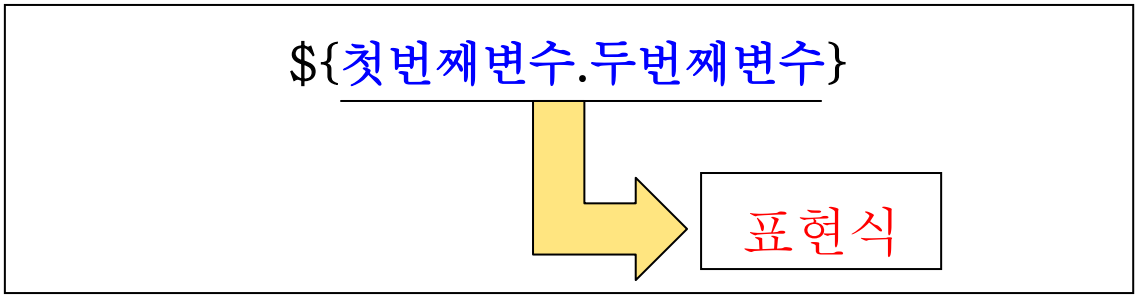
4.EL(Expression Language)

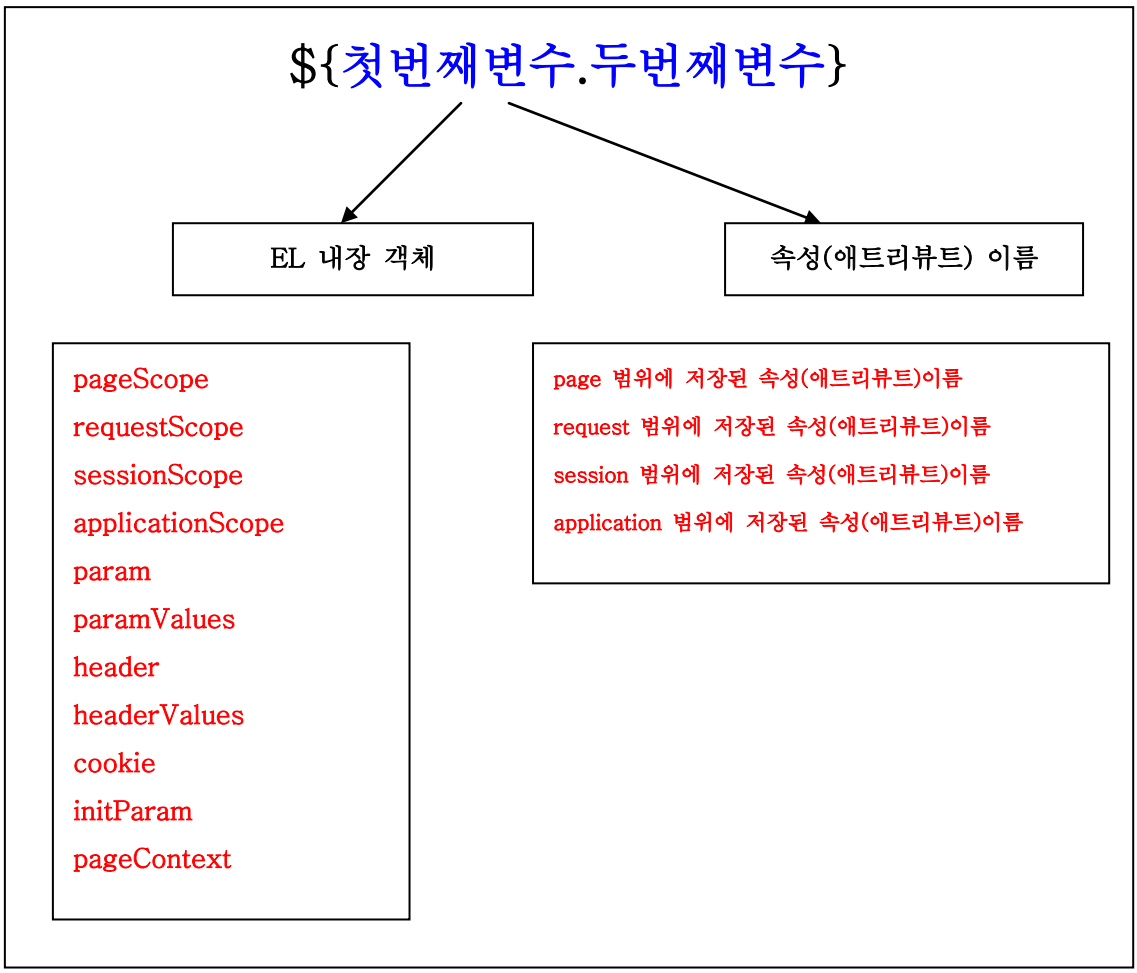
JSP 스펙 2.0부터는 표현식 언어(Expression Language)가 추가 되었다. **EL은 특정 scope 객체에 명시된 애트리뷰트(Attribute)의 property나 property의 property를 좀더 쉽게 표현하고 나온 것이다**. 다음 장에서 배울 JSTL과 EL이 결합된다면 훌륭한 View 페이지를 만들 수 있을 것이다. 이제 본격적으로 EL의 사용법에 대해 알아보도록 하자.

3.1 EL 사용법

EL의 사용방법과 구성은 매우 간단하다.



EL은 $로 시작하고 "{" 시작하여 "}" 끝나게 된다. 중간에 들어갈 "첫번째변수.두번째변수"를 **표현식**이라 하는데 EL을 학습한다는 것은 표현식을 배우게 되는 것이다.



