JSP

Back-end Programming

김기정 (bangry313@gmail.com)

목차 (Table of Contents)

1. JSP(Java Server Page) 개요

1.1 JSP 소개

✓ JSP는 Java Server Page의 약자이며, 서블릿과 달리 HTML 페이지에 Java 코드를 넣어(Scripting) 동적으로 웹 페이지를 생성하기 위한 자바 언어 기반의 Server Side Script 언어이다.

- 대표적인 서버 사이드 스크립트 언어 : JSP, ASP, PHP
- 클라이언트 사이트 스크립트 언어 : JavaScript

✓ JSP는 Servlet과 마찬가지로 서블릿(웹) 컨테이너에 의해 관리되고 실행된다.

- Servlet이 Java 코드에서 스트림 API를 기반으로 HTML을 동적 출력하는 것과 달리, JSP는 HTML 페이지에서 자바 코드를 넣는 방식으로 개발한다. - JavaScript와 유사하게 개발
- JSP의 확장자는 반드시 *.jsp 이어야 한다.

✓ JSP 탄생 배경

- Servlet만으로도 웹 애플리케이션을 충분히 개발할 수 있지만, 동적 웹 페이지(HTML, CSS, JavaScript 등) 출력이 번거롭다.
  - 서블릿 코드영역에서 HTML을 출력함으로 써 개발 생산성이 떨어지는 단점을 가진다.
- Servlet과 반대로 JSP는 HTML 페이지에서 동적 출력이 필요한 영역에 자바 코드를 포함시켜 사용함으로써 동적 웹페이지를 구성하는데 매우 유용하고, 개발 생산성이 현격히 높아진다.
- 또한 화면 출력(Presentation Logic)과 자바 비즈니스 코드(Business Logic)를 분리(캡슐화)하여 개발함으로써 유지보수가 훨씬 수월해 진다.

1.2 Servlet vs JSP 비교

✓ Servlet이 Java 코드에서 HTML을 출력하는 것과 달리 JSP는 HTML 문서에 Java 코드를 넣는 방식으로 개발한다.

```
public class 서블릿 extends HttpServlet {

protected void doGet(HttpServletRequest req,
HttpServletResponse resp) … {

    …
        HTML 출력
    …

}
```

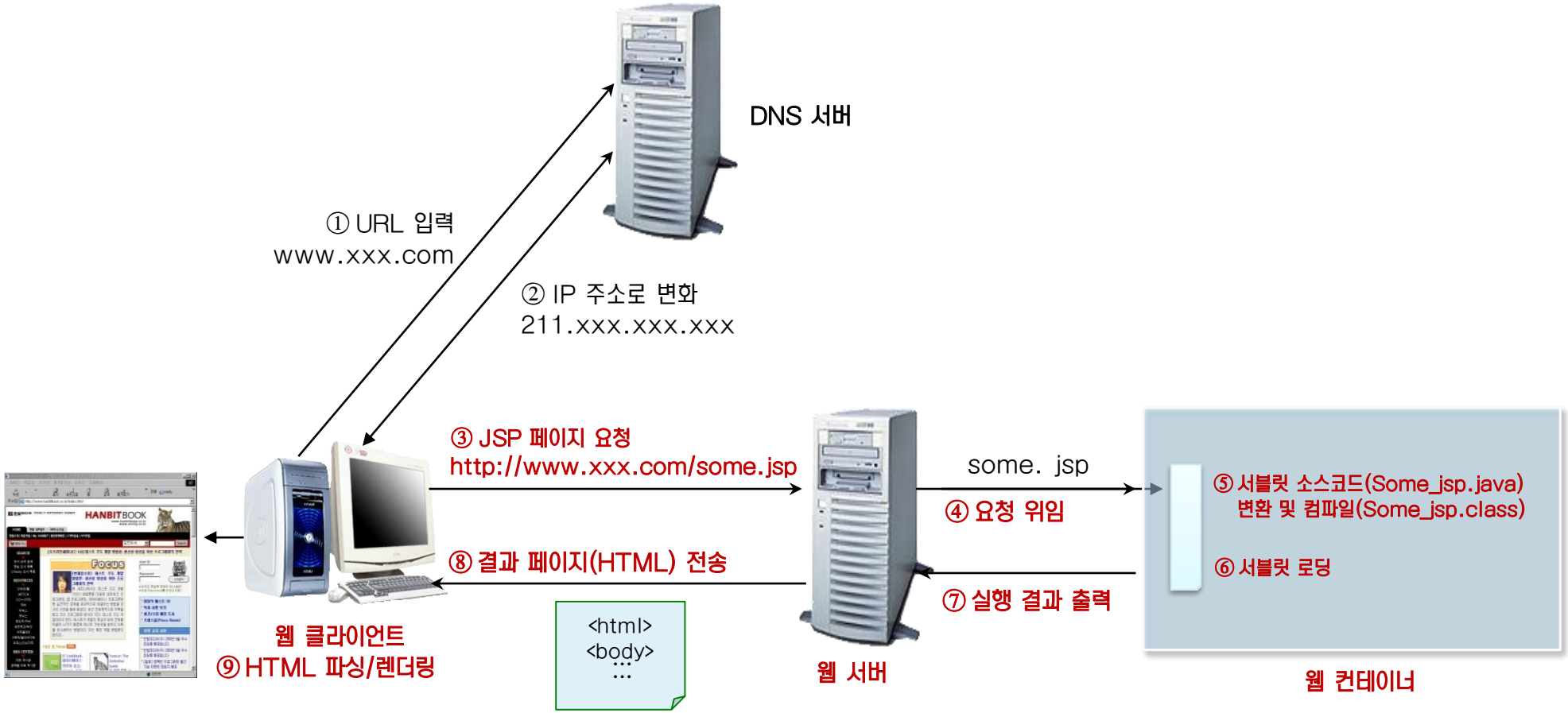
Servlet(Java 코드)에 포함된 HTML

```
<html>
  <head>…</head>
  <body>
  …
  <%
        Java 코드
  %>
  <br>
  …
  </body>
</html>
```

JSP(HTML 페이지)에 스크립팅된 Java 코드

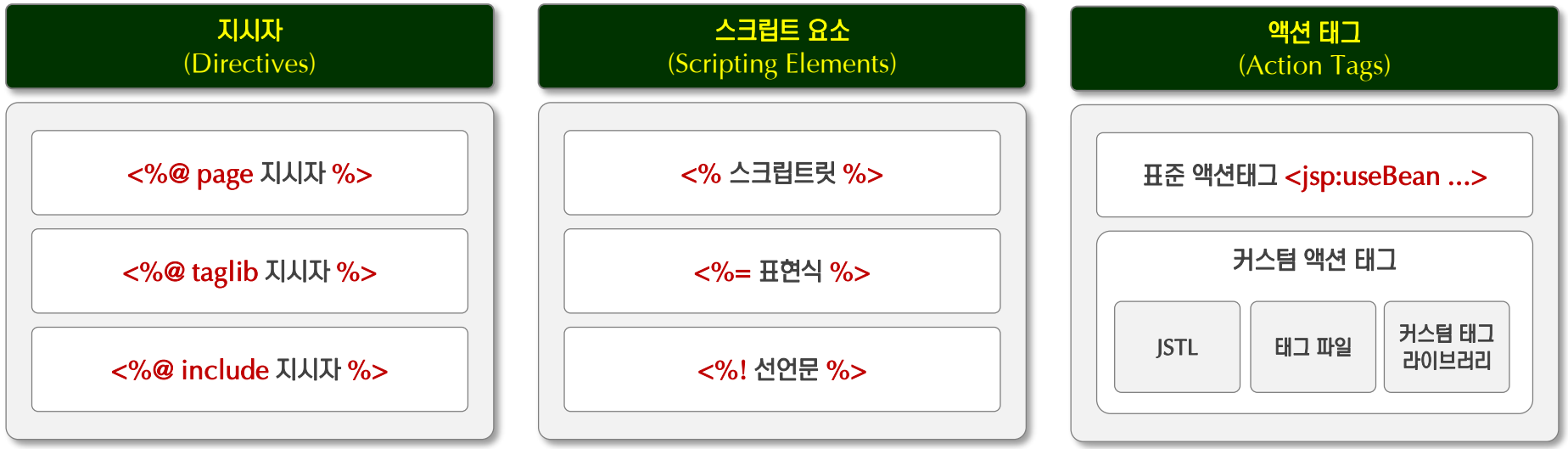
1.3 JSP 처리 과정

✓ JSP는 서블릿(웹) 컨테이너에 의해 자동으로 서블릿 소스코드로 변환되며, 컴파일 되고, 실행되어 요청을 처리한다.



