JSP 개요

1. JSP(Java Server Pages)는 Java를 이용하여 **동적인 웹 페이지를 만들기 위해 선 마이크로 시스템사에서 개발한 서버 스크립트 언어**이다.

2. Java EE
  2-1 Java EE(Java Enterprise Edition)는 자바의 기본적인 기능을 정의한 Java SE에 웹 서버 역할을 추가한 것으로 자바 웹애플리케이션을 동작하게 하는 컨테이너(톰캣) 등을 표준화한 것이다.

  2-2 Java EE 기술로는 웹 프로그래밍에서 사용하는 jsp, Servlet과 비즈니스 모듈, 데이터베이스를 연동하는 JDBC, 서버의 자원을 관리하는 JNDI 를 통한 커넥션 풀 관리등이 있다. JNDI(Java Naming and Directory Interface)는 디렉터리 서비스에서 제공하는 데이터 및 객체를 발견(discover)하고 참고(lookup)하기 위한 자바 API다.

  2-3 WAS(Web Application Sever) 서버 종류
    2-3-1 상용 제품으로 한국 티맥스 스프트의 제우스, 미국 오라클의 웹로직 등이 있다.
    2-3-2 오픈 소스(무료)로는 아파치 소프트웨어의 아파치 톰캣이 있다.

3. 1999년 썬 마이크로시스템즈에 의해 배포되었으며 이와 비슷한 구조로 PHP, ASP, ASP.NET 등이 있다. 모두 서버쪽에서 실행되는 서버 스크립트 언어이다.
  3-1 jsp는 1998년 첫 번째 API가 발표되었다.

4. jsp는 뷰(View) 페이지를 만들 때 사용한다.

5. JSP 실행순서
  *.jsp -> 서블릿 자바로 변환 : *.java -> 컴파일 -> *.class 서블릿 클래스로 변환된 뒤에 서블릿 컨테이너(톰캣)에 의해서 실행된다.

HTML과 JSP 주석문 기호

1. **html 주석문** : **<!-- html 주석문 -->**
2. **JSP 주석문** : **<%-- jsp 주석문 --%>**

<%@ .. %>

1. **<%@ 를 지시자 또는 지시어**라 한다. **JSP 파일 내에서 jsp를 실행할 때 웹서버에게 해당 페이지를 어떻게 처리할 것인지에 대한 정보를 지정**해 주는 데 사용한다.

JSP에서 **외부 패키지 임포트** 하는법

1. **<%@ page import= " java.util.*" %>** : **java.util패키지의 모든 클래스등을 임포트 즉 읽어 들인다.**

2. **<%@ page import="java.util.*, model.*" %>** : **java.util패키지의 모든 클래스와 model 패키지의 모든 클래스를 쉼표로 구분해서 복수개의 패키지를 임포트** 한다.

page 디렉티브 주요속성

1. session 속성 : session ="true or false"
주어진 jsp 페이지가 세션 관리 처리 여부를 지정하고자 할 때 사용한다. 기본값은 true이다. 그래서 모든 페이지가 자동으로 세션을 생성한다. 특별히 세션 처리를 하지 않게 하려면 속성값을 false로 지정한다.

2. buffer
jsp 페이지의 출력 버퍼 크기를 지정한다. none일때는 출력 버퍼를 사용하지 않으며 ,8kb라고 한 경우 8킬로바이트 크기의 출력 버퍼를 사용한다.

3. autoFlush
출력 버퍼가 다 찼을 경우 자동으로 버퍼에 있는 데이터를 출력 스트림에 보내고 비울지 여부를 나타낸다. true 인 경우는 버퍼의 내용을 웹 브라우저에 보낸 후 버퍼를 비우며, false인 경우는 에러를 발생시킨다. 기본값은 true이다.

4. contentType
"text/html;charset=UTF-8" 은 문서 타입과 문자코드를 지정(파일형식과 언어코딩 타입) 한다.

5. info
jsp 페이지에 대한 설명

6. language ="java"
이 항목을 생략하면 jsp는 기본적으로 자바 언어로 간주한다.

7. isELIgnored
true 일 경우는 표현언어를 해석하지 않고 문자열로 처리하며, ,false 일 경우는 표현언어를 지원한다. 기본값은 false이다.

JSP에서 **외부 jsp파일을 읽어들이는 법**

1. **<%@ include file="header.jsp" %>**: **header.jsp 외부 jsp 파일을 읽어들임**.
2. **<jsp:include page="footer.jsp" />** : **jsp:include 액션태그로 외부 jsp 포함파일을 읽어들임.**

JSP 선언문, 스크립트릿, 표현식 문법(스크립트 요소)

1. **선언문**: **<%!**

```
자료형 변수명 선언;

접근지정자 자료형 사용자 정의 메서드(){

}
%>
```

형식과 같이 주로 **변수 선언 또는 메서드 선언 용도로 선언문이 사용**된다.
즉, **<%! 라고 시작하면 JSP에서는 선언문**이라 한다.