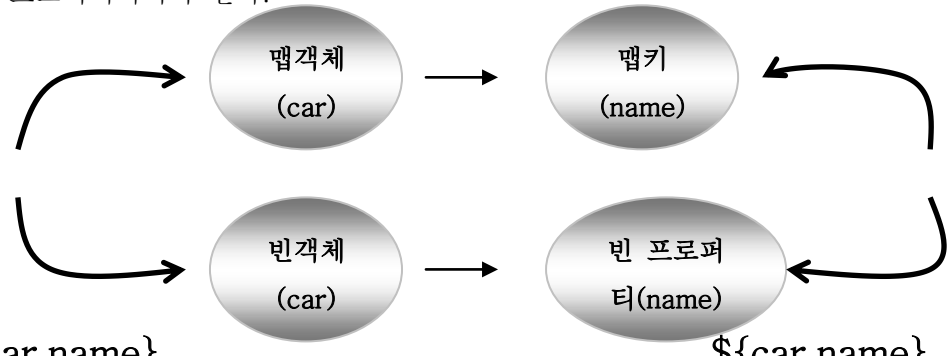
EL에서 표현식중 “첫번째변수”는 **EL 내장 객체나 속성이름**이 올 수 있다. EL 내장 객체는 모두 11개로 이루어 졌으며, 이중 pageContext를 제외한 나머지 모든 객체는 맵 객체이다. 맵 객체란 키와 값으로 구성된 데이터를 저장할 수 있는 컬렉션(Collection)이다. 그리고 속성이름은 JSP에서 이야기하는 4개의 범위(page, request, session, application)에 저장된 이름을 사용한다.

```
Car c = new Car();
c.setName(“BMW”);
request.setAttribute(“car”,c);

당신의 차이름 : ${car.name} 입니다.
```

EL 형식에서 “첫번째변수”에 저장된 객체는 **맵 객체이거나 빈 객체**이어야 한다. 만약, “첫번째변수”에 저장된 객체가 **맵 객체인 경우**는 “두번째변수”는 **맵의 키**가 와야 하고, “첫번째변수”에 저장된 객체가 **자바 빈**인 경우에는 “두번째변수” 는 **빈의 프로퍼티**가 와야 한다. 이 규칙은 “첫번째변수”가 EL 내장 객체이든 속성이름이든 똑 같이 적용된다.

1. 표현식에서 “첫번째변수”에 저장된 객체는 맵 객체이거나 빈 객체이어야 한다.
2. “두번째변수”는 “첫번째변수”에 저장된 객체가 맵 객체이면 맵의 키, 자바 빈이면 빈의 프로퍼티이어야 한다.



이 두 가지를 경우에 대한 JSP 페이지를 만들어 보자.

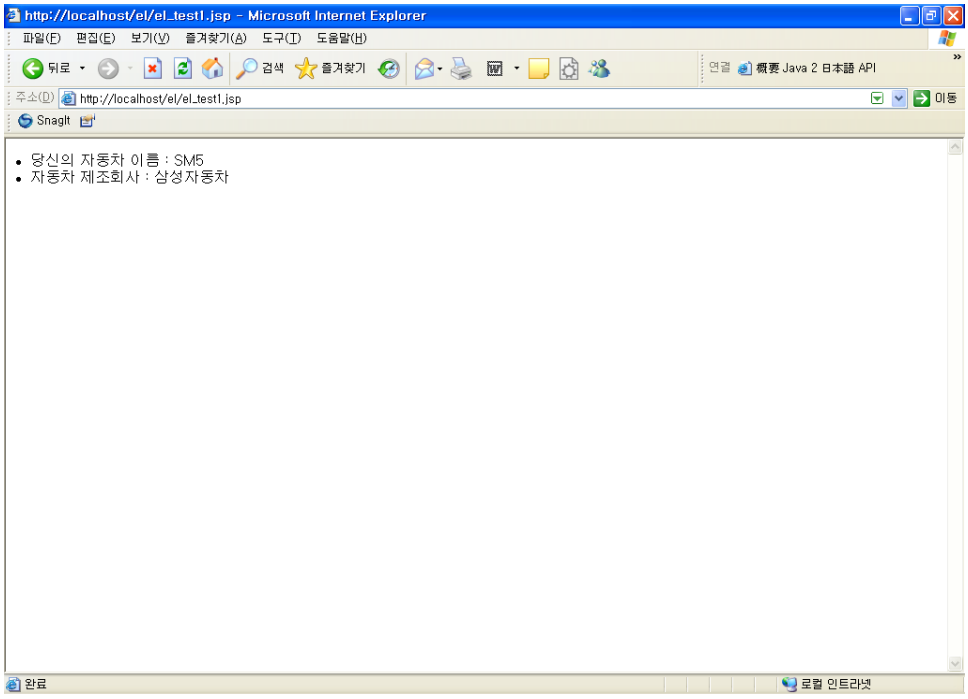
첫 번째 예제는 빈 객체를 저장하는 경우이다.

<파일 Car.java>

```
01. package el;
02. public class Car {
03.             private String name;
04.             private String company;
05.             private int doorCount;
06.             public String getCompany() {
07.                         return company;
08.             }
09.             public void setCompany(String company) {
10.                         this.company = company;
11.             }
12.             public int getDoorCount() {
13.                         return doorCount;
14.             }
15.             public void setDoorCount(int doorCount) {
16.                         this.doorCount = doorCount;
17.             }
18.             public String getName() {
19.                         return name;
20.             }
21.             public void setName(String name) {
22.                         this.name = name;
23.             }
24. }
```

