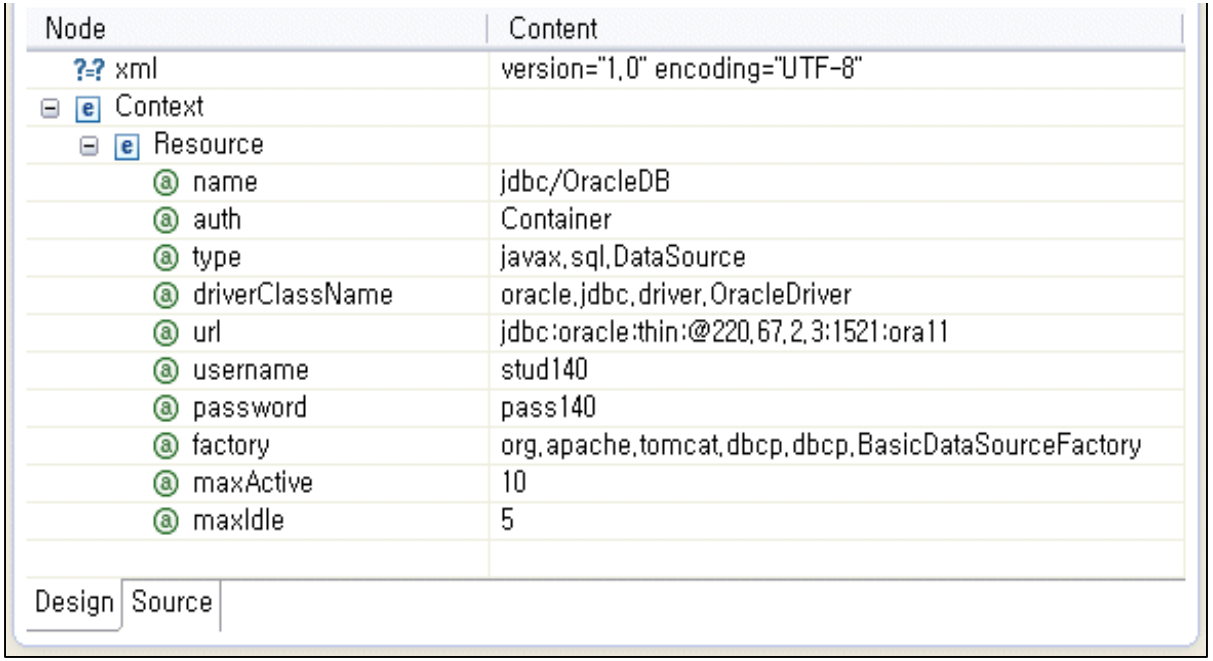
복 습 하 기

- JNDI 등록과 DBCP 이용한 데이터베이스 연동 작업
  - ① tomcat-dbcp.jar의 DBCP 관련 파일
  - ② 데이터베이스 서버, 계정, 암호 등의 "context.xml" 파일 생성
- WebContent/META-INF/context.xml 파일 생성 [Oracle 예]



Node	Content
?=? xml	version="1.0" encoding="UTF-8"
Context	
Resource	
name	jdbc/OracleDB
auth	Container
type	javax.sql.DataSource
driverClassName	oracle.jdbc.driver.OracleDriver
url	jdbc:oracle:thin:@220.67.2.3:1521:ora11
username	stud140
password	pass140
factory	org.apache.tomcat.dbcp.dbcp.BasicDataSourceFactory
maxActive	10
maxIdle	5

Design Source

복 습 하 기

DBCP을 이용한 데이터베이스 연동 프로그램

1단계

```
import="java.sql.*, javax.sql.*, javax.naming.*" %>
```

2,3단계

ⓐ Context init = new InitialContext();
ⓑ DataSource ds = (DataSource)init.lookup("java:/comp/env/jdbc/OracleDB");
ⓒ Connection con = ds.getConnection();

Part2. 기초 프로그래밍

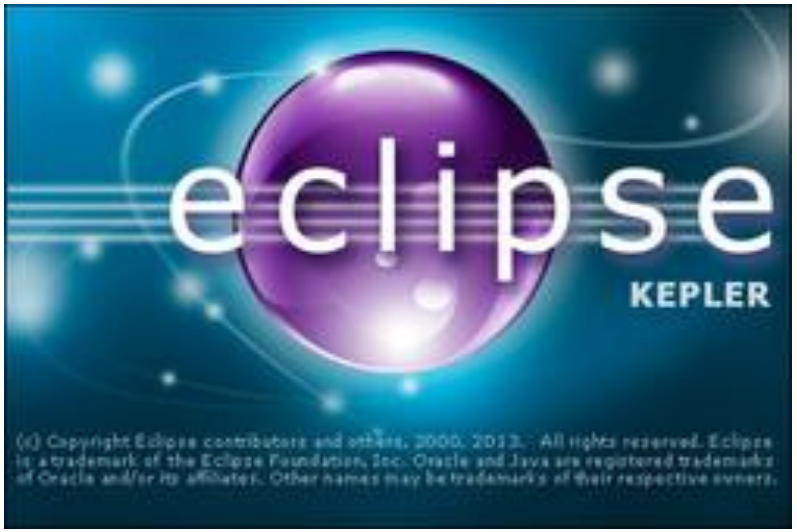
- Chapter4. JSP 기본문법
- Chapter5. JSP 내장객체
- Chapter6. JSP 입력 폼 설계
- Chapter7. JSP와 DB 연동
- Chapter8. 자바빈과 액션태그
- Chapter9. 쿠키와 세션
- Chapter10. 서블릿
- Chapter11. DBCP
- Chapter12. EL
- Chapter13. JSTL

문법과
기초 프로그래밍 실습

제12장 EL(Expression Language)

