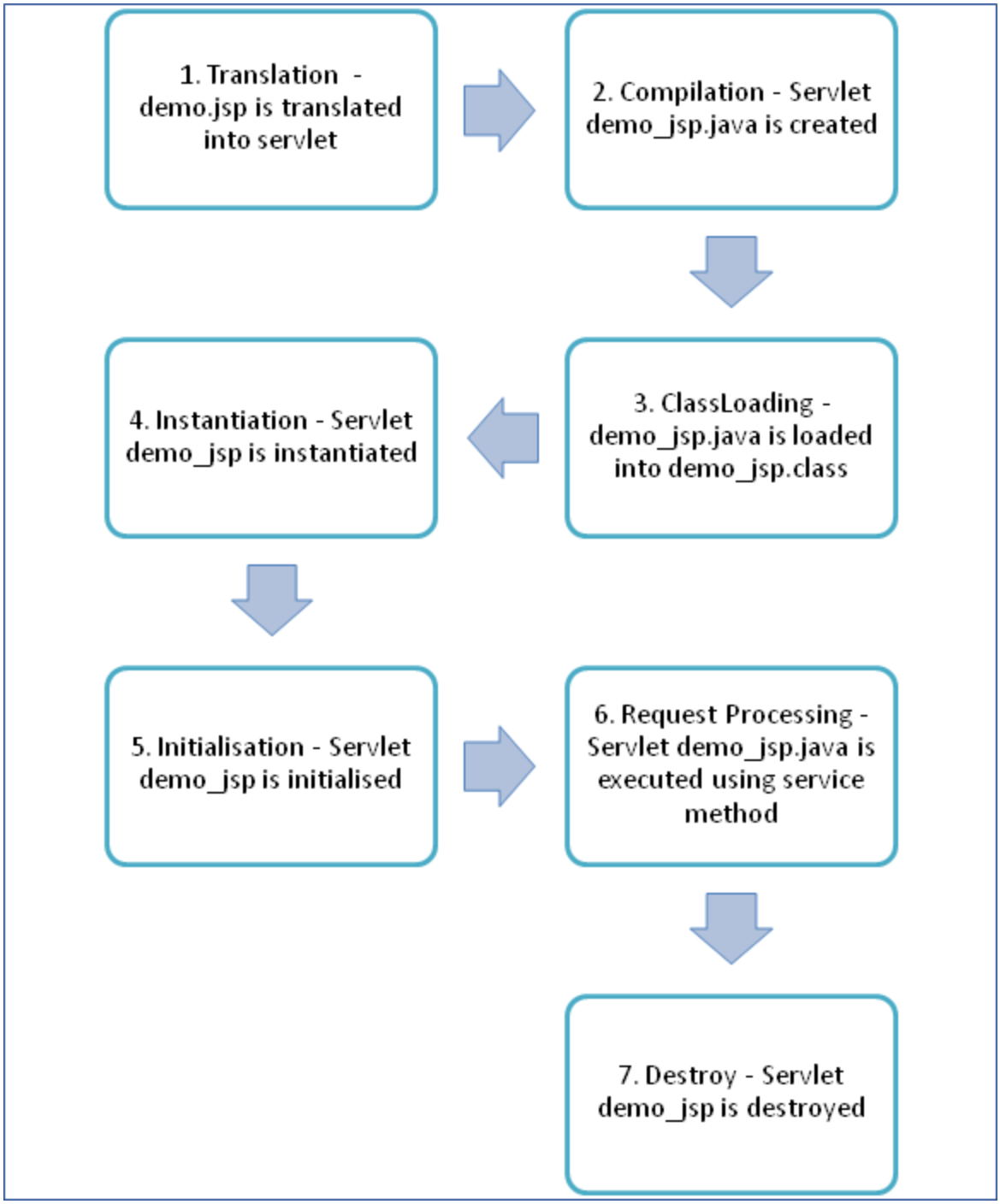
Apache Tomcat Server

JSPs, like servlets, are server-side programs run inside a *Java-capable* HTTP server. Apache Tomcat Server (@ http://tomcat.apache.org) is the official reference implementation (RI) for Java servlet and JSP, provided free by Apache (@ http://www.apache.org) - an open-source software foundation.