2. **스크립트 릿**: jsp 페이지가 요청될 때마다 수행되어야 하는 자바 코드를 추가하고자 할 때 사용한다.

```
<%
//이 영역을 스크립트 릿이라 한다. 스크립트 릿은 자바 문법을 따라 간다.
//주석문도 자바 주석문이랑 같다.
%>
```

3. **표현식**: <%= 변수명%> 과 같은 방법으로 **<%= 문법을 표현식**이라 한다. 주로 **표현식은 출력용으로 많이 사용**한다.

out.println();

1. **out.pritnln(출력될 값);** => **<%= 표현식 문법과 같이 출력 메서드**로 활용된다.

jsp:forward 액션 태그

1. forward 액션 태그는 다른 페이지로 프로그램의 제어를 이동할 때 사용되는 것으로 페이지의 흐름을 제어한다.

2. forward 액션 태그는 페이지의 흐름을 제어하는 것으로, jsp 페이지 내에서 forward 액션태그를 만나면 지정한 페이지로 이동한다. 사용자가 입력한 값에 따라 각 페이지로 이동해야 할 경우에 사용하면 좋다.

3. forward 액션 태그를 잘 이해하면 모델 2에서 컨트롤러를 훨씬 더 쉽게 이해할 수 있다. 모델 2에서는 컨트롤러가 forward 액션 태그와 같이 페이지 흐름을 제어한다.

4. forward 액션 태그 사용 형식
형식) <jsp:forward page="이동할 페이지명" />
주의사항) 포워드 액션 태그로 페이지를 이동하면 웹브라우저 주소창에 보이는 주소값과 실제 보이는 본문 내용이 다르다. 이 방식은 모델 2와 같다. **forward액션태그는 서블릿에서 RequestDispatcher.forward() 메서드와 기능이 같다.**

5. forward 액션 태그에서 param 액션 태그를 사용해서 프로그램 제어가 이동할 페이지에 피라미터 값을 전달한다.
형식) <jsp:forward page="이동할 페이지명">
<jsp:param name="paramName01" value="값" />
<jsp:param name="paramName02" value="값" />
</jsp:forward>

서블릿 개념

1. 서블릿은 **서버쪽에서 실행되는 자바 기반으로 이루어진 웹 프로그래밍 언어**이다.

2. 서블릿은 **Sun Microsystems 사에서 자바언어를 이용하여 쉽게 웹 어플리케이션을 개발하기 위해 만든 표준API**(라이브러리, 클래스의 집합)이다.

2-1 서블릿1 사양은 Pavni Diwanji가 썬 마이크로시스템즈에서 일하는 동안 개발한 것으로, 버전 1.0은 1997년 6월 완성되었다.

2-2 서블릿은 컨트롤러(Controller)를 만들 때 사용한다.

3. 서블릿 클래스는 **온라인 웹상에서 누구나 다 접속 가능하게 public 접근지정자로 선언**한다.

4. 서블릿 클래스를 만들기 위해서는 **HttpServlet 서블릿 클래스로부터 상속**을 받아야 한다.

5. **get으로 접근할 때는 doGet() 메서드를 오버라이딩 해서 호출**한다.

6. **post로 접근할때는 doPost()메서드를 오버라이딩해서 호출**하면 된다.

7. 서블릿 클래스를 호출해서 실행할때 **get 또는 post 에 상관없이 실행하려면 service() 메서드를 오버라이딩 해서 호출**한다.

8. **HttpServletRequest 서블릿**은 **사용자 폼에서 입력한 정보를 서버로 가져오는 역할을 한다.**

9. **HttpServletResponse 서블릿**은 **서버의 가공된 정보를 사용자 웹브라우저에 응답**할 때 사용한다.

10. **response.setContentType("text/html;charset=UTF-8");**
    => 웹브라우저에 출력되는 파일형식과 언어코딩 타입을 설정

11. **PrintWriter out=response.getWriter();**
    => 출력스트림 객체 **out**생성

12. **request.setCharacterEncoding("UTF-8") 코드는**
    => **폼태그에서 method=post방식으로 서버로 넘어가는 한글자료를 서버에서 받을때 안깨지게 하는 역할**을 한다.
    method=get방식으로 넘어가는 한글 자료는 안깨지게 하는 역할을 못한다.