<파일 el_test1.jsp>

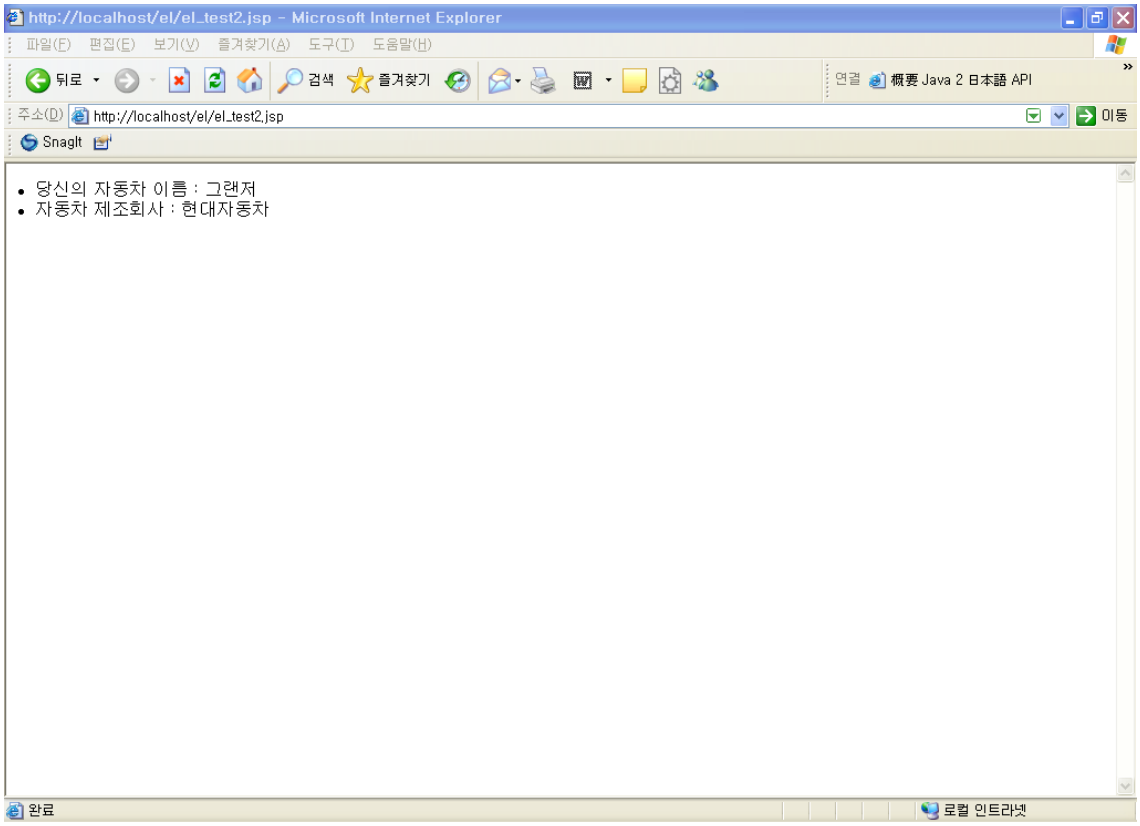
```
01. <%@ page pageEncoding="euc-kr" %>
02. <%@ page import="el.*" %>
03. <%@ page import="java.util.*" %>
04. <%
05.             Car c = new Car();
06.             c.setName("SM5");
07.             c.setCompany("삼성자동차");
08.
09.             request.setAttribute("car",c);
10. %>
11.
12. <li> 당신의 자동차 이름 : ${car.name}<br>
13. <li> 자동차 제조회사 : ${car.company}<br>
```



두 번째 예제는 맵 객체를 저장하는 경우이다.

<파일 el_test2.jsp >

```
01. <%@ page pageEncoding="euc-kr" %>
02. <%@ page import="book.ch10.*" %>
03. <%@ page import="java.util.*" %>
04. <%
05.         Car c = new Car();
06.         c.setName("그랜져");
07.         c.setCompany("현대자동차");
08.         HashMap hm = new HashMap();
09.         hm.put("car",c);
10.         request.setAttribute("hm",hm);
11. %>
12.
13. <li> 당신의 자동차 이름 : ${hm.car.name}<br>
14. <li> 자동차 제조회사 : ${hm.car.company}<br>
```



우리는 두 가지 예제에서 “첫번째변수”에 저장된 객체가 맵 객체이거나 자바 빈일 경우를 살펴 봤다. 하지만 “첫번째변수”에 저장된 객체가 배열이거나, Collection의 List 객체일 경우는 위의 방법으로는 불가능하다. 이 두 가지 경우를 위해 [] 연산자가 나오게 된 것이다. 다음 절에서 [] 연산자의 사용법에서 구체적으로 살펴 보자.

3.1 [] 연산자 사용법

EL 표현식에서 “첫번째변수”에 저장된 객체가 맵 객체이거나 자바 빈일 경우는 일반적으로 dot(.) 연산자를 이용한다. 하지만 [] 연산자를 이용하여도 동일한 결과를 얻을 수 있다.

${car.name} == ${car[“name”]}

그렇다면 [] 연산자를 언제 사용하는 것일까?

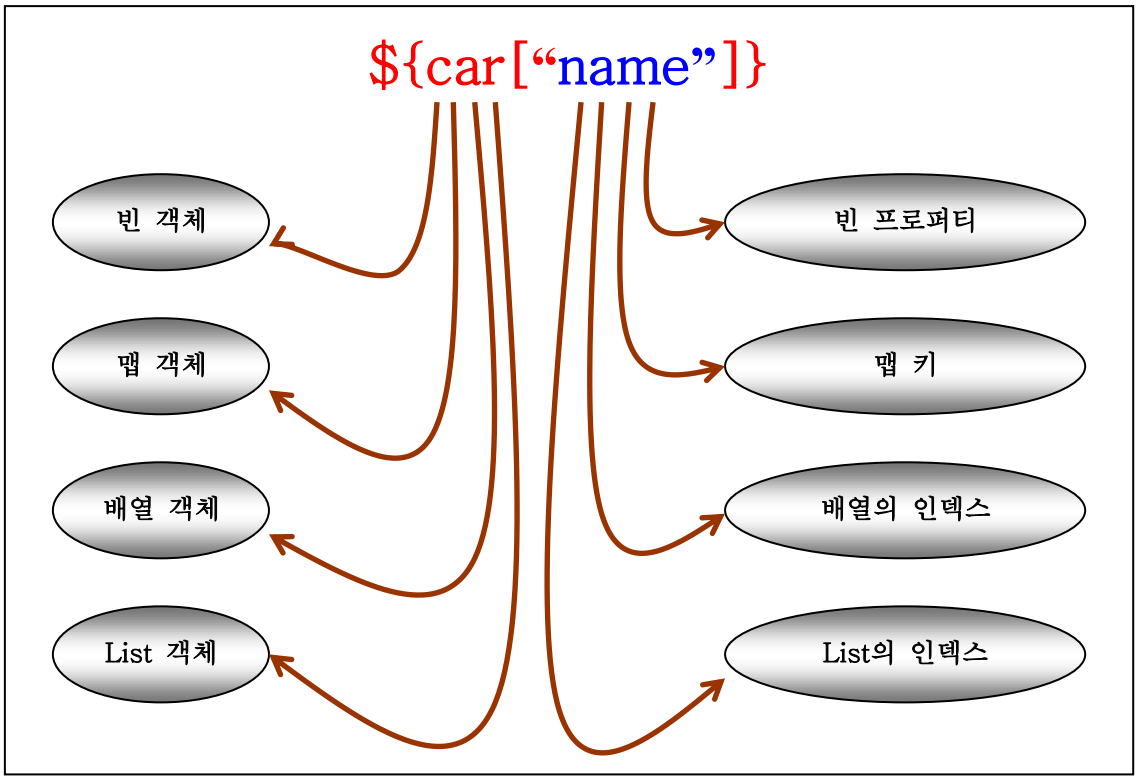
그 해답은 EL 표현식에서 “첫번째변수”에 저장된 객체가 Collection의 List 객체이거나 배열 객체인 경우는 dot(.) 연산자로는 표현할 수 가 없다. 다시 말해서 **“첫번째변수”에 저장된 객체가 List 객체이거나 배열 객체인 경우 “두번째변수”는 List 객체의 인덱스이거나 배열**

