데이타베이스 시스템 연구실
Database Systems Lab.

11. JSP I

충남대학교 컴퓨터공학과
데이타베이스시스템 연구실

Overview

- http://www.tutorialspoint.com/jsp/index.htm

- **What is JavaServer Pages?**
  - JavaServer Pages (JSP) is a server-side programming technology that enables the creation of dynamic, platform-independent method for building Web-based applications.
  - 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 %>.

Overview (cont'd)

- **Advantages of 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 Servelts API, so like Servelts, 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

CGI(Common Gateway Interface)

- 최초의 웹 애플리케이션 기술
- 웹 애플리케이션을 독립적인 프로그램 형태로 작성
- 문제점
  - 각 프로그램이 하나의 프로세스로 수행됨
  - 독립적인 프로그램 실행을 하기 위해 시스템 자원이 많이 필요
  - 그 결과 웹 서버로 많은 요청이 한꺼번에 들어오면 너무 많은 CGI 프로그램의 프로세스가 동시에 실행되어 컴퓨터 전체가 다운되는 일이 빈번히 발생

서블릿(Servlet)

- 자바를 기반으로 하는 웹 애플리케이션 프로그래밍 기술
- 처음에 Sun Microsystems의해 개발되었지만, 그 후에 자바 표준 개발을 전담하는 기관인 JCP(Java community Process)로 이관되어서 발전
- 자바 클래스 형태로 웹 애플리케이션 작성
  - 이 클래스를 서블릿 클래스라 함

서블릿 클래스의 예

doGet 또는 doPost 매서드안에 서블릿 클래스가 해야 할 일을 써 넣음

HttpServlet 클래스를 상속

```
import  javax.servlet.*;
import  javax.servlet.http.*;
import  java.io.*;
Public  class HundredServlet  extends  HttpServlet  {
   public  void  doGet(HttpServletRequest  request, HttpServletResponse response)
            throws  ServletException, IOException {
       printWriter  out = response.getWriter();
            out.println(“<HTML>”);
            out.println(“<HEAD><TITLE>Sum of 1 to 100</TITLE></HEAD>”);
            out.println(“<BODY>”);
            int    total = 0;
            for(int  cnt =1;cnt <=100; cnt++)
               total += cnt ;
            out.println(“1+2+3+ ...+100 = “+total);
            out.println(“</BODY>”);
            out.println(“</HTML>”);
   }
}
```

HttpServletResponse 파라미터를 이용해서 HTML 코드를 출력함

서블릿 장점과 단점

- **장점**
  - 자바의 플랫폼 독립성: 서블릿 실행 코드를 다양한 운영체제에 서 수행가능
  - 네트워크 환경에서 보안이 용이
  - 시스템 부하가 적은 멀티스레드 기능이 지원됨
- **단점**
  - 프로그래밍 작업의 효율성이 떨어짐
    - HTML 코드가 자바 코드 안으로 들어가는 구조로 인함
    - 이러한 구조는 서블릿 클래스가 출력하는 동적 HTML 문서의 구조를 이해하기 어렵게 함
    - 또한, 웹 페이지의 디자인 작업을 위해서도 소스 코드에 손을 대게 만드는 비 효율성을 낳음
- **자바의 유용한 장점을 살리면서 서블릿 기술의 단점을 보완하는 새로운 기술 JSP가 개발됨**

JSP (JavaServer Pages)

- **자바를 기반으로 한 웹 애플리케이션**
- JSP는 웹 애플리케이션 서버의 서블릿 컨테이너에서 서블릿 원시코드로 변환된 후에 서블릿 원시코드는 바로 컴파일된 후 실행
- **JSP페이지는 HTML 문서에 자바 코드가 삽입되는 구조**

```
<%@  page   contentType="text/html;  charset=UTF-8"%>
<HTML>
<HEAD><TITLE>1부터 100까지의 합</TITLE></HEAD>
<BODY>
<% int  total = 0;
            for (int cnt=1; cnt <= 100; cnt++)
               total += cnt;
%>
1부터 100까지의 합은?  <%=total %>
</BODY>
</HTML>
```

