EL (Expression Language)

< 1. EL의 이해 >

JSP에서 자바 영역 즉, 변수를 선언하거나 if문 또는 for문을 사용하기 위해서는 표현식(<%%>)을 사용했다. 그러나 **MVC 패턴**에서는 뷰단과 프로세스단을 분리해야 하는데 이를 위해서 **EL**을 사용한다! **EL**은 **표현식**과 **액션태그**를 간략하게 표현한 것이다.

< 2. 표현식의 EL 표기법과 사용가능한 연산자 >

표기법 : EL표기법 (${value }) = 표현식 (<%= value %>)

연산자 : 1) 산술 : + , - , * , / , %

2) 관계 : == , != , < , > , <= , >=

3) 조건 : a(조건) ? : b : c => 삼항연산자

-> (예시) : ${(1 > 2) ? "1>2입니다" : "1>2가 아닙니다" }

4) 논리 : && , ||

5) ${ **empty** 값} => Null이야? 라고 물었을 때 Null이면 true(참)

-> (예시) : ${**empty** validMember ? "로그인전":"로그인후"}

6) ${**not empty** 값} => Null이 아니야? 라고 물었을 때 Null이 아니면 true(참)

< 3. 액션태그로 사용되는 EL >

<jsp:getProperty name="member" property="name"/>을 ${ member.name }로 표현이 가능!

(예시)

```
<jsp:useBean id="member" class="com.lec.ex.MemberDto" scope="page"/>
<jsp:setProperty property="*" name="member"/>
```

⇨ 우선 자바 빈 생성과 속성을 설정한다

```
<h2>이름 <jsp:getProperty name="member" property="name"/> </h2> 와
<h2>이름 : ${member.name }</h2> 는 같은 역할이다
```

< 4. EL의 내장객체들 >

1. Scope에 관한 객체

1) **pageScope** : page객체의 attributte를 참조하는 객체

2) **requestScope** : request객체의 attribute를 참조하는 객체

➔ ${requestScope.requestName } = <%=request.getAttribute("requestName") %>

➔ ${requestName } = <%=request.getAttribute("requestName") %>

3) **sessionScope** : session객체의 attribute를 참조하는 객체

➔ ${sessionName } = <%=session.getAttribute("sessionName") %>

4) **applicationScope** : application객체의 attribute를 참조하는 객체

★ 참고로 Scope 객체에 한하여 scope 변수 생략이 가능하다!

2. 그 외 유용한 객체

1) **param** : 요청 파라미터를 참조하는 객체(웹 브라우저로부터 입력된 데이터의 집합)

(예시)

```
<% String name = request.getParameter("name"); %>

<h2>이름 : <%=name %></h2> = <h2>이름 : ${param.name }</h2>
```

2) **paramValues** : 요청 파라미터(배열)을 참조하는 객체

3) **initParam** : 초기화 파라미터를 참조하는 객체

➔ 특정 데이터를 공유해야 할 경우 web.xml에 <context-param> 태그 안에 <param-name>과 <param-value>를 설정하고 여러 서블릿에서 initparam 객체로 파라미터 받음!

(예시)

```
<context-param>
   <param-name>contextId</param-name>
   <param-value>scott</param-value>
</context-param>                                    => web.xml 안에 표기
```

<%=getServletContext().getInitParameter("contextId") %> 와

${initParam.contextId } 는 scott이라는 같은 속성을 불러온다.

```
public int reply(BoardDto dto)
```

➔ SQL문 : "INSERT INTO BOARD (NUM, WRITER, SUBJECT, CONTENT, EMAIL, PW, REF, RE_STEP, RE_LEVEL, IP) " + " VALUES ((SELECT NVL(MAX(NUM),0)+1 FROM BOARD), ?, ?, ?, ?, ?, ?, ?, ?, ?)"

➔ Connection 객체를 null 값으로 초기화하기 전에

preReplyStepA(dto.getRef(), dto.getRe_step()); 메소드 호출

➔ pstmt.setInt(7, dto.getRe_step()+1); / pstmt.setInt(8, dto.getRe_indent()+1);

➔ DB에 저장될 dto 내용은 dto.setRe_step(dto.getRe_step()+1);

dto.setRe_indent(dto.getRe_indent()+1); 로 세팅

< 5. list.jsp에서 답변글의 경우 공간 확보 >

```
if(dto.getRe_indent()>0){
  int width = dto.getRe_indent() * 10;
  out.println("<img src='"+conPath+"/img/level.gif' width='"+width +"'height='10' />");
  out.println("<img src='"+conPath+"/img/re.gif' />");
}
```

< 6. writeForm.jsp에서 dto에 원글에 대한 정보 저장 >

```
if(request.getParameter("num")!= null) {
  num = Integer.parseInt(request.getParameter("num"));
              BoardDao dao = BoardDao.getInstance();
```

```
                dto = dao.getBoardOneLine(num);
                ref = dto.getRef(); // 원글의 ref
                re_step = dto.getRe_step(); // 원글의 re_step
                re_indent = dto.getRe_indent(); // 원글의 re_indent
}
```

그리고 hidden 타입으로 pageNum/num/ref/re_step/re_indent 파라미터 값을 writePro.jsp로 넘겨준다

(예시) : `<input type="hidden" name="ref" value="<%=ref %>" />`

< 7. writePro.jsp에서 원글쓰기 / 답글쓰기 나눠주기 >

```
if(dto.getNum()==0){
        result= dao.insertBoard(dto); //새글
}else{
        result = dao.reply(dto); // 답글쓰기
}
```