Life Cycle of JSP

==========

When a JSP page is first requested, Tomcat translates the JSP into a servlet, compiles the servlet, load, and execute the servlet. The best way to understand JSP is to check out the generated servlet and study the JSP-to-Servlet translation. The generated servlet for "first.jsp" is kept under Tomcat's "work" directory ("<CATALINA_HOME>\work\Catalina\localhost\hellojsp\...").



JSP Life Cycle is defined as translation of JSP Page into servlet as a JSP Page needs to be converted into servlet first in order to process the service requests. The Life Cycle starts with the creation of JSP and ends with the disintegration of that.

1. Translation of JSP page
2. Compilation of JSP page(Compilation of JSP page into _jsp.java)
3. Classloading (_jsp.java is converted to class file _jsp.class)
4. Instantiation(Object of generated servlet is created)
5. Initialisation(_jspinit() method is invoked by container)
6. Request Processing(_jspservice() method is invoked by the container)
7. Destroy (_jspDestroy() method invoked by the container)

Both Servlet and JSP are based the important feature of Java that is Multi Threading.

For every user request a copy of that servlet or jsp object is served via a thread.

JSP Elements

- JSP Declaration
- JSP Scriptlet
- JSP Expression
- JSP Comments
- Creating a simple JSP Page
- How to run simple JSP Page
- Directory Structure of JSP

JSP Declaration

- A declaration tag is a piece of Java code for declaring variables, methods and classes. If we declare a variable or method inside declaration tag it means that the declaration is made inside the servlet class but outside the service method.
- We can declare a static member, an instance variable (can declare a number or string) and methods inside the declaration tag.

Syntax of declaration tag:

<%! Dec var %>

Here Dec var is the method or a variable inside the declaration tag.

Example:

In this example, we are going to use the declaration tags

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

    pageEncoding="ISO-8859-1"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>Guru Declaration Tag</title>

</head>

<body>

<%! int mark =100; %>

<% out.println("The Number is " +mark); %>

</body>

</html>
```

JSP Scriptlet

- Scriptlet tag allows to write Java code into JSP file.(Java +html)
- JSP container moves statements in _jspservice() method while generating servlet from jsp.
- For each request of the client, service method of the JSP gets invoked hence the code inside the Scriptlet executes for every request.
- A Scriptlet contains java code that is executed every time JSP is invoked.

Syntax of Scriptlet tag:

```
<% java code %>

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

    pageEncoding="ISO-8859-1"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>Guru Scriplet</title>

</head>

<body>

<% int a=10;

   int b=20;

   int c = a+b;

   out.println(" Sum is " +c);

%>

</body>

</html>
```

JSP Expression

- Expression tag evaluates the expression placed in it.
- It accesses the data stored in stored application.
- It allows create expressions like arithmetic and logical.
- It produces scriptless JSP page.

Syntax:

```
<%= expression %>
```

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

    pageEncoding="ISO-8859-1"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>JSP Expression</title>

</head>

<body>

<% out.println("The expression number is "); %>

<% int num1=10; int num2=10; int num3 = 20; %>

<%= num1*num2+num3 %>

</body>

</html>
```

(Ref my codes in class for font size dynamically increasing)

JSP Comments

Comments are the one when JSP container wants to ignore certain texts and statements.

When we want to hide certain content, then we can add that to the comments section.

Syntax:

<% -- JSP Comments %>

Key Difference between Servlet and JSP

- Servlet can accept all protocol requests, including HTTP, while JSP can only accept HTTP requests.
- In MVC architecture, servlet works as a controller, while JSP works as a view for displaying output.
- Servlet should be used when there is more data processing involved, whereas JSP is generally used when there is less involvement of data processing.
- Servlets run faster than JSP, on the other hand, JSP runs slower than servlet as it takes time to compile the program and convert it into servlets.
- You can override the service() method in servlet, but in JSP, you can't override the service() method.
- In Servlet, you have to implement both business logic and presentation logic in a single file. Whereas in JSP, business logic is split from presentation logic using JavaBeans.

