EL(Expression Language) ; 표현식 또는 액션태그를 대신해서 값을 표현

1. EL (Expression Language)란?

   ✓ EL(Expression Language)란, 표현식(<%=~ %>) 를 조금 더 간략하게 대신해서 값을 표현하는 언어. 액션 태그를 조금 다르게(간략하게) 표현한 것.

   ✓ 표현식 (<%= value %>) → EL 표기법(${ value }) 사용법 : 빈 태그나 jstl태그로 인해 생성된 객체만 el표기법 안에 사용

```
<body> 코드를 보면 물론 HTML은 다르다
        <h1>${10 }</h1>
        <h1>${99.99}</h1>
        <h1>${"ABC" }</h1>
        <h1>${true}</h1>
</body>
```

```
<h2>EL표기법을 이용(표현식 안에 쓸 수 있는 연산자는 다 사용 가능)</h2>
        <p>산술연산 : 10+1 = ${10+1 }</p>
        <p>산술연산 : 10/0.0 = ${10/0.0 }</p> <!-- 0으로 나누면 에러, 0.0으로 나누면 무한대 -->
        <p>관계연산 : 1==2 => ${1 eq 2 }></p>
        <p>관계연산 : 1!=2 => ${1!=2 }</p>
        <p>관계연산 : 1 &lt;2 => ${1<2 }</p> <!-- <라는 태그라고 인식할 수도 있음 -->
        <p>조건연산 : ${(1>2)? "1이 2보다 큽니다" : "1이 2보다 크지 않습니다" }</p>
        <p>논리연산 : (1>2)||(1<2)=${(1>2)||(1<2) }</p>
        <p>로그인 여부 ${customer ==null ? "로그인전":"로그인후" }</p>
        <p>로그인 여부 ${customer eq null ? "로그인전":"로그인후" }</p>
        <p>로그인 여부 ${empty customer ? "로그인전":"로그인후" }</p>
        <p>로그인 여부 ${not empty customer ? "로그인후":"로그인전" }</p>
```

➔ Session이 내장객체인데 ,

```
        <p>pageNum 파라미터 여부 ${ empty param.pageNum ? "pagenNum 안옴":"pageNum옴" }</p>
        <p>name 파라미터 : ${param.name }</p> //param이라는 EL의 내장객체 이용
```

${ eq , not empty , empty} 를 mvc패턴을 위해 익숙해져야한다.

<%%> 대신 ${}를 써준다.

빈태그 이용 (mvc패턴에서는 잘 안쓴다)

--빈태그를 활용하기 위해 html 만들고 form action에 ex.jsp 넣어주고 dto도 자바에 생성

--ex.jsp에서 usebean과 setProperty해주기

-- bean태그를 이용한 정보 출력

```
<h2>el 표기법을 이용한 정보 출력 </h2>
        <h2>이름: ${member.name }</h2>
        <h2>ID: ${member.id }</h2>
```

```
<h2>PW: ${member.pw }</h2>
```

처럼 이용 할 수 있다. Getter을 알아서 찾아가서 호출한다.

객체 이름을 생략 할 수 있다.

내장 객체

- ✓ Scope에 관한 객체
  - ① pageScope : page객체의 attributte를 참조하는 객체(page attribute의 집합) Map 타입
  - ② requestScope : request객체의 attribute를 참조하는 객체(rquest attribute의 집합) Map 타입
  - ③ sessionScope : session객체의 attribute를 참조하는 객체(session attribute의 집합) Map 타입
  - ④ applicationScope : application객체의 attribute를 참조하는 객체(application attribute의 집합) Map 타입

innerObject.jsp

```
//이 페이지를 벗어나면 없어지는 속성
pageContext.setAttribute("pageName", "pageValue");
//새로운 request객체를 생성하기 전까지 유효
request.setAttribute("requestName", "requestValue");
//브라우저를 닫기전까지 유효
session.setAttribute("sessionName", "sessionValue");
//메모리에서 이 어플리케이션이 사라지기 전까지 유효(톰캣을 껐다 키기 전까지)
application.setAttribute("applicationName", "applicationValue");
// forward 할 객체 (dispatcher)
RequestDispatcher dispatcher = request.getRequestDispatcher("ex02_scope.jsp?name=kim&id=aaa");
dispatcher.forward(request, response);
```

Submit을 실행하면 페이지가 넘어가고 새로운 url을 받아오게 되므로 request가 새로 생성되면서 페이지 속성과 request속성은 사라진다. (만약 새로운 request객체 생성을 원하지 않으면 forward해주자)

url 창에 기존과 다른 url이 생기면 새로운 request가 생긴것이다.

Forward를 해주면 request를 갖고 ex02_scope로 넘어간다

Forward를 하고싶으면 무조건 위에서 session을 생성 해줘야 한다

Forward는 다음 장에서 더 자세히 설명

- ✓ 그 외 유용한 객체
  - ① **param** : 요청 파라미터를 참조하는 객체(웹 브라우저로부터 입력된 데이터의 집합) Map 타입
  - ② paramValues : 요청 파라미터(배열)를 참조하는 객체(웹 브라우저로부터 입력된 데이터의 집합-똑 같은 이름의 데이터가 여럿일 때 사용) Map 타입

③ initParam : 초기화 파라미터를 참조하는 객체(웹 어플리케이션의 초기화 파라미터의 집합) Map 타입

④ cookie : cookie객체를 참조하는 객체(웹 브라우저로부터 전송된 쿠키의 집합)

데이터 공유 : 여러 서블릿에서 특정 데이터를 공유해야 할 경우 context Parameter를 이용해서 web.xml에 데이터를 기술하고 공유하면서 사용할 수 있다.

```
<body>
  context 초기화 파라미터<br>
    contextId:<%=getServletContext().getInitParameter("contextId") %><
    br>
    contextPw:<%=getServletContext().getInitParameter("contextPw") %><
br>
    contextPath:<%=getServletContext().getInitParameter("contextPath")
%><br>
  <hr>
    ${initParam.contextId }<br>
    ${initParam.contextPw }<br>
    ${initParam.contextPath }
</body>
```

Web.xml

```
<context-param>
    <param-name>contextId</param-name>
    <param-value>scott</param-value>
  </context-param>
  <context-param>
    <param-name>contextPw</param-name>
    <param-value>tiger</param-value>
  </context-param>
```