자바 코드

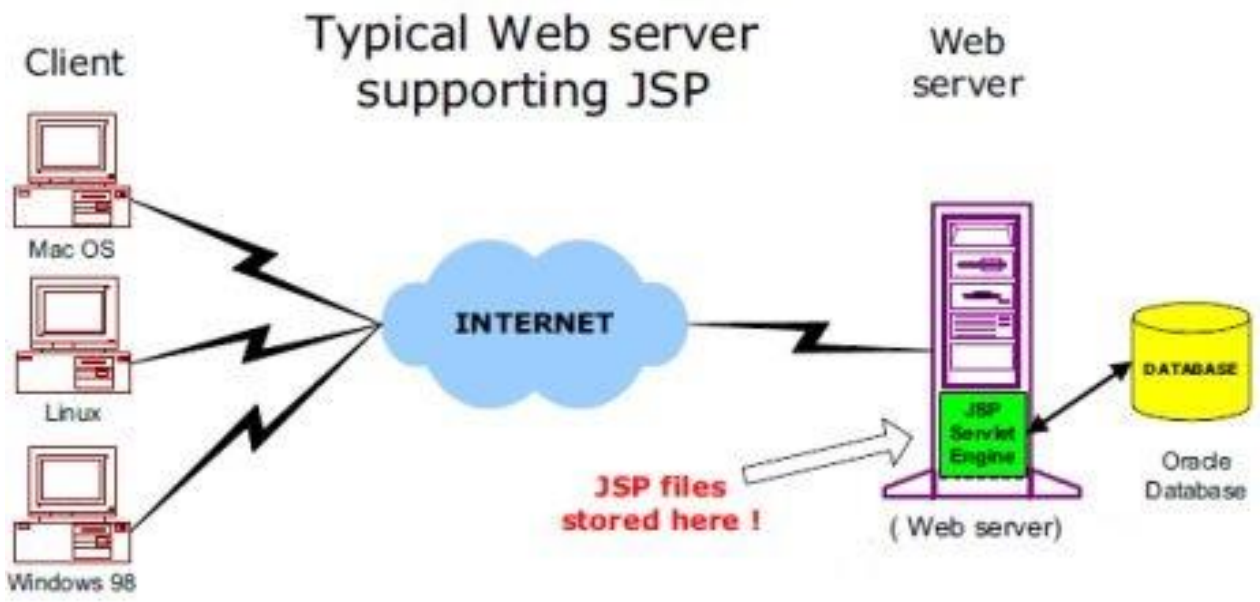
JSP (JavaServer Pages) (cont'd)

- **<% …. %>, <%= … %>**
  - JSP 문법
  - 이 태그 안에 자바 프로그래밍 언어로 작성된 코드가 있음
  - 이 코드는 웹 서버에서 실행됨
  - <%..%> : 1부터 100까지의 수를 더해서 total 변수에 대입
  - <%= …%> : total 변수의 값을 웹 브라우저로 전송

- **웹 디자이너가 그대로 가져다가 디자인 작업을 할 수 있고, 디자인 작업이 끝나면 프로그래머가 다시 가져다가 자바 부분을 수정할 수 있음**

