Introduction to JSP

JavaServer Pages (JSP) technology enables you to mix regular, static HTML with dynamically generated content. You simply write the regular HTML in the normal manner, using familiar Web-page-building tools. You then enclose the code for the dynamic parts in special tags, most of which start with <% Java Code %>.

Servlets can be considered as Java code with HTML inside and JSP can be considered as HTML with Java code inside. Despite the large apparent differences between JSP pages and servlets, behind the scenes they are the same thing. JSP pages are translated into servlets, the servlets are compiled, and at request time it is the compiled servlets that execute. So, writing JSP pages is really just another way of writing servlets.

Need of JSP

Servlets have the following deficiencies when it comes to generating the output:

1- It is hard to write and maintain the HTML. Using print statements to generate HTML? Hardly convenient: you have to use parentheses and semicolons, have to insert backslashes in front of embedded double quotes, and have to use string concatenation to put the content together.

2- You cannot use standard HTML tools. All those great Web-site development tools you have are of little use when you are writing Java code.

3- The HTML is inaccessible to non-Java developers. If the HTML is embedded within Java code, a Web development expert who does not know the Java programming language will have trouble reviewing and changing the HTML.
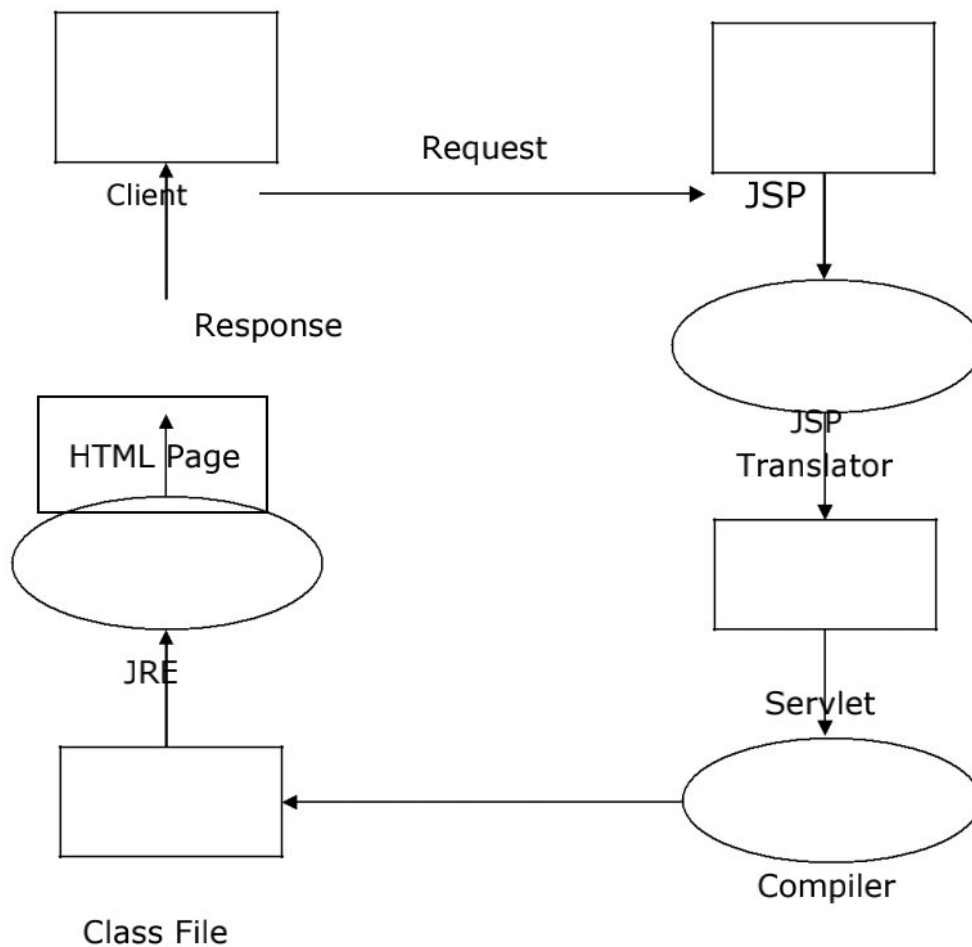
Example

```
<html>
    <body>
            <% out.print("Java Server Pages");
    %> </body>
</html>
```

The Lifecycle of a JSP Page

The JSP pages follow these phases:
1. Translation of JSP Page to Servlet.
2. Follow Servlet life cycle

```
   ┌──────────┐            Request         ┌──────────┐
   │          │ ─────────────────────────> │          │
   │  Client  │                            │   JSP    │
   └──────────┘                            └──────────┘
        ↑                                       │
     Response                                   ↓
                                          ⬯ JSP
   ┌──────────┐                            Translator
   │HTML Page │                                │
   └──────────┘                                ↓
        ↑                                 ┌──────────┐
     ⬯                                    │          │
     JRE                                  └──────────┘
        ↑                                   Servlet
   ┌──────────┐          <───────────         │
   │          │                               ↓
   └──────────┘                             ⬯
   Class File                              Compiler
```

A JSP page is converted into a servlet. The servlet is compiled, loaded into the server's memory, initialized, and executed. But which step happens when? Remember two points:

1- The JSP page is translated into a servlet and compiled only the first time it is accessed after having been modified.

2- Loading into memory, initialization, and execution follow the normal rules for servlets.