1.4 JSP 구성 요소

1. HTML 요소 : 템플릿 텍스트(<html></html>)
   - 서블릿 컨테이너에 의해 해석되지 않는 모든 텍스트를 말한다(HTML, CSS, JavaScript 등)
2. JSP 지시자, JSP 스크립트 요소, JSP 액션 태그 : 동적 컨텐츠 생성
   - JSP 지시자(Directives) : JSP가 서블릿으로 변환되는 과정에서 필요한 정보를 서블릿 컨테이너에게 제공한다.
   - JSP 스크립트 요소(Scripting Elements) : 동적 컨텐츠 생성을 위해 Java 코드를 사용할 수 있는 요소이다.
   - JSP 액션 태그(Action Tags) : Java 코드를 사용하지 않고, JSP에서 제공하는 표준 액션태그와 커스텀 액션태그를 사용할 수 있다.



2. JSP(Java Server Page) 기본 구문

2.1 JSP 지시자 (1/4) – 종류

✓ 지시자(Directives)는 JSP가 서블릿으로 변환하는 과정에서 서블릿 컨테이너에 필요한 변환 정보를 제공한다.

- page 지시자는 JSP 페이지 관련 변화정보를 제공하며 문자 인코딩, 컨텐츠 타입 등의 속성을 정의한다.
- include 지시자는 현재 페이지에 다른 페이지를 포함시킬 때 사용한다.
- taglib 지시자는 JSP 페이지 내에서 태그 라이브러리를 사용하기 위해 선언한다.

page 지시자	`<%@ page contentType="text/html;charset=utf-8" pageEncoding="utf-8" %>` JSP 페이지가 UTF-8로 인코딩 되었음을 컨테이너에 알려주고, 응답 처리 시 Content-Type 헤더에 MIME 타입을 text/html로 설정
include 지시자	`<%@ include file="copyright.jspf" %>` JSP 페이지에 copyright 문구를 정의하고 있는 서브문서를 포함시킴, copyright.jspf 의 확장자를 통해 이 문서가 단편적인 JSP 문서임을 알 수 있다
taglib 지시자	`<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>` JSP 페이지 안에서 JSTL의 core tag library를 접두어 c를 사용할 수 있도록 선언

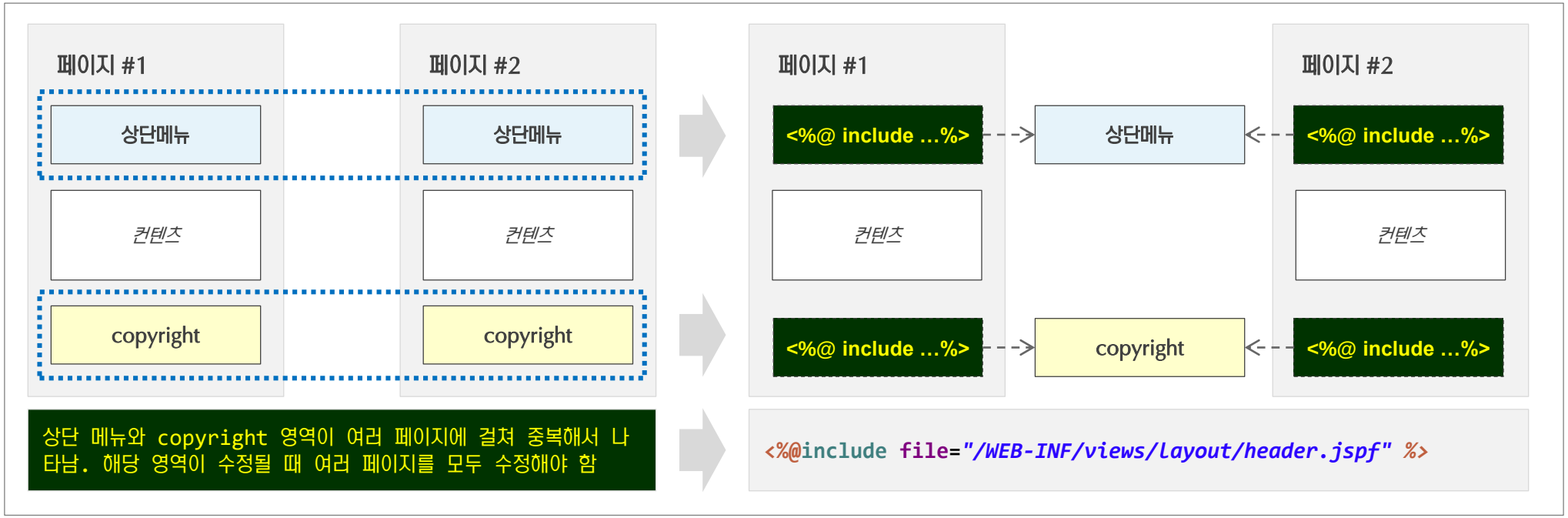
2.1 JSP 지시자 (2/4) – page 지시자

✓ page 지시자 관련 속성

속성	설명
contentType	MIME 타입과 문자 인코딩 방식을 설정 (예: contentType="text/html; charset=UTF-8")
pageEncoding	JSP 문자 인코딩 방식을 설정 (디폴트 utf-8)
import	생성될 서블릿 클래스에 추가될 자바 import문을 정의
errorPage	이 페이지에서 잡지 못한 예외사항을 보낼 오류 페이지 URL을 정의
isErrorPage	오류 처리용 페이지 여부. true면 내장 예외객체(exception)를 사용할 수 있음 (디폴트 false)
isELIgnored	페이지를 서블릿으로 전환할 때 EL 표현식을 무시할 것인지를 결정
session	내장 session 객체를 가질지 여부 (디폴트 true)
autoFlush	자동적으로 버퍼링된 출력 결과를 비울지 여부 (디폴트 true)
buffer	응답객체에 값을 출력할 때 버퍼링을 사용할 지 여부 (디폴트 값은 8kb)
language	스크립트릿에서 사용할 언어를 지정 (디폴트 java)

2.1 JSP 지시자 (3/4) – include 지시자

- ✓ include 지시자는 지정한 파일을 JSP파일에 삽입할 때 사용한다.
- ✓ include 된 파일들은 소스코드 내용이 그대로 복사되어 포함되고 이에 대한 서블릿 코드는 하나만 생성된다.
- ✓ 필요한 만큼 중첩하여 사용할 수 있다. 즉, 포함되는 JSP 내부에서 또 다른 파일을 include 할 수 있다.
- ✓ 상단 메뉴나 하단 문구 등과 같은 공통적인 부분을 분리하여 관리하기 위해 주로 사용한다 - 페이지 모듈화



include 지시자를 사용한 페이지 구성

2.1 JSP 지시자 (4/4) – taglib 지시자

- ✓ taglib 지시자는 커스텀 태그를 JSP에서 사용하기 위한 정보를 제공한다.
  - 기본적인 JSP 내장 기능을 확장하기 위해 정의한 커스텀 태그들의 집합을 태그 라이브러리라 한다.
- ✓ uri 속성에는 TLD(Tag Library Descriptor) 파일에 정의한 URI를 설정한다.
- ✓ prefix 속성에는 사용할 커스텀 태그들의 네임스페이스(접두어)를 지정한다.

