# 자바 웹프로그램 개발 입문 과정

33일차 (보강)

# 오늘의 키워드

- EL

# EL

- EL (Expression Language)
  - JSP 2.0 부터 지원되는 스크립트 언어
  - JSP 스크립팅 요소보다 함축적인 코드 사용
    간결하고 편리하게 특정 데이터 값 출력
  - EL 이 없다고 JSP 프로그래밍이 불가능한 건 아니다
    - 특정식을 통해서 데이터 값을 출력해주는게 목적이니깐

```
${ 식 }
```

# EL

- 표현식 VS EL

StartWithoutEL.jsp

```
<%@ page contentType="text/html; charset=utf-8"%>
<html>
    <head>
        <title>EL 없이 데이터 출력</title>
    </head>
    <body>
        전달받은 데이터는 [<%=request.getParameter("data")%>]
    </body>
</html>
```

```
<%@ page contentType="text/html; charset=utf-8"%>
<html>
    <head>
        <title>EL 없이 데이터 출력</title>
    </head>
    <body>
        전달받은 데이터는 [${param.data}]
    </body>
</html>
```

StartWithEL.jsp

# EL

- EL 식의 데이터 이름
  - setAttribute
  - getAttribute
  - removeAttribute
    - page, request, session, application

```
<- 사용범위 작음          사용범위 넓음 ->
page -> request -> session -> application
```

# EL

- EL 식의 데이터 이름 ELNameScope.jsp

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

<%
    request.setAttribute("dataname", "request Attribute");
    application.setAttribute("dataname", "application Attribute");
%>

<html>
    <head>
        <title>EL 데이터 이름 사용</title>
    </head>
    <body>
        <h3>request 내장객체의 dataname 속성값 : </h3>
        [<%=request.getAttribute("dataname") %>] <hr/>
        <h3>application 내장객체의 dataname 속성값 : </h3>
        [<%=application.getAttribute("dataname") %>] <hr/>
        <h3>EL 에서 출력되는 dataname 의 값 </h3>
        [${dataname}]
    </body>
</html>
```

# EL

- EL의 내장객체

| 분류 | 내장 객체 | 설명 | 타입 |
|---|---|---|---|
| JSP 내장객체 | pageScope | JSP의 **page** 내장객체와 동일 | Map |
| | requestScope | JSP의 **request** 내장객체와 동일 | Map |
| | sessionScope | JSP의 **session** 내장객체와 동일 | Map |
| | applicationScope | JSP의 **application** 내장객체와 동일 | Map |
| | pageContext | JSP의 **pageContext** 내장객체와 동일 | PageContext |
| 파라미터 | param | 요청 파라미터 데이터 접근시 사용<br>request.getParameter(paramName) 과 동일 | Map |
| | paramValues | 요청 파라미터 데이터들을 배열로 접근시 사용<br>request.getParameterValues(paramName) 과 동일 | Map |
| | initParam | 컨텍스트 초기화 파라미터 접근시 사용<br>application.getInitParameter(paramName)과 동일 | Map |
| 쿠키 | cookie | 쿠키 객체 참조시 사용 | Map |
| 헤더 | header | 요청 헤더 정보 데이터 접근시 사용<br>request.getHeader(headerName) 과 동일 | Map |
| | headerValues | 요청 헤더 정보 데이터들을 배열로 접근시 사용<br>request.getHeaders(headerName)과 동일 | Map |

# EL

- EL을 이용한 JSP 내장객체 데이터 출력

| 식 | 설명 |
|---|---|
| ${pageScope.ATTRIBUTE} | page 내장객체의 데이터 출력 |
| ${requestScope.ATTRIBUTE} | request 내장객체의 데이터 출력 |
| ${sessionScope.ATTRIBUTE} | session 내장객체의 데이터 출력 |
| ${applicationScope.ATTRIBUTE} | application 내장객체의 데이터 출력 |

# EL

- EL을 이용한 JSP 내장객체 데이터 출력

ELImplicitObject.jsp

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

<%
    request.setAttribute("dataname", "request Attribute");
    application.setAttribute("dataname", "application Attribute");
%>

<html>
    <head>
        <title>EL을 이용한 JSP 내장객체 사용</title>
    </head>
    <body>
        <h3>request 내장객체의 dataname 속성값 : </h3>
        [<%=request.getAttribute("dataname") %>] <hr/>
        <h3>application 내장객체의 dataname 속성값 : </h3>
        [<%=application.getAttribute("dataname") %>] <hr/>
        <h3>EL에서 출력되는 dataname 의 값 </h3>
        [${dataname}]<hr/>
        <h3>EL에서 출력되는 request JSP 내장객체의 dataname 의 값 </h3>
        [${requestScope.dataname}]<hr/>
        <h3>EL 에서 출력되는 application JSP 내장객체의 dataname 의 값 </h3>
        [${ applicationScope.dataname}]

    </body>
</html>
```