의 인덱스를 표현해야 한다. 이럴 경우 dot 연산자를 이용하면 에러가 발생된다.

${car.1} ➔ Error ${car[“1”]} , ${car[1]} ➔ OK

[] 연산자를 정리하면 아래와 같다.

첫 번째, [] 연산자는 “첫번째변수”에 저장된 객체가 자바 빈, 맵 객체, List 객체, 배열 객체인 경우 모두 사용할 수 있으며, 특히 List 객체나 배열 객체인 경우는 dot(.) 연산자로 표현이 불가능하므로 반드시 [] 연산자를 이용해야 한다.



위의 경우는 “두번째 변수”에 빈 프로퍼티나 맵 키이면 **문자열을 사용**해야 하며, 배열의 인

덱스나 List의 인덱스이면 **문자열을 이나 숫자를 사용**할 수 있다. 이 때 문자열은 숫자로 변환된다.

${car[“1”]} == ${car[1]}

두 번째, [] 연산자는 “두번째변수”를 사용할 때 자바 이름 규칙을 이용해야 하는데 **자바 이름 규칙을 벗어나는 경우**에도 사용할 수 있다.

${car[“name.company”]}

예제를 통해 [] 연산자를 익히도록 하자. 이번 예제는 “첫번째변수”에 빈 객체, 맵 객체, List 객체, 배열 객체를 이용한 예제이다.

<파일 el_test3.jsp>

```
01. <%@ page pageEncoding="euc-kr" %>
02. <%@ page import="el.*" %>
03. <%@ page import="java.util.*" %>
04.
05. <%
06.         Car c = new Car();
07.         c.setName("젠트라");
08.         request.setAttribute("car",c);
09. %>
10. <li>이름 : ${car.name}
11. <li>이름 : ${car["name"]}
12. <hr>
13. <%
14.         String[] fruit = {"banana","melon","strawberry","apple"};
15.         request.setAttribute("fruitList" , fruit);
16. %>
17.
18. <li>과일 : ${fruitList[0]}
19. <li>과일 : ${fruitList[1]}
20. <li>과일 : ${fruitList["2"]}
21. <li>과일 : ${fruitList["3"]}
```

```
35. <%
36.             Map people = new HashMap();
37.             people.put("first","노무형");
38.             people.put("second","이멍박");
39.             people.put("center.chief","박군해");
40.             request.setAttribute("peopleList",people);
41.
42. %>
43. <li>사람 : ${peopleList.first}
44. <li>사람 : ${peopleList["second"]}
45. <li>사람 : ${peopleList["center.chief"]}
46. <hr>
```

세 번째, EL에서 표현식의 “첫번째변수”에 저장된 객체가 자바 빈이거나, 맵 객체인 경우 [] 연산자 안에 문자열이 아닌 경우는 [] 연산자 안에 있는 이름으로 속성 찾아 반환한다.

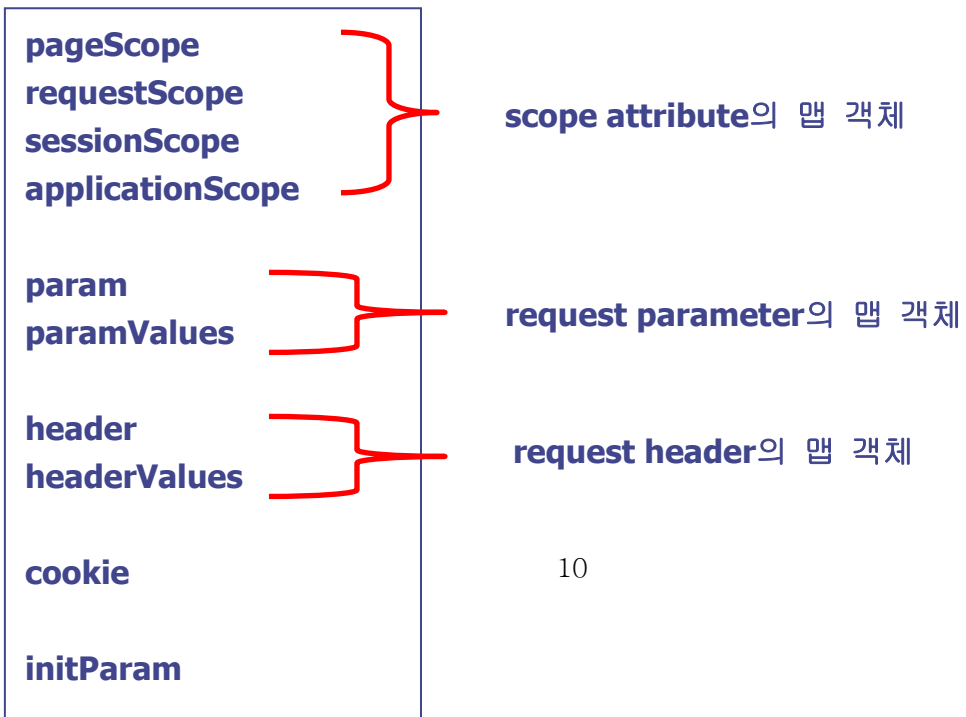
```
01. <%
02.         Map map = new HashMap();
03.         map.put("banana","good");
04.         request.setAttribute("fruit",map);
05.         request.setAttribute("first","banana");
06. %>
07. 바나나 : ${fruit[first]}
```