13. **method=get방식으로 한글 자료가 서버로 넘어갈때 인코딩 되어져서 깨져서 넘어간다.**이 **한글 자료를 서버에서 받을때 안깨지게 할려면 String클래스의 getBytes()메서드를 사용하여 원래 언어코딩 타입인 UTF-8로 바꿔주면 서버에서 받을때 한글이 안깨진다.**

14. **RequestDispatcher 서블릿 특징**

가. **request.setAttribute("키이름", 값);**

메서드로 키이름에 저장된 값을 유지할려면 **RequestDispatcher를 사용해서 뷰 페이지 이동**을 해야한다.

만약 **response.sendRedirect() 메서드로 이동**하면 **기존 url-pattern 주소값**도 **잃어 버리고, 키이름에 저장된 값도 잃어 버린다.**

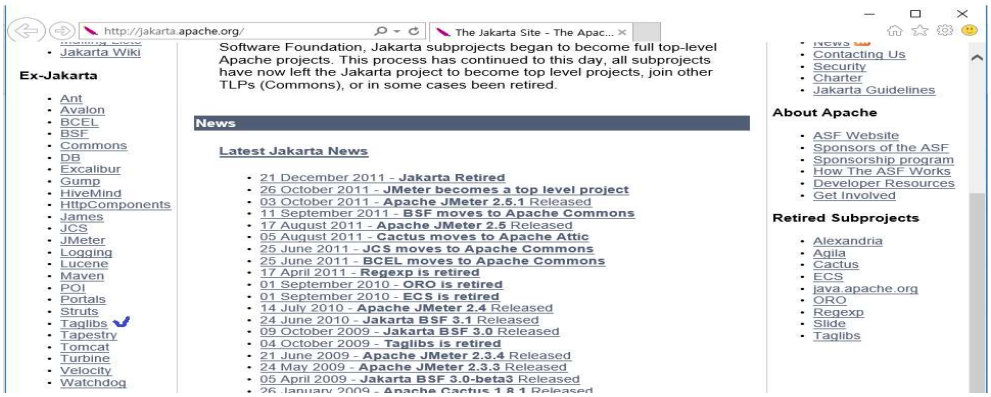
나. 로그인 인증할때 사용하는 **session.setAttribute("키이름", 값) 메서드로 키이름에 저장된 값은 RequestDispatcher을 사용**하나, **response.sendRedirect() 메서드로 이동하나 키이름에 저장된 값을 잃어 버리지 않는다.**

표현 언어(Expression Language:EL) 출력법

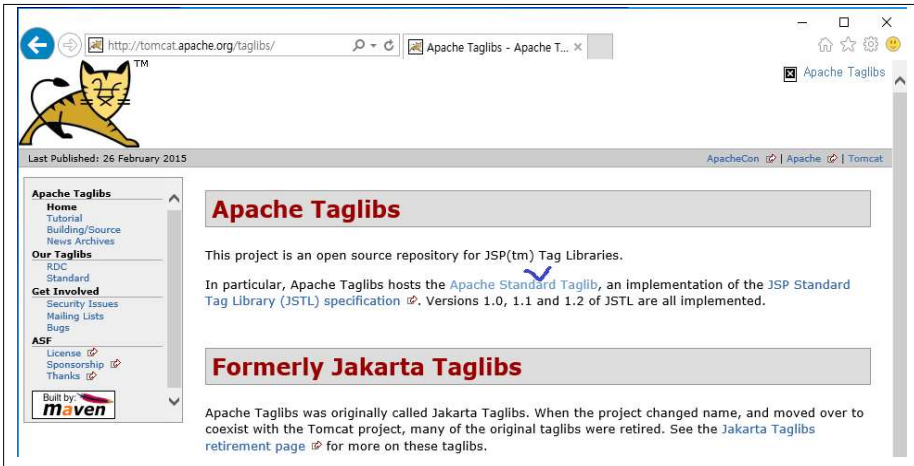
${출력할 값}

JSTL 라이브러리 다운로드 순서

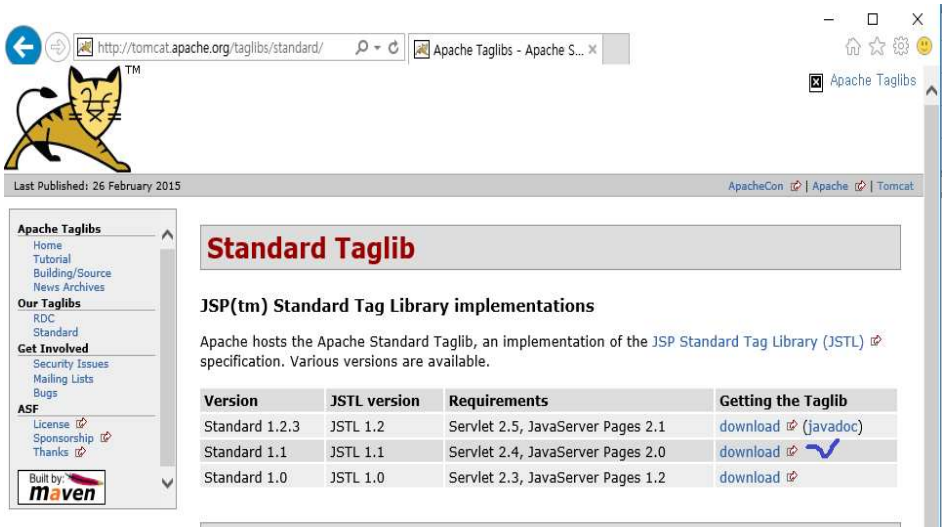
1. http://jakarta.apache.org 사이트로 접속한다.
2. 왼쪽 하단 밑에서 6번째 Taglibs 메뉴를 선택한다.



