Java Web Programming 입문

(Servlet/JSP)

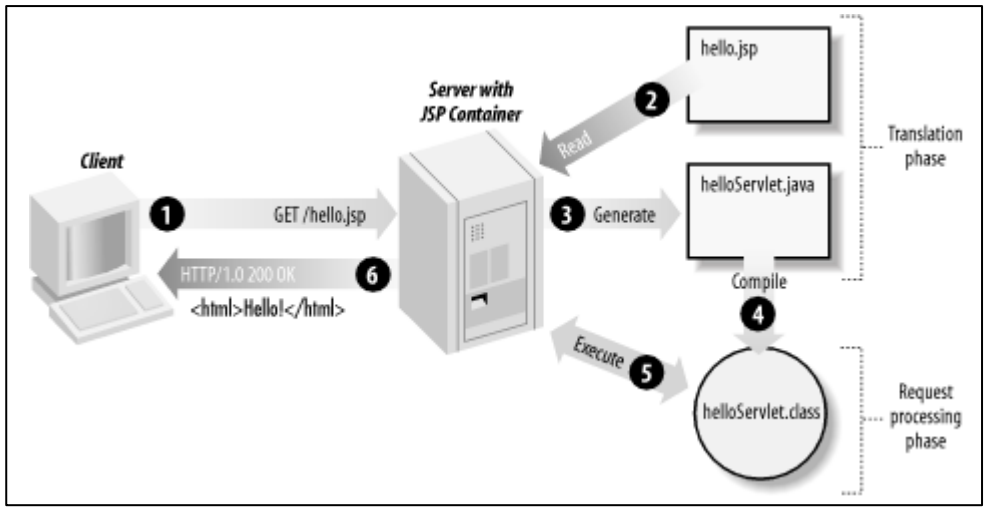
22일차

오늘의 키워드

- JSP
  - JSP의 동작
  - JSP의 생명주기
  - JSP 파일
  - JSP 작성
    - JSP -> Java
  - 지시문 (Directive)
  - Script 요소
    - Script 요소 -> Servlet

JSP

- JSP의 동작
  - Client의 요청에 해당하는 Object를 Servlet Container 에서 검색
  - Client의 요청을 처리할 Object가 없을 경우 JSP 파일을 Servlet 파일로 변환
  - Servlet 파일을 Class 파일로 Compile
  - Compile된 Object를 Loading 후 요청 처리

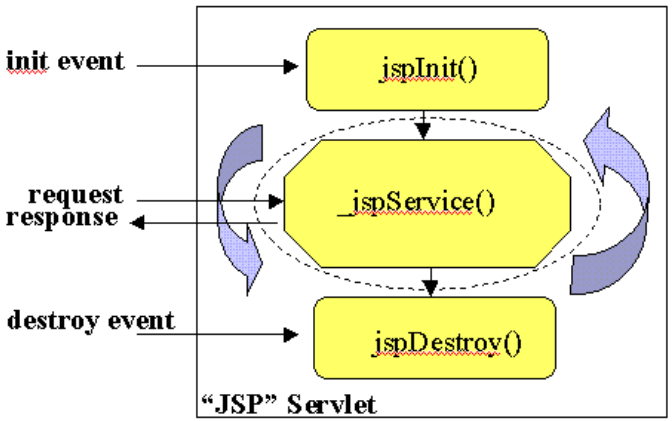


JSP

- JSP의 생명주기
  - Servlet은?
    - init(), service(), destroy()
  - JSP는?
    - jspInit(), _jspService(), jspDestroy()
    - jspInit() Method
      - JSP에 첫 요청이 왔을때 Servlet으로 변환 후, 한번 호출
      - Servlet의 초기화 및 Service를 시작하직 위한 준비
      - _jspService() Method 호출
    - _jspService() Method
      - Client 요청을 처리하는 Method
      - GET / POST 방식 처리
    - jspDestroy() Method
      - Servlet Object가 Memory에서 제거될때 호출
    - jspInit(), jspDestroy() Method는 한번 호출
    - _jspService() Method는 Client의 요청이 있을때마다 호출

JSP

- JSP 동작과 파일
  - JSP 실행 후 생성 파일
    - 자바(.java) 파일
      - JSP Page가 Servlet 형태의 Java 파일로 변경된 파일
    - 클래스(.Class) 파일
      - Java 파일이 Compile된 후 생성된 파일
  - Java 파일과 Class 파일
    - …/work/Catalina/localhost/



