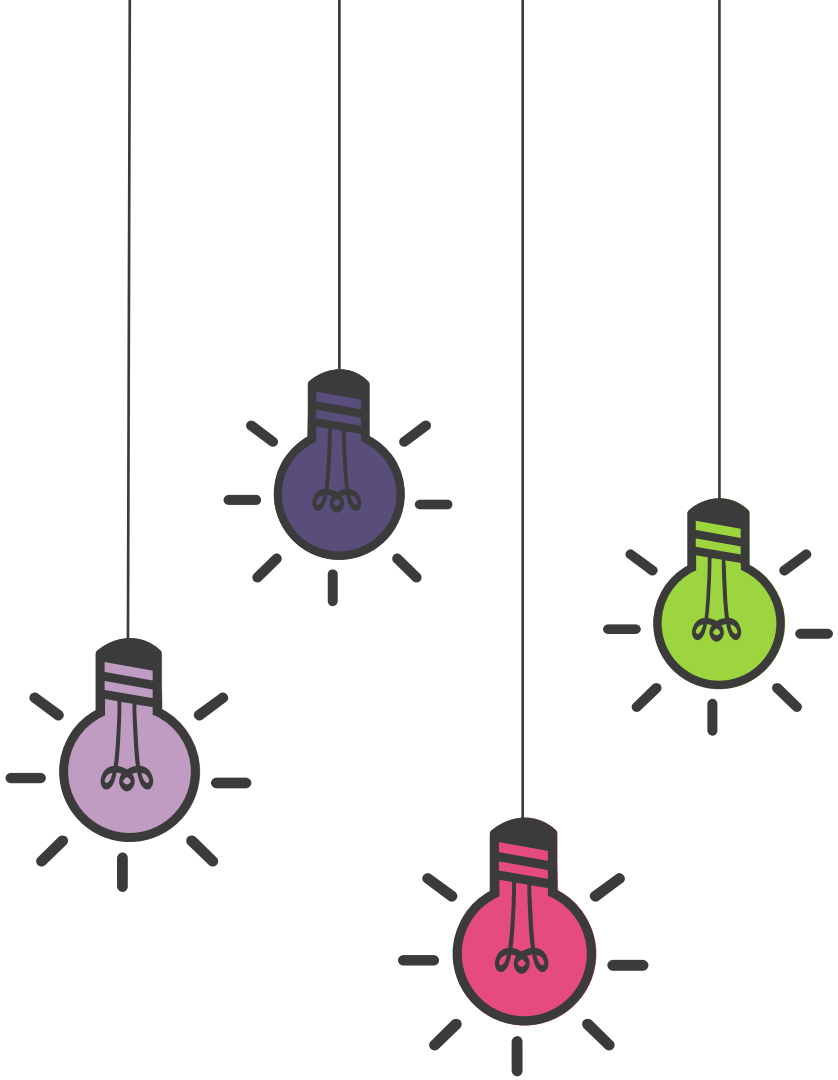


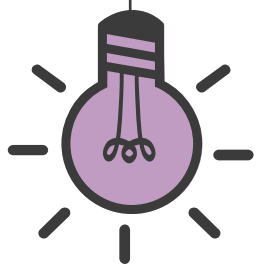
JSP 프로그래밍

J S P 프 로 그 래 밍

Index

CONTENTS

01 JSP 개요



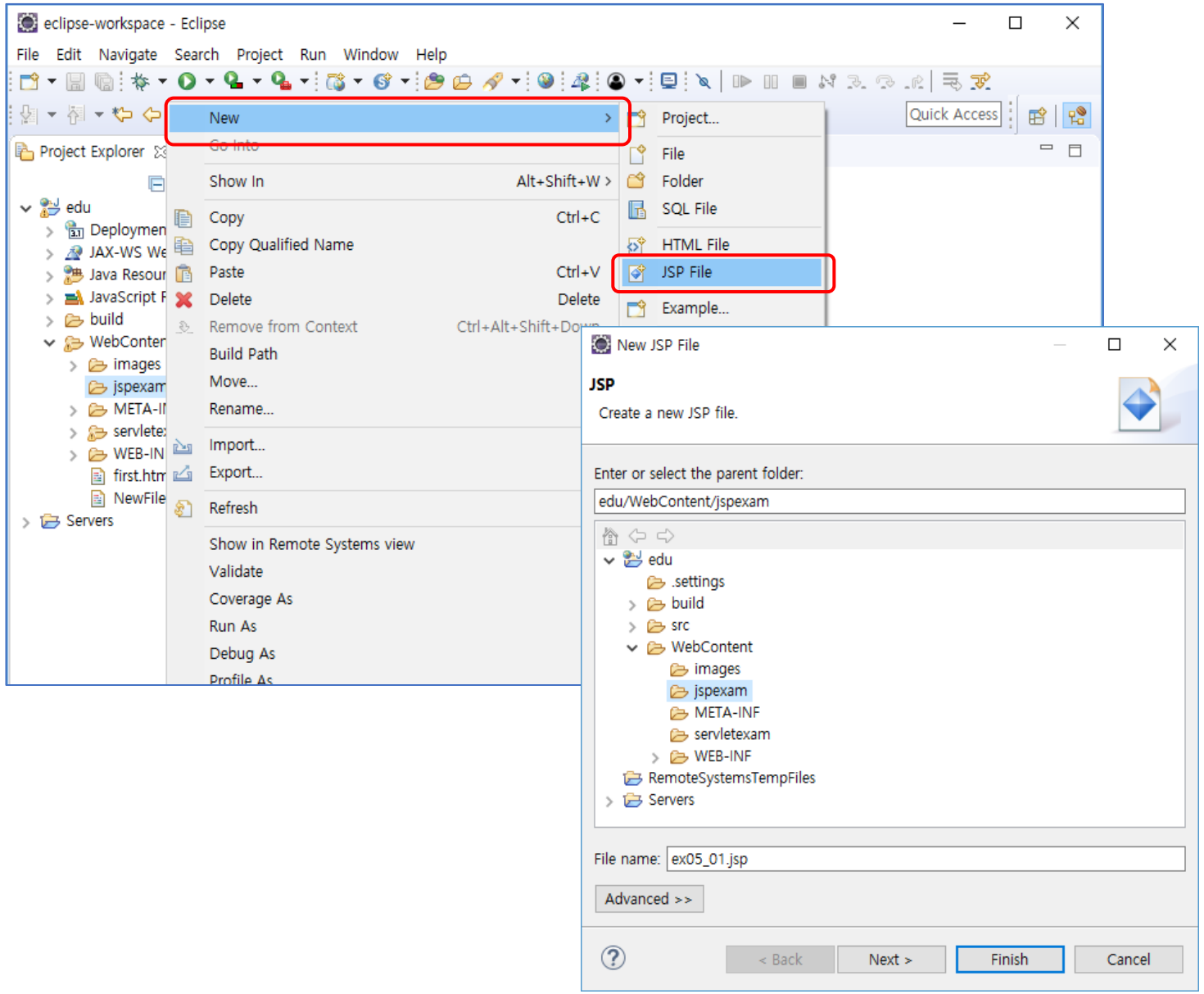
JSP의 개요

- ✓ JSP는 JavaServer Pages의 약어로서 HTML 또는 XML 기반의 동적인 웹 콘텐츠를 개발하는 기술
- ✓ Servlet 기술이 발표되고 1년 후인 1999년 6월에 썬 마이크로시스템즈㈜에서 발표
- ✓ Servlet과 JSP 모두 웹서버단에서 수행되는 기술이라는 것은 동일하지만 구현 방식이 다름
- ✓ JSP는 웹페이지의 콘텐츠를 만들면서 정적인 내용은 HTML 또는 XML 기술로 작성하고, 동적인 내용은 JSP 태그와 스크립트 코드로 작성하는 기술
- ✓ 동적인 내용이라는 것은 클라이언트로부터 JSP가 요청될 때마다 서버에서 수행되고 수행 결과가 응답 내용에 포함되는 것임

- ✓ Servlet은 Java로 구현된 프로그램으로서 수행 결과를 HTML로 응답하게 구현하는 기술이며 JSP는 HTML 문서 안에 JSP 태그와 동적인 처리를 담당하는 Java 코드를 삽입하여 구현하는 기술임