# EL

- 요청 파라미터 데이터 출력

| 파라미터의 종류 | 사용 구분 |
|---|---|
| 단일값을 가지는 파라미터 | ${param.ParamName} |
| | ${param[ “ParamName” ]} |
| 다중값을 가지는 파라미터 | ${paramValues.ParamName[index]} |
| | ${paramValues{ “ParamName[index]} |

# EL

- 요청 파라미터 데이터 출력 (HTML) ELRequestParam.html

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
        <title>EL에 요청 파라미터 데이터 전달하기</title>
    </head>
    <body>
        <form action="ELRequestParam.jsp" method="post">
            이름 <input type="text" name="name"><hr/>
            사용언어
            <table>
                <tr>
                    <td><input type="checkbox" name="language" value="Java"></td>
                    <td>Java</td>
                </tr>
                <tr>
                    <td><input type="checkbox" name="language" value="C#"></td>
                    <td>C#</td>
                </tr>
                <tr>
                    <td><input type="checkbox" name="language" value="Phython"></td>
                    <td>Phython </td>
                </tr>
                <tr>
                    <td><input type="checkbox" name="language" value="Ruby"></td>
                    <td>Ruby</td>
                </tr>
            </table><hr/>
            <input type="submit" value="전송">
        </form>
    </body>
</html>
```

# EL

- 요청 파라미터 데이터 출력 (JSP)

```jsp
<%@ page contentType="text/html; charset=utf-8"%>
<%
    request.setCharacterEncoding("utf-8");
%>
<html>
    <head>
        <title>EL 요청 파라미터 데이터 사용</title>
    </head>
    <body>
        이름 : ${param.name}<br/>
        선택 언어:  ${paramValues.language[0]}
                    ${paramValues.language[1]}
                    ${paramValues.language[2]}
                    ${paramValues.language[3]}
    </body>
</html>
```

ELRequestParam.jsp

# EL

- 요청 파라미터 데이터 출력

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
                      http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  version="3.0"
  metadata-complete="true">

  <context-param>
    <param-name>initParamName</param-name>
    <param-value>initParamValue</param-value>
  </context-param>

</web-app>
```

```
${initParam.PARAM_NAME}
${initParam["PARAM_NAME"]}
```

# EL

- 요청 파라미터 데이터 출력 (web.xml)

```
<%@ page contentType="text/html; charset=utf-8"%>
<html>
    <head>
        <title>EL 웹 애플리케이션 초기화 파라미터 사용</title>
    </head>
    <body>
        초기화 파라미터  initParamName 값 출력<hr/>
        [${initParam.initParamName}]<br/>
        [${initParam["initParamName"]}]<br/>
    </body>
</html>
```

ELInitParam.jsp

# EL

- 쿠키정보 출력

```
${ cookie.COOKIE_NAME.property}

${ cookie.COOKIE_NAME["property "]}

${ cookie["COOKIE_NAME"].property}

${ cookie["COOKIE_NAME"]["property "]}
```

# EL

- 쿠키정보 출력

```jsp
<%@ page contentType="text/html; charset=utf-8"%>
<%
    Cookie cookie = new Cookie("cookieName", "cookieValue");
    response.addCookie(cookie);
%>

<html>
    <head>
        <title>EL 쿠키 생성</title>
    </head>
    <body>
        쿠키 생성이 완료되었습니다.
    </body>
</html>
```