1. EL(Expression Language)
2. EL의 식
3. EL의 연산자
4. EL의 내장개체

${ }



오늘의 수업 주제

1. EL(Expression Language)

${ }

2. EL의 식

3. EL의 연산자

4. EL의 내장개체

강의 목표

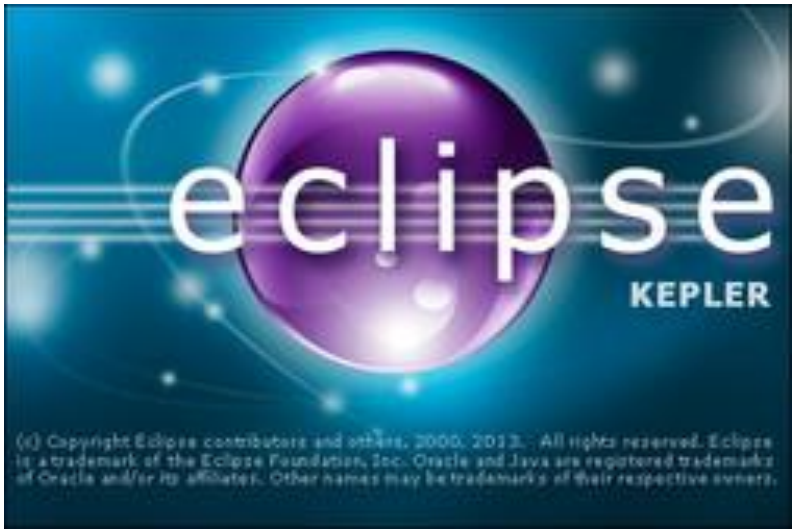
- 1. EL(Expression Language)
- 2. EL의 식
- 3. EL의 연산자
- 4. EL의 내장개체

${ }

12. EL

${ }

- 1 EL(Expression Language)
- 2 EL의 식
- 3 EL의 연산자
- 4 EL의 내장개체



1. EL(Expression Language)

${ }

- EL(Expression Language)
  - 표현식 언어 또는 익스프레션 언어
  - JSTL 1.0에 소개, JSP 2.0/JSTL 1.1에 추가된 스크립트 언어
  - 아파치 탐캣 5.0부터 사용 가능
  - JSP 페이지에서 자바코드 대신 액션태그 엘리먼트의 속성 값을 지정하는 역할
  - 표현식(<%= %>) 대용 효과
- 주요기능
  - 리터럴 데이터 출력
  - 다양한 연산자와 연산결과 출력 지원
  - 4개의 scope[page, request, session, application) 속성 값 출력
  - JSTL과 연동
- JSP 모델 1 소스 코드의 단점
  - HTML, 스크릿트릿, 표현식(<%= %>) 중첩으로 소스 프로그램 가독성 저하
  - 디버깅 어려움

1. EL의 식

${ }

- EL의 식
  - 표기법 : ${}
    - {}속의 값을 웹브라우저로 출력
    - {}에 리터럴, 변수, 연산식, 객체의 프로퍼티, 리스트 계열의 배열
    - 예:

문자, 숫자, 부울린

구 분	표 기 예	설 명
리터럴	${"Hello"}, ${10}	문자 또는 숫자 리터럴이 출력
	${10+20}	연산식의 값이 출력
변수	${변수}, {변수 +1}	변수 또는 산술식 값이 출력
프로퍼티	${객체.속성}, ${객체["속성"}	특정객체의 프로퍼티 값이 출력
배열	${배열[번호]}, ${배열["번호"]}	배열 번호의 요소 값을 출력

- 모델 1 소스 코드 방식
  - 표현식
    - <%= 변수명 %>
    - <%= 객체명.속성명 %>

3. EL의 연산자

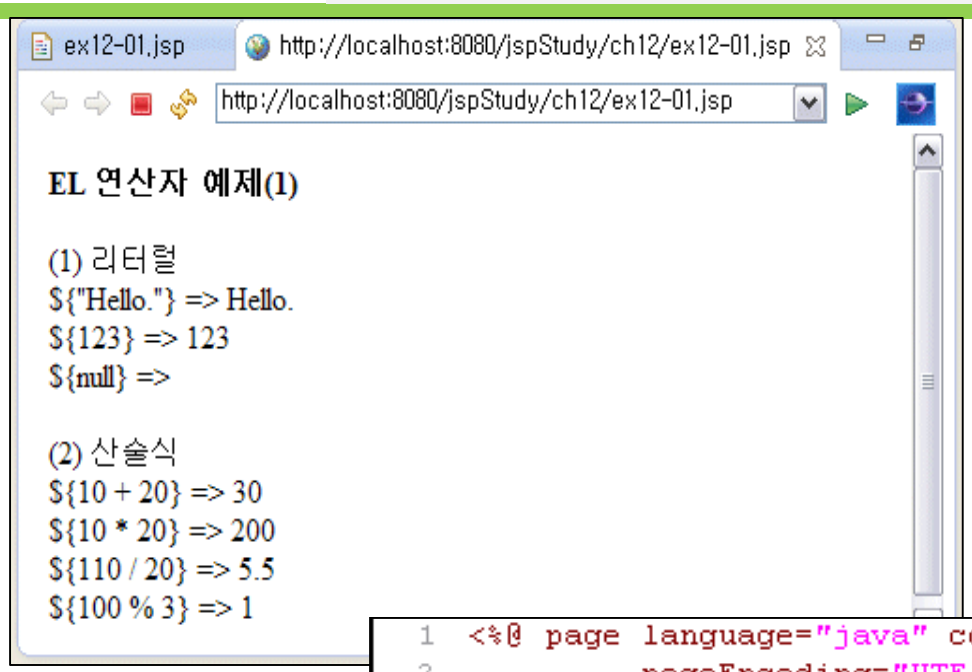
${ }

- EL의 연산자
  - 종류
    - 산술연산자
    - 비교연산자
    - 논리연산자
    - empty 연산자
    - 기타

구분	연산자	예	결과
산술연산자	+	${20 + 10}	30
	–	${20 – 10}	10
	*	${20 * 10}	200
	/ 또는 div	${20/10} 또는 ${20 div 10}	2
	% 또는 mod	${10 % 3} 또는 ${10 mod 3}	1