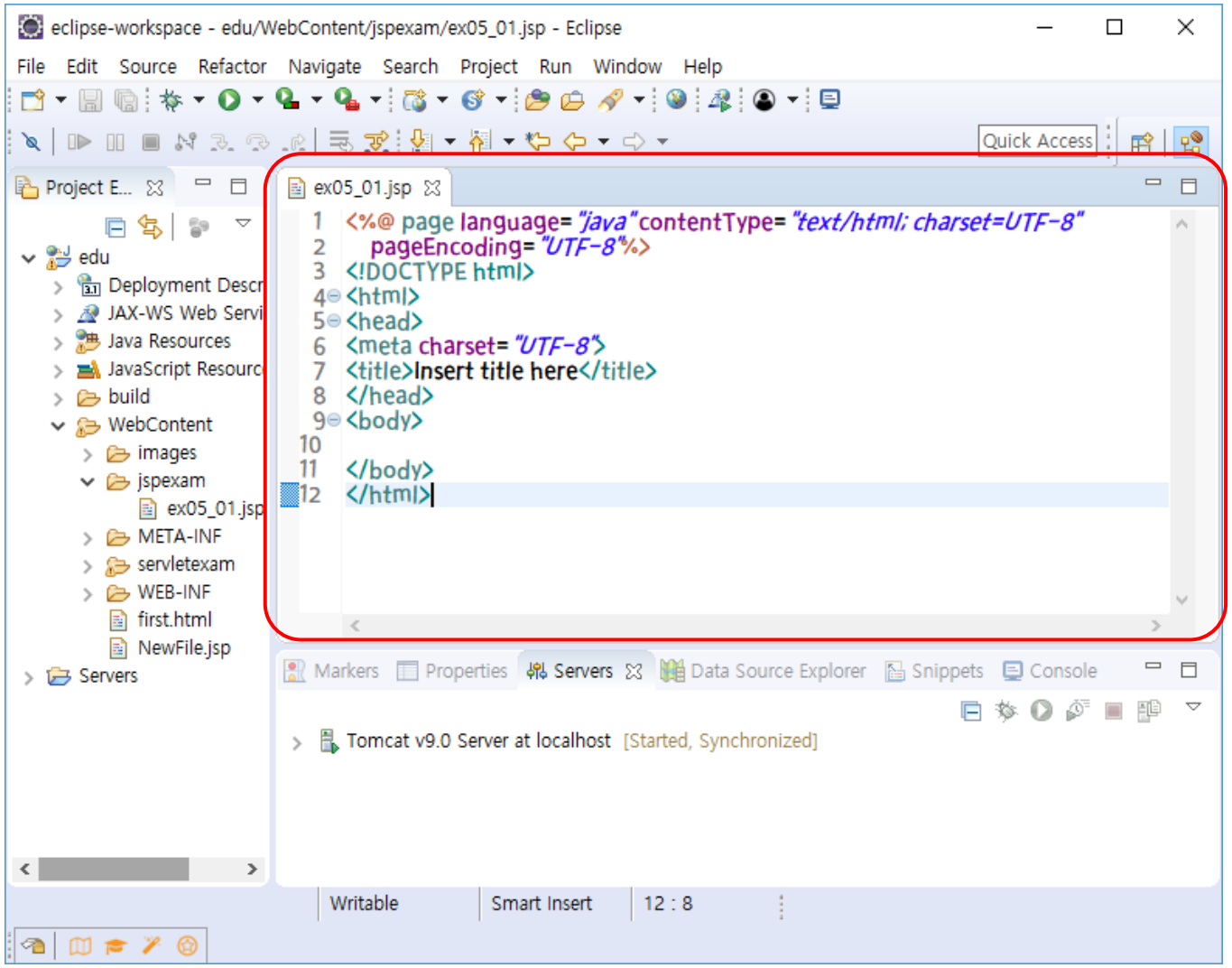
1-1. Eclipse에서 JSP 생성 프로세스

- ✓ JSP의 경우에도 Eclipse를 사용하면 좀 더 편하게 작성하고 테스트할 수 있음
- ✓ 화면 왼쪽의 Project Explorer뷰에서 edu라는 dynamic web project를 오픈
- ✓ WebContent 폴더에 생성된 jspexam 폴더를 선택하고 오른쪽 버튼을 클릭한 후 New 메뉴를 선택하고 서브메뉴에서 JSP File을 선택
- ✓ File name 항목에 ex05_01.jsp를 입력하고 Next 버튼을 클릭