Advantages of JSP over Servlet

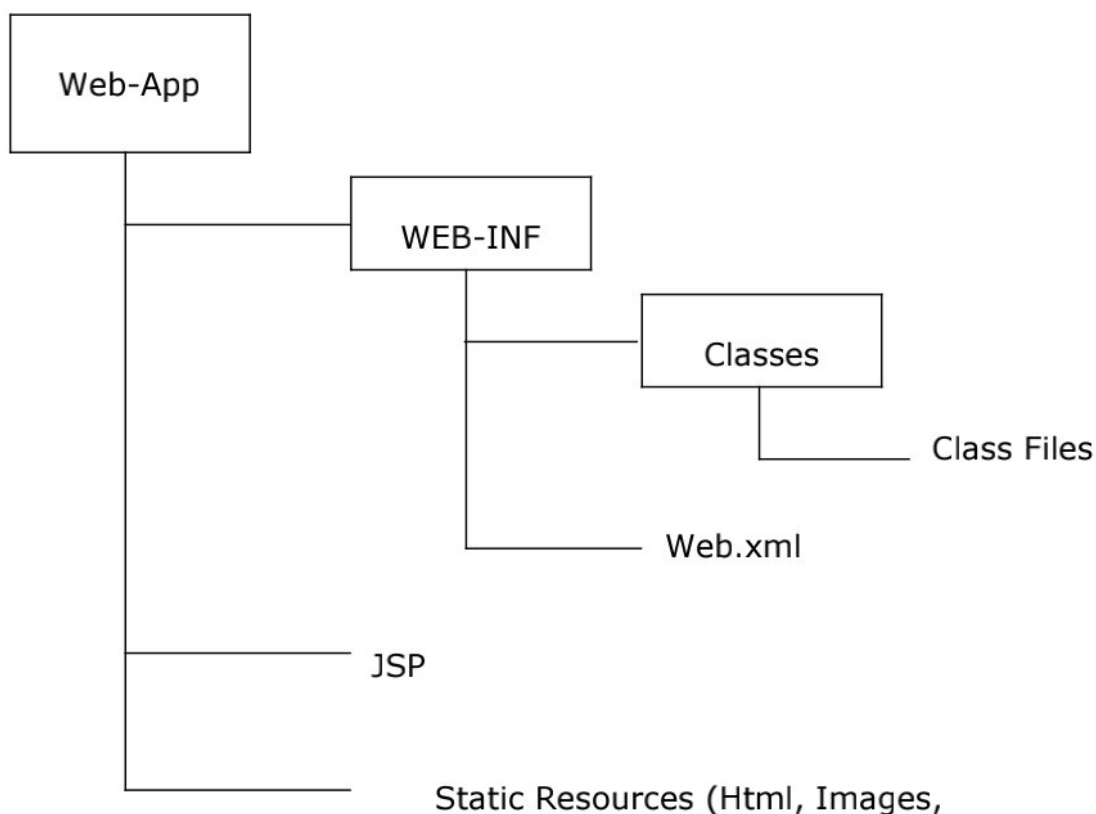There are many advantages of JSP over the Servlet. They are as follows:

1- Extension to Servlet: - JSP technology is the extension to Servlet technology. We can use all the features of the Servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy.

2- Easy to maintain: - JSP can be easily managed because we can easily separate our business logic with presentation logic. In Servlet technology, we mix our business logic with the presentation logic.

3- **Fast Development: - No need to recompile and redeploy:** If JSP page is modified, we don't need to recompile and redeploy the project. The Servlet code needs to be updated and recompiled if we have to change the look and feel of the application.

4- **Less code than Servlet: -** In JSP, we can use many tags such as action tags, JSTL, custom tags, etc. that reduces the code. Moreover, we can use EL, implicit objects, etc.

The Directory structure of JSP

The directory structure of JSP page is same as Servlet. We contain the JSP page outside the WEB-INF folder or in any directory.

```
Web-App
 │
 ├──────────────── WEB-INF
 │                    │
 │                    ├──────────── Classes
 │                    │                │
 │                    │                └──────── Class Files
 │                    │
 │                    └──────────── Web.xml
 │
 ├──────────── JSP
 │
 └──────────── Static Resources (Html, Images,
```

JSP Scripting elements

The scripting elements provide the ability to insert java code inside the HTML Tags. There are three types of scripting elements:
1. Scriptlet tag.
2. Expression tag.
3. Declaration tag.

Scripting Element	Example
Comment	<%-- comment --%>
Directive	<%@ directive %>
Declaration	<%! declarations %>
Scriptlet	<% scriplets %>
Expression	<%= expression %>

Types of JSP Scripting Elements

JSP scripting elements let you insert Java code into the servlet that will be generated from the JSP page. There are three forms:

1- Expressions of the form, which are evaluated and inserted into the servlet's output.
2- Scriptlets of the form, which are inserted into the servlet's _jspService method (called by service).
3- Declarations of the form, which are inserted into the body of the servlet class, outside any existing methods.

JSP Expressions

A JSP expression is used to insert values directly into the output. It has the following form:

<% = Java Expression %>.

The expression is evaluated, converted to a string, and inserted in the page. This evaluation is performed at runtime (when the page is requested) and thus has full access to information about the request.

For example, the following shows the date/time that the page was requested.
Current time: <% = java.util.Date() %>

Predefined Variables

To simplify these expressions, you can use a number of predefined variables (or "implicit objects"). The system simply tells you what names it will use for the