${fruit[first]} 여기서 [] 연산자 안에 있는 first의 의미는 "first" 이름으로 속성을 찾게 된다. 05 라인에서 "first" 이름으로 request 속성에 "banana"를 저장했기 때문에 "banana"를 반환하게 된다. 따라서 ${fruit [first]}는 ${fruit ["banana"]} 로 의미가 바뀌게 되고 fruit는 맵 객체이므로 맵의 키가 "banana"를 찾아 "good"를 반환하게 된다.

${fruit[first]} ➔ ${fruit["banana"]}➔ "good"

3.1 EL implicit Object

EL은 11가지 내장 객체를 가지고 있다. EL 내장 객체는 JSP 내장 객체와 다르다. JSP 내장 객체는 스크립트릿 이나 표현식에서만 사용하지만 EL 내장객체는 스크립트릿 이나 표현식에서 사용할 수 없고 EL의 표현식에서만 가능하다. EL 내장객체는 pageContext 객체를 제외한 나머지 모든 객체가 **맵 객체**이다.



cookie의 맵 객체

context init parameter의 맵 객체

유일하게 **map** 객체가 아니다. 이 객체는 빈 객체이며 실제 **pageContext**의 **reference** 이다.따라서 **pageContext**의 모든 메서드를 **API**에서 찾아볼 수 있다.

11가지 EL 내장 객체의 사용법에 대해 알아보자.

1. EL에서 request parameter 가져오기

request parameter가 1개 인 경우

표현식	EL
<%=request.getParameter('name")%>	${param.name} or ${param["name"]}

표현식	EL

request parameter가 2개 이상인 경우

<%=request.getParameterValues(‘name”)%>	${paramValues.name[0] } or ${paramValues[“name”][0]}

Request parameter가 2개 이상인 경우는 EL 표현 방법은 ${paramValues.name}를 사용한다. 하지만 이 경우 결과값이 String[] 이기 때문에 실제 화면에 출력하면 우리가 원하는 결과값을 얻을 수 없다. 따라서 EL를 표현할 때는 배열의 인덱스를 명시해야 한다.

예) ${paramValues.name[0]}

2. EL에서 request header 정보 가져오기

Request Header의 1개의 정보를 가져오는 경우

표현식	EL
<%=request.getHeader(“host”)%>	${header.host} or ${header[“host”]}

Request Header의 2개이상 정보를 가져오는 경우

표현식	EL
<%=request.getHeaders(‘accept”)%>	${headerValues.accept[0]} or ${headerValues[“accept”][0]}

3. EL에서 request method 정보 가져오기

request의 method 정보를 가져오는 경우

표현식	EL
<%=request.getMethod()%>	${pageContext.request.method} or ${ pageContext.request.[“method”] }

pageContext는 EL 내장객체 중에 유일하게 맵이 아닌 객체이다. 실제 pageContext 객체에 대한 reference 이다. 그리고 이 객체는 빈 객체이며, API 문서에서 PageContext 접근

자(getter)가 어떤 것이 있는지 살펴보고 프로퍼티를 정하면 된다.
PageContext에 있는 접근자 : getErrorData(), getPage(), getRequest(), getResponse(), getServletConfig(), getServletContext(), getSession()
JspContext로부터 상속받은 접근자 : getAttribute(), getOut() 등이 있다.

```
01. <%@ page pageEncoding="euc-kr" %>
02. <html>
03.         <body bgcolor="#FFFFFF">
04.                 <form action="el_test5_1.jsp" method="post">
05.                 이름 : <input type="text" name="name"><br>
06.                 아이디 : <input type="text" name="id"><br>
07.                 <p>
08.                 음식1 : <input type="text" name="food"><br>
09.                 음식2 : <input type="text" name="food"><br>
10.                 <input type="submit">
11.                 </form>
12.         </body>
13. </html>
```

```
01. <%@ page pageEncoding="euc-kr" %>
02. <%request.setCharacterEncoding("euc-kr");%>
03. <li> 이름  : ${param.name}
04. <li> 아이디  : ${param.id}
05. <li> 음식1 : ${param.food}
06. <li> 음식1 : ${paramValues.food[0]}
07. <li> 음식2 : ${paramValues.food[1]}
08. <li> 이름  : ${paramValues.name[0]}
09. <li> 호스트 : ${header.cookie}
10. <li> Accept : ${headerValues.accept[0]}
11. <li> HTTP 메서드 : ${pageContext.request.method}
```