예제 12.1

EL로 그림 12.1의 출력 결과와 같이 리터럴과 산술식을 이용하여 출력하는 프로그램을 작성해 보시오.




```
1  <%@ page language="java" contentType="text/html; charset=UTF-8"
2           pageEncoding="UTF-8" %>
3  <h4>EL 연산자 예제(1)</h4>
4  ${"(1) 리터럴"}<br>
5  \${"Hello."} => ${"Hello."} <br>
6  \${123} => ${123}    <br>
7  \${null} => ${null} <p>
8
9  ${"(2) 산술식"}<br>
10 \${10 + 20}  => ${10 + 20} <br>
11 \${10 * 20}  => ${10 * 20} <br>
12 \${110 / 20} => ${110 / 20} <br>
13 \${100 % 3}  => ${100 % 3}
```

${}식 앞에 "\"를 붙이면 문자열로 처리

3. EL 연산자

${ }

EL연산자

구분	연산자	예	결과
비교연산자	> 또는 gt	${20 > 10 } 또는 ${20 gt 10}	참
	>= 또는 ge	${20 >= 10 } 또는 ${20 ge 10}	참
	< 또는 lt	${20 < 10 } 또는 ${20 lt 10}	거짓
	<= 또는 le	${20 <= 10 } 또는 ${20 le 10}	거짓
	== 또는 eq	${20 == 10 } 또는 ${20 eq 10}	거짓
	!= 또는 ne	${20 != 10 } 또는 ${20 ne 10}	참

구분		연산자	예	결과
논리연산자	논리곱	&& 또는 and	${true && true} 또는 ${true and true}	참
	논리합	|| 또는 or	${true || false} 또는 ${true or false}	참
	부정	! 또는 not	${!true} 또는 ${not true}	거짓

3. EL 연산자

${ }

EL 연산자

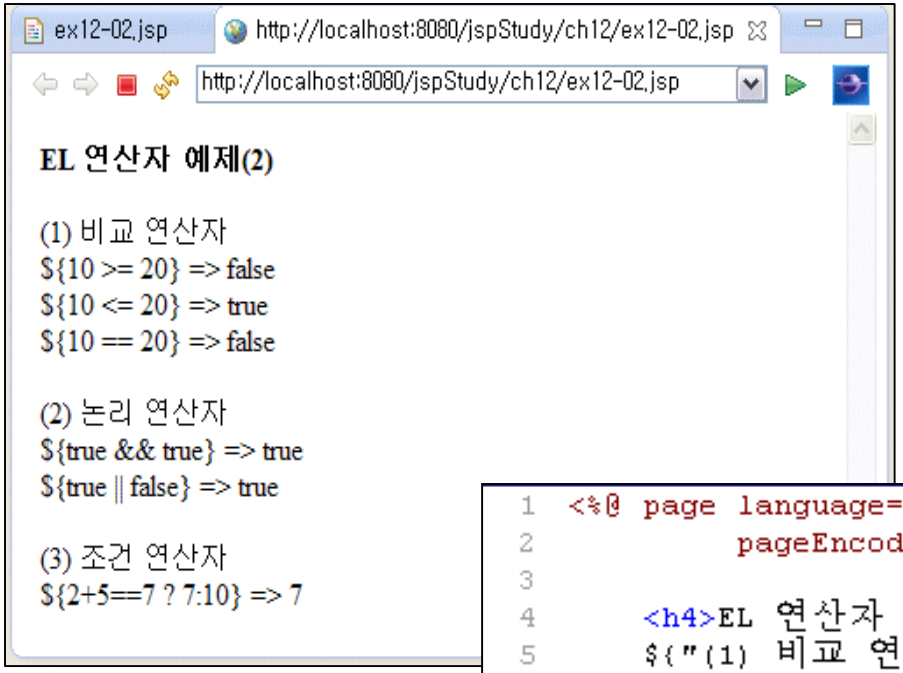
구분	연산자	예	결과
Empty 연산자	empty <값>	${empty a}	a가 null 또는 길이가 0이면 참
		${not empty a}	a가 null 또는 길이가 0이 아니면 참

구분	연산자	예	결과
조건연산자	<수식>?<값1>:<값2>	${2+5==7} ? 7:10	{2+5==7}이 참이 되어 7을 반환

<수식>이 참이면 <값1>을 반환하고, 그렇지 않으면 <값2>를 반환

예제 12.2

예제 12.2와 같이 출력결과를 얻을 수 있도록 비교연산자, 논리 연산자, 조건연산자를 사용하여 프로그램을 작성해 보시오.




```
1  <%@ page language="java" contentType="text/html; charset=UTF-8"
2          pageEncoding="UTF-8" %>
3
4      <h4>EL 연산자 예제(2)</h4>
5      ${"(1) 비교 연산자"} <br>
6      \${10 >= 20} => ${10 >= 20} <br>
7      \${10 <= 20} => ${10 <= 20} <br>
8      \${10 == 20} => ${10 == 20} <p>
9
10     ${"(2) 논리 연산자"}<br>
11     \${true && true} => ${true && true} <br>
12     \${true || false} => ${true || false} <p>
13
14     ${"(3) 조건 연산자"}<br>
15     \${2+5==7 ? 7:10} => ${2+5==7 ? 7:10} <br>
```