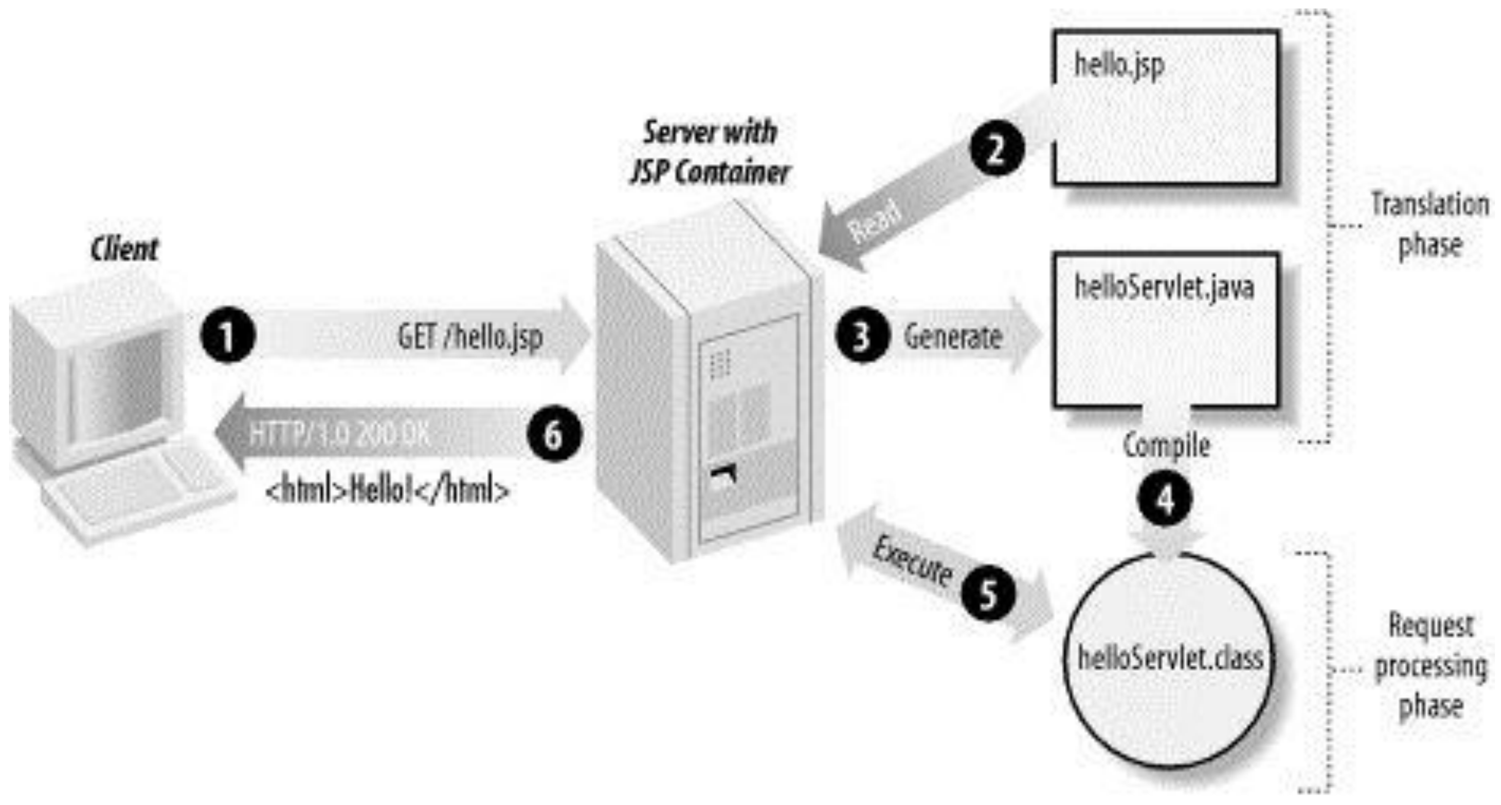
Architecture

- **The web server needs a JSP engine ie. container to process JSP pages. JSP container is responsible for intercepting requests for JSP pages.**
  - Apache has built-in JSP container(Tomcat) to support JSP pages development



Architecture (cont'd)

- JSP Processing



Architecture (cont'd)

- 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.
  - All template text is converted to println( ) statements
  - 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.

Architecture (cont'd)

- 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.

JSP 기초 문법

- **JSP 문법의 형태**
  1. **<%...... %>**
  2. ${.........}
  3. XML 태그 형태 : <jsp:forward>나 <c:if>

- **<%....%>로 끝나는 문법**
  - 지시자(directive)
  - 스크립팅 요소(scripting elements) : scriptlet과 expression

- **${...}로 끝나는 문법**
  - 익스프레션 언어(expression language)

- **XML 태그 형태로 기술**
  - 액션(Action)

JSP의 문법 - 지시자와 스크립트 요소 예

지시자(directive)

```
<%@  page   contentType="text/html;  charset=UTF-8"%>
<HTML>
            <HEAD><TITLE>1부터 100까지의 합</TITLE></HEAD>
            <BODY>
              <%
                        int total = 0;
                        for (int cnt=1; cnt <= 100; cnt++)
                      total += cnt;
              %>
1부터 100까지의 합은?       <%=total %>
</BODY>
</HTML>
```

스크립틀릿(scriptlet)

익스프레션(expression)

Syntax :Scriptlet

- **The Scriptlet**
  - Contain any number of JAVA language statements, variable or method declarations, or expressions that are valid in the page scripting language.
  - The syntax of Scriptlet

```
<% code fragment %>
```

  - Example   *Try it!*

```
<html>
<head><title>Hello World</title><head>
<body>
Hello World!<br>
<%     out.println("Your IP address is " + request.getRemoteAddr());   %>
</body>
</head>
```

Syntax : JSP Declaration

- **JSP Declarations**
  - Declares one or more variables or methods that you can use in Java code later in the JSP files.
  - The syntax of JSP Declarations

```
<%!   declaration;  [declaration; ] + …. %>
```

  - Example

```
<%!  int  i=0; %>
<%!  int  a, b, c; %>
<%!  Circle  a= new Circle(2.0) ; %>
```

Syntax : JSP Expression

- **JSP Expression**
  - JSP expression element contains a scripting language expression that is evaluated, converted to a String, and inserted where the expression appears in the JSP file.
  - The syntax of JSP Expression

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

  - Example   *Try it*

```
<html>
<head><title>A Comment Test</title><head>
<body>
<p>   Today's date: <%= (new java.util.Date()).toLocaleString()%>  </p>
</body>
</head>
```