4. EL에서 Cookie 정보 가져오기

Cookie 정보를 가져오는 경우

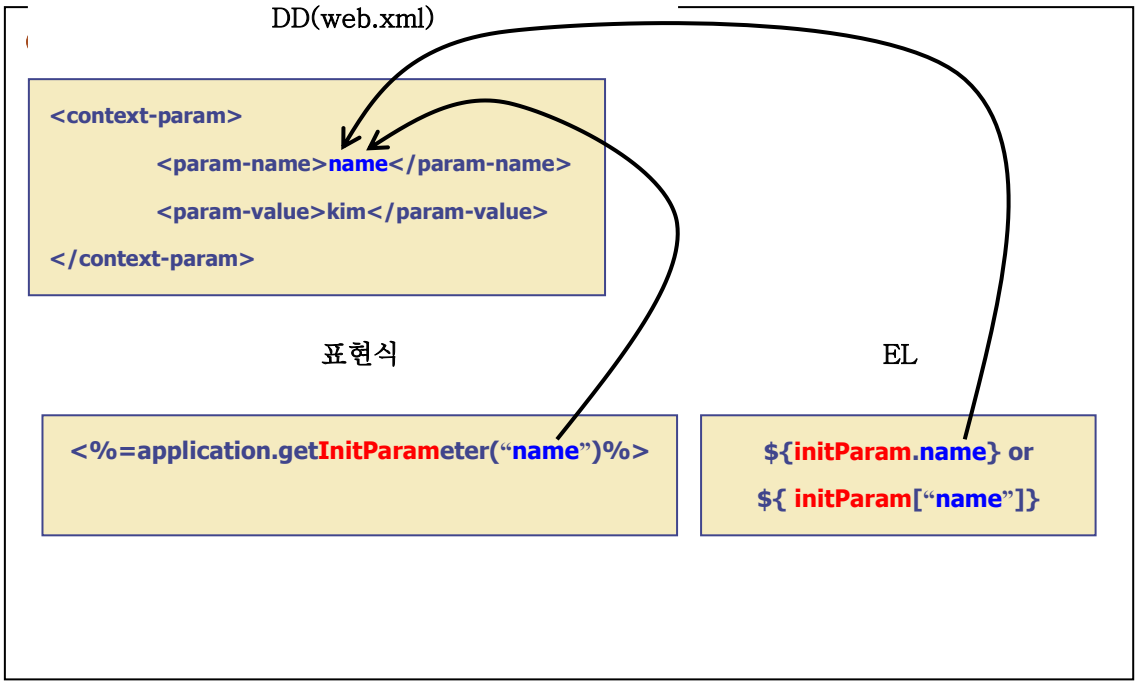
스크립트릿	EL

<%
Cookie[] cookies = request.getCookies();
for(int i=0;i<cookies.length;i++){
if(cookies[i].getName().equals("userName")){
out.println(cookies[i].getValue());
}
}
%>
	${cookie.userName.value}

```
01. <%
02.         Cookie cookie = new Cookie("userName","kim");
03.         cookie.setMaxAge(60);
04.         response.addCookie(cookie);
05.         response.sendRedirect("el_test6_1.jsp");
06. %>
```

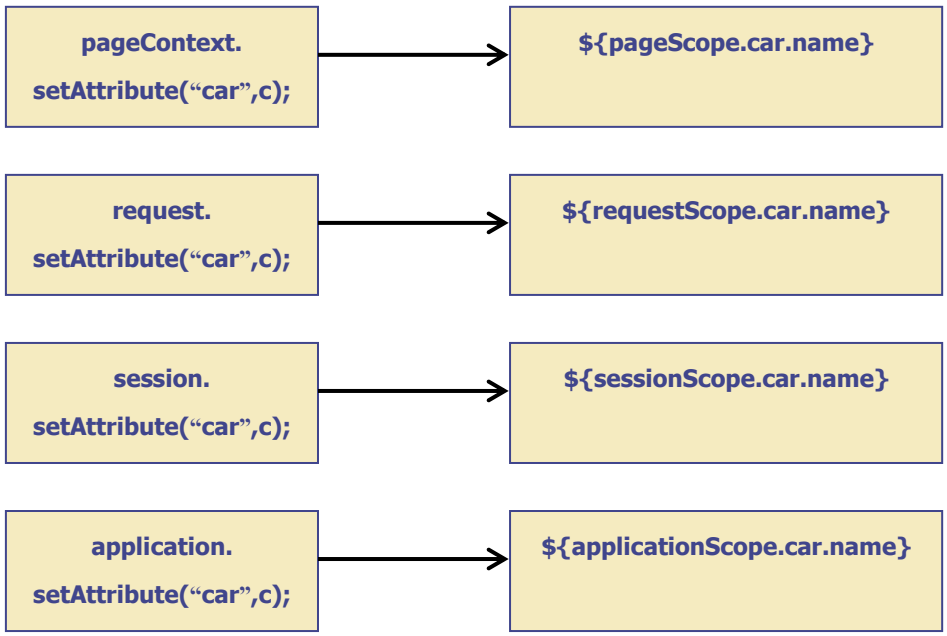
```
01. <%@ page pageEncoding="euc-kr" %>
02. <html>
03.         <body>
04.         <%
05.                 Cookie[] cookies = request.getCookies();
06.                 for(int i=0; i< cookies.length ; i++){
07.                     if(cookies[i].getName().equals("userName")){
08.                         out.println("이름 :
    "+ cookies[i].getValue()+ "<br>");
09.                     }
10.                 }
11.         %>
12.         이름 : ${cookie.userName.value} <br>
```