JSP

- 지시문(Directive)
  - JSP Page에 대한 설정 정보를 지정
  - page 지시문
    - JSP Page의 MIME Type
    - Encoding 정보

```
<%@ page contentType = "text/html; charset=euc-kr" %>
```

- JSP Page내 HTML

```
<html>
    <head>
        <title>jsp 페이지 구성</title>
    </head>
    <body>
    ...
    </body>
</html>
```

- 스크립틀릿 (Scriptlet)

```
<%
    String name = "홍길동";
    String job  = "학생"  ;
%>
```

- 표현식 (Expression)
  - 변수의 값이나 Method의 Return 값을 출력
  - <%= … %>

```
<%= name %>은 <%= job %> 입니다.
```

JSP

- basicTest.jsp

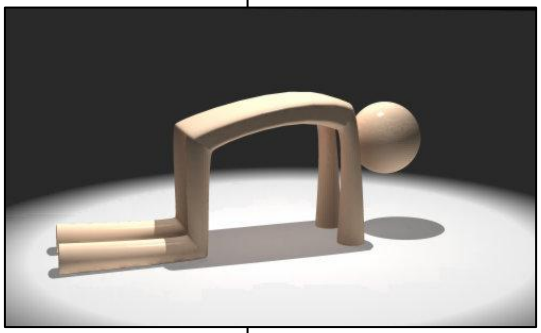
```jsp
<%@ page contentType = "text/html; charset=euc-kr" %>

<html>
    <head>
        <title>jsp 페이지 구성</title>
    </head>
    <body>
        <%
            String name = "홍길동";
            String job  = "학생"  ;
        %>

        <%= name %>은 <%= job %> 입니다.
    </body>
</html>
```

  - ../webapps/MySample/
  - http://localhost:8080/MySample/bastTest.jsp
  - http://127.0.0.1:8080/MySample/bastTest.jsp
  - ../work/Catalina/localhost/MySample/org/apache/jsp

JSP

- basicTest_jsp.java

```
package org.apache.jsp;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;

public final class basicTest_jsp extends org.apache.jasper.runtime.HttpJspBase implements org.apache.jasper.runtime.JspSourceDependent {
    private static java.util.List _jspx_dependants;

    public Object getDependants() {
        return _jspx_dependants;
    }

    public void _jspService(HttpServletRequest request, HttpServletResponse response) throws java.io.IOException, ServletException {
        JspFactory _jspxFactory = null;
        PageContext pageContext = null;
        HttpSession session = null;
        ServletContext application = null;
        ServletConfig config = null;
        JspWriter out = null;
        Object page = this;
        JspWriter _jspx_out = null;
        PageContext _jspx_page_context = null;

        try {
            _jspxFactory = JspFactory.getDefaultFactory();
            response.setContentType("text/html; charset=euc-kr");
            pageContext = _jspxFactory.getPageContext(this, request, response, null, true, 8192, true);
            _jspx_page_context = pageContext;
            application = pageContext.getServletContext();
            config = pageContext.getServletConfig();
            session = pageContext.getSession();
            out = pageContext.getOut();
            _jspx_out = out;

            out.write("<html><head><title>jsp 페이지 구성</title></head><body>\r\n");

            String name = "홍길동";
            String job  = "학생";

            out.write('\r');
            out.write('\n');
            out.print( name );
            out.write("은 ");
            out.print( job );
            out.write(" 입니다.\r\n");
            out.write("</body></html>");
        } catch (Throwable t) {
            if (!(t instanceof SkipPageException)){
                out = _jspx_out;
                if (out != null && out.getBufferSize() != 0)
                    out.clearBuffer();
                if (_jspx_page_context != null) _jspx_page_context.handlePageException(t);
            }
        } finally {
            if (_jspxFactory != null) _jspxFactory.releasePageContext(_jspx_page_context);
        }
    }
}
```



JSP

- basicTest_jsp.java 뜯어보기
  - Package 선언

```
package org.apache.jsp;
```

  - Class Import

```
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;
```

  - basicTest_jsp Class

```
public final class basicTest_jsp
    extends org.apache.jasper.runtime.HttpJspBase
    implements org.apache.jasper.runtime.JspSourceDependent {
```