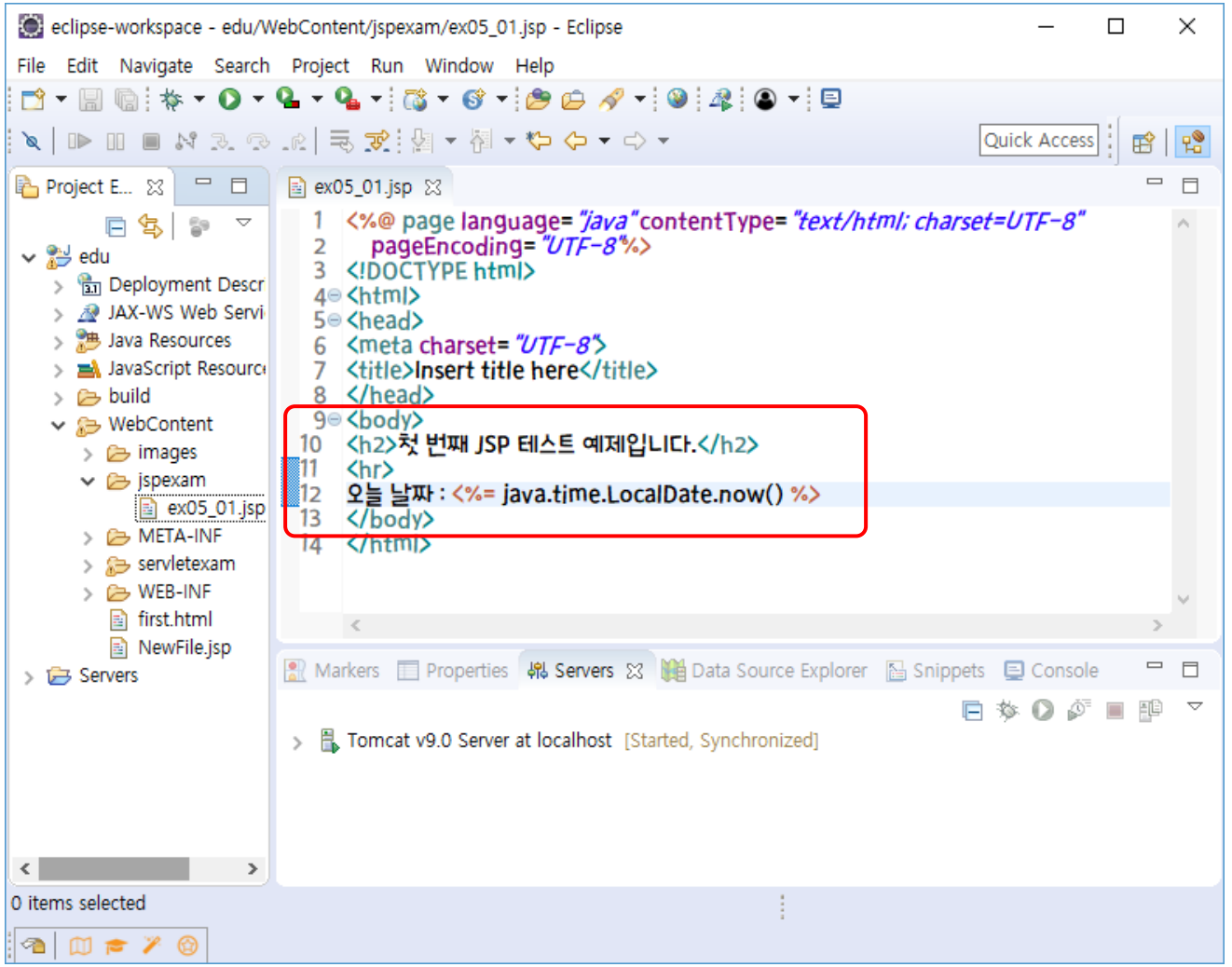
1-1. Eclipse에서 JSP 생성 프로세스

✓ 다음과 같이 선택된 템플릿 내용으로 구성된 JSP 파일이 생성된 것을 볼 수 있음



1-1. Eclipse에서 JSP 생성 프로세스

✓ <body> 태그 안에 다음 코드를 추가합니다

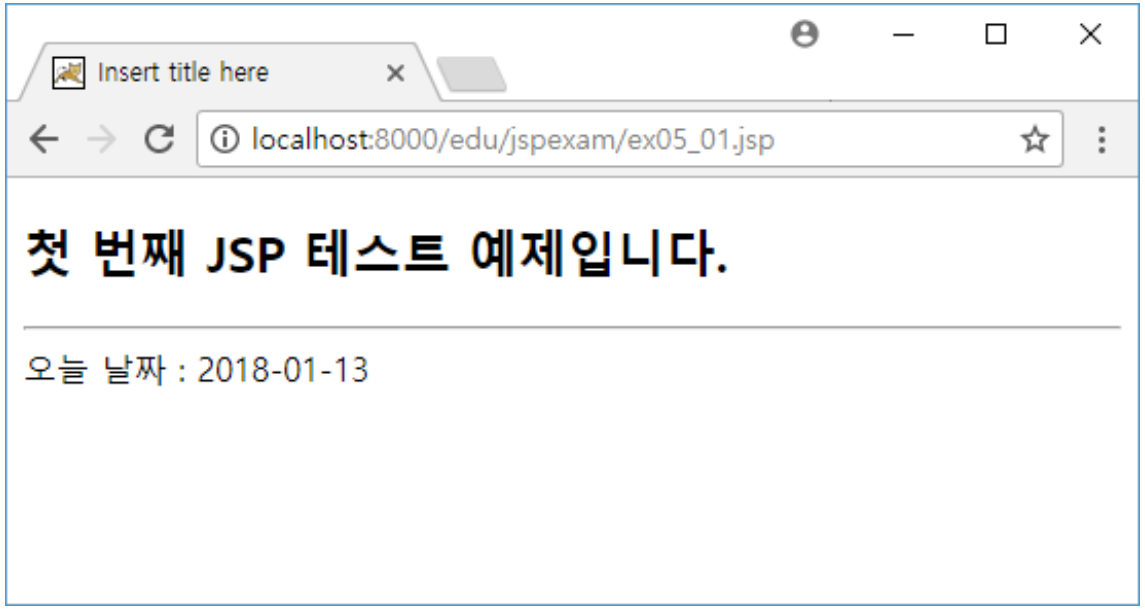


1-1. Eclipse에서 JSP 생성 프로세스

✓ 저장한 후 Tomcat 서버를 기동시킨 다음 Chrome 브라우저에서 다음과 같이 입력하여 요청

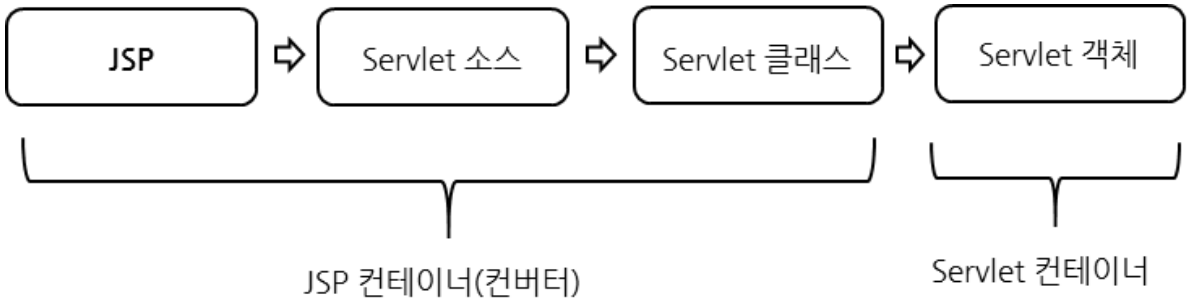
http://localhost:8000/edu/jspexam/ex05_01.jsp

✓ 다음과 같이 ex05_01.jsp의 수행 결과가 응답



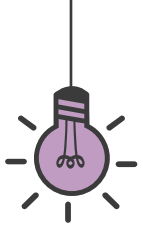
1-2. JSP 처리 방식

- ✓ JSP는 웹 클라이언트로부터의 요청 시 웹서버 단에서 수행되고 그 결과는 요청을 보내온 클라이언트에게 응답되는 기술
- ✓ 웹서버의 JSP 컨테이너(컨버터)에 의해 Servlet 소스 코드로 변환되고 수행 가능한 Servlet 클래스로 변환되며 변환된 이후부터는 Servlet 컨테이너에 의해 객체가 생성되고 수행



JSP 개발 시 선택할 수 있는 구현 요소들의 종류

- 표준 지시자(Standard Directives)
- 스크립팅(Scripting)
- 템플릿 콘텐츠(Template Contents)
- 표준 액션(Standard Actions)
- 표현 언어(Expression Language)
- JSTL(JSP Standard Tag Library)

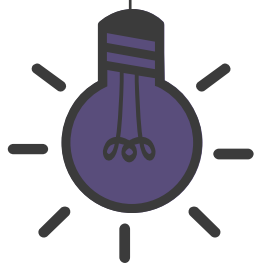


1-3. JSP의 구현 요소

- ✓ JSP 구현 시 선택할 수 있는 구현 요소들은 표준 지시자, 스크립팅, 템플릿 콘텐츠, 표준 액션, 표현 언어 그리고 태그 확장 메커니즘까지 다양
- ✓ 템플릿 콘텐츠란 정적인 콘텐츠라고도 할 수 있는 텍스트 내용입니다. 주로 HTML이나 XML 태그로 구성
- ✓ 서버에서 수행되는 코드가 아니며 웹 클라이언트에 그대로 전달되는 내용
- ✓ 서버단에서 처리할 기능으로써 JSP의 스크립트 태그와 Java
- ✓ JSP에서 제공되는 액션 태그를 사용하여 구현하는 방법
- ✓ 서버단에서 처리하려는 기능을 일정 규격의 Java 클래스로 만들고 객체 생성은 액션 태그를 사용
- ✓ 객체의 getter 메서드를 호출하고 수행 결과를 출력하는 부분은 EL을 사용하여 좀 더 간단하게 구현할 수 있음
- ✓ JSP의 액션 태그는 두 종류가 있음
- ✓ JSP에서 기본으로 지원하는 표준 액션 태그와 만들어서 사용하는 커스텀 액션 태그
- ✓ JSTL은 JSP Standard Tag Library의 약어로서 자주 사용되는 기능의 커스텀 액션 태그들을 모아놓은 것
- ✓ JSP에서 표준으로 지원되는 태그들이 아니므로 JSTL을 사용할 수 있게 지원하는 라이브러리를 설치하여 환경을 만들어야 사용할 수 있음

02

JSP의 스크립트 태그와 내장객체



2-1. 스크립트 태그

- ✓ JSP의 스크립트 태그란 JSP의 동적인 처리에 필요한 Java 코드를 포함시키는 용도의 태그
- ✓ 스크립트 태그로는 표현식 태그, 수행문 태그, 선언문 태그 그리고 지시자 태그 등이 있음
- ✓ JSP의 표현식 태그는 동적인 데이터를 응답 결과에 포함시키기 위해 사용
- ✓ JSP 수행을 요청할 때마다 서버에서 수행시킨 Java 코드의 수행 결과를 웹페이지 내에 출력하는 기능을 제공

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

- ✓ 수행문 태그는 JSP 수행이 요청될 때마다 서버에서 수행해야 하는 Java 코드를 추가하고자 할 때 사용하는 스크립트 태그
- ✓ <% 와 %>를 사용하여 템플릿 콘텐츠와 구분
- ✓ 멤버(멤버 변수 또는 메서드) 정의와 관련된 내용을 제외하고 어떠한 수행 코드든(예를 들어 제어문, 변수 선언문 등) 올 수 있음