3. EL 연산자

${ }

EL 연산자

구분	연산자	예	결과
기타연산자	.	${param.id}	전송된 id 값
	[ ]	${score[0]}	Score 배열의 0번째

스크립트릿(<% %>)의 배열, "java.util.*" 표준 라이브러리로 선언하는 List 객체, Map 객체, 그리고 자바빈의 프로퍼티의 요소를 구분하는 연산자

구 분	표 기 예
배열	${movies[1]}
List 객체	${department[2]}
Map 객체	${map["C0802"]} 또는 ${map.C0802}
자바빈	${LoginBean.id} 또는 ${LoginBean["id"]}

```
5  <jsp:useBean id="test" class="ch08.LoginBean" scope="page" />
6      <jsp:setProperty name="test" property="id" />
7      <jsp:setProperty name="test" property="pw" />
8
9      <h4> 로그인 정보 </h4>
10     아 이 디 : ${test.id}<p>
11     비밀번호 : ${test["pw"]}
```

자바빈 프로퍼티 반환

4. EL의 내장객체

${ }

EL 내장객체 [EL 식에서만 사용]

구분	내장객체	설명
Scope	pageScope	page 영역에 존재하는 객체의 참조
	requestScope	request 영역에 존재하는 객체의 참조
	sessionScope	session 영역에 존재하는 객체의 참조
	applicationScore	application 영역에 존재하는 객체의 참조
요청 파라메타	param	요청 파라메타 값을 단일 값으로 반환
	paramValues	요청 파라메타의 값을 배열로 반환
헤더 값	header	요청 헤더명의 정보를 단일 값으로 반환
	headerValues	요청 헤더명의 정보를 배열로 반환
쿠키 값	Cookies	쿠키명의 값을 반환
JSP	pageContext	PageContext 객체를 참조할 때
초기 파라메타	initParam	컨텍스트의 초기화 매개변수명의 값을 반한