ELMakeCookie.jsp

# EL

- 쿠키정보 출력

ELPrintCookie.jsp

```
<%@ page contentType="text/html; charset=utf-8"%>
<html>
    <head>
        <title>EL 쿠키 출력</title>
    </head>
    <body>
        <h3>cookie 작성 후 출력</h3>
        ${cookie} <hr/>

        <h3>cookie.cookieName 작성 후 출력</h3>
        ${cookie.cookieName} <hr/>

        <h3>cookie.cookieName의 value 출력</h3>
        ${cookie.cookieName.value}<br/>
        ${cookie.cookieName["value"]}<br/>
        ${cookie["cookieName"].value}<br/>
        ${cookie["cookieName"]["value"]}<br/>

    </body>
</html>
```

# EL

- 헤더 정보 출력

```
${ header.HEADER_NAME }
${ header.["HEADER_NAME"] }

${ headerValues.HEADER_NAME[index] }
${ headerValues["HEADER_NAME"][index] }
```

ELHeader.jsp

```
<%@ page contentType="text/html; charset=utf-8"%>
<html>
    <head>
        <title>EL 헤더 출력</title>
    </head>
    <body>
        <h3>header.cookie</h3>
        ${header.cookie}<hr/>

        <h3>header.connection</h3>
        ${header["connection"]}<hr/>

        <h3>header.accept-language</h3>
        ${header["accept-language"]}<hr/>

        <h3>headerValues.host[0]</h3>
        ${headerValues.host[0]}<hr/>

        <h3>headerValues.accept[0]</h3>
        ${headerValues["accept"][0]}
    </body>
</html>
```

# EL

- EL의 pageContext 내장객체

| 사용 | 설명 |
|---|---|
| ${pageContext.request}<br>${pageContext[ “request” ]} | request 내장객체를 가져옴<br>PageContext 클래스의 **getRequest()** 메소드와 동일 |
| ${pageContext.response}<br>${pageContext[ “response” ]} | response 내장객체를 가져옴<br>PageContext 클래스의 **getResponse()** 메소드와 동일 |
| ${pageContext.page}<br>${pageContext[ “page” ]} | page 내장객체를 가져옴<br>PageContext 클래스의 **getPage()** 메소드와 동일 |
| ${pageContext.exception}<br>${pageContext[ “exception” ]} | exception 내장객체를 가져옴<br>PageContext 클래스의 **getException()** 메소드와 동일 |
| ${pageContext.errorData}<br>${pageContext[ “errorData” ]} | 에러정보를 가져옴<br>PageContext 클래스의 **getErrorData()** 메소드와 동일 |
| ${pageContext.servletConfig}<br>${pageContext[ “servletConfig” ]} | servletConfig 인스턴스를 가져옴<br>PageContext 클래스의 **getServletConfig()** 메소드와 동일 |
| ${pageContext.servletContext}<br>${pageContext[ “servletContext” ]} | servletContext 인스턴스를 가져옴<br>PageContext 클래스의 **getServletContext()** 메소드와 동일 |
| ${pageContext.session}<br>${pageContext[ “session” ]} | session 내장객체를 가져옴<br>PageContext 클래스의 **getSession()** 메소드와 동일 |

# EL

- EL의 pageContext 내장객체 ELPageContext.jsp

```
<%@ page contentType="text/html; charset=utf-8"%>
<html>
    <head>
        <title>EL pageContext 사용</title>
    </head>
    <body>
        <h3>pageContext.request</h3>
        ${pageContext.request}<hr/>

        <h3>pageContext.request.requestURI</h3>
        ${pageContext.request.requestURI}<hr/>

        <h3>pageContext.request["requestURI"]</h3>
        ${pageContext.request["requestURL"]}<hr/>

        <h3>pageContext["request"]["requestedSessionId"]</h3>
        ${pageContext["request"]["requestedSessionId"]}<hr/>

        <h3>pageContext["request"].queryString</h3>
        ${pageContext["request"].queryString}


    </body>
</html>
```