2-1. 스크립트 태그

- ✓ JSP에 구현되는 Java 코드 중에서 멤버 변수 선언이나 메서드 정의는 선언문 태그를 사용
- ✓ 선언문 태그에 작성된 내용은 Servlet 소스로 변환될 때 Servlet 클래스의 멤버가 됨
- ✓ JSP는 최초 수행 전에 Servlet으로 변환되어 처리되므로 JSP의 수행은 Servlet과 동일
- ✓ JSP의 경우에도 멤버 변수는 지역 변수에 비해 많이 사용하지 않지만 멤버 변수를 선언할 일이 있다면 선언문 태그를 사용
- ✓ 메서드를 정의해서 사용하려는 경우에도 선언문 태그를 사용

```
<%! int member_v = 0; %>
<%!
int calculate(int n1, int n2, String oper) {
        int result=0;
        switch(oper) {
                case "addition" : result = n1 + n2; break;
                case "subtraction" : result = n1 - n2; break;
                case "multiplication" : result = n1 * n2; break;
                case "division" : result = n1 / n2;
        }
        return result;
}
%>
```

2-1. 스크립트 태그

- ✓ JSP에서 사용할 수 있는 주석은 3가지가 있dma
- ✓ HTML 주석 태그, Java 언어 주석 그리고 JSP 전용 주석 태그임

JSP에서 사용 가능한 주석	설 명
<%-- --%>	JSP 주석 태그로서 JSP 태그를 포함하여 주석 처리하려는 경우에 사용합니다. JSP 주석 태그에 작성된 코드는 JSP가 Servlet으로 변환될 때 제외됩니다.
<!-- -->	HTML 주석 태그로서 HTML 태그를 포함하여 일반 텍스트를 브라우저 화면에 출력되지 않게 하려는 경우 사용합니다. HTML 주석 태그에 작성된 코드는 서버에서 처리되고 브라우저로 전달되며 브라우저에 의해 제외됩니다.
/* */-다중행, //-단일행	Java 주석으로서 JSP의 스크립팅 태그 내에 작성되는 Java 코드의 일부를 주석 처리하려는 경우 또는 코드에 대한 설명을 작성하려는 경우에 사용합니다. JSP가 Servlet으로 변환될 때 Servlet 소스에 포함되지만 컴파일 시 제외됩니다.

2-1. 스크립트 태그

- ✓ 지시자 태그는 JSP가 Servlet으로 변환될 때 JSP 컨테이너에게 정보를 제공하는 역할을 지원
- ✓ 지시자 태그에는 Java 코드를 직접 작성하지 않지만 Servlet으로 변환하는데 영향을 주는 전반적인 구성 정보를 설정
- ✓ 지시자 이름 위치에 어떤 예약어가 오느냐에 따라 다양한 지시자 태그들이 사용됨

```
<%@ 지시자이름 속성1="값1" 속성2="값2"..... %>
```

지시자의 종류	작성 구문
page 지시자	<%@ page {attr = value ..} %>
include 지시자	<%@ include {attr = value ..} %>
taglib 지시자	<%@ taglib {attr = value ..} %>

2-1. 스크립트 태그

✓ page 지시자는 컨테이너가 참조하는 다양한 정보들 중에서 JSP에 종속적인 설정 정보들을 알려주기 위한 수단으로 사용

```
<%@ page page_directive_attr_list %>
page_directive_attr_list ::= { language="scriptingLanguage"}
{ extends="className" }
{ import="importList" }
{ session="true|false" }
{ buffer="none|sizekb" }
{ autoFlush="true|false" }
{ isThreadSafe="true|false" }
{ info="info_text" }
{ errorPage="error_url" }
{ isErrorPage="true|false" }
{ contentType="ctinfo" }
{ pageEncoding="peinfo" }
{ isELIgnored="true|false" }
{ deferredSyntaxAllowedAsLiteral="true|false"}
{ trimDirectiveWhitespaces="true|false"}
```

2-1. 스크립트 태그

- ✓ contentType 속성은 JSP가 생성하는 응답의 MIME(Multipurpose Internet Mail Extensions) 타입을 지정하기 위한 속성

```
<%@ page contentType="text/html; charset=UTF-8%>
```

- ✓ import 속성은 JSP에서 자주 사용되는 속성으로서 JSP에서 Java API를 사용하는 경우 사용되는 Java 클래스들에 대한 패키지를 import 하도록 지정

```
<%@ page import="java.util.*, java.lang.*" %>
<%@ page import="java.util.Hashtable, java.util.List" %>
```

- ✓ errorPage 속성은 수행 오류 발생 시 대신 응답되는 JSP를 설정하는 속성

```
<%@ page errorPage="errorPage.jsp" %>
```

2-1. 스크립트 태그

- ✓ isErrorPage 속성은 errorPage 속성과 연관된 속성으로서 errorPage.jsp와 같이 오류 페이지를 응답하는 용도의 JSP에서는 이 속성을 true로 설정
- ✓ isErrorPage 속성의 값을 true로 지정하면 exception이라는 JSP의 내장 객체를 사용할 수 있음
- ✓ exception 내장 객체는 이 페이지가 수행되기 전에 발생한 예외 상황에 대한 정보를 담고 있는 예외 객체를 참조

```
<%@ page isErrorPage="true" %>
```

- ✓ session 속성은 JSP에서 HttpSession 객체를 사용할지의 여부를 설정
- ✓ 기본값은 true이기 때문에 모든 페이지들은 자동으로 HttpSession 객체를 생성
- ✓ HttpSession 객체를 사용하지 않는 JSP는 이 속성의 값을 false로 지정하여 필요 없는 HttpSession 객체를 생성하지 않게함

```
<%@ page session="false" %>
```

2-1. 스크립트 태그

- ✓ buffer 속성은 JSP의 버퍼 출력을 조정합
- ✓ 버퍼를 사용하지 않고 HTTP 응답을 클라이언트로 바로 보내려면 이 속성의 값을 none으로 할당해 줌
- ✓ buffer 속성의 기본값은 8kb이므로 대부분 이 속성의 값은 설정하지 않음

- ✓ autoFlush 속성도 버퍼 출력과 연관되는 속성으로서 출력 버퍼가 Full이 되었을 때의 동작을 설정

- ✓ language 속성은 JSP에서 사용되는 스크립트 언어의 종류를 지정할 때 사용합니다. 이 항목이 생략되면 JSP 컨테이너는 기본적으로 Java 언어로 간주

- ✓ pageEncoding 속성은 응답되는 결과의 문자 집합에 대한 정보를 지정하는 용도로 사용합니다. 생략하면 기본값은 "ISO-8859-1"이므로 출력 결과에 한글이 포함되어 있는 경우에는 "UTF-8"을 설정

- ✓ isELIgnored 속성은 JSP에서 EL 표현식의 사용 여부를 결정

2-1. 스크립트 태그

- ✓ deferredSyntaxAllowedAsLiteral 속성은 JSP 2.1에서 추가된 속성으로서 JSP(Java Server Faces) 표현식 작성 시 사용되는 #{ 기호를 템플릿 콘텐츠의 내용으로 사용 가능한지 여부를 결정하는 속성

- ✓ trimDirectiveWhitespaces 속성은 JSP 2.1에서 추가된 속성으로서 trimDirectiveWhitespaces 속성의 값을 true로 지정하면 선언문 태그에 의해서 생성되는 빈 행들이 제거

2-2. 내장 객체

✓ JSP는 동적인 처리 기능 즉 서버단에서 수행시키는 기능을 Java로 구현하는데 있어서 자주 사용되는 API들을 내장 객체라는 요소로 제공

✓ 내장 객체를 통해서 처리할 수 있는 대표적인 기능들은 다음과 같습니다.