5. EL에서 Context Init Parameter 가져오기



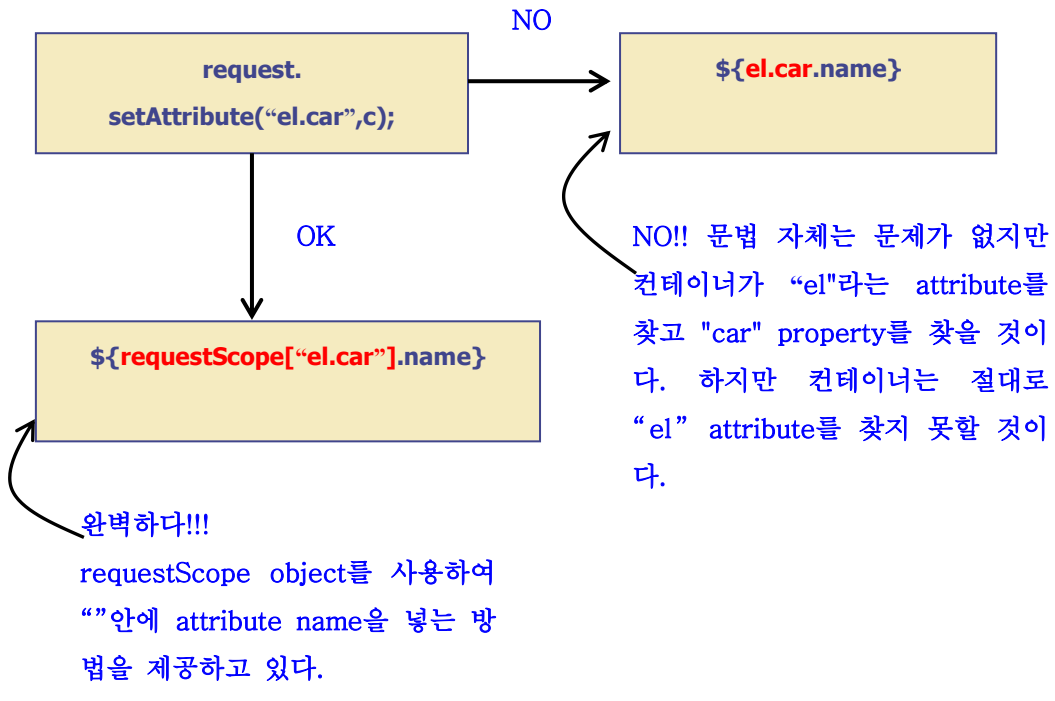
6. EL에서 4가지 scope에 저장되어 있는 attribute의 property 가져오기

4가지 scope에 저장되어 있는 attribute의 property 가져오는 경우



7. Attribute name이 자바 변수 이름 규칙을 따르지 않을 경우

Attribute name이 자바 변수 이름 규칙을 따르지 않는 경우



```
01. <%@ page pageEncoding="euc-kr" %>
02. <%
03.             Float f1 = new Float(10.11);
04.             Float f2 = new Float(20.12);
05.             Float f3 = new Float(30.13);
06.             Float f4 = new Float(40.14);
07.             Float f5 = new Float(50.15);
08.             pageContext.setAttribute("first",f1);
09.             request.setAttribute("first",f2);
10.             session.setAttribute("first",f3);
11.             application.setAttribute("first",f4);
12.             application.setAttribute("first.fifth",f5);
13. %>
14. <li> pageContext : ${pageScope.first}
15. <li> request : ${requestScope.first}
16. <li> session : ${sessionScope.first}
17. <li> application : ${applicationScope.first}
18. <li> fifth : ${applicationScope["first.fifth"]}
```

3.4 EL Function

EL Function이란 EL에서는 사용자가 메서드를 만들어 사용하기 위한 기능이다. 사용자가 메서드를 만들기 위해서는 아래의 같이 4가지 단계를 거쳐야 한다.

1. public static 메서드로 java class를 작성한다.

메서드는 반드시 public static 메서드로 작성해야 하며, 작성후에는 /WEB-INF/classes 디렉토리에 적당히 패키지로 컴파일 해야 한다.

2. Tag Library Descriptor(TLD)파일을 작성한다.

TLD 파일은 메서드를 정의해 놓은 자바 클래스와 메서드를 호출하는 JSP 파일을 연결해 주는 교량 역할을 한다. 이 파일은 /WEB-INF 디렉토리에 tld 확장자를 가지고 이름을 정하여 저장하면 된다. tld 파일의 위치는 /WEB-INF 디렉토리에 있든 /WEB-INF의 서브 디렉토리에 있든 상관없다.

3. JSP 페이지에 taglib 디렉티브를 넣는다.

taglib directive는 Container에게 tld에 정의된 클래스의 메서드를 호출할 수 있도록 prefix와 name를 정할 수 있다.

4. EL를 사용해서 메서드를 호출한다.

${prefix:functionName()}를 사용하여 메서드를 호출할 수 있다.

EL Function을 만드는 4단계를 알아봤다. 이제 예제를 만들어 보자. 총 3개의 파일을 만들어야 한다. java class 파일, tld 파일, jsp 파일이다. 첫 번째로 java class 파일부터 생성해 보자.

<파일경로> ELFunction.java</파일경로>

```
01. package book.ch10;
02. public class ELFunction {
03.         public static String rotto(){
04.             int[] array = new int[6];
05.             first: for(int i=0;;){
06.                 if(i==6) break;
07.                 int random = 1 + (int)(Math.random()*45);
08.                 if(i==0){
09.                     array[i] = random;
10.                     i++;
11.                 }else{
12.                     for(int j=0;j<i;j++){
13.                         if(random == array[j])
14.                             continue first;
15.                     }
16.                     array[i] = random;
17.                     i++;
```

두 번째로 tld 파일을 생성해 보자.