Syntax : JSP Comments

- **JSP Comments**
  - The syntax of JSP Comments

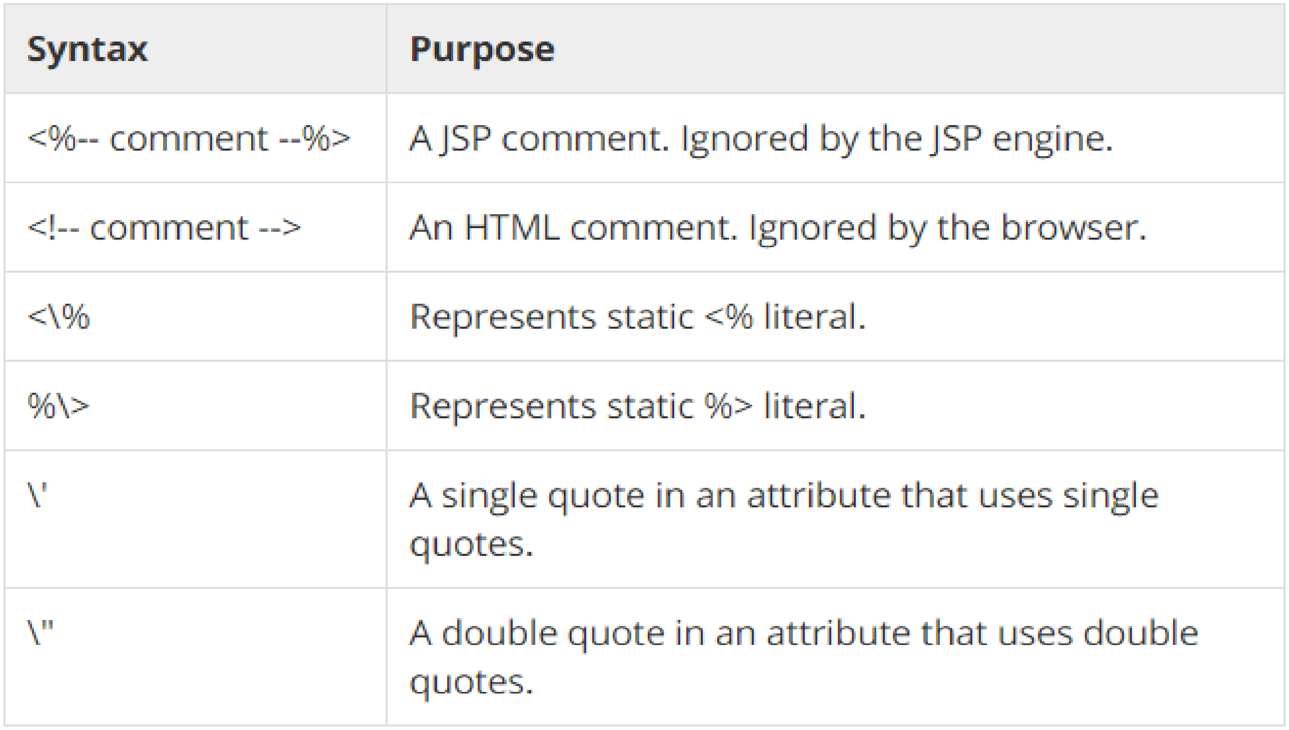
```
<%--  This is JSP comment -- %>
```

  - Example   *Try it!*

```
<html>
<head><title>A Comment Test</title><head>
<body>
<h2> A Test of Comments </h2>
<%-- This comment will not be visible in the page source --%>
</h2>
</body>
</head>
```

Syntax : JSP Comments (cont'd)

- A small numbers of special constructs you can use in various cases to insert comments or characters that would otherwise be treated specially



Syntax	Purpose
<%-- comment --%>	A JSP comment. Ignored by the JSP engine.
<!-- comment -->	An HTML comment. Ignored by the browser.
<\%	Represents static <% literal.
%\>	Represents static %> literal.
\'	A single quote in an attribute that uses single quotes.
\"	A double quote in an attribute that uses double quotes.

Syntax : Control-Flow Statement

- **Control-Flow Statements**
  - You can use all the APIs and building blocks of Java in your JSP programming including decision making statements, loops etc
- **Decision-Making Statements**
  - **If…else** block. *Try it!*

```
<%! int day = 3; %>
<html>
<head><title>IF…ELSE Example</title></head>
<body>
<%  if (day == 1 | day ==7) { %>
            <p> Today is weekend</p>
<% } else {%>
            <p> Today is not weekend</p>
<% } %>
</body>
</html>
```

Syntax : Control-Flow Statement (cont'd)