- 요청 파라미터(Query 문자열) 읽어오기
- 응답 결과 클라이언트로 전송하기
- 세션 객체 사용하기
- 설정 정보 읽어오기

객체 변수	클래스 및 인터페이스
request	javax.servlet.http.HttpServletRequest
response	javax.servlet.http.HttpServletResponse
pageContext	javax.servlet.jsp.PageContext
session	javax.servlet.http.HttpSession
application	javax.servlet.ServletContext
config	javax.servlet.ServletConfig
out	javax.servlet.jsp.JspWriter
page	java.lang.Object
exception	java.lang.Throwable

2-2. 내장 객체

- ✓ JSP에서 사용되는 내장 객체들은 기능과 역할에 따라서 객체의 스코프가 정해져 있음
- ✓ 스코프라는 것은 객체가 생성되어 메모리상에 유지되는 동안으로서 4가지 스코프가 존재

스코프(scope)	설명
page scope	요청된 JSP가 수행되는 동안입니다. JSP 수행이 끝나면 더 이상 이 객체는 사용할 수 없습니다.
request scope	요청이 끝날 때까지 객체 상태를 유지합니다. 그러므로 forward나 include 관계에 있는 다른 JSP와 객체를 공유할 수 있습니다.
session scope	요청이 종료되어 응답되어도 일정 시간 동안 객체 상태를 계속 유지합니다. 최대 유지 시간은 브라우저가 기동되어 있는 동안입니다. 그리고 웹 클라이언트인 브라우저별로 객체가 생성되고 개별적으로 유지하고 사용됩니다.
application scope	서버가 기동되어 있는 동안 객체 상태를 유지합니다. 모든 클라이언트가 공유하게 되는 객체입니다.

2-2. 내장 객체

✓ request, session, application 내장 객체들은 Servlet과 JSP 사이에서 객체 상태의 데이터를 주고받을 수 있게 지원

메서드	설명
setAttribute(key,value)	주어진 key(이름 등)로 속성값(객체)을 설정합니다.
getAttribute(key)	주어진 key로 설정된 속성값을 얻어냅니다.
getAttributeNames( )	설정된 모든 속성들의 key(이름)을 추출합니다.
removeAttribute(key)	주어진 key로 설정된 속성값을 제거합니다.

2-2. 내장 객체

- ✓ **request객체와 response 객체**는 각각 HttpServletRequest타입 객체와 HttpServletResponse 타입 객체를 참조하는 변수

```
public void _jspService(HttpServletRequest request, HttpServletResponse response)
```

- ✓ HttpServletRequest와 Http ServletResponse 객체를 사용하고자 하는 경우 각각 request와 response 변수로 사용할 수 있음
- ✓ request는 Servlet과 마찬가지로 요청 정보나 Query 문자열을 추출하는 용도로 사용할 수 있음
- ✓ response는 응답 헤더 설정이나 Cookie 설정 그리고 redirect 방식의 요청 재지정 용도로 사용할 수 있음

2-2. 내장 객체

- ✓ **out은** PageContext 객체의 getOut( ) 메서드를 사용하여 JSP 컨테이너에 의해 자동 생성되고 초기화됨
- ✓ JspWriter 타입의 객체로서 JspWriter는 java.io.Writer 클래스를 상속받아 생성된 클래스
- ✓ 출력 기능을 담당하는 write( ), print( ) 메서드를 사용할 수 있음

- ✓ **session은** HttpSession 객체임
- ✓ 상태 정보 유지 기술에 사용된 HttpSession 객체를 JSP에서도 동일한 용도로 사용할 수 있음
- ✓ JSP에서는 객체 생성이나 추출을 직접 구현할 필요가 없으며 JSP에서 자동 생성되는 HttpSession이라는 내장 객체를 session이라는 변수명으로 사용하면 됨

2-2. 내장 객체

- ✓ **application은** javax.servlet.ServletContext 타입의 객체를 참조
- ✓ 서버에 등록된 웹 애플리케이션마다 ServletContext 객체가 한 개씩 생성되며 서버는 이 객체를 통해서 웹 애플리케이션을 관리하고 처리
- ✓ 서버가 기동될 때 객체가 생성되고 서버가 종료될 때 객체가 해제되므로 가장 긴 라이프타임을 갖는 내장 객체
- ✓ Servlet 컨테이너에 대한 정보를 추출하거나 서버가 종료될 때까지 필요한 객체를 보관하는 용도로 사용할 수 있는 내장 객체

- ✓ **pageContext는** javax.servlet.jsp.PageContext 타입의 객체를 참조
- ✓ pageContext 객체는 다른 내장 객체를 액세스할 수 있게 하는 객체

2-2. 내장 객체

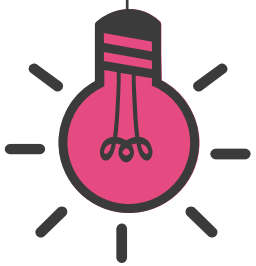
- ✓ **page는** JSP에 의해 생성되는 Servlet 객체에 대한 참조 변수
- ✓ 자기 자신에 대한 Servlet 객체를 참조할 수 있는 내장 객체입니다. Java 프로그래밍에서 this를 사용하는 것과 동일

- ✓ **config는** ServletConfig 객체
- ✓ ServletConfig 객체는 주로 초기화 파라미터를 추출하는 용도로 사용

- ✓ **exception은** java.lang.Throwable 클래스의 객체
- ✓ 다른 객체들과 다르게 JSP 처리 중 수행 오류가 발생했을 때 사용할 수 있는 내장 객체
- ✓ 유일하게 디폴트로는 사용할 수 없는 내장 객체로서 page 지시자에 isErrorPage 속성의 값을 true로 설정해야만 사용할 수 있음

03

JSP의 표준 액션 태그와 EL



JSP의 표준 액션 태그와 EL

- ✓ 액션 태그란 JSP를 개발할 때 자주 구현하는 기능을 제공하는 JSP의 고유 태그
- ✓ 자주 구현하는 기능을 미리 태그로 만들어 놓으면 일일이 스크립트 코드를 작성하지 않아도 되므로 JSP 구현을 좀 더 편하게 할 수 있음
- ✓ 액션 태그도 표준 액션 태그와 커스텀 액션 태그로 나뉨
- ✓ 표준 액션 태그는 JSP 스펙에 정의된 기능의 액션 태그로서 모든 JSP 컨테이너가 지원하므로 JSP 구현 시 언제든지 사용할 수 있음
- ✓ 커스텀 액션 태그는 개발자가 직접 구현하여 만드는 액션 태그로서 액션 태그의 작성 방법에 따라서 직접 정의하여 사용
- ✓ EL이란 Expression Language의 약어입니다. 표현 언어라고도 하는데 JSP의 표현식 태그와 거의 동일한 기능을 지원
- ✓ JSP 2.0부터 추가된 JSP의 구현 방식 중 하나로서 특정 스코프 영역에 보관된 객체를 추출하고 getter 메서드를 호출한 결과를 표현할 때 표현식 태그보다 훨씬 간단

3-1. 표준 액션 태그

- ✓ 표준 액션은 XML 구문을 기반으로 구현 방식과 태그명이 정해져 있으며 태그명 앞에 jsp라는 접두어를 붙여서 처리
- ✓ 대소문자가 구분되며 속성에 값을 할당할 때는 인용 부호를 붙여 주어야 함
- ✓ 시작 태그와 종료 태그 사이에 내용이 없다면 빈 요소 형식 (<태그명 ... />)으로 작성해야 함

표준 액션 태그	기능
<jsp:include>	JSP의 수행 결과 내에 다른 자원의 내용 또는 수행 결과를 포함합니다.
<jsp:forward>	요청된 JSP 대신 다른 자원의 내용 또는 수행 결과를 대신해서 클라이언트로 응답합니다.
<jsp:param>	<jsp:include> 또는 <jsp:forward>와 함께 사용되는 표준 액션 태그로서 include 또는 forward 되는 대상 페이지에 name과 value로 구성되는 데이터를 전달하는 용도의 태그입니다.
<jsp:useBean>	주어진 JavaBeans 클래스의 객체를 생성하거나 이미 생성된 객체를 추출합니다.
<jsp:getProperty>	JavaBeans 객체의 프로퍼티 값을 추출하여 출력합니다. (getter 메서드가 호출됩니다.)
<jsp:setProperty>	JavaBeans 객체의 프로퍼티에 값을 설정합니다. (setter 메서드가 호출됩니다.)