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%> ➔ JSTL의 Core Tag Library 를 네임스페이스 c로 접근하도록 함
<%@ taglib prefix="mytag" tagdir="/WEB-INF/tags" %>  ➔ tags 폴더 하위에 있는 Tag 파일들을 네임스페이스 mytag로 접근하도록 함
<%@ taglib prefix="namoo" uri="http://namoo.com/tag" %>  ➔ 커스텀 태그 라이브러리를 네임스페이스 namoo로 접근함
```

*TLD 파일 : Tag Library Descriptor, 커스텀 태그에 대한 정보를 제공하고, 태그가 유효한 것인지를 검증하는데 사용하는 XML

2.2 JSP 스크립팅 요소 (1/4) – 스크립트릿 (<% %>)

- ✓ 스크립트릿(Scriptlet)은 JSP 페이지에 자바 코드를 삽입하기 위해 사용한다.
- ✓ page 지시자에 선언된 language 속성에 해당하는 언어로 작성한다. 생략 시 Java 언어가 기본값이다
- ✓ 외부 패키지의 클래스를 사용할 때는 page 지시자에 import 속성을 지정하거나 패키지명 포함 풀클래스명을 사용한다.
- ✓ request, response, out, session, config 등과 같은 JSP 내장객체를 사용할 수 있다

```
<%
   User loginUser = (User) session.getAttribute("loginUser");
   if (loginUser == null) {
%>
    <a href="/user/login.do">로그인</a>
<%
   } else {
%>
   <b><% out.print(loginUser.getName()); %></b> 님!! 환영합니다.
<%
   }
%>
```

2.2 JSP 스크립팅 요소 (2/4) – 표현식 [<%= %>]

- ✓ 표현식(Expression)은 JSP 페이지에서 출력할 내용을 나타내기 위해 사용한다.
- ✓ 표현식으로 작성한 내용은 컨테이너가 out.print(" ") 메소드 호출로 변환한다 .
- ✓ 표현식은 문장의 마지막에 세미콜론(;)은 붙이지 않는다.

표현식	`<%= loginUser.getName() %>`
스크립트릿	`<% out.print(loginUser.getName()); %>`

두 코드는 동일한 결과를 출력한다

2.2 JSP 스크립팅 요소 (3/4) – 선언문 (<%! %>)

- ✓ 선언문(Declarations) 은 JSP 페이지에서 사용할 인스턴스 변수나 인스턴스 메소드를 정의하기 위해 사용한다.
- ✓ 선언문은 서블릿 컨테이너가 JSP를 서블릿 코드로 변환하는 과정에서 인스턴스 변수, 인스턴스 메소드로 변환한다.
- ✓ 싱글톤 기반 서블릿은 멀티 스레드 환경에서 동작하므로 동기화 문제가 발생하지 않도록 선언문을 신중하게 사용해야 한다.

```
<%! int counter = 0; %>

<%! int getNextCounter() {
    return ++counter;
} %>
```

선언문으로 변수와 메소드를 선언함

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

 int counter = 0;
 int getNextCounter() {
    return ++counter;
}
```

JSP 컨테이너에 의해 서블릿으로 변환된 코드

2.2 JSP 스크립팅 요소 (4/4) – 주석 선언문 (<%-- --%>)

✓ JSP 주석은 <%-- Comment -->와 같이 작성하며 서블릿으로 변환될 때 자바 소스 코드 주석으로 변환된다.

✓ HTML 주석과 차이점

- HTML 주석(<!-- Comment -->)은 응답 메시지에 포함되기 때문에 브라우저의 [소스보기]를 통해 주석 내용을 확인할 수 있다.
- JSP 주석은 클라이언트 응답에 포함되지 않으므로 브라우저의 [소스보기]를 통해 확인할 수 없다.

2.3 JSP 디폴트 내장객체 (1/3) – 생존범위와 내장객체 (9개)

✓ JSP에는 개발 편의성을 위해 서블릿 컨테이너에 의해 미리 생성되어 제공되는 9개의 디폴트 내장 객체를 제공한다.

✓ JSP 디폴트 내장객체는 사용할 수 있는 4개의 유효범위(Scope)를 가진다.

- page Scope : 해당 페이지 내에서만 유효
- request Scope : 클라이언트 요청 범위에서만 유효
- session Scope : 클라이언트 세션에서만 유효
- context Scope : 웹 애플리케이션(ServletContext)에서 유효

객체이름	데이터 타입	의미	유효범위
request	HttpServletRequest	클라이언트의 요청	요청
response	HttpServletResponse	요청에 대한 응답	현재 서블릿
out	JspWriter	문자 출력 스트림	현재 서블릿
pageContext	PageContext	현재 JSP에 대한 실행 환경 정보	현재 서블릿
session	HttpSession	클라이언트 상태정보 저장을 위한 세션	동일 브라우저
application	ServletContext	실행 환경 정보 저장 및 데이터 공유	현재 컨테이너
config	ServletConfig	초기 설정 정보 제공	현재 서블릿
page	Object	요청을 처리하고 있는 현재 Servlet (this)	현재 서블릿
exception	Throwable	실행 시 발생하는 Throwable 예외 객체	현재 서블릿

2.3 JSP 디폴트 내장객체 (2/3) – 4개의 Scope 내장 객체 공통 메소드

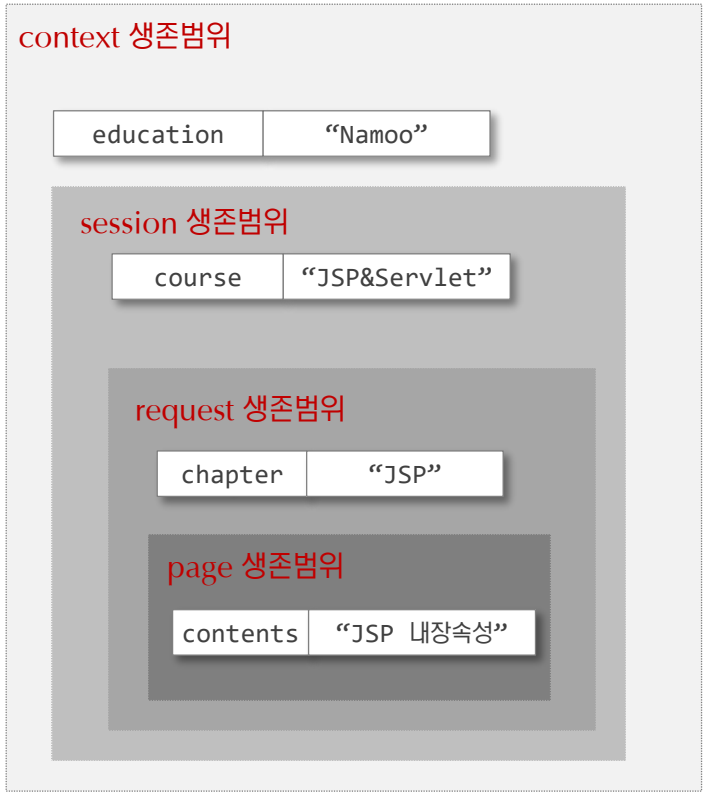
✓ pageContext, request, session, application 4개의 객체를 Scope 객체라 하며, JSP(서블릿)간 데이터 공유를 위해 속성을 설정하거나 조회할 수 있는 공통 메소드를 제공한다.