# EL

- EL의 산술 연산자

ELArithmeticOp.jsp

```
<%@ page contentType="text/html; charset=utf-8"%>
<html>
    <head>
        <title>EL의 산술연산자</title>
    </head>
    <body>
        <h3>param1 + param2</h3> ${param.param1 + param.param2}<hr/>

        <h3>param1 - param2</h3>
        ${param.param1 - param.param2}<hr/>

        <h3>param1 * param2</h3>
        ${param.param1 * param.param2}<hr/>

        <h3>param1 / param2</h3>
        ${param.param1 / param.param2}<hr/>

        <h3>param1 div param2</h3>
        ${param.param1 div param.param2}<hr/>

        <h3>param1 % param2</h3>
        ${param.param1 % param.param2}<hr/>

        <h3>param1 mod param2</h3>
        ${param.param1 mod param.param2}

    </body>
</html>
```

| 연산자 | 기능 |
|---|---|
| + | 덧셈 |
| - | 뺄셈 |
| * | 곱셈 |
| / | 나눗셈 |
| div | |
| % | 나머지연산 |
| mod | |

# EL

- EL의 비교 연산자

| 연산자 | 설명 |
|---|---|
| == | 양측 피연산자의 결과값이 같으면 **true,** 아니면 **false** 반환 |
| eq | |
| != | 양측 피연산자의 결과값이 다르면 **true,** 아니면 **false** 반환 |
| ne | |
| > | 왼쪽 피연산자의 결과값이 오른쪽 피연산자의 결과값보다 크면 **true** 아니면 **false** 반환 |
| gt | |
| >= | 왼쪽 피연산자의 결과값이 오른쪽 피연산자의 결과값과 같거나 크면 **true** 아니면 **false** 반환 |
| ge | |
| < | 왼쪽 피연산자의 결과값이 오른쪽 피연산자의 결과값보다 작으면 **true** 아니면 **false** 반환 |
| lt | |
| <= | 왼쪽 피연산자의 결과값이 오른쪽 피연산자의 결과값과 같거나 작으면 **true** 아니면 **false** 반환 |
| le | |

# EL

- EL의 비교 연산자

ELComparisonOp1.jsp

```
<%@ page contentType="text/html; charset=utf-8"%>
<html>
    <head>
        <title>EL의 비교연산자</title>
    </head>
    <body>

        <h3>param.param == 3 : ${param.param == 3}</h3>
        <h3>param.param eq 3 : ${param.param eq 3}</h3><hr/>

        <h3>param.param != 3 : ${param.param != 3}</h3>
        <h3>param.param ne 3 : ${param.param ne 3}</h3><hr/>

        <h3>param.param > 3 : ${param.param > 3}</h3>
        <h3>param.param gt 3 : ${param.param gt 3}</h3><hr/>

        <h3>param.param >= 3 : ${param.param >= 3}</h3>
        <h3>param.param ge 3 : ${param.param ge 3}</h3><hr/>

        <h3>param.param < 3 : ${param.param < 3}</h3>
        <h3>param.param lt 3 : ${param.param lt 3}</h3><hr/>

        <h3>param.param <= 3 : ${param.param <= 3}</h3>
        <h3>param.param le 3 : ${param.param le 3}</h3>

    </body>
</html>
```

# EL

- EL의 비교 연산자 (문자열) ELComparisonOp2.jsp

```jsp
<%@ page contentType="text/html; charset=utf-8"%>
<html>
    <head>
        <title>EL의 비교연산자</title>
    </head>
    <body>

        <h3>param.skill == "JSP" : ${param.skill == "JSP"}</h3>
        <h3>param.skill eq "JSP" : ${param.skill eq "JSP"}</h3><hr/>

        <h3>param.skill != "JSP" : ${param.lang1 != "JSP"}</h3>
        <h3>param.skill ne "JSP" : ${param.lang1 ne "JSP"}</h3><hr/>

        <h3>param.skill > "JSP" : ${param.skill > "JSP"}</h3>
        <h3>param.skill gt "JSP" : ${param.skill gt "JSP"}</h3><hr/>

        <h3>param.skill >= "JSP" : ${param.skill >= "JSP"}</h3>
        <h3>param.skill ge "JSP" : ${param.skill ge "JSP"}</h3><hr/>

        <h3>param.skill < "JSP" : ${param.param < "JSP"}</h3>
        <h3>param.skill lt "JSP" : ${param.param lt "JSP"}</h3><hr/>

        <h3>param.skill <= "JSP" : ${param.skill <= "JSP"}</h3>
        <h3>param.skill le "JSP" : ${param.skill le "JSP"}</h3>

    </body>
</html>
```