3-1. 표준 액션 태그

- ✓ <jsp:forward> 액션 태그는 동일 컨텍스트에 존재하는 또 다른 리소스(JSP 페이지나 Servlet으로)로 제어를 옮겨서 대신 응답하려는 경우 사용
- ✓ forward 액션 태그의 구문입니다. relativeURLspec이라는 것은 동일 컨텍스트 내에 존재하는 대상에 대한 URL을 의미하는 것으로 지정 시 컨텍스트 패스를 제외하고 나머지만 설정해야 함

```
<jsp:forward page="relativeURLspec"/>
또는
<jsp:forward page="urlSpec"> { <jsp:param .... /> }* </jsp:forward>
```

3-1. 표준 액션 태그

- ✓ <jsp:include> 표준 액션 태그는 다른 자원의 수행 결과를 포함하는 기능을 지원
- ✓ JSP 구현 시 사용할 수 있는 include 기능은 include 지시자와 include 표준 액션 태그가 있음
- ✓ 다음 표는 이 두 가지 방법의 기능을 비교 설명한 내용

항목	Include 지시자	Include 액션
구문	<%@ include file=... %>	<jsp:include page=... />
경로	JSP 파일의 상대 경로	JSP 파일의 상대 경로
대상	정적	정적 혹은 동적 (포함하는 대상에 따라)
설명	대상의 소스를 JSP 안에 포함하여 Servlet 소스로 변환하므로 JSP의 구문에 위배되는 내용은 포함할 수 없음.	대상을 독립적으로 수행시키고 수행 결과만을 포함하기 때문에 JSP가 아닌 다른 기술로 만들어진 자원이어도 대상이 될 수 있음.
수행	JSP가 Servlet으로 변환되는 시점	요청을 수행하는 시점. 요청 시마다 include가 처리됨.

3-1. 표준 액션 태그

- ✓ urlSpec에는 <jsp:forward> 액션 태그처럼 동일 컨텍스트 내에 존재하는 대상에 대한 URL만 설정해야 함
- ✓ <jsp:param>이라는 태그를 콘텐츠로 정의하여 include 되는 대상 페이지에 name과 value로 구성된 파라미터를 전달할 수 있음
- ✓ <jsp:include> 표준 액션 태그에는 flush 속성을 true 또는 false로 설정할 수 있음

```
<jsp:include page="urlSpec" flush="true|false"/>
또는
<jsp:include page="urlSpec" flush="true|false"> { <jsp:param .... /> } * </jsp:include >
```

3-1. 표준 액션 태그

- ✓ <jsp:useBean> 액션 태그는 JavaBeans 클래스의 객체를 생성하거나 이미 생성된 JavaBeans 객체를 추출하는 기능을 수행하는 태그
- ✓ JavaBeans 클래스란 컴포넌트로서 정해진 규격의 클래스라고 할 수 있음
- ✓ 다음에 제시된 규격을 준수하여 클래스를 구현해야 함
  - JSP에서 사용하는 Java 클래스는 패키지화되어야 하므로 패키지를 선언합니다.
  - 매개 변수가 없는 생성자 메서드를 정의합니다.
  - 객체가 수행하는데 필요한 값 그리고 수행 결괏값을 보관하는 멤버 변수(프로퍼티)를 private 타입으로 선언합니다.
  - 객체가 수행하는데 필요한 값(프로퍼티)은 setter 메서드를 통해서 설정할 수 있게 적당한 setter 메서드를 정의합니다.
  - 객체의 수행 결괏값(프로퍼티)을 getter 메서드를 통해서 추출할 수 있게 적당한 getter 메서드를 정의합니다.
- ✓ JavaBeans 클래스에 setter 메서드와 getter 메서드를 정의할 때는 각각 설정 기능과 추출 기능을 구현
- ✓ 다음과 같이 첫 글자를 대문자로 만든 프로퍼티명에 set 또는 get을 앞에 붙여서 정의함

```
private int number;
public void setNumber(int number) {
        this.number = number;
}
public int getNumber( ) {
        return number;
}
```

```
[ 경우1 ]
<jsp:useBean id="name" scope="page|request|session|application" typeSpec />
[ 경우2 ]
<jsp:useBean id="name" scope="page|request|session|application" typeSpec >
    body
</jsp:useBean>
```

3-1. 표준 액션 태그

- ✓ scope 속성은 생성되는 JavaBeans 객체가 저장될 영역을 지정
- ✓ page, request, session 그리고 application 중 하나를 설정할 수 있으며 기본값은 page 임

- ✓ <jsp:getProperty> 액션 태그는 JavaBeans 객체의 getter 메서드를 호출하여 객체의 프로퍼티의 값 즉, 수행 결괏값을 추출하여 출력하는 기능을 제공
- ✓ 다음에 제시된 구문을 적용하여 엠티 태그(emptytag) 형식으로만 사용할 수 있음

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

3-1. 표준 액션 태그

- ✓ <jsp:setProperty> 액션 태그는 JavaBeans 객체의 setter 메서드를 호출하여 객체에서 사용하는 데이터 값인 프로퍼티의 값을 설정하는 기능을 제공
- ✓ <jsp:setProperty> 태그는 다음에 제시된 구문을 적용하여 엠티 태그(emptytag) 형식으로만 사용할 수 있음

```
<jsp:setProperty name="beanName" prop_expr />
prop_expr ::= property="*" |
              property="propertyName"|
              property="propertyName" param="parameterName"|
              property="propertyName" value="propertyValue"
              propertyValue ::= string
              propertyValue ::= expr_scriptlet
```

사용 속성명	기능
value	프로퍼티 값을 직접 설정합니다.
param	Query 문자열에서 param 속성에 할당된 값과 동일한 이름으로 전달되는 value 값을 추출해서 설정합니다.
모두 생략	Query 문자열에서 property 속성에 할당된 값과 동일한 이름으로 전달되는 value 값을 추출해서 설정합니다.

3-2. EL(Expression Language)

- ✓ JSP 2.0부터 지원된 EL(Expression Language)은 JSP 구현 시 수행 결과를 표현(출력)하려는 위치에 사용되는 JSP의 또 다른 구현 방식
- ✓ 모든 기능을 EL로 구현할 수 있는 것은 아님
- ✓ 특정 스코프 영역에 보관되어 있는 객체를 추출하여 이 객체의 값 즉, 프로퍼티 값을 추출하여 출력하려는 경우 그리고 Query 문자열의 value 값이나 요청 헤더 값 그리고 간단한 연산 결괏값을 출력하려는 경우에 사용
- ✓ 대부분 JSP의 표현식 태그와 중복되는 기능이지만 EL을 사용하면 좀 더 간단하게 구현하는 것이 가능
- ✓ EL의 구현 방법은 다음과 같이 $ 기호와 블록({ })을 사용

```
${ EL식 }
#{ EL식 }
```

3-2. EL(Expression Language)

- ✓ ${ EL식 }은 JSP의 스크립트 태그(선언문, 표현식, 수행문)를 제외한 나머지 부분에서 사용할 수 있음
- ✓ #{ EL식 }은 Deferred Expression이라고 합니다. JSP 2.1 버전부터 새롭게 지원하는 구문으로 JSF(JavaServer Faces)에서 사용되던 표현 언어 구문
- ✓ 표현식이 실행되는 시점에 바로 식을 계산하는 ${ EL식 }과 다르게 #{ EL식 }은 여러 단계로 분할하여 수행되는 JSF의 수행 방식에 알맞게 지연된 평가와 계산을 지원
- ✓ Query 문자열을 추출하여 출력하는 경우도 다음과 같이 스크립트 태그를 사용하는 것보다 간단하게 구현

구문	기능
수행문 태그	<% out.println(request.getParameter("q")); %>
표현식 태그	<%= request.getParameter("q") %>
EL	${param.q} 또는 ${param["q "]}

3-2. EL(Expression Language)