  - _jspService() Method
    - Client의 요청에 의해 실행
    - Servlet에서의 service() Method와 동일한 역할
    - Client의 요청방식인 GET / POST 모두 처리 가능

```
public void _jspService ( HttpServletRequest request,
                          HttpServletResponse response )
            throws java.io.IOException, ServletException {
   ...
}
```

JSP

- basicTest_JSP.java 뜯어보기
  - 내장객체 (Implicit / Internal / Built-in - Object)
    - JSP Page에서 기본으로 제공되어, 선언하지 않아도 바로 쓸 수 있는 Object

```
out.println("out 객체는 내장 객체 입니다.");
```

    - 대표적인 내장객체 out Object
      - out Object는 선언없이, 출력문 작성을 위해 사용 가능
    - 그러나?? 그딴건 Java World 에선 사실 불가능하다!

```
JspFactory      _jspxFactory          = null;
PageContext     pageContext           = null;
HttpSession     session               = null;
ServletContext  application           = null;
ServletConfig   config                = null;
JspWriter       out                   = null;
Object          page                  = this;
JspWriter       _jspx_out             = null;
PageContext     _jspx_page_context    = null;
```

JSP

- basicTest_JSP.java 뜯어보기
  - 내장객체 (Implicit / Internal / Built-in - Object)
    - 게다가?? 초기화까지!

```
_jspxFactory          = JspFactory.getDefaultFactory();
pageContext           = _jspxFactory.getPageContext(this, request, response, null, true, 8192, true);
_jspx_page_context    = pageContext;
application           = pageContext.getServletContext();
config                = pageContext.getServletConfig();
session               = pageContext.getSession();
out                   = pageContext.getOut();
_jspx_out             = out;
```

    - 이렇게 선언/초기화된 Object 변수들은
      - JSP Page 내 <% ~ %> Java Code에서 선언없이 사용 가능

JSP

- basicTest_JSP.java 뜯어보기
  - JSP Page -> Java Code

```
<%@ page contentType = "text/html; charset=euc-kr" %>
```

```
response.setContentType("text/html; charset=euc-kr");
```

```
<html>
    <head>
        <title>jsp 페이지 구성</title>
    </head>
    <body>
    ...
    </body>
</html>
```

```
response.setContentType("text/html; charset=euc-kr");


out.write("<html>\r\n");
out.write("\t<head>\r\n");
out.write("\t\t<title>jsp 페이지 구성</title>\r\n");
out.write("\t</head>\r\n");
out.write("\t<body>\r\n");
...
out.write("\t</body>\r\n");
out.write("</html>");
```

JSP

- basicTest_JSP.java 뜯어보기
  - JSP Page -> Java Code

```
<%
   String name = "홍길동";
   String job  = "학생"  ;
%>
```

```
String name = "홍길동";
String job  = "학생";
```

```
<%= name %>은 <%= job %> 입니다.
```

```
out.print( name );
out.write("은 ");
out.print( job );
out.write(" 입니다.\r\n");
```

- 결론
  - JSP 내 HTML, <% ~ %> 등의 모든 Code 들이 _jspService() 내 작성
  - Servlet
    - HttpServlet Class 상속
    - init(), service(), destroy() Method에 의해 실행
  - JSP
    - HttpJspBase Class 상속
    - jspInit(), _jspService(), jspDestroy() Method 에 의해 실행

JSP

- 지시문 (Directive)
  - JSP Page 내 속성 지정, 특정 Page 포함과 같은 Page 차원의 작업을 위해 사용
  - Client 요청에 의해 JSP Page가 실행될 때 필요한 정보를 기술하기 위해 사용

```
<%@ 지시문_이름  속성_이름="속성_값"    속성_이름="속성_값"  ... %>
```

- 지시문(Directive)의 종류
  - page
    - JSP Page의 속성을 지정
    - language, import, extends, etc…

```
<%@ page contentType = "text/html; charset=euc-kr" %>
```

  - include
    - 다른 JSP 파일을 현재의 Page에 삽입
    - 대상 JSP Page 내 구문들이 현재 Page에 그대로 포함되어 실행

```
<%@include file="include.jsp"%>
```