- Scope 객체에 속성 값을 설정하거나 조회할 때는 setAttribute() 및 getAttribute() 메소드를 사용한다.
- Scope 객체의 getAttributeNames() 메소드는 모든 속성의 이름을 반환한다
- Scope 객체의 removeAttribute() 메소드는 해당하는 속성을 삭제할 때 사용한다.

메소드	설명
public Object getAttribute(String key)	key값으로 등록된 속성을 Object로 반환하고 해당 key값이 존재하지 않을 경우 null을 반환
public void setAttribute(String key, Object value)	속성 값을 key값으로 등록
public Enumeration getAttributeNames()	해당 생존범위의 모든 속성들의 이름을 Enumeration으로 반환
public void removeAttributte(String key)	key 값에 해당하는 속성을 생존범위에서 삭제

2.3 JSP 디폴트 내장객체 (3/3) – pageContext 객체

- ✓ pageContext는 현재 페이지에 대한 정보를 제공하며, 추가적으로 속성을 설정하고 조회할 수 있다.
- ✓ pageContext는 추가적으로 다른 Scope 객체의 속성에도 접근할 수 있는 기능을 제공한다.
  - getAttribute() 메소드의 두 번째 인자로 scope를 명시하면 해당 Scope 객체의 속성 값을 조회할 수 있다.
  - setAttribute() 메소드의 세 번째 인자로 scope를 명시하면 해당 Scope 객체의 속성 값을 설정할 수 있다.
  - findAttribute() 메소드는 가장 작은 생존범위 부터 넓은 범위 순으로 속성을 검색한다.



pageContext 생존범위 속성 값 조회

```
<%= pageContext.getAttribute("contents") %>
```

request 생존범위속성 값 조회

```
<%= pageContext.getAttribute("chapter",
                             PageContext.REQUEST_SCOPE) %>
```

session 생존범위의 속성 값 설정

```
<%= pageContext.setAttribute("course", "SpringMVC",
                             PageContext.SESSION_SCOPE) %>
```

생존범위를 알 수 없는 속성 찾기

```
<%= pageContext.findAttribute("education") %>
```

2.4 JSP 표준액션태그 (1/13) – 소개 (1/2)

✓ JSP 표준액션태그는 스크립팅 요소(<% 자바코드 %>)를 사용하지 않고 XML 문법을 사용하는 태그만으로 JSP 페이지를 작성할 수 있는 방법을 제공한다.

- 표준액션태그는 서블릿 컨테이너가 기본 지원하기 때문에 특별한 설정 없이 바로 사용할 수 있다.

✓ JSP 표준액션태그 종류

- <jsp:forward> : 클라이언트의 요청과 응답에 대한 제어권을 JSP 페이지 사이에 이동시킬 수 있다.
- <jsp:include> : 정적 또는 동적인 자원을 현재 페이지에 포함시킬 수 있다.
- <jsp:useBean>, <jsp:getProperty>, <jsp:setProperty> : 자바 빈(JavaBean)을 사용할 수 있다.

✓ 표준액션태그 사용시 주의 사항

- 표준액션태그는 XML 문법을 따르기 때문에, 대소문자를 구분한다.
- 태그명 앞에 반드시 jsp 접두어(prefix)를 붙여 사용하여야 한다.
- 시작태그(<jsp:some>)가 있으면 반드시 끝나는 태그(</jsp:some>)가 있어야 한다.
  - 내용이 없는 단일태그 사용 시 태그의 마지막 부분을 "/>"로 마무리 할 수 있다.
- 태그의 속성값을 할당할 때는 반드시 인용부호("")를 사용하여야 한다.

2.4 JSP 표준액션태그 (2/13) – 소개 (2/2)

스크립트릿을 통한 Java Bean 접근

```
<html>
<body>
<%
    com.namoo.Course course = (com.namoo.Course) request.getAttribute("course");
%>
Person is : <%= course.getName() %>
</body>
</html>
```

HTML 안에 포함된 Java 코드는 HTML 구조를 한눈에 파악하기 어렵게 한다

JSP 표준액션을 사용한 Java Bean 접근

```
<html>
<body>
   <jsp:useBean id="course" class="com.namoo.Course" scope="request" />
   <jsp:getProperty name="course" property="name" />
</body>
</html>
```

2.4 JSP 표준액션태그 (3/13) – <jsp:useBean /> (1/5)

✓ JSP 표준액션태그에는 Java Bean 객체를 다룰 수 있는 3가지 표준액션이 존재한다.

- <jsp:useBean> 은 지정한 생존범위(page, request, session, application)에 속성 Bean을 선언하고 초기화하는 태그이다.
- <jsp:getProperty> 은 속성 Bean 객체의 프로퍼티를 읽어오는 태그이다.
- <jsp:setProperty> 은 속성 Bean 객체의 프로퍼티에 값을 설정하는 태그이다.

<jsp:useBean> 표준액션태그

```
<jsp:useBean id = "course"
    class = "com.namoo.Course"
    scope = "request" />
```

course 식별자를 가진 Course 타입의 빈 객체를 request 생존범위의 속성으로 선언한다. 생존 범위를 생략하는 경우 생존범위는 page가 된다. 생존범위에 기존에 동일한 이름의 빈이 있으면 해당 빈이 사용되고, 없으면 새로운 빈 객체가 생성된다.

<jsp:getProperty> 표준액션태그

```
<jsp:getProperty name="course" property="name" />
```

course 라는 빈 객체의 name 프로퍼티 값을 출력한다.

<jsp:setProperty> 표준액션태그

```
<jsp:setProperty name="course" property="name"
                 value="JSP&Servlet" />
```

course 라는 빈 객체의 name 프로퍼티 값에 “JSP&Servlet” 을 설정한다

2.4 JSP 표준액션태그 (4/13) – <jsp:useBean /> (2/5)

✓ Java Bean

- 웹 애플리케이션 작성 시 데이터 저장을 목적으로 사용되는 재사용 가능한 컴포넌트를 말한다
- Java Bean Spec(규약)에 따라 디폴트 생성자와 공개된 속성(프로퍼티)을 제공해야 한다

Java Bean 규약

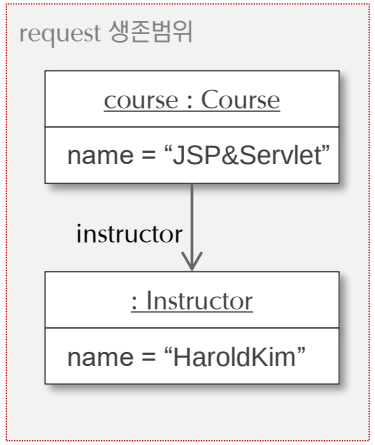
```
public class BeanClassName {
    /** 속성(프로퍼티) 선언 */
    private String value;

    /** 디폴트 생성자 */
    public BeanClassName() {  }

    /** public getter 메소드 */
    public String getValue() {
        return value;
    }

    /** public setter 메소드 */
    public void setValue(String value) {
        this.value = value;
    }
}
```

2.4 JSP 표준액션태그 (5/13) – <jsp:useBean /> (3/5)

- ✓ <jsp:useBean> 표준액션태그는 내용을 가질 수 있다.
- ✓ 내용은 <jsp:useBean> 액션태그를 통해 Bean 객체가 새로 생성되는 경우에만 동작한다.
- ✓ 예를 들어, 새로운 Bean 객체를 만들면서 디폴트 값을 설정할 경우 내용에 <jsp:setProperty> 를 정의하면 된다.
- ✓ useBean 표준액션태그는 연관된 객체의 프로퍼티의 값을 출력할 수 없는 단점이 있다. 이러한 경우에는 표현식이나 EL을 사용해서 처리하여야 한다.