✓ 다음은 EL에서만 사용할 수 있는 내장 객체들을 소개하는 표

EL 내장 객체	기능
pageScope	Page 영역에 존재하는 객체를 참조할 때 사용
requestScope	Request 영역에 존재하는 객체를 참조할 때 사용
sessionScope	Session 영역에 존재하는 객체를 참조할 때 사용
applicationScope	Application 영역에 존재하는 객체를 참조할 때 사용
param	파라미터 값(Query 문자열)을 얻어올 때 사용
paramValues	파라미터 값(Query 문자열)을 배열로 얻어올 때 사용
header	요청 헤더 정보를 얻어올 때 사용
headerValues	요청 헤더 정보를 배열로 얻어올 때 사용
cookie	쿠키 정보를 추출할 때 사용
initParam	컨텍스트의 초기화 파라미터를 추출할 때 사용
pageContext	pageContext 객체를 참조할 때 사용

3-2. EL(Expression Language)

- ✓ EL 표현에서 사용할 수 있는 연산자들 리스트이며 대부분의 연산자들은 Java 언어에서도 볼 수 있는 연산자들임
- ✓ /, %, >, >=, <, <=, &&, || 그리고 ! 연산자들은 기호뿐만 아니라 div, mod, gt, ge, lt, le, and, or 그리고 not 등의 명칭으로도 사용할 수 있음
- ✓ 점(.) 연산자와 [ ] 연산자 그리고 empty 연산자는 Java 언어에는 존재하지 않거나 사용 방법이 다르므로 자세히 다뤄볼 필요가 있음

연산자	기능
.	JavaBeans 객체의 getter 메서드 호출 Map 객체에 저장된 데이터 추출
[ ]	배열 또는 리스트에 접근 . 연산자의 기능도 지원
( )	연산할 때 연산자 우선순위 조정
logical expr ? expr1:expr2	logical expr의 연산 결과가 true면 expr1을 false면 expr2를 사용
empty	주어진 변수의 값이 비었는지 점검 값이 null 이거나 “ ” 이면 true
+,-,*,/(div),%(mod)	사칙연산과 나머지 연산
+=	문자열 결합
&&(and)	두 개의 항이 모두 true인 경우만 true
||(or)	두 개의 항 중에서 하나라도 true이면 true
!(not)	주어진 항이 true면 false, false면 true로 변경
==(eq), !=(ne)	두 항의 값이 동일한지, 서로 다른지 점검
<(lt), >(gt), <=(le), >=(ge)	두 항의 값을 비교하여 대소 관계 점검

3-2. EL(Expression Language)

✓ . 연산자는 객체의 getter 메서드를 호출하거나 주어진 key 값으로 저장된 데이터 값을 추출하려는 경우에 사용하는 연산자

```
${ param.addr }                    ${ param["addr"] }
${ requestScope.msg }              ${ requestScope["msg"] }
${ sessionScope.count }            ${ sessionScope["count"] }
${ header.user-agent } - 오류                ${ header["user-agent"] }
```

✓ EL식을 작성할 때 다음과 같이 EL 내장 객체에 . 연산자를 사용하고 적당한 명칭(xxx)을 붙이게 됨

✓ 이 때의 xxx는 . 연산자 앞에 사용된 EL 내장 객체의 타입에 따라 그 적용 규칙이 달라짐

```
${ EL내장객체.xxx }
```

1) EL 내장 객체가 일반 Java 객체(JavaBeans)이면 getXxx( )를 호출한 결과가 됩니다.
2) EL내장 객체가 Map 객체이면 get("xxx")을 호출한 결과가 됩니다.
3) 추출된 것이 null 이거나 아무것도 없으면 아무것도 표현(출력)하지 않습니다

3-2. EL(Expression Language)

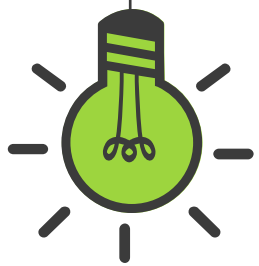
✓ EL 표현식에서 특정 스코프 영역에 보관된 객체를 추출하기 위해 다음과 같이 각 스코프에 따라서 정해진 EL내장 객체와 . 연산자 뒤에 추출하려는 객체의 이름(객체가 저장될 때 부여된 이름)을 붙여서 식으로 만듬

스코프	기능
page	${ pageScope.이름 }
request	${ requestScope.이름 }
session	${ sessionScope.이름 }
application	${ applicationScope.이름 }

✓ EL 내장 객체를 생략하고 ${ 이름 }과 같이 추출하려는 객체의 이름만 사용할 수도 있음

✓ "이름"에 해당하는 객체를 제일 먼저 page 영역에서 찾고 없으면 request 영역, session 영역 그리고 application 영역으로 차례대로 찾아서 제일 먼저 찾아지는 객체를 사용

04 JSP의 JSTL 활용



JSP의 JSTL 활용

- ✓ 표준 액션 태그는 JSP가 내장하고 있는 액션 태그로서 사용할 수 있는 태그의 종류와 기능이 정해져 있음
- ✓ 커스텀 액션 태그는 개발자가 직접 만들어 사용하는 액션 태그이므로 종류와 기능을 얼마든지 확장해 갈 수 있음
- ✓ JSTL(JavaServer Pages Standard Tag Library)은 JSP를 구현할 때 자주 필요한 기능의 커스텀 태그들을 모아놓은 것으로 Jakarata Project에서 제공
- ✓ 지원 기능에 따라 core 라이브러리, xml 라이브러리, format 라이브러리, sql 라이브러리 그리고 functions 라이브러리로 구성

4-1. core 라이브러리

- ✓ core 라이브러리는 변수 선언, 조건문, 반복문 등 간단한 프로그램 로직 구현을 대신할 수 있는 기능의 커스텀 태그들이 지원되는 라이브러리임
- ✓ JSTL의 core 라이브러리를 사용하려면 다음과 같이 taglib 지시자 태그를 정의해야 함

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

- ✓ <c:set> 태그는 JSTL 태그에서 사용되는 EL 변수를 만들고 값을 설정할 때 또는 EL 변수의 프로퍼티 값을 설정할 때 사용
- ✓ EL 변수란 EL 식에서 사용되는 변수로서 page, request, session 그리고 application의 4개 스코프에 저장된 객체의 이름

```
<c:set target="대상" property="프로퍼티이름" value="값"/>
<c:set target="대상" property="프로퍼티이름">값</c:set>
```

```
<c:remove var="varName" [scope="영역"]/>
```

4-1. core 라이브러리

- ✓ <c:out> 태그는 JspWriter 객체를 이용하여 데이터를 출력할 때 사용

```
<%--방법 1 --%>
<c:out value="value" [escapeXml="(true | false)"] [default="defaltValue"]/>
<%--방법 2 --%>
<c:out value="value" [escapeXml="(true | false)"] >
        defaltValue
</c:out>
```

- ✓ <c:if> 태그는 Java 언어의 if 문과 동일한 기능을 제공
- ✓ else 절 기능을 지원하지 않으며 단순 if 문만을 지원

```
<c:if test="조건">
...
</c:if>
```

4-1. core 라이브러리

✓ <c:choose> 태그는 Java의 switch 구문과 if-else 또는 if-else if 구문을 혼합한 형태로서 다수의 조건문을 하나의 블록에서 수행하고자 할 때 사용

```
<c:choose>
    <c:when test="${member.level == 'trial'}">
        ...
    </c:when>
    <c:when test="${member.level == 'regular'}">
        ...
    </c:when>
    <c:otherwise>
        ...
    </c:otherwise>
</c:choose>
```

4-1. core 라이브러리

✓ <c:forEach> 태그는 배열, Collection 또는 Map 객체에 저장되어 있는 값들을 순차적으로 처리할 때 사용할 수 있는 태그

```
<c:forEach var="i" begin="1" end="10" step="2">
        ${ i } 사용
</c:forEach>
```

```
<c:forEach var="item" items="<%= someItemList %>" varStatus="status">
        ${status.index + 1 } 번째 항목 ${item.name }
</c:forEach>
```

✓ <c:forTokens> 태그는 java.util.StringTokenizer 클래스와 동일한 기능을 제공

```
<c:forTokens var="token" items="문자열" delims="구분자">
        ${ token }의 사용
</c:forTokens>
```