  - taglib
    - JSP Page가 사용할 '사용자 정의 태그' (Custom Tag) 를 명시
      - Servlet Container 에게
    - Custom Tag를 정의한 태그 라이브러리 서술파일 (TLD, Tag Library Description)의 URI를 지정

```
<%@taglib uri="MySample/sampletag" prefix="samplePrefix"%>
```

JSP

- page 지시문(Directive)
  - JSP Page에 대한 정보를 지정
  - 문서 Type, Class Import, Error Page, etc…
  - 문서 맨 윗줄에 작성

```
<%@ page    속성_이름="속성_값"  속성_이름="속성_값"    ... %>
```

  - 속성

속성	설명	기본값
language	스크립트 내에 사용되는 언어 명을 지정	"Java"
contentType	페이지의 MIME 타입과 문자 인코딩을 지정	"text/html;charset=ISO-8859-1"
extends	상속받는 클래스를 지정	없음
import	JSP에서 사용할 자바 클래스를 지정	없음
info	JSP 페이지에 대한 설명을 입력	없음
session	세션의 사용여부 결정	Session = "true"
buffer	버퍼의 크기를 지정	Buffer = "8kb"
autoFlush	자동 flush 기능	autoFlush = "true"
errorPage	에러 페이지 지정	없음
isErrorPage	에러 페이지 사용여부 결정	isErrorPage = "true"
pageEncoding	페이지의 캐릭터 인코딩 값	없음
isThreadSafe	SingleThreadModel Interface 구현 여부 결정	isThreadSafe = "false"

JSP

- page 지시문(Directive)의 속성
  - language
    - JSP Page의 Programming Language 지정
      - 사실상 java 라고 밖에 쓰지 않음
      - language = “java”
  - contentType
    - Clinet에 응답할 문서의 MIME Type과 Encoding 지정
    - contentType=“text/html;charset=ISO=8859-1”
  - extends
    - JSP Page 요청시 생성되는 Servlet의 Parent Class 지정
    - extends = “parent.class”
    - Servlet Container 제품군에서는 알아서 적합한 Parent Class 사용하여 동적으로 Servlet을 생성
      - 사실상 쓸일이 없는 속성
  - Info
    - JSP Page에 대한 설명을 지정
    - getServerInfo() Method를 이용하여 지정한 정보 출력가능
    - info = “jsp 예제1”
  - import
    - JSP Page가 요청되어 변환된 Servlet이 import 할 Class를 지정
    - java.lang.*, javax.servlet.*, javax.servlet.jsp.*, javax.servlet.http.* 외
    - import = “java.util.*”

JSP

- page 지시문(Directive)의 속성
  - session
    - JSP Page가 HttpSession을 사용할지 여부 지정
      - true / false
      - true
        - Session이 이미 존재한다면 Session을 유지
        - Session이 존재하지 않는다면 새로운 Session을 생성해서 연결
      - false
        - Session 연결되지 않음
  - buffer
    - Client로 전송을 담당하는 out Object(JspWriter)의 Buffer Size를 설정
    - 지정된 Buffer Size 만큼의 Data 단위로 Client에 전송
    - buffer = “30kb”, buffer=“none”
  - autoflush
    - 출력 Buffer의 용량이 다 찼을때 제어할 방법을 지정
    - true / false
    - true
      - 출력 Buffer를 Client로 보낸 후, Buffer를 비움
    - false
      - Buffer를 비우지 않고 Exception 발생시킴
    - buffer 속성의 값이 none 일 경우, autoflush 속성은 true가 될 수 없음

JSP

- page 지시문(Directive)의 속성
  - isThreadSafe
    - JSP에서 만들어진 Servlet이 SingleThreadModel Interface를 구현하는지 여부 지정
      - SingleThreadModel은 Servlet에서 Thread로 자원에 접근할 때 공유로 인한 문제점이 발생하면 한번에 하나의 Thread만 자원에 접근하도록 하기위한 방법
      - true (기본값) / false
      - false
        - SingleThreadModel 구현하려면 false
  - errorPage
    - JSP Page에서 Exception이 발생할 경우 해당 Error를 처리할 Page를 지정
    - Exception이 발생하면 errorPage 속성에 지정된 페이지로 exception Object가 Error을 가진고 넘어감
    - errorPage = “errManage.jsp”
  - isErrorPage
    - JSP Page가 Error처리를 위한 Page인지를 지정
    - errPage 속성으로 Exception을 넘겼을때, 그것을 받아 처리할 것인가 여부 지정
    - isErrorPage = “true”
  - pageEncoding
    - JSP Page의 문자 Encoding 기술
    - contentType 속성을 이용한 Encoding은 Response Message에 대한 Encoding
    - pageEncoding 속성을 이용한 Encoding은 JSP Page를 읽을 때의 Encoding 밥벙 지정
    - pageEncoding = “euc-kr”