<jsp:getProperty> 표준액션태그

```
<jsp:useBean id="course" class="com.namoo.Course" scope="request">
    <jsp:setProperty name="course" property="name" value="JSP&Servlet" />
</jsp:useBean>
```



[Bean 표준 액션태그의 단점] 연관된 객체의 프로퍼티에 접근할 수 없다

```
<jsp:useBean id="course" class="com.namoo.Course" scope="request" />
<jsp:getProperty name="course" property="instructor" />
```

➔ instructor 객체에는 접근할 수 있지만 instructor 객체의 name 프로퍼티에 접근할 방법이 없다

[해결방안 #1] 표현식 사용하기

```
<%= course.getInstructor().getName() %>
```

[해결방안 #2] EL 사용하기

```
${course.instructor.name}
```

2.4 JSP 표준액션태그 (6/13) – <jsp:useBean /> (4/5)

✓ <jsp:useBean /> 은 JSP에서 자바빈을 선언하고 초기화 하는 표준 액션 태그이다.

✓ 사용 형식

- <jsp:useBean id="beanName" class="bean Class Full Name" [scope="scope 객체"]/>
  - id : 빈을 scope 객체에 저장할 때 사용할 식별자
  - class : 패키지 이름을 포함한 빈 클래스 이름
  - scope : page, request, session, application 중 하나를 값으로 설정 (디폴트 : page)

표준액션태그를 사용하지 않고서 (스크립팅으로)

```
<html>
<body>
<%
    foo.Person p = (foo.Person) request.getAttribute("person");
%>
Person is : <%= p.getName() %>
</body>
</html>
```

표준액션태그를 사용하면(스크립팅 사용 안 함)

```
<html>
<body>
   <jsp:useBean id="person" class="foo.Person" scope="request" />
   <jsp:getProperty name="person" property="name" />
</body>
</html>
```

2.4 JSP 표준액션태그 (7/13) – <jsp:useBean /> (5/5)

- ✓ id 속성에 해당하는 Bean이 지정한 scope 객체에 존재할 경우
  - 존재하는 Bean 반환
- ✓ id 속성에 해당하는 Bean이 지정한 scope 객체에 존재하지 않을 경우
  - class 속성에 지정한 Bean 생성 및 지정한 scope 객체에 저장하고 반환

<jsp:useBean /> 동작 원리

```
<jsp:useBean id="person" class="foo.Person" scope="request" />
```

```
// _jspService() 메소드 내에 다음과 같은 코드로 변환
foo.Person person = null;
person = (foo.Person)pageContext.getAttribute("person", PageContext.REQUEST_SCOPE);
if(person == null){
          person = Class.forName(“foo.Person”).newInstance();
          pageContext.setAttribute("person", PageContext.REQUEST_SCOPE);
}
```

2.4 JSP 표준액션태그 (8/13) – <jsp:setProperty /> (1/2)

✓ Bean의 프로퍼티 값 설정

- setter 메소드 호출

<jsp:setProperty />

```
<jsp:setProperty  name="bean Name"  property="property Name"  value="property Value" />

<jsp:setProperty  name="bean Name"  property="property Name"  param="param Name" />

<jsp:setProperty  name="bean Name"  property="property Name" />
 - 요청 파라미터 이름과 빈 프로퍼티 이름이 동일한 프로퍼티 설정

<jsp:setProperty  name="bean Name"  property="*" />
 - 요청 파라미터 이름과 빈 프로퍼티 이름이 동일한 모든 프로퍼티 자동 설정
 - 폼 데이터를 빈에 설정할 때 유용하게 사용
```

2.4 JSP 표준액션태그 (9/13) – <jsp:setProperty /> (2/2)

✓ Bean 프로퍼티 자료형에 따른 자동 형 변환

프로퍼티 타입	형 변환	기본 값
boolean | Boolean	Boolean.valueOf(String)	False
byte | Byte	Byte.valueOf(String)	(byte) 0
short | Short	Short.valueOf(String)	(short) 0
char | Character	Character.valueOf(String)	(char) 0
int | Integer	Integer.valueOf(String)	0
long | Long	Long.valueOf(String)	0L
double | Double	Double.valueOf(String)	0.0
float | Float	Float.valueOf(String)	0.0f

2.4 JSP 표준액션태그 (10/13) – <jsp:getProperty />

✓ Bean의 프로퍼티 값 읽어와 출력

- getter 메소드 호출

<jsp:getProperty />

```
<jsp:getProperty  name="bean Name"  property="property Name" />
```

2.4 JSP 표준액션태그 (11/13) – <jsp:include /> (1/2)

- ✓ <jsp:include> 액션태그는 현재 페이지에 다른 웹 리소스의 실행 결과를 포함시킬 때 사용하는 표준액션태그 이다.
  - 제어권을 include 되는 리소스로 넘겼다가 그 리소스의 처리가 끝나면 현재 페이지로 제어권을 반환한다.
- ✓ <jsp:include> 의 몸체에 <jsp:param>을 선언하면 포함되는 페이지에 파라미터를 전달할 수 있다.
- ✓ <jsp:include> 액션태그는 **페이지 모듈화**에 주로 사용된다.

- ✓ <%@ include %> 지시자와 차이점
  - include 지시자를 사용하면 서블릿으로 변환되는 단계에 소스 그대로 미리 포함되지만, include 표준액션태그는 요청을 처리하는 시점에 실행된 결과가 포함된다.

<jsp:include> 표준액션태그 사용하기

```
<%@ page contentType="text/html" pageEncoding="utf-8"%>
<!DOCTYPE html>
<html>
<head>
<title>namoo</title>
</head>
<body>
<jsp:include page="header.jsp">
    <jsp:param name="courseName" value="JSP&Servlet" />
</jsp:include>
JSP&Servlet을 소개합니다.
...
</body>
</html>
```