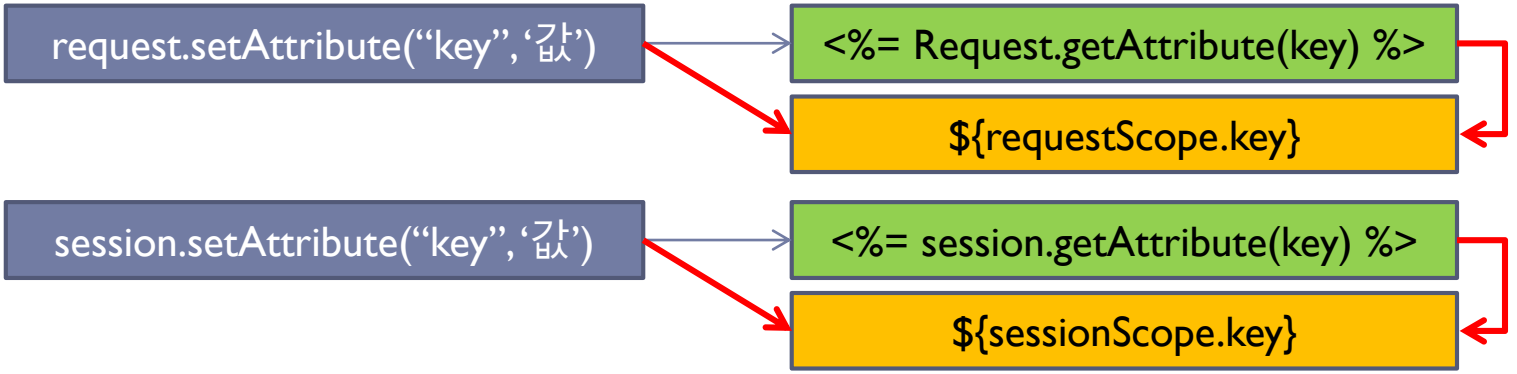
4. EL의 내장객체

${ }

4.1 scope의 표기

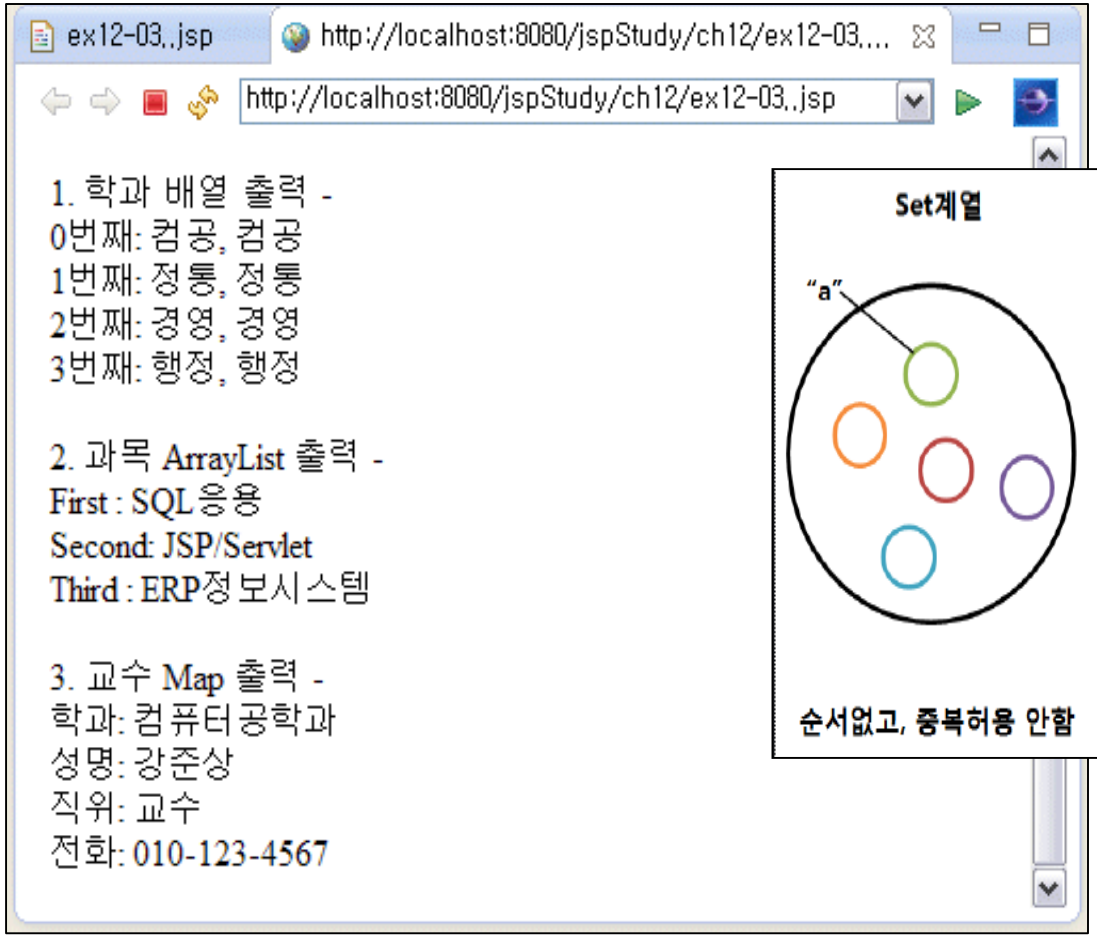
- scope는
  - request.setAttribute("키명", 값)으로 설정한 값을 ${"키명"}으로 얻을 때 각각의 scope 영역에서 정의할 수 있는 내장객체
- 해석순서는 page, request, session, application

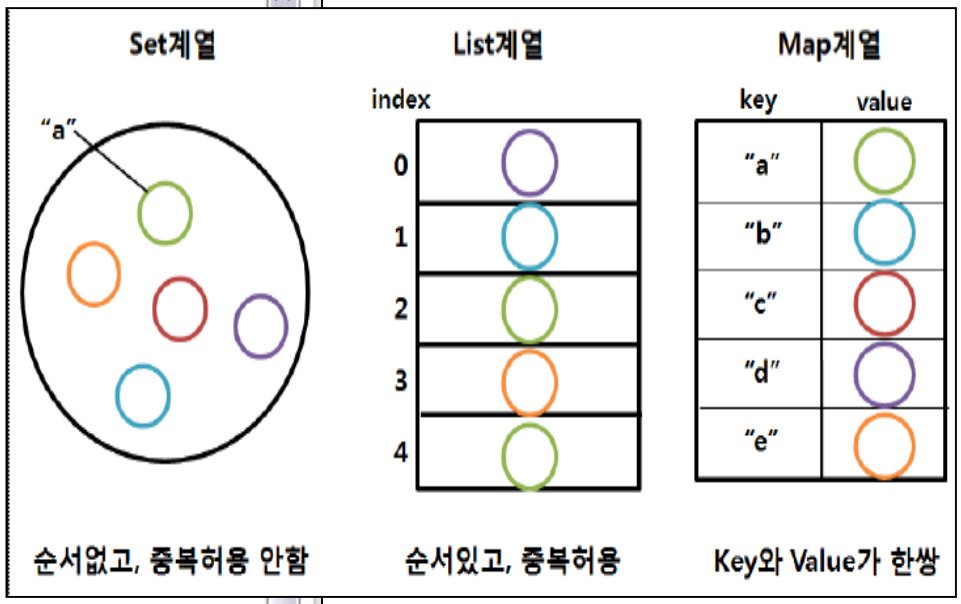
표기법		
	${pageScope.키명}	// page scope
	${requestScope.키명}	// request scope
	${sessionScope.키명}	// session scope
	${applicationScope.키명}	// application scope



예제 12.3

다음과 같이 학과 배열과 과목 ArrayList, 교수정보 Map 객체를 생성하여 그 값들을 출력하는 프로그램을 작성하시오.





콜렉션 객체 선언과 데이터 저장

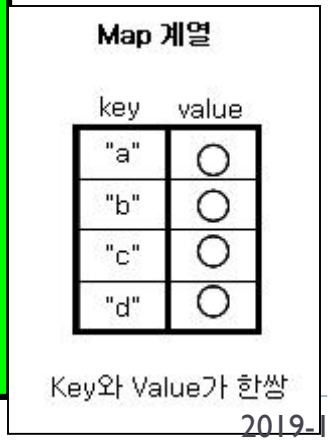
- 배열(Array)
  - 정적 배열
  - 데이터를 순서대로 연속해서 저장
  - 크기 고정
  - 선언 및 데이터 저장
    - String [ ] ar = { “값1” ,” 값2” };

- Vector ArrayList
  - 가변 배열
  - 데이터를 연속해서 순서대로 저장
  - 크기 가변
  - 선언 및 데이터 저장

```
List <String> list= new ArrayList<String>();
        list.add( “값1” );
        list.add( “값2” ;
        list.add( “값3” );
```