- **switch…case** block *Try it!*

```
<%! int day = 3; %>
<html>
<head><title>SWITCH…CASE
Example</title></head>
<body>
<%
switch(day) {
case 0:
        out.println("It\'s Sunday.");
        break;
case 1:
        out.println("It\'s Monday.");
        break;
case 2:
        out.println("It\'s Tuesday.");
        break;
```

```
case 3:
        out.println("It\'s Wednesday.");
        break;
case 4:
        out.println("It\'s Thursday.");
        break;
case 5:
        out.println("It\'s Friday.");
        break;
default:
        out.println("It\'s Saturday.");
}
%>
</body>
</html>
```

Syntax : Control-Flow Statement (cont'd)

- **Loop Statements**
  - You can also use three basic types of looping blocks in Java **: for, while, and do…while** blocks in your JSP programming.
  - **for** loop *Try it!*

```
<%! int fontSize; %>
<html>
<head><title>FOR LOOP Example</title></head>
<body>
<%for (fontSize =1;fontSize <=3;fontSize++) { %>
            <font color="green" size="<%=fontSize %>">
            JSP Tutorial
            </font><br>
<% } %>
</body>
</html>
```

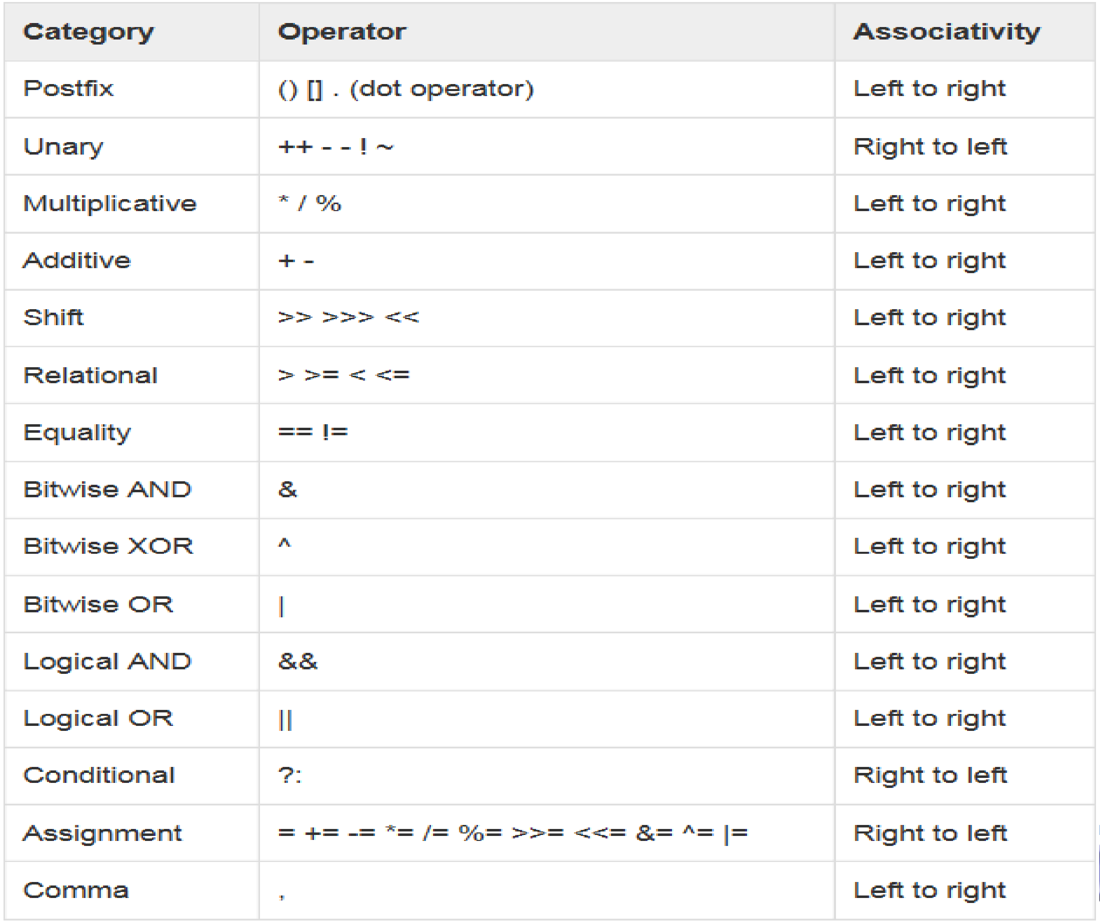
Syntax : Control-Flow Statement (cont'd)

- **while** loop

```
<%! int fontSize; %>
<html>
<head><title>FOR LOOP Example</title></head>
<body>
<%while (fontSize <=3) { %>
        <font color="green" size="<%=fontSize %>">
        JSP Tutorial
        </font><br>
<%fontSize++; %>
<%}%>
</body>
</html>
```