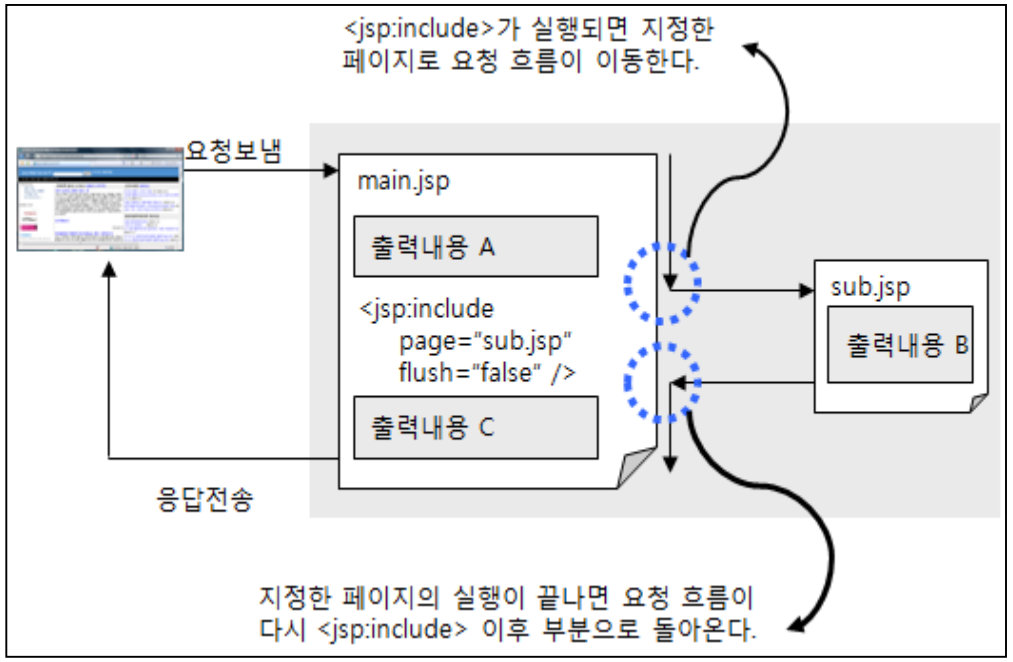
header.jsp

```
<%@ page contentType="text/html" pageEncoding="utf-8" %>
<img src="./images/next.png" /><br/>
<h2>
  Namoo Course -
  <span style="color:darkblue">
    ${param.courseName}
  </span>
</h2>
```

2.4 JSP 표준액션태그 (12/13) – <jsp:include /> (2/2)

✓ <jsp:include> 액션태그를 이용한 페이지 모듈화

- 여러 페이지에 공통적으로 사용되는 메뉴를 독립적인 JSP 파일로 작성하고 포함한다.




```
<jsp:include page="top.jsp" .. />
...
<jsp:include page="left.jsp" .. />
...
...
이 페이지가 보여줄 내용을 생성
...
...
<jsp:include page="bottom.jsp" .. />
```

top.jsp
상단코드

left.jsp
좌측메뉴코드

bottom.jsp
하단코드

2.4 JSP 표준액션태그 (13/13) – <jsp:forward />

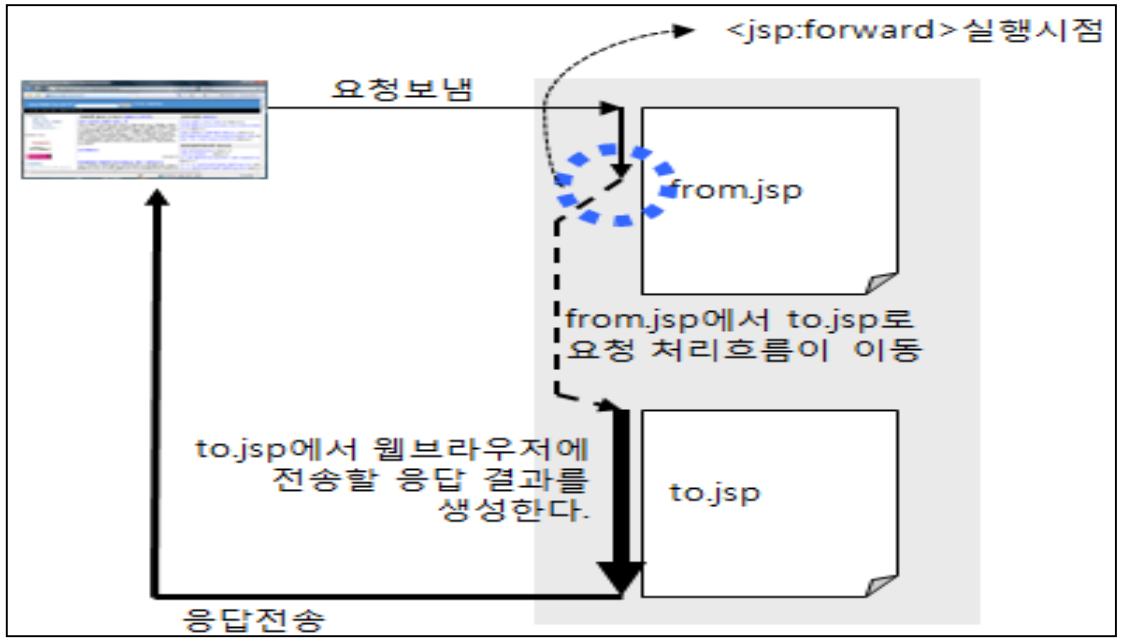
✓ 클라이언트의 요청을 웹 컨테이너내의 다른 리소스(HTML, JSP 등)에 위임(dispatch)한다.

✓ 주로 페이지 흐름을 제어할 때 사용한다.

✓ redirect(<% response.sendRedirect("**URL**") %>)와 구별된다.

<jsp:forward> 표준액션태그

```
<jsp:forward page="to.jsp">
    <jsp:param name="paramName1" value="paramValue1" />
    <jsp:param name="paramName2" value="paramValue2" />
</jsp:include>
```



2.5 Expression Language (1/8) – 개요 (1/2)

✓ **EL(Expression Language)은 스크립트 요소인 스크립트릿(<%자바코드%>)이나 표현식(<%=자바코드%>)을 대신해 JSP 스코프(컨텍스트) 객체의 속성을 쉽고, 간결하게 출력할 수 있다.**

- 스크립트릿이나 표현식을 사용할 경우 자바 구문을 알아야 하기 때문에 비 자바개발자(웹디자이너 등)가 다루기 어렵고,
- HTML 템플릿과 자바코드(스크립트릿, 표현식)의 혼재로 가독성이 떨어지고, 유지보수가 용이하지 않다.

✓ **EL은 JSP의 표현식(<%= %>)을 대체하고, 액션태그(<jsp:getProperty>)를 보완하기 위해 사용된다.**

✓ **주요 기능**

- 데이터의 연산자(산술, 관계, 논리 등) 사용이 가능하며, EL만의 디폴트 내장 객체(11개)를 제공한다.
- JSP 4개의 스코프 객체(pageContext, request, session, application)에 설정된 자바 빈의 프로퍼티나 배열 및 Collection(Set, List, Map) 객체의 요소를 쉽게 접근할 수 있다.

✓ **기본 구문**

- EL 표현식에서 도트(.) 연산자 왼쪽은 반드시 Java Bean 이거나 java.util.Map 객체여야 한다.
- EL 표현식에서 도트(.) 연산자 오른쪽은 반드시 Java Bean의 프로퍼티이거나 맵의 키이거나 여야 한다.
- 오른쪽에 오는 값은 식별자로서 일반적인 자바 명명 규칙을 따라야 한다

스크립트릿

```
<%=  ((foo.Person)
request.getAttribute("person")).getDog().getName() %>
```

Expression Language

```
${person.dog.name}
```

[EL 문법] Map을 사용하는 경우

[EL 문법] Java Bean을 사용하는 경우

2.5 Expression Language (2/8) – 개요 (2/2)

- ✓ EL에는 Dot(.) 표기법 외에 [ ] 연산자를 사용하여 객체의 값에 접근할 수 있다.
- ✓ [ ] 연산자의 왼편에는 배열, Java Bean, Map, List가 올 수 있다.
- ✓ [ ] 안의 값이 문자열인 경우, 이것은 맵의 키가 될 수도 있고, Bean 프로퍼티나 리스트 및 배열의 인덱스가 될 수 있다.
  - 배열과 리스트인 경우, 문자로 된 인덱스 값은 숫자로 변경하여 처리한다.