# EL

- EL의 논리 연산자

| 연산자 | 기능 |
|---|---|
| && | 논리 **AND** 연산자로 양측 피연산자의 결과값이 모두 **true** 일때 **true** 아니면 **false**를 반환 |
| and | |
| \|\| | 논리 **OR** 연산자로 양측 피연산자의 결과값이 모두 **false** 일대 **fasel** 아니면 **true**를 반환 |
| or | |
| ! | 논리 부정 연산자로 피연산자의 결과값이 **true**면 **false, false**면 **true**를 반환 |
| not | |

# EL

- EL의 논리 연산자 ELLogicalOp.jsp

```
<%@ page contentType="text/html; charset=utf-8"%>
<html>
    <head>
        <title>EL의 논리연산자</title>
    </head>
    <body>

        <h3>(param.param1 > 3) && (param.param2 > 3) :
        ${(param.param1 > 3) && (param.param2 > 3)}</h3>
        <h3>(param.param1 > 3) and (param.param2 > 3) :
        ${(param.param1 > 3) and (param.param2 > 3)}</h3><hr/>

        <h3>(param.param1 < 10) && (param.param2 < 10) :
        ${(param.param1 < 10) && (param.param2 < 10)}</h3>
        <h3>(param.param1 < 10) and (param.param2 < 10) :
        ${(param.param1 < 10) and (param.param2 < 10)}</h3><hr/>

        <h3>(param.param1 < 5) || (param.param2 < 5) :
        ${(param.param1 < 5) || (param.param2 < 5)}</h3>
        <h3>(param.param1 < 5) or (param.param2 < 5) :
        ${(param.param1 < 5) or (param.param2 < 5)}</h3><hr/>

        <h3>(param.param1 < 3) || (param.param2 < 3) :
        ${(param.param1 < 3) || (param.param2 < 3)}</h3>
        <h3>(param.param1 < 3) or (param.param2 < 3) :
        ${(param.param1 < 3) or (param.param2 < 3)}</h3><hr/>

        <h3>!(param.param1 == param.param2) :
        ${!(param.param1 == param.param2)}</h3>
        <h3>not(param.param1 == param.param2) :
        ${not(param.param1 == param.param2)}</h3>

    </body>
</html>
```

# EL

- EL의 삼항 연산자

```
<%@ page contentType="text/html; charset=utf-8"%>
<html>
    <head>
        <title>EL의 삼항연산자</title>
    </head>
    <body>

        param.gender == "male" ? "남자" : "여자" <br/>
        <h3>결과 : ${param.gender == "male" ? "남자" : "여자"}</h3>

    </body>
</html>
```

ELTernaryOp.jsp

# EL

- empty 연산자

```
${empty DATA}
```

ELEmptyOp.jsp

```jsp
<%@ page contentType="text/html; charset=utf-8"%>
<html>
    <head>
        <title>EL의 empty 연산자</title>
    </head>
    <body>

        <h3>요청 파라미터 param1 데이터가 빈 상태(empty) 입니까? <br/>
        ${empty param.param1}</h3><hr/>
        <h3>요청 파라미터 param2 데이터가 빈 상태(empty) 입니까? <br/>
        ${empty param.param2}</h3>

    </body>
</html>
```

ELEmptyOp2.jsp

```jsp
<%@ page contentType="text/html; charset=utf-8"%>
<html>
    <head>
        <title>EL의 empty 연산자</title>
    </head>
    <body>

        <h3>${empty param.id ? "방문자" : param.id}님, 환영합니다.</h3>

    </body>
</html>
```

# EL

- 연산자의 우선순위

| 연산자 | 우선순위 |
| --- | --- |
| [] | 높음<br>낮음 |
| () | |
| -(단항) ! not empty | |
| * / div % mod | |
| + - | |
| < > <= >= lt gt le ge | |
| == != eq ne | |
| && and | |
| \|\| or | |
| ? : | |

# EL

- EL을 이용한 자바 인스턴스 메소드 사용

```
package MethodTest;

public class InstanceMethod {

    private int data;

    public void setData(int data) {
        this.data = data;
    }

    public int getData() {
        return data;
    }

    public int getAddData(int a, int b){
        this.data = (a + b);
        return this.data;
    }

    public int[] getArrayData (int a, int b){
        int[] result = new int[2];
        result[0] = a;
        result[1] = b;
        return result;
    }
}
```