Syntax : Operators

- **JSP Operators**
  - Following table give a list of all the operators with the highest precedence appear at the top of the table, those with the lowest appear at the bottom



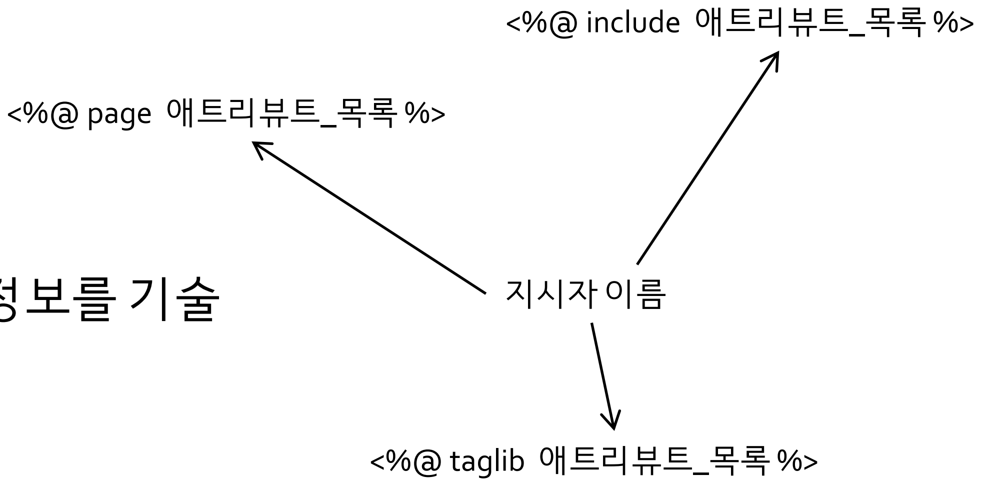
Category	Operator	Associativity
Postfix	() [] . (dot operator)	Left to right
Unary	++ - - ! ~	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	>> >>> <<	Left to right
Relational	> >= < <=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR	|	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %= >>= <<= &= ^= |=	Right to left
Comma	,	Left to right

Syntax : Literals

- JSP Literals
  - Boolean : true and false
  - Integer : as in Java
  - Floating point : as in Java
  - String : with single and double quotes; " is escaped as \", 'is escaped as \', and \ is escaped as \\
  - Null : null

지시자의 문법

- **목적**
  - 웹 컨테이너가 JSP 페이지를 서블릿 클래스로 변환할 때 필요한 여러 가지 정보들을 기술하기 위해 사용하는 문법
- **구성**
  - page 지시자
    - JSP 페이지 전체에 적용되는 정보를 기술
  - include 지시자
    - JSP 페이지에 포함시킬 파일
  - taglib 지시자
    - 해당 페이지가 taglib library를 사용할 것인지의 여부
    - Tag library의 URI