- 맵(Map)
  - 키에 의해 원소를 저장하고 접근
  - 임의 객체를 키로 사용
  - 크기 가변
  - 선언 및 데이터 저장

```
Map <String, String> map= new HashMap<String, String>();
        map.put( “키1” ,  “값1” );
        map.put( “키2” ,  “값2” );
        map.put( “키3” ,  “값3” );
```



```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
         pageEncoding="UTF-8"
         import="java.util.ArrayList"
         import="java.util.HashMap" %>
<%
    String[] dept={"컴공","정통","경영","행정"};
    request.setAttribute("dept", dept);

    ArrayList <String> list = new ArrayList();
    list.add("SQL응용");
    list.add("JSP/Servlet");
    list.add("ERP정보시스템");
    request.setAttribute("course", list);

    HashMap <String, String> map = new HashMap();
    map.put("department", "컴퓨터공학과");
    map.put("name","강준상");
    map.put("position", "교수");
    map.put("telephone","010-123-4567");
    request.setAttribute("professor", map);
%>
  1. 학과 배열 출력 -<br>
 0번째: ${dept[0]}, ${requestScope.dept[0]}<br>
 1번째: ${dept[1]}, ${requestScope.dept[1]}<br>
 2번째: ${dept[2]}, ${requestScope.dept[2]}<br>
 3번째: ${dept[3]}, ${requestScope.dept[3]}<p>

  2. 과목 ArrayList 출력 -<br>
 First : ${course[0]}<br>
 Second: ${course[1]}<br>
 Third : ${course[2]}<p>

  3. 교수 Map 출력 -<br>
  학과: ${professor.department}<br>
  성명: ${professor.name}<br>
  직위: ${professor.position}<br>
  전화: ${professor.telephone}
```

4. EL의 내장객체

${ }

4.2 요청 파라메타의 표기

1) param

- 웹 브라우저의 입력 폼에서 전송된 단일 값을 얻을 때 사용하는 내장객체
- JSP 내장객체의 request.getParameter() 메서드와 동일

표기법	${param.필드명} 또는 ${param["필드명"]}

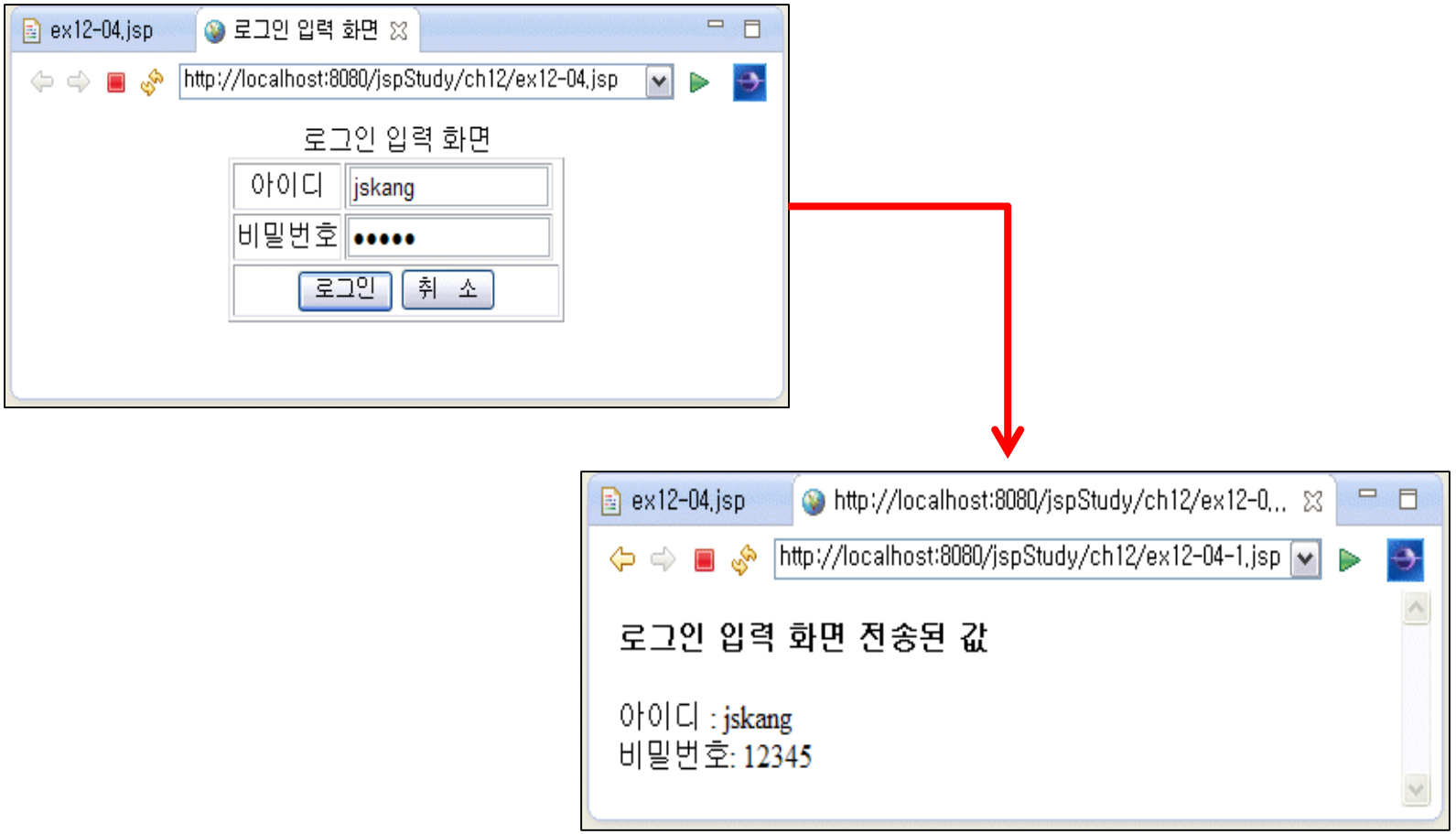
2) paramValues

- 웹 브라우저의 입력 폼에서 check 태그나 select 태그로 전송된 배열 값을 얻을 때 사용하는 내장객체
- 인덱스는 0부터 시작
- JSP 내장객체의 request.getParameterValues() 메서드와 동일