Advantages of JSP

Here are the advantages of using JSP:

- It is very much convenient to modify the regular HTML.
- We can write the servlet code into the JSP.
- It is only intended for simple inclusions which can use form data and make connections.
- You can easily connect with JSP with the MySQL database.
- The performance and scalability of JSP are good because JSP allows embedding of dynamic elements in HTML pages.
- JSP is built on Java technology hence it is platform-independent and not depending on any operating systems.
- JSP includes the feature of multithreading of java.
- We can also make use of exception handling of java into JSP.
- It enables to separate presentation layer with the business logic layer in the web application.
- It is easy for developers to show as well as process the information.

Disadvantages of JSP

Here are the disadvantages for using JSP:

- It is hard to trace JSP pages error because JSP pages are translated to servlet.
- As JSP output is HTML, it is not rich in features.
- It is very hard to debug or trace errors because JSP pages are first translated into servlets before the compilation process.
- Database connectivity is not easy.
- JSP pages require more disk space and time to hold JSP pages as they are compiled on the server.

Implicit objects in JSP need not be declared or instantiated by the JSP author. They are automatically instantiated by the container which is accessed by using standard variables. Hence, they are called implicit objects. These are parsed by the container and inserted into the generated servlet code. They are only available within the JSP service method and not in any declaration.

Types of Implicit Objects in JSP

For every JSP file when the JSP compiler generates the servlet program it will create the following 9 implicit objects inside the _jspService() method.

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

(ref my codes in class room training)

Request Object in JSP

It is an instance of javax.servlet.http.HttpServletRequest object. This implicit object is used to process the request sent by the client. It is the HttpServletRequest object associated with the request. When a client requests a page the JSP engine creates a new object to represent that request.

Response Object in JSP:

This is the HttpServletResponse object associated with the response to the client. This implicit object is used to process the request and send the response to the client. We can send content-type as well as error reports and text. The object is by default available to scriptlets and expressions to get response-related information and to set response-related information. If a JSP wants to redirect a request from this server to another server then it can use the response object and sendRedirect method call.

PageContext Implicit Object in JSP:

pageContext is an implicit object in JSP Technology, it can be used to get all the JSP implicit objects even in a Non-JSP environment. It is created for pageContext class which is used for creating all the implicit variables like a session, ServletConfig, out, etc… While preparing Tag handler classes in custom tags container will provide pageContext object as part of its life cycle, from this we are able to get all the implicit objects. In JSP applications, by using the pageContext object we are able to perform some operations with the attributes in the JSP scopes page, request, session, and application like adding an attribute, removing an attribute, getting an attribute, and finding an attribute. This encapsulates the use of server-specific features like higher performance JSPWriters. The pageContext reference points to pageContext class sub-class object.

Session Object in JSP:

This is the HttpSession object associated with the request. a session is an implicit object which is created for HttpSession class using which we can store and access data. The session reference point HttpSession implementation object. By default, this reference is available to scriptlet, expression tags. These tags can access session related info and they can manage session scope attributes. By using this reference these tags can delete session objects by calling session.invalidate() and these reference tags can set session timeout.

Application Object in JSP

This is the ServletContext object associated with the application context. Application is an implicit object which is created for ServletContext class and used to access the data of the ServletContaxt object same as in servlets. The application reference point ServletContext implementation object. By using this reference, a JSP can access application-related information such as managing application scope attributes, getting the application init parameters, etc.

Implicit Config Object in JSP:

This is the ServletContext object associated with the page. This implicit object is created for ServletConfig class and used to access the data of the ServletConfig object same as in servlets. The config reference points ServletConfig implementation object. By using this config reference a JSP page can get initialization parameters of JSP configuration, get initialization parameters names, get servlet names, and can get servlet context reference.

Following is the only method of config object:

1. getServletName(): This method returns the servlet name which is the string contained in the <servlet-name> element defined in the web.xml configuration file.
2. getServletContext(): It returns a reference to the Servlet Context.

Out Object in JSP:

This is the PrintWriter object used to send output to the client. The out-reference points JSPWriter subclass object. By using the object, we can print HTML tags and plain text data on the browser.