InstanceMethod.java

# EL

- EL을 이용한 자바 인스턴스 메소드 사용

ELInstanceMethod.jsp

```
<%@ page contentType="text/html; charset=utf-8"%>
<%@ page import="MethodTest.InstanceMethod" %>

<%
    InstanceMethod instanceMethod = new InstanceMethod();
    request.setAttribute("usingMethod", instanceMethod);
%>

<html>
    <head>
        <title>EL의 인스턴스 메소드 사용</title>
    </head>
    <body>

        <h3>instanceMethod 객체의 setData 메소드 사용<br/>
        ${usingMethod.setData(10)}
        메소드 호출 완료 </h3><hr/>

        <h3>instanceMethod 객체의 getData 메소드 사용<br/>
        메소드 호출 결과 : ${usingMethod.getData()}</h3><hr/>

        <h3>instanceMethod 객체의 다중 입력 파라미터를 사용하는 getAddData 메소드 사용<br/>
        메소드 호출 결과 : ${usingMethod.getAddData( usingMethod.getData(), 3 )}</h3><hr/>

        <h3>instanceMethod 객체의 배열 반환 getArrayData 메소드 사용<br/>
        메소드 호출 결과 : ${usingMethod.getArrayData( usingMethod.getData(), 3) }</h3><hr/>

        <h3>instanceMethod 객체의 getArrayData 메소드를 통해 반환된 배열 사용<br/>
        0번 인덱스 값 : ${usingMethod.getArrayData( usingMethod.getData(), 3)[0] }<br/>
        1번 인덱스 값 : ${usingMethod.getArrayData( usingMethod.getData(), 3)[1] }<br/>
        2번 인덱스 값 : ${usingMethod.getArrayData( usingMethod.getData(), 3)[2] }</h3>

    </body>
</html>
```

# EL

- EL을 이용한 자바 정적 메소드 사용

StaticMethod.java

```
package MethodTest;

public class StaticMethod {

    public static int getData(int data) {
        return data;
    }
}
```

static-mathod.tld

```
<taglib xmlns="http://java.sun.com/xml/ns/javaee" version="2.1">
    <description>EL의 자바 정적 메소드 사용</description>
    <tlib-version>1.0</tlib-version>
    <short-name>ELStaticMethod</short-name>

    <function>
        <name>useStaticMethod</name>
        <function-class>MethodTest.StaticMethod</function-class>
        <function-signature>int getData(int)</function-signature>
    </function>
</taglib>
```

# EL

- EL을 이용한 자바 정적 메소드 사용

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
                      http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  version="3.0"
  metadata-complete="true">

    <jsp-config>
        <taglib>
            <taglib-uri>/WEB-INF/tlds/static-method.tld</taglib-uri>
            <taglib-location>/WEB-INF/tlds/static-method.tld</taglib-location>
        </taglib>
    </jsp-config>

</web-app>
```

```
<%@ taglib prefix="태그라이브러리 구분용 접두어" uri="TLD 파일의 URI" %>

${ 접두어 : TLD 파일의 <function> 태그 내 <name> 태그로 지정된 함수명 ( 입력 파라미터 )}
```

# EL

- EL을 이용한 자바 정적 메소드 사용

```
<%@ page contentType="text/html; charset=utf-8"%>
<%@ taglib prefix="elstaticmethod" uri="/WEB-INF/tlds/static-method.tld" %>

<html>
    <head>
        <title>EL의 정적 메소드 사용</title>
    </head>
    <body>
        <h3>StaticMethod 클래스의 getData 정적 메소드 호출 결과<br/>
        ${elstaticmethod:useStaticMethod(10)}</h3>
    </body>
</html>
```

ELStaticMethod.jsp

# EL

- EL 의 비활성화
  - page 지시자를 이용한 비활성화

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

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

  - web.xml의 <jsp-property-group> 태그를 이용

```
<jsp-config>
    <jsp-property-group>
        <url-pattern>/notUsingEL.jsp</url-pattern>
        <el-ignored>true</el-ignored>
    </jsp-property-group>
</jsp-config>
```

# 오늘 숙제

- 지금까지 작성했던 모든 출력 예제들
  - EL로 변경