3. Apache Standard Taglib을 선택한다.



4. Standard 1.1에서 download를 선택한다.



5. binaries 폴더를 선택한다.

6. 아래 메뉴목록에서 밑에서 6번째 압축파일을 다운받아 압축을 푼 후 lib폴더에 있는

jstl.jar와 standard.jar파일을 복사하여 WEB-INF/lib 폴더 하위에 붙여넣기 하면 된다.

jakarta-taglibs-standard-1.1.0.tar.gz.asc	2004-01-28 20:11	304
jakarta-taglibs-standard-1.1.0.zip	2004-01-28 20:11	2.8M
jakarta-taglibs-standard-1.1.0.zip.asc	2004-01-28 20:11	304
jakarta-taglibs-standard-1.1.1.tar.gz	2004-07-19 21:53	872K
jakarta-taglibs-standard-1.1.1.tar.gz.asc	2004-07-19 21:53	186
jakarta-taglibs-standard-1.1.1.zip	2004-07-19 21:53	931K
jakarta-taglibs-standard-1.1.1.zip.asc	2004-07-19 21:53	186
jakarta-taglibs-standard-1.1.2.tar.gz	2004-10-25 20:57	873K
jakarta-taglibs-standard-1.1.2.tar.gz.asc	2004-10-25 20:57	186
jakarta-taglibs-standard-1.1.2.zip	2004-10-25 20:57	933K
jakarta-taglibs-standard-1.1.2.zip.asc	2004-10-25 20:57	186
jakarta-taglibs-standard-oldxml-compat.tar.gz	2002-06-21 22:59	1.1M
jakarta-taglibs-standard-oldxml-compat.tar.gz.asc	2002-06-21 22:59	232
jakarta-taglibs-standard-oldxml-compat.zip	2002-06-21 23:01	1.1M
jakarta-taglibs-standard-oldxml-compat.zip.asc	2002-06-21 23:01	232

JSTL 중요한 문법 구조

1. **jstl 출력법**: **<c:out value="출력할 값" />**

2. **c:if 조건문** 형식:

```
<c:if test="조건식">
 조건식이 맞으면 실행;
</c:if>
```

주의사항: else 문은 없다.

3. **다중 조건문** 형식:

```
<c:choose>
   <c:when test="조건 1">조건 1이 만족하면 실행 </c:when>
   <c:when test="조건 2">조건 2가 만족하면 실행 </c:when>
   ...중략...
   <c:otherwise>해당 사항 없으면 실행</c:otherwise>
</c:choose>
```

4. 반복문

형식) **<c:forEach var="변수명" begin="1" end="10" step="2">**
1부터 10 까지 2씩 증가하면 홀수값 출력
</c:forEach>

web.xml 서블릿 설정 형식

```
<servlet>
  <servlet-name>Test</servlet-name>
  <servlet-class>model.Test</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>Test</servlet-name>
  <url-pattern>/t</url-pattern>
</servlet-mapping>
```

1. servlet-name 에 지정한 Test는 서블릿 클래스명
2. servlet-class 에 지정한 model은 패키지명, Test는 서블릿 클래스명
3. /t는 웹주소에서 실행되는 매핑주소.

결론적으로 웹주소에서 /t 매핑주소가 실행되면 model패키지의 서블릿 자바 클래스 Test가 로드 되어져서 웹상에 실행된다.

4. 서블릿 2.5 버전까지는 web.xml 파일의 설정을 통해서만 서블릿에 접근할 수 있었다. 톰캣 6은 서블릿 2.5을 지원한다. 서블릿 3.0부터는 @WebServlet 애노테이션이 추가되어져서 쉽게 web.xml에 설정하지 않아도 매핑주소를 등록해서 서블릿에 접근이 가능하다. 톰캣 7.0부터 서블릿 3.0을 지원한다. 그러므로 이 애노테이션을 통한 서블릿 접근은 톰캣 7.0부터 가능하다.

cos.jar 다운로드 주소: www.servlets.com/cos