[] 연산자를 이용한 객체 프로퍼티 접근

```
${person["name"]}
```

Dot 표기법을 이용한 객체 프로퍼티 접근

```
${person.name}
```

리스트나 배열 요소에 접근

```
// Scope 객체에 속성 설정
String[] courses = {"JSP/Servlet", "jQuery", "SpringMVC"};
request.setAttribute("courses", courses);

// EL in JSP
${courses[0]}   // JSP/Servlet 이 출력된다
${courses["1"]} // 문자열인 인덱스 값이 숫자로 바뀌어 courses[1]의 결과인 jQuery 가 출력된다
```

2.5 Expression Language (3/8) – EL 연산자

✓ 산술연산자
- +, -, *, / (div), % (mod)
- 문자열과 숫자 연산 시 숫자로 자동 형변환 된다.

✓ 논리연산자
- &&(and), ||(or), !(not)

✓ 관계연산자
- == (eq), != (ne), < (lt), > (gt), <= (le), >= (ge)

✓ 기타 연산자들
- empty
  - null, 빈 문자열, 빈 콜렉션 검증

2.5 Expression Language (4/8) – EL 내장 객체 (1/3)

✓ EL에서는 JSP 페이지의 EL 표현식에서 사용할 수 있는 11개의 내장 객체를 제공한다.

✓ EL 내장객체에는 생존 범위 속성 맵, 요청 파라미터 맵, 요청 헤더 맵, 쿠키 맵, 컨텍스트 초기화 파라미터 맵 등이 있다.

✓ pageContext는 EL 내장객체 중 유일하게 맵이 아닌 Java Bean 객체이다.

구분	타입	내장 객체명
생존범위 속성	Map	pageScope, requestScope, sessionScope, applicationScope
요청 파라미터	Map	param, paramValues
요청 헤더	Map	header, headerValues
쿠키	Map	cookie
컨텍스트 초기화 파라미터	Map	initParam
pageContext 객체 참조	Java Bean	pageContext

2.5 Expression Language (5/8) – EL 내장 객체 (2/3)

- ✓ requestScope 는 request 생존범위에 저장되어 있는 속성(Attributes)에 대한 단순한 맵이다.
- ✓ EL에서는 JSP request 객체에 직접 접근할 수 없고 pageContext를 통해서 접근할 수 있다.
- ✓ 이름 중첩(연관 객체)이 발생하는 경우 생존범위 내장 객체를 통해 명시적으로 값을 조회해야 한다

EL에서 request 객체 접근

```
// request의 method 프로퍼티 출력
Method is : ${pageContext.request.method}
```

생존범위 내장객체를 통한 명시적인 속성 접근

```
// In servlet
request.setAttribute("foo.person", p);

// Case #1 : 에러
${foo.person.name} // foo 라는 속성은 존재하지 않음

// Case #2 : request 생존범위 내장객체에서 []연산자를 통해 속성 접근
${requestScope["foo.person"].name}
```

2.5 Expression Language (6/8) – EL 내장 객체 (3/3)

✓ JSP에서 쿠키 값을 출력하기 위해서는 스크립트릿을 사용하거나 EL 내장객체인 cookie를 사용할 수 있다.

✓ initParam EL 내장객체를 통해 컨텍스트 초기화 파라미터 값을 출력할 수 있다.

스크립트릿을 통한 쿠키 값 출력

```
Cookie[] cookies = request.getCookies();
for(Cookie cookie : cookies) {
if(cookie.getName().equals("userName")) {
    out.println(cookie.getValue());
}
```

EL 내장객체를 통한 쿠키 값 출력

```
${cookie.userName.value}
```

web.xml 에 설정된 컨텍스트 초기화 파라미터

```
<context-param>
   <param-name>contactEmail</param-name>
   <param-value>support@namoosori.com</param-value>
</context-param>
```

스크립트릿

```
<%= application.getInitParameter("contactEmail") %>
```

EL 내장객체

```
${initParam.contactEmail}
```

2.5 Expression Language (7/8) – Bean의 메소드 호출

✓ EL에서 Bean의 메소드 호출 가능하다

- setter/getter 메소드 뿐만 아니라 반환형이 void 이거나 매개변수가 한 개 이상 존재하는 메서드 호출 가능하다

빈의 메소드 호출

```
<body>
<%
Person person = new Person();
request.setAttribute("person", person);
%>
${person.setName("김기정")}
이름 : ${person.getName()}
</body>
```

2.5 Expression Language (8/8) – EL 비활성화

✓ JSP 지시어를 이용한 비활성화

page 지시어

```
<%@ page isELIgnored=“true”%>
```

✓ web.xml 설정을 통한 비활성화

page 지시어

```
<jsp-config>
    <jsp-property-group>
        <url-pattern>/oldVersion/*<url-pattern>
        <el-ignored>true</el-ignored>
    </jsp-property-group>
</jsp-config>
```

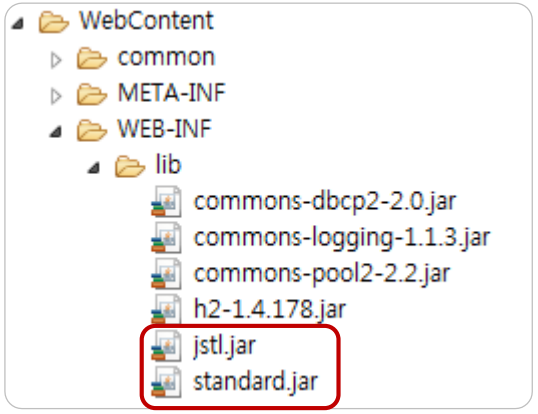
2.6 JSTL (1/8) – JSTL 개요 (1/2)

- ✓ JSP를 사용하여 프리젠테이션(화면) 영역을 개발하는 많은 개발자는 Java와 같은 프로그래밍 언어에 익숙하지 않다(?)
  - JSP 페이지에서 동적인 컨텐츠를 생성하기 위해 스크립트 요소에 Java 언어를 사용해야 한다
- ✓ JSTL(JSP Tag Library)은 JSP 페이지에서 스크립트릿(Java 코드)을 사용하지 않고 태그 만을 이용해 간단하게 처리할 수 있는 방법을 제공한다
- ✓ JSTL에는 다양한 액션태그가 있으며, EL과 함께 사용하면 JSP를 간결하게 작성할 수 있다
- ✓ JSTL 태그 종류

2.6 JSTL (2/8) – JSTL 개요 (2/2)

- ✓ JSTL은 JSP 표준 태그 라이브러리로 자바 코드를 대신해 반복문, 조건문, 포맷팅 및 XML 처리 등을 HTML 태그처럼 사용할 수 있도록 지원한다
- ✓ JSTL을 사용하려면 jstl.jar와 standard.jar 를 /WEB-INF/lib 아래에 설치해야 한다
- ✓ JSTL을 사용하기 위해 JSP 페이지 상단에 taglib 지시자로 사용하려는 태그 라이브러리를 선언해야 한다

1. 웹 프로젝트에 JSTL 설치



Jstl.jar, standard.jar 파일은 어디에서 찾을 수 있나?

인터넷에서 다운로드 받아서 설치하거나 톰캣을 사용 중이라면 톰캣의 예제 프로젝트에 포함된 것을 사용하면 된다.
톰캣 설치경로의 webapps/examples/WEB-INF/lib 폴더를 확인하자.