JSP

- page 지시문(Directive) contentType
  - JSP Page의 MIME Type 및 문자 Encoding 지정

```
<%@ page    contentType="MIME_타입;charset=인코딩_정보"%>
```

    - MIME Type
      - 현재 문서의 Contents 를 나타내는 속성
      - JSP Page는 Text형태의 HTML 문서
        - 보통 “text/html” 값을 가짐
    - Encoding
      - Response Message의 Encoding 방법
      - 문자 해석 방식을 지정
      - JSP Page에 한글을 사용할때는 “euc-kr” 사용
    - contentType 속성의 기본 값
      - text/html;charset=ISO-8859-1

JSP

- wrong_encoding.jsp

```
<%@ page contentType="image/gif; charset=iso-8859-1"%>

<html>
    <head>
        <title>잘못된 한글 인코딩</title>
    </head>
    <body>
        <h2>한글이 안보여요.</h2>
    </body>
</html>
```

  - HTTP Header
    - Content-Type: image/gif;charset=iso-8859-1

- right_encoding.jsp

```
<%@ page contentType="text/html; charset=euc-kr"%>

<html>
    <head>
        <title>제대로 작성한 한글 인코딩</title>
    </head>
    <body>
        <h2>한글이 너무 잘 보여요.</h2>
    </body>
</html>
```

  - Http Header
    - Content-Type: text/html;charset=euc-kr

JSP

- page 지시문(Directive) import
  - JSP Page 내 사용할 Java Class를 import
  - Servlet이 기본적으로 import
    - java.lang.*
    - javax.servlet.*
    - javax.servlet.jsp.*
    - javax.servlet.http.*
    - 이 외에 필요한 Class/Package는 import

```
<%@ page   import="사용할 패키지, 사용할 패키지, ..."%>
```

JSP

- use_import.jsp

```jsp
<%@ page contentType="text/html; charset=euc-kr"%>
<%@ page import="java.util.*"%>
<html>
    <head>
        <title>page 지시문의 import 속성</title>
    </head>
    <body>
    <%
        String str = "Hello JSP~~~!";
        Vector vector = new Vector();
        vector.addElement(str);
        out.println("<h2>" + vector.elementAt(0) + "</h2>");
    %>
    </body>
</html>
```

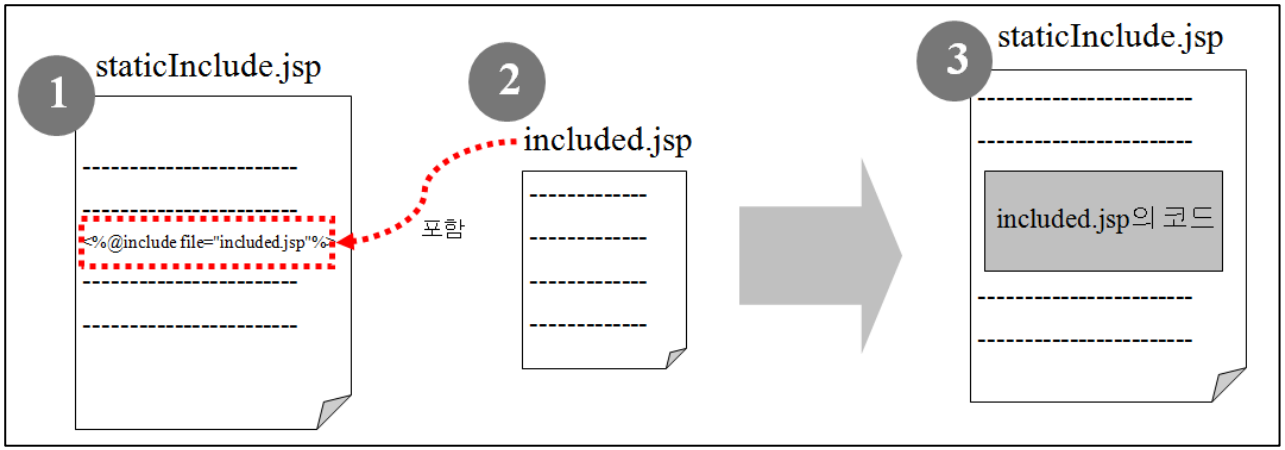
```jsp
<%@ page import="java.util.*"%>
```