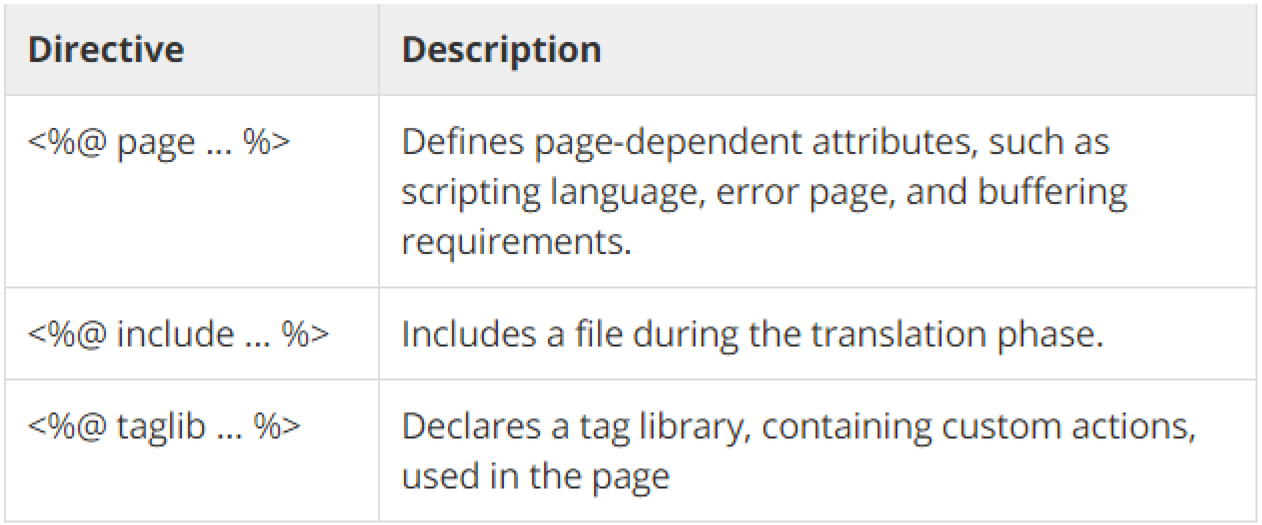

<%@ include 애트리뷰트_목록 %>
<%@ page 애트리뷰트_목록 %>
지시자 이름
<%@ taglib 애트리뷰트_목록 %>

Syntax : JSP Directives

- **JSP Directives**
  - JSP directive affects the overall structure of the servlet class
  - Usually has the following form

```
<%@  directive attribute ="value" %>
```

  - Three types of directive tag



Directive	Description
<%@ page ... %>	Defines page-dependent attributes, such as scripting language, error page, and buffering requirements.
<%@ include ... %>	Includes a file during the translation phase.
<%@ taglib ... %>	Declares a tag library, containing custom actions, used in the page

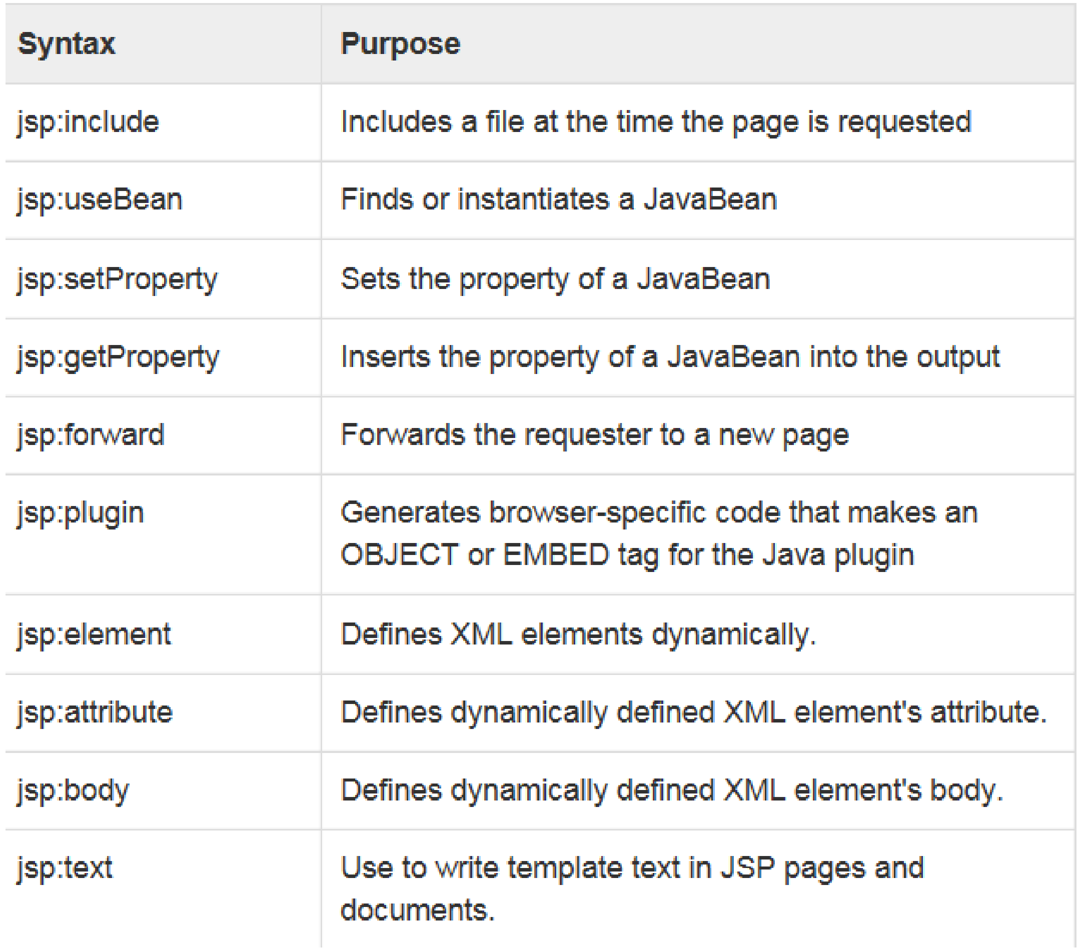
Syntax : JSP Actions

- JSP Actions
  - Use constructs in XML syntax to control the behavior of the servlet engine.
  - You can dynamically insert a file, reuse JavaBeans components, forward the user to another page, or generate HTML for the Java plugin
  - Syntax

```
<jsp:action_name  attribute="value">
```

Syntax : JSP Actions (cont'd)