Methods of out object

1. out.print(dataType dt): It is used to print a data type value.
2. out.println(dataType dt): It is used to print a data type value then terminate the line with a new line character.
3. out.flush(): It is used to flush the stream.

Page Object in JSP

This is simply a synonym for this and is used to call the methods defined by the translated servlet class. The page reference points current JSP page object.

Exception Object in JSP

The Exception object allows the exception data to be accessed by a designated JSP. This reference points to any exception object raised in the .jsp page and provides that exception object to the error JSP page.

Methods of Exception implicit object

1. getMessage(): It gives a detailed message about the exception that has occurred. This is initialized in the Throwable constructor.
2. getCause(): It returns the cause of the exception as represented by a Throwable object.
3. toString(): It gives the name of the class concatenated with the result of getMessage().
4. printstackTrace(): It prints the result of toString() along with the stack trace to System.err.
5. getStackTrace(): It returns an array containing each element on the stack trace.
6. fillnStackTrace(): It fills the stack trace of this Throwable object with the current

(ref code in class room training)

JDBC code Refer our class room training

Directives

Directives supply directions and messages to a JSP container. The directives provide global information about the entire page of JSP. Hence, they are an essential part of the JSP code. These special instructions are used for translating JSP to servlet code.

There are 3 types of directives:

1. Page directive
2. Include directive
3. Taglib directive

The page directive is used for defining attributes that can be applied to a complete JSP page. You may place your code for Page Directives anywhere within your JSP page. However, in general, page directives are implied at the top of your JSP page.

The basic syntax of the page directive is:

<%@ page attribute = "attribute_value" %>

The XML equivalent for the above derivation is:

<jsp:directive.page attribute = "attribute_value" />

The page directive is used for defining attributes that can be applied to a complete JSP page. You may place your code for Page Directives anywhere within your JSP page. However, in general, page directives are implied at the top of your JSP page.

The basic syntax of the page directive is:

<%@ page attribute = "attribute_value" %>

The XML equivalent for the above derivation is:

<jsp:directive.page attribute = "attribute_value" />

The attributes used by the Page directives are:

1. buffer: Buffer attribute sets the buffer size in KB to control the JSP page's output.

2. contentType: The ContentType attribute defines the document's MIME (Multipurpose Internet Mail Extension) in the HTTP response header.

3. autoFlush: The autofill attribute controls the behavior of the servlet output buffer. It monitors the buffer output and specifies whether the filled buffer output should be flushed automatically or an exception should be raised to indicate buffer overflow.

4. errorPage: Defining the "ErrorPage" attribute is the correct way to handle JSP errors. If an exception occurs on the current page, it will be redirected to the error page.

5. extends: extends attribute used for specifying a superclass that tells whether the generated servlet has to extend or not.

6. import: The import attribute is used to specify a list of packages or classes used in JSP code, just as Java's import statement does in a Java code.

7. isErrorPage: This "isErrorPage" attribute of the Page directive is used to specify that the current page can be displayed as an error page.

8. info: This "info" attribute sets the JSP page information, which is later obtained using the getServletInfo() method of the servlet interface.

9. isThreadSafe: Both the servlet and JSP are multithreaded. If you want to control JSP page behavior and define the threading model, you can use the "isThreadSafe" attribute of the page directive.

10. Language: The language attribute specifies the programming language used in the JSP page. The default value of the language attribute is "Java".

11. Session: In JSP, the page directive session attribute specifies whether the current JSP page participates in the current HTTP session.

12. isELIgnored: This isELIgnored attribute is used to specify whether the expression language (EL) implied by the JSP page will be ignored.

13. isScriptingEnabled: This "isScriptingEnabled" attribute determines if the scripting elements are allowed for use or not.

The JSP "include directive" is used to include one file in another JSP file. This includes HTML, JSP, text, and other files. This directive is also used to create templates according to the developer's requirement and breaks the pages in the header, footer, and sidebar.

To use this Include Directive, you have to write it like:

<%@ include file = "relative url" >

The XML equivalent of the above way of representation is:

<jsp:directive.include file = "relative url" />

More to continue….