JSP

- page 지시문(Directive) include
  - 현재 Page에 포함될 파일을 Text 형태 그대로 삽입 후 처리

```
<%@ include   file="include될 파일의 URL"%>
```

```
<%@include file="included.jsp"%>
```



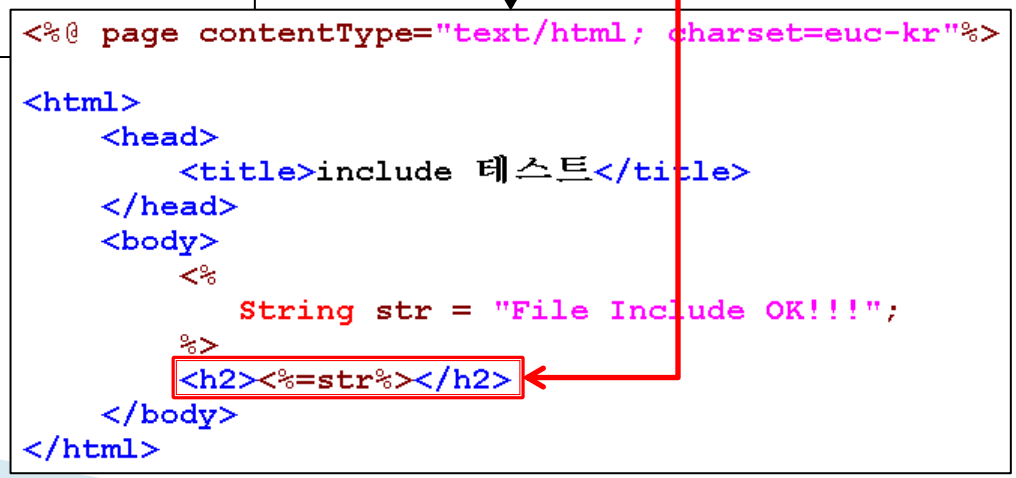
JSP

- included.jsp

```jsp
<h2><%=str%></h2>
```

- staticInclude.jsp

```jsp
<%@ page contentType="text/html; charset=euc-kr"%>

<html>
    <head>
        <title>include 테스트</title>
    </head>
    <body>
        <%
            String str = "File Include OK!!!";
        %>
        <%@include file="included.jsp"%>
    </body>
</html>
```



```jsp
<%@ page contentType="text/html; charset=euc-kr"%>

<html>
    <head>
        <title>include 테스트</title>
    </head>
    <body>
        <%
            String str = "File Include OK!!!";
        %>
        <h2><%=str%></h2>
    </body>
</html>
```

JSP

- Script 요소
  - JSP는 Server측에서 실행되는 Script Language
  - JSP는 Script 요소들을 이용하여 Java Code 작성
  - JSP 의 핵심 요소
    - Script 요소로 작성된 Code는
      - Clietn의 요청에 의해 Servlet Code로 자동 변환
  - JSP의 Script 요소
    - 스크립틀릿 (Scriptlet)
      - JSP의 Java Code를 작성
      - <% ~ %>
    - 선언문 (Declaration)
      - JSP Page 전체에서 참조될 Member 변수 / Member Method 선언
      - <%! ~ %>
    - 표현식 (Expression)
      - Code에 사용된 변수의 값을 출력하기 위해 사용
      - <%= ~ %>

JSP

- Script 요소 -> Serlvet
  - 지시문 (Directive) <%@ ~ %>

```
<%@ page contentType = "text/html; charset=euc-kr" %>
```

  - 선언문 (Declaration) <%! ~ %>
    - Page 전체에서 사용될 Member 변수 / Member Method 선언

```
<%!
    String name = "jsp";
    int a = 5;
    int b = 15;
    public int sum(){
        return a + b;
    }
%>
```

  - 표현식 (Expression) <%= ~ %>

```
<%= sum() %>
```

```
<% out.print(sum()); %>
```