4-1. core 라이브러리

- ✓ <c:import> 태그는 특정 URL의 결과를 읽어와 현재 위치에 삽입하거나 EL 변수에 저장할 때 사용
- ✓ <jsp:include>가 동일 웹 애플리케이션 내 다른 자원의 수행 결과를 포함하는 기능이라면, <c:import> 태그는 동일 웹 애플리케이션 뿐만 아니라 외부의 다른 자원을 읽어와 포함시킬 수 있게 함

```
<c:url value="URL" [var="varName"] [scope="영역"]>
      <c:param name="이름" value="값" />
</c:url>
```

- ✓ <c:url> 태그는 URL을 생성해 주는 기능을 제공

```
<c:url value="URL" [var="varName"] [scope="영역"]>
      <c:param name="이름" value="값" />
</c:url>
```

4-1. core 라이브러리

✓ <c:redirect> 태그는 response.sendRedirect( )처럼 지정한 페이지로 리다이렉트 시키는 기능을 제공

```
<c:redirect url="URL" [context="콘텍스트 경로"]>
   <c:param name="이름" value="값" />
</c:redirect>
```

✓ <c:catch> 태그는 발생된 예외를 EL 변수에 저장할 때 사용하는 태그로서, 다음과 같이 사용

```
<c:catch var="exName">
   예외가 발생할 수 있는 코드
</c:catch>
```

4-2. xml 라이브러리

- ✓ XML 문서에서 자주 사용하는 기능들을 커스텀 태그로 모아 놓은 것으로 JSP에서 JSTL의 xml 라이브러리를 사용하려면 다음과 같이 taglib 지시자 태그를 정의해야 함

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

- ✓ <x:out>은 파싱한 XML 문서에서 지정된 XPath 표현식에 대응되는 태그의 내용을 출력하는 태그

```
<x:out select="XPath표현식" escapeXml="true/false">
```

4-2. xml 라이브러리

- ✓ XPath란 XML Path Language를 의미합니다. XPath는 XML 문서의 특정 요소나 속성에 접근하기 위한 경로를 지정하는 언어
- ✓ 예제에서 사용된 XPath 표현식에 대한 소개

예제에서 사용된 XPath 표현식	기능 설명
$xdata//학생[1]/이름	xdata라는 EL 변수에 저장된 DOM 객체에서 첫 번째 <학생> 태그의 <이름>이라는 자식 태그를 찾고 이 태그의 콘텐츠를 사용합니다.
$xdata//학생[1]/@번호	xdata라는 EL 변수에 저장된 DOM 객체에서 첫 번째 <학생> 태그의 번호라는 속성을 찾고 이 속성의 값을 사용합니다.
$xdata//이름[@성별='남']	xdata라는 EL 변수에 저장된 DOM 객체에서 성별 속성의 값이 '남'인 <이름> 태그를 찾고 이 태그의 콘텐츠를 사용합니다.
$xdata//학생[@번호='st0004']	xdata라는 EL 변수에 저장된 DOM 객체에서 번호 속성의 값이 'st0004'인 <학생> 태그를 찾고 이 태그의 콘텐츠를 사용합니다.
$wrss/rss/channel/title	wrss라는 EL 변수에 저장된 DOM 객체에서 <rss>라는 최상위 태그를 찾고 <channel>이라는 자식 태그를 찾은 다음 이 태그의 <title>이라는 자식 태그를 찾아서 이 태그의 콘텐츠를 사용합니다.
$xdata//headerCd	xdata라는 EL 변수에 저장된 DOM 객체에서 <headerCd>라는 태그를 찾고 이 태그의 콘텐츠를 사용합니다.

4-2. xml 라이브러리

✓ <x:set>은 파싱한 XML 문서에서 지정된 XPath 표현식에 대응되는 태그의 내용을 지정된 변수에 저장하는 태그

```
<x:if var="변수명" select="XPath 표현식"/>
```

✓ <x:if>는 core 라이브러리 <c:if>와 유사한 기능을 지원

✓ 파싱한 XML 문서에 select 속성에 설정된 XPath 표현식 조건에 알맞은 태그가 존재하는지 여부에 따라 시작 태그와 종료 태그 사이의 코드의 수행 여부를 결정

```
<x:choose>
<x:when select="XPath 표현식" >참일 때 수행 코드</x:when>
    <x:otherwise>거짓일 때 수행 코드</x:otherwise>
</x:choose>
```

4-2. xml 라이브러리

✓ <x:forEach>는 파싱한 XML 문서에 select 속성에 설정된 XPath 표현식 조건에 알맞은 태그가 2개 이상인 경우 사용되는 태그로서 XML 문서에 대한 반복 처리가 필요할 때 사용

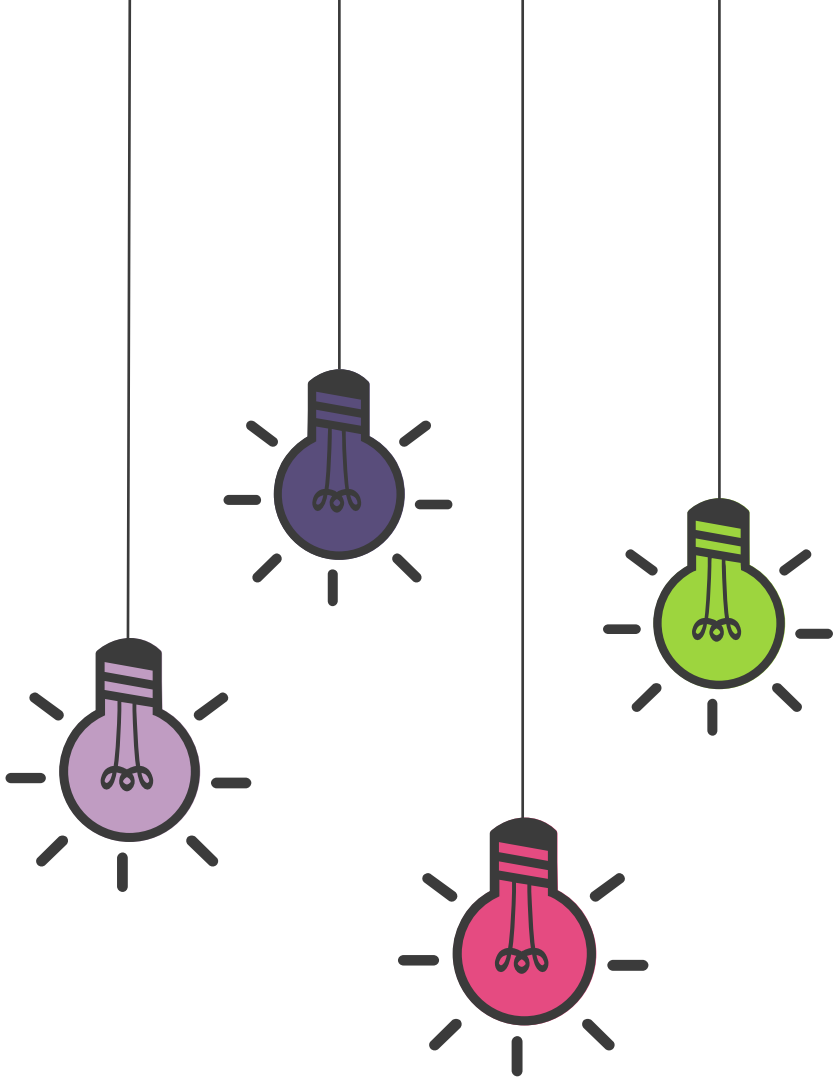
```
<x:forEach var="변수명" select="XPath 표현식" begin="시작 인텍스" end="끝 인덱스" step="증감식">
```

✓ <x:parse>는 XML 문서를 파싱할 때 사용하는 태그로서 XML 문서를 읽어서 DOM 객체를 생성

```
<x:parse xml="EL 변수의 표현식" varDom="변수명" scopeDom="범위" >
```

✓ <x:transform>은 XML 문서를 XSL 스타일 시트를 이용하여 새로운 문서로 변형시키는 역할

```
<x:transform var="변수명" scope="범위" doc="source" xslt="XSLTStyleSheet
```



감사합니다

THANK YOU FOR WATCHING