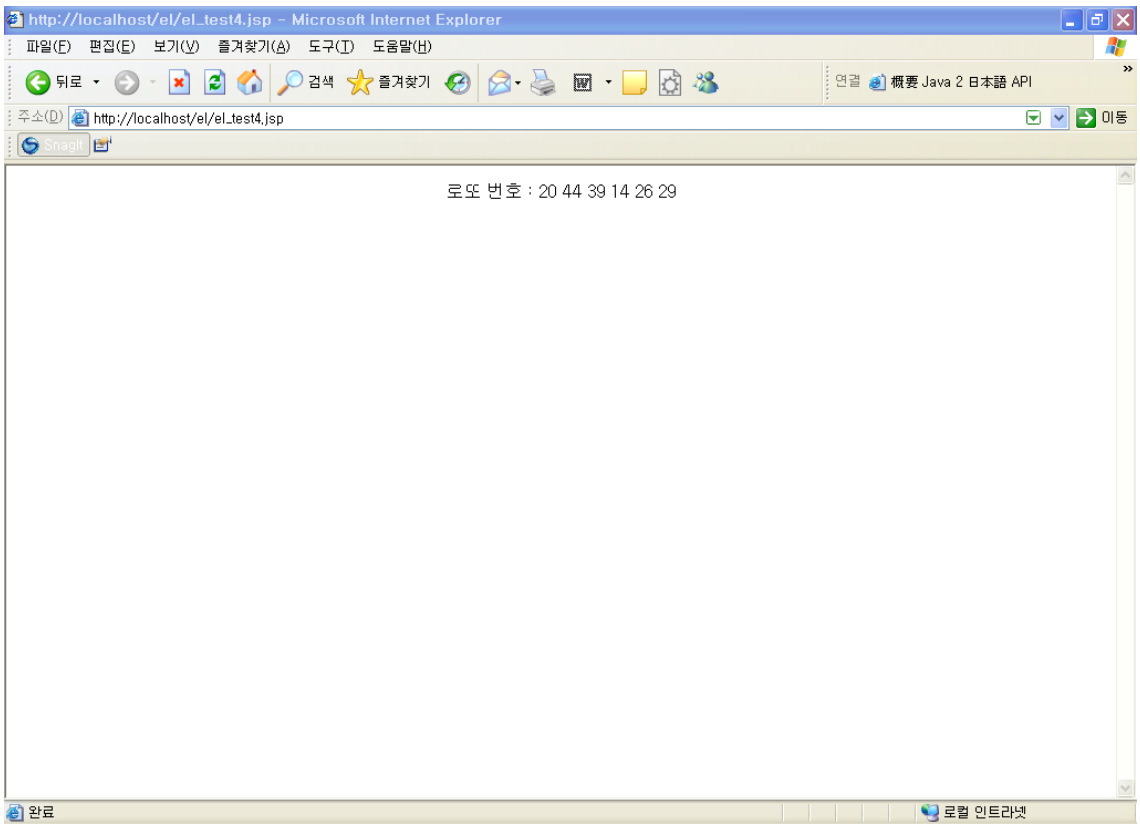
<파일경로>WEB-INF₩rotto.tld</파일경로>

```
01. <?xml version="1.0" encoding="UTF-8" ?>
02. <taglib xmlns="http://java.sun.com/xml/ns/j2ee"
03.       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
04.       xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-jsptaglibrary_2_0.xsd"
05.       version="2.0">
06.            <tlib-version>1.2</tlib-version>
07.            <uri>RottoFunction</uri>
08.            <function>
09.                  <name>goRotto</name>
10.                  <function-class>book.ch10.ELFunction</function-class>
11.                  <function-signature>
12.                  java.lang.String rotto()
13.                  </function-signature>
14.            </function>
15. </taglib>
```

세 번째로 jsp 파일을 만들어 보자.

<파일경로> el_test4.jsp</파일경로>

```
01. <%@ page pageEncoding="euc-kr" %>
02. <%@ taglib prefix="mine" uri="RottoFunction" %>
03. <html>
04.         <body>
05.                 <center>로또 번호 : ${mine:goRotto()}</center>
06.         </body>
07. </html>
```



3.5 EL Operator

EL은 다른 언어와 마찬가지로 연산자들을 지원한다. **EL의 연산자로는 산술 연산자, 논리 연산자, 관계 연산자, empty 연산자**가 있다. 이런 EL 연산자들은 자체적으로는 별로 의미가 없지만 JSTL(JSP Standard Tag Library)와 함께 사용될 때 의미가 있을 것이다. 앞으로 설명이나 결과화면에 의문을 가지 고 후에 배울 JSTL을 기다리기 바란다.

논리 연산자(3)
AND : && 또는 and
OR : || 또는 or
NOT : ! 또는 not

산술 연산자(5)
더하기 : +
빼기 : -
곱하기 : *
나누기 : / 또는 div
나머지 : % 또는 mod

EL에서는 0으로 나눌 수 없다
0으로 나누게 되면 Exception이 발생된다.

관계 연산자(3)

동등 : == 또는 eq (equals)

다름 : != 또는 ne (not equals)

작다 : < 또는 lt (less than)

크다 : > 또는 gt (greater than)

작거나 같다 : <= 또는 le (less than or equals to)

크거나 같다 : >= 또는 ge(greater than or equals to)

empty 연산자

empty <값>

<값>이 null 이면 true를 반환한다.

<값>이 빈 문자열(“”)이면 true를 반환한다.

<값>이 빈 Map이면 true를 반환한다.

<값>이 빈 Collection이면 true를 반환한다.

EL에서는 16가지 단어를 예약어로 지정했기 때문에 식별자로 사용할 수 없다. 연산자에서 사용한 12가지와 추가적으로 4가지를 더 제공한다.

div	mod	and	or	not
eq	ne	lt	gt	le
ge	empty	true	false	null
instanceof				

파랑색은 EL 연산자이며, 빨강색은 추가 단어이다. 이렇게 외면 좀더 쉽지 않겠는가!!

JSP에서도 연산자의 우선순위가 있듯이 EL에서도 연산자의 우선순위가 있다. 즉 우선순위

가 높은 순서대로 처리된다. EL의 연산자 우선순위를 보는 방법은 위에서 아래로, 우선순위가 같을 경우는 좌에서 우로 우선순위가 높다.

```
연산자 우선순위
[]
()
-  not  !  empty
/  div  %  mod
+  -
<  >  <=  >=  lt  gt  le  ge
==  !=  eq  ne
&&  and
```

```
01. <%@ page pageEncoding="euc-kr" %>
02. <%
03.             request.setAttribute("first",new Integer(10));
04.             request.setAttribute("second",new Integer(20));
05.             request.setAttribute("true",new Boolean("true"));
06.             request.setAttribute("false",new Boolean("false"));
07.             request.setAttribute("emp",new int[0]);
08. %>
09. <li> 10 +  20 : ${first +  second}
10. <li> 10 / 20 : ${first div second}
11. <li> 10 % 20 : ${first mod second}
12. <li> true && false : ${true and false}
13. <li> true || false : ${true or false}
14. <li> 10 > 20 : ${first gt second}
15. <li> 10 <= 20 : ${first le second}
16. <li> emp : ${empty(emp)}
```