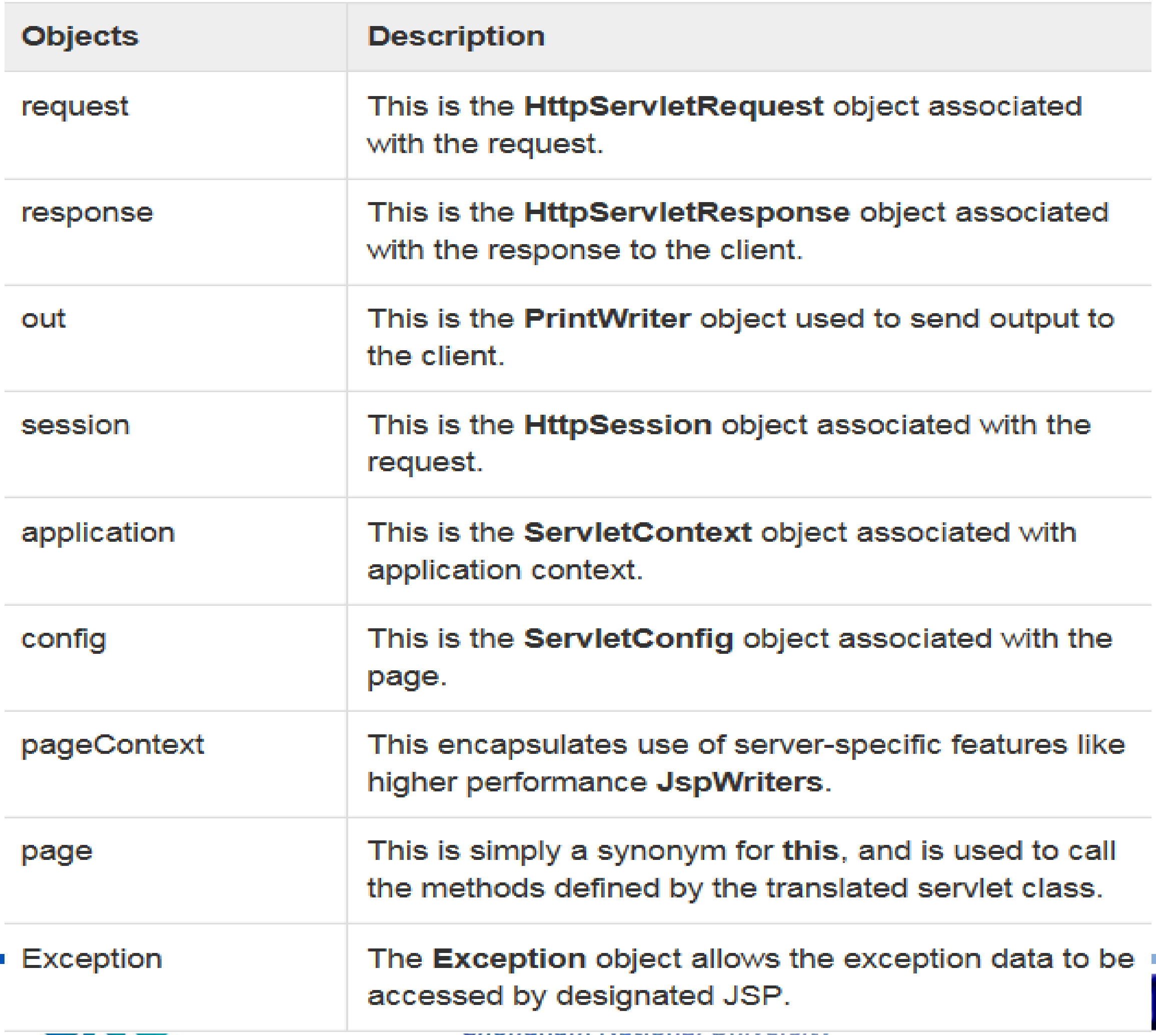
- Basically predefined functions



Syntax	Purpose
jsp:include	Includes a file at the time the page is requested
jsp:useBean	Finds or instantiates a JavaBean
jsp:setProperty	Sets the property of a JavaBean
jsp:getProperty	Inserts the property of a JavaBean into the output
jsp:forward	Forwards the requester to a new page
jsp:plugin	Generates browser-specific code that makes an OBJECT or EMBED tag for the Java plugin
jsp:element	Defines XML elements dynamically.
jsp:attribute	Defines dynamically defined XML element's attribute.
jsp:body	Defines dynamically defined XML element's body.
jsp:text	Use to write template text in JSP pages and documents.

Syntax : Implicit Object

- **JSP Implicit Objects**



Objects	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.