  - 스크립틀릿 (Scriptlet) <% ~ %>

```
<%
    int i;
    for(i=0; i<2; i++) {
%>
```

```
<%= name %>
```

```
<%
        }
%>
```

JSP

- scriptServlet

```
<%@ page contentType="text/html; charset=euc-kr" %>
<html>
    <body>
    <%!
        String name = "jsp";
        int a = 5;
        int b = 15;

        public int sum(){
          return a + b;
        }
    %>
    <%= sum() %><p>
    <%
        int i;
        for(i=0; i<2; i++) {
    %>
        <%= name %><p>
    <%  }   %>
    </body>
</html>
```

JSP

- scriptServlet_jsp.java

```
package org.apache.jsp;
import javax.servlet.*; import javax.servlet.http.*;    import javax.servlet.jsp.*;

public final class scriptServlet_jsp extends org.apache.jasper.runtime.HttpJspBase implements org.apache.jasper.runtime.JspSourceDependent {
    String name = "jsp";    int a = 5;  int b = 15;
    public int sum(){
        return a + b;
    }
    private static java.util.List _jspx_dependants;
    public Object getDependants() {
        return _jspx_dependants;
    }
    public void _jspService(HttpServletRequest request, HttpServletResponse response) throws java.io.IOException, ServletException {
        JspFactory _jspxFactory = null;     PageContext pageContext = null; HttpSession session = null;
        ServletContext application = null;  ServletConfig config = null;    JspWriter out = null;
        Object page = this;                 JspWriter _jspx_out = null;     PageContext _jspx_page_context = null;
        try {
            _jspxFactory = JspFactory.getDefaultFactory();  response.setContentType("text/html; charset=euc-kr");
            pageContext = _jspxFactory.getPageContext(this, request, response, null, true, 8192, true);
            _jspx_page_context = pageContext;   application = pageContext.getServletContext();
            config = pageContext.getServletConfig();    session = pageContext.getSession();
            out = pageContext.getOut(); _jspx_out = out;
            out.write("<html><body> \r\n"); out.write("\r\n");  out.print( sum() ); out.write("<p> \r\n");
            int i;
            for(i=0; i<2; i++) {
                out.write("\r\n");  out.write(" \t");   out.print(name);    out.write("<p> \r\n");
            }
            out.write("</body></html>");
            } catch (Throwable t) {
                if (!(t instanceof SkipPageException)){
                    out = _jspx_out;
                    if (out != null && out.getBufferSize() != 0)    out.clearBuffer();
                    if (_jspx_page_context != null) _jspx_page_context.handlePageException(t);
                }
            } finally {
                if (_jspxFactory != null) _jspxFactory.releasePageContext(_jspx_page_context);
        }
    }
}
```

JSP

- Script 요소 -> Serlvet
  - 선언문 (Declaration)

```
<%!
    String name = "jsp";
    int a = 5;
    int b = 15;
    public int sum(){
        return a + b;
    }
%>
```

```
public final class scriptServlet_jsp
                extends ...
                implements ...        {
    String name = "jsp";
    int a = 5;
    int b = 15;
    public int sum(){
        return a + b;
    }
    ...
}
```

    - 선언문을 제외한 나머지 Script 요소는 _jspService() Method 내부에 작성
  - 지시문 (Directive)

```
<%@ page contentType = "text/html; charset=euc-kr" %>
```

```
response.setContentType("text/html; charset=euc-kr");
```

  - 스크립틀릿 (Scriptlet) 과 표현식 (Expression)

```
<%= sum() %>
<%
    int i;
    for(i=0; i<2; i++) {
%>
    <%= name %>
<%  }   %>
```

```
out.print( sum() );
int i;
for(i=0; i<2; i++) {
    out.print(name);
}
```

JSP

- Script 요소 -> Serlvet
  - 선언문 (Declaration)
    - JSP에서 사용할 Member 변수 / Member Method 선언
    - Servlet 변환 시 Class의 Member 변수 / Member Method로 선언
    - 현재는 자주 사용되지는 않음 (Java Beans 이용)
  - 스크립틀릿 (Scriptlet)
    - Java Code 작성
    - Servlet 변환 시 _jspService() Method 내부에 작성
  - 표현식 (Expression)
    - 선언된 변수, Method의 Return값을 출력
    - 사실은 out.print 구문

오늘 숙제

- 집에서 해보자!