표기법	${paramValues.배열명[인덱스]} 또는 ${paramValues["배열명"][인덱스]}

예제 12.4

로그인 입력 화면에서 전송된 값을 출력하는 프로그램을 작성해 보시오.



```
 9  <body>
10    <center>로그인 입력 화면
11    <form method=post action=ex12-04-1.jsp>
12    <table border="1">
13       <tr>
14           <td>아이디</td>
15           <td><input type="text" name="id" size=15></td>
16       </tr>
17       <tr>
18           <td>비밀번호</td>
19           <td><input type="password" name="pw" size=17></td>
20       </tr>
21       <tr align="center">
22           <td colspan="4">
23              <input type="submit" value="로그인">
24              <input type="reset"  value="취   소">
25           </td>
26       </tr>
27    </table>
28    </form></center>
29  </body>
```

```
1  <%@ page language="java" contentType="text/html; charset=UTF-8"
2           pageEncoding="UTF-8"%>
3     <h4>로그인 입력 화면 전송된 값</h4>
4     아이디 : ${param.id}<br>
5     비밀번호: ${param.pw}<br>
```

4. EL의 내장객체

${ }

4.3 헤더의 표기

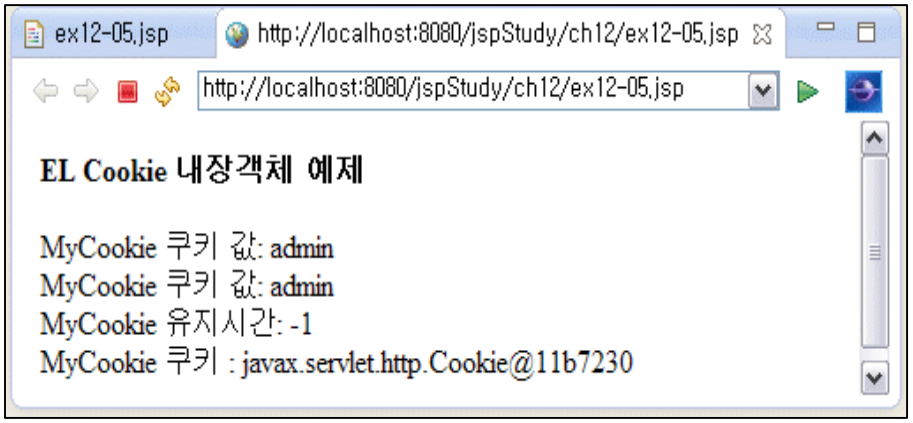
표기법	${header.http헤더명} 또는 ${header["http헤더명"]}
	${headerValue.http헤더명} 또는 ${httpValue["http헤더명"]}
	${header['User-Agent']} // 사용자 웹 브라우저
	${header['host']} // 사용자의 호스트명

4.4 쿠키의 표기

표기법	${cookie.쿠키명} 또는 ${cookie['쿠키명']} // 쿠키
	${cookie.쿠키명.value} 또는 ${cookie['쿠키명'].value} // 쿠키 값
	${cookie.쿠키명.domain} // 쿠키의 도메인명
	${cookie.쿠키명.maxAge} // 쿠키의 설정시간

예제 12.5

"MyCookie"명으로 "admin"을 1시간 동안 저장되는 쿠키를 설정하고, "MyCookie" 값을 출력하는 프로그램을 작성해 보시오. [초기에 쿠키 값이 출력되지 않으면 "새로고침" 버튼을 클릭함]




```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
        pageEncoding="UTF-8" %>
<%
    Cookie cookie = new Cookie("MyCookie", "admin");
    cookie.setMaxAge(60*60);
    response.addCookie(cookie);
%>
    <h4>EL Cookie 내장객체 예제</h4>
    MyCookie 쿠키 값: ${cookie.MyCookie.value} <br>
    MyCookie 쿠키 값: ${cookie['MyCookie']['value']} <br>
    MyCookie 유지시간: ${cookie['MyCookie']['maxAge']} <br>
    MyCookie 쿠키 : ${cookie.MyCookie} <br>
```

4. EL의 내장객체

${ }

4.5 pageContext의 EL 내장객체 표기

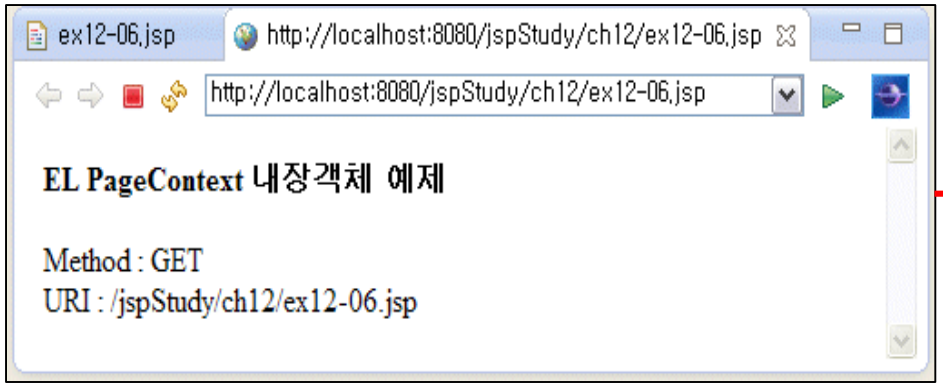
표기법	${pageContext.request.method} // 요청 메서드
	${pageContext.request.requestURI} // 요청 uri

- ${pageContext.request.method}
  - 웹브라우저의 전송 메서드 반환 출력
- ${pageContext.request.requestURI}
  - 웹브라우저의 요청 uri 반환 출력

예제 12.6

요청 메서드, 요청 uri를 출력하는 프로그램을 작성해 보시오.