2. JSP 페이지 상단에 tablib 지시자로 사용하려는 태그라이브러리 선언

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

3. JSP 페이지에서 JSTL 사용하기

```
<c:choose>

  <c:when test="${loginUser == null}">
     <a href="${ctx}/user/login.do">로그인</a> |
     <a href="${ctx}/user/signup.do">회원가입</a>
  </c:when>

  <c:otherwise>
   <b>${loginUser.name}</b> 님!! 환영합니다.
   [<a href="${ctx}/user/logout.do">로그아웃</a>]
  </c:otherwise>

</c:choose>
```

2.6 JSTL (3/8) – <c:out> 액션

✓ <c:out> 액션태그는 value 속성에 지정된 표현식을 평가하여 그 결과를 JspWriter를 통해 출력하는 태그이다

- value 속성에는 출력하려는 표현식을 적는다. 생략할 경우 빈 문자열("")을 출력한다
- escapeXml 속성은 표현식의 결과에 <, >, &, ', " 문자가 포함된 경우 escape 할지 여부를 설정한다(디폴트 true)
- default 속성 또는 액션태그 내용에 문자열을 설정하면, value의 결과가 null인 경우 기본 값으로 사용된다

1. JSP 상단에 core 태그라이브러리 사용을 위한 taglib 지시자 선언

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

In Servlet

```
String html = "<span style=\"color:red\">escapeXml attribute</span>";
request.setAttribute("html", html);
request.setAttribute("someValue", null);
```

2-1. default 속성으로 디폴트 값 설정하기

```
<c:out value="${someValue}" default="Empty"/>
```

3-1. <c:out>

```
<c:out value="${html}" />
```

웹브라우저 출력결과

escapeXml attribute

3-2. escapeXml 속성을 false로 설정한 경우

```
<c:out value="${html}" escapeXml="false" />
```

웹브라우저 출력결과

escapeXml attribute

2-2. 액션태그 Body에 디폴트 값 설정하기

```
<c:out value="${someValue}">Empty Body</c:out>
```

웹브라우저 출력결과

Empty Body

2.6 JSTL (4/8) – <c:set> 액션

✓ <c:set> 액션은 4개의 스코프 객체에 속성을 설정하거나 JavaBea의 프로퍼티에 값을 설정할 때 사용한다

- value 속성이나 내용의 컨텐츠로 값을 설정할 수 있다
- var 속성은 스코프 객체에 속성명을 나타내며, 생존범위는 scope 속성으로 설정한다 (디폴트는 page)
- JavaBeran의 프로퍼티에 값을 할당할 때는 target 속성에 객체를 설정하고 property에 프로퍼티명을 설정한다

[문법1] value 속성을 이용하여 생존범위 변수 값 할당

```
<c:set value="value" var="varName" [scope="{page|request|session|application}"]/>
```

[문법2] 액션의 Body 컨텐츠를 사용하여 생존범위 변수 값 할당

```
<c:set var="varName" [scope="{page|request|session|application}"]>
body content
</c:set>
```

[문법3] value 속성을 이용하여 대상 객체의 프로퍼티 값 할당

```
<c:set value="value" target="target" property="propertyName"/>
```

[문법4] 액션의 Body 컨텐츠를 사용하여 대상 객체의 프로퍼티 값 할당

```
<c:set target="target" property="propertyName">
body content
</c:set>
```

2.6 JSTL (5/8) – <c:catch> 액션

- ✓ 기본적으로 JSP 페이지는 예외가 발생하면 지정된 오류페이지를 통해 처리한다
- ✓ <c:catch> 액션은 JSP 페이지에서 예외가 발생할 만한 코드를 오류페이지로 넘기지 않고 직접 처리할 때 사용한다
  - var 속성에는 발생한 예외를 담을 page 생존범위 변수를 지정한다
  - <c:catch>와 <c:if> 액션을 함께 사용하여 Java 코드의 try~catch 와 같은 기능을 구현할 수 있다

try ~ catch 구문

```
try {
    String str = null;
    out.println("Length of string : " +
                str.length());
} catch(Throwable ex) {
    out.print(ex.getMessage());
}
```

<c:catch>와 <c:if> 사용하기

```
<%@ page contentType="text/html" pageEncoding="UTF-8"
         errorPage="error.jsp" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<c:catch var="ex">
<%
    String str = null;
    out.println("Length of string : " + str.length()); // 예외 발생!!!
%>
</c:catch>

<c:if test="${ex != null}">
    예외가 발생하였습니다. ${ex.message}
</c:if>
```

2.6 JSTL (6/8) – 조건문 액션

✓ <c:if> 액션은 test 속성에 지정된 표현식을 평가하여 결과가 true인 경우 액션의 내용을 수행한다

- <c:if> 액션의 var 속성은 표현식의 평가 결과인 Boolean 값을 담을 스코프 객체의 속성을 나타내며, 생존범위는 scope 속성으로 설정한다 (디폴트는 page)
- <c:choose><c:when><c:otherwise> 액션을 사용하면 if, else if, else 와 같이 처리할 수 있다

<c:if> 액션 사용 예

```
<c:if test="${userType eq 'member'}">
    <jsp:include page="memberOnly.jsp" />
</c:if>
```

<c:if> 액션의 var 속성

```
<c:if test="${userType eq 'member'}" var="accessible">
    <jsp:include page="memberOnly.jsp" />
</c:if>

...
<c:if test="${accessible && userDevice == 'iPhone'}">
   User type is member.
</c:if>
```

<c:choose>, <c:when>, <c:otherwise> 액션 사용 예

```
<c:choose>
    <c:when test="${userDevice == 'iPhone'}">
        실행 할 내용….
   </c:when>

   <c:when test="${userDevice == 'Android'}">
        실행 할 내용…
   </c:when>

   <c:otherwise>
        해당 조건이 맞지 않을 때의 디폴트 실행문
   </c:otherwise>
</c:choose>
```

2.6 JSTL (7/8) – 반복문 액션

✓ <c:forEach> 액션은 배열이나 컬렉션에 있는 항목들에 대하여 액션의 내용을 반복하여 수행한다

- var 속성에는 반복에 대한 현재 항목을 담는 스코프 객체의 속성을 지정하고 items 속성은 반복할 항목들을 갖는 컬렉션을 지정한다
- varStatus 속성에 지정한 속성값을 통해 현재 반복의 상태정보를 알 수 있다

courses

0	Java Fundamental
1	Java IO
2	Java Socket
3	JSP&Servlet
4	Spring MVC
5	JavaScript
6	jQuery Basic
7	MyBatis

<c:forEach> 액션의 다양한 활용

```
<b>1) courses 리스트를 반복하면서 과정명 출력하기</b><br/>
<c:forEach var="course" items="${courses}">
    ${course.name}<br/>
</c:forEach>

<b>2) courses 리스트를 반복하면서 순번과 과정명 출력하기</b><br/>
<c:forEach var="course" items="${courses}" varStatus="varStatus">
    ${varStatus.count}. ${course.name}<br/>
</c:forEach>

<b>3) 짝수번째 과정명만 출력하기</b><br/>
<c:forEach var="course" items="${courses}" begin="0" end="5" step="2">
    ${course.name}<br/>
</c:forEach>

<b>4) 숫자 1부터 5까지 출력하기</b><br/>
<c:forEach var="value" begin="1" end="5" step="1">
    ${value}<br/>
</c:forEach>
```

실행결과

```
1) courses 리스트를 반복하면서 과정명 출력하기
Java Fundamental
Java IO
Java Socket
JSP&Servlet
Spring MVC
JavaScript
jQuery Basic
MyBatis
2) courses 리스트를 반복하면서 순번과 과정명 출력하기
1. Java Fundamental
2. Java IO
3. Java Socket
4. JSP&Servlet
5. Spring MVC
6. JavaScript
7. jQuery Basic
8. MyBatis
3) 짝수번째 과정명만 출력하기
Java Fundamental
Java Socket
Spring MVC
4) 숫자 1부터 5까지 출력하기
1
2
3
4
5
```

End of Document

✓ Q&A

감사합니다...