생략




```
1  <%@ page language="java" contentType="text/html; charset=UTF-8"
2              pageEncoding="UTF-8" %>
3
4     <h4>EL PageContext 내장객체 예제</h4>
5     Method : ${pageContext.request.method} <br>
6     URI    : ${pageContext.request.requestURI}
```

강 의 내 용 요 약 정 리

- EL(Expression Language)
  - 표현식 언어 또는 익스프레션 언어
  - JSTL 1.0에 소개, JSP 2.0/JSTL 1.1에 추가된 스크립트 언어
  - 아파치 탐캣 5.0부터 사용 가능
  - JSP 페이지에서 자바코드 대신 액션태그 엘리먼트의 속성에 값을 지정하는 역할
  - 표현식(<%= %>) 대용 효과

<%= %> 대용 ${ }

- 주요기능
  - 리터럴 데이터 출력
  - 다양한 연산자와 연산결과 출력 지원
  - 4개의 scope[page, request, session, application) 속성 값 출력
  - JSTL과 연동

- JSP 모델 1 소스 코드의 단점
  - HTML, 스크릿트릿, 표현식(<%= %> 중첩으로 소스 프로그램 가독성 저하
  - 디버깅 어려움

강 의 내 용 요 약 정 리

- EL의 연산자

  ${ }

  - 종류
    - 산술연산자
    - 비교연산자
    - 논리연산자
    - empty 연산자
    - 기타

강 의 내 용 요 약 정 리

EL 내장객체 [EL의 식에서만 사용]

${ }

구분	내장객체	설명
Scope	pageScope	**page 영역에 존재하는 객체의 참조**
	requestScope	**request 영역에 존재하는 객체의 참조**
	sessionScope	**session 영역에 존재하는 객체의 참조**
	applicationScore	**application 영역에 존재하는 객체의 참조**
요청 파라메타	param	**요청 파라메타 값을 단일 값으로 반환**
	paramValues	**요청 파라메타의 값을 배열로 반환**
헤더 값	header	**요청 헤더명의 정보를 단일 값으로 반환**
	headerValues	**요청 헤더명의 정보를 배열로 반환**
쿠키 값	Cookies	**쿠키명의 값을 반환**
JSP	pageContext	**PageContext 객체를 참조할 때**
초기 파라메타	initParam	**컨텍스트의 초기화 매개변수명의 값을 반한**

강 의 내 용 요 약 정 리

${ }

요청 파라메타의 표기

1) param

- 웹 브라우저의 입력 폼에서 전송된 단일 값을 얻을 때 사용하는 내장객체
- JSP 내장객체의 request.getParameter() 메서드와 동일

표기법	${param.필드명} 또는 ${param["필드명"]}

2) paramValues

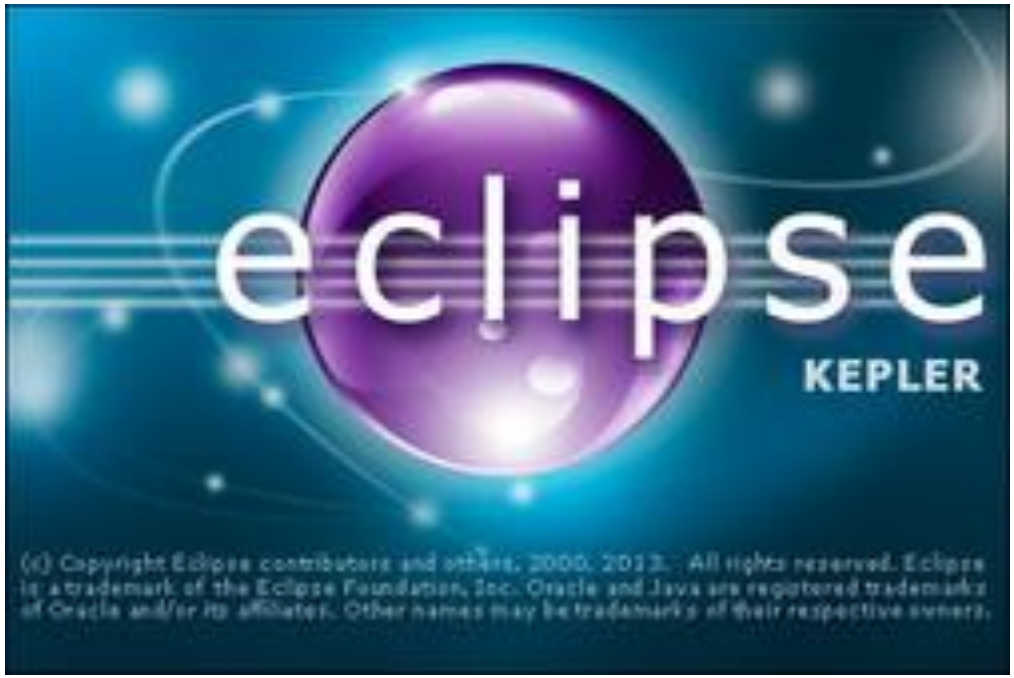
- 웹 브라우저의 입력 폼에서 check 태그나 select 태그로 전송된 배열 값을 얻을 때 사용하는 내장객체
- 인덱스는 0부터 시작
- JSP 내장객체의 request.getParameterValues() 메서드와 동일

표기법	${paramValues.배열명[인덱스]} 또는 ${paramValues["배열명"][인덱스]}


다음

장 주제

13장. JSTL


eclipse
KEPLER
(c) Copyright Eclipse contributors and others. 2000, 2013. All rights reserved. Eclipse is a trademark of the Eclipse Foundation, Inc. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.