※ 지금까지 공부한 내용 정리

● HTML내에서 자바를 사용할 수 있는 방법 : 스크립트 릿, 표현식, 선언부

● JSP의 문법 : page지시자(자바 패키지를 import 할 때 사용, 에러 처리 페이지를 지정할 때 사용)

● JSP에서 사용할 수 있는 내장 객체들 : request, response, out, session, applicatoin

HTML안에서 자바의 사용이 불편하지 않나요?

● JSP의 작성

- JSP는 HTML내에 자바 코드를 작성할 수 있는 기법
- 그런데, 자바 코드가 길어질수록 점점 자바 코드 작성이 불편해진다. 특히, "}"문자 처리가 불편하다.
- 또한, 자바를 모르면 JSP를 작성할 수 없는데, 자바를 몰라도 JSP를 작성할 수 있도록 해야 한다.
- 따라서, 자바 코드를 대체하는 다양한 태그가 JSP에 도입되었다.
- 이러한 태그에는 다음가 같은 종류가 있다.

① 표현언어(expression language)

② 표준 액션(standard action tag)

③ 커스텀 액션(custom action tag) 와 JSTL(JSP Standard Tag Library)

● 표현 언어(EL)

- 자바의 표현식을 대체하는 태그로서 "${어떤식}"의 형태로 사용한다.
- EL과 표현식의 비교

· EL : ${cnt + 1}

· 표현식 : <%= cnt + 1 %>

- 표현식에서 사용할 수 있는 변수는 자바에서 정의한 변수만 해당되지만, EL에서 사용할 수 있는 변수는 자바에서 정의한 변수 뿐만 아니라, 다양한 파라미터의 이름도 사용할 수 있다.
- 다음의 예를 비교해 본다.

· Problem.jsp의 내용

```
<%
        int sum = 0;
        for(int cnt = 1; cnt <= 100; cnt++) {
                sum = sum + cnt;
        }
        request.setAttribute("RESULT", new Integer(sum));
        RequestDispatcher dispatcher = request.getRequestDispatcher("Result.jsp);
        dispatcher.forward(request, response);
%>
```

· 위의 결과를 사용자에게 보여주는 Result.jsp

```
<%@ page contentType = "text/html; charset = euc-kr" %>
<HTML>
        <HEAD><TITLE>1부터 100까지의 합</TITLE></HEAD>
        <BODY>
```

```
            1 부터 100까지의 합은 ? <%= request.getAttribute("RESULT") %>
        </BODY>
</HTML>
```

· 위와 같은 결과를 EL을 사용해서 작성한 Result.jsp

```
<%@ page contentType = "text/html"; charset = "euc-kr" %>
<HTML>
        <HEAD><TITLE>1부터 100까지의 합</TITLE></HEAD>
        <BODY>
            1 부터 100까지의 합은 ? ${RESULT}
        </BODY>
</HTML>
```

- 이와 같이 EL은 편리할 뿐만 아니라, 파라미터의 값을 출력할 때도 request객체를 사용할 필요가 없다.

● ${NAME} 일 경우, NAME은 request, session, applicatoin 중에 어디에서 설정된 파라미터 이름일까?

- NAME은 다음과 같이 파라미터로 설정할 수 있다.

· requst객체에서 설정한 경우 : request.setAttribute("NAME", "John");

· session객체에서 설정한 경우 : session.setAttribute("NAME", "John);

· application객체에서 설정한 경우 : application.setAttribute("NAME", "John");

· 현재 page에 설정하는 경우 : pageContext.setAttribute("NAME", "John");

※ 주의 : pageContext는 현재 page를 객체형태로 가지고 있는 객체로, page안에 포함되어 있는 request, session, applicatoin 등의 객체도 간접적으로 보유하고 있는 객체입니다.

- "NAME"이라는 동일한 이름으로 위와 같이 모두 4개의 객체에서 속성으로 설정할 수 있으므로, 순서대로 "NAME"이라는 속성을 찾는다. 순서는 다음과 같다.

① 페이지 → ② request → ③ session → ④ application

- 다음은 현재 페이지의 RESULT 속성에 1 부터 1000까지의 합을 설정하고, 다시 RESULT 속성을 읽어서 출력한다.

```
<%@ page contentType = "text/html; charset = euc-kr" %>
<%
        int sum = 0;
        for(int cnt = 1; cnt <= 1000; cnt++) sum = sum + cnt;
        pageContext.setAttribute("RESULT", new Integer(sum));
%>
<HTML>
        <HEAD><TITLE>1 부터 1000까지의 합</TITLE></HEAD>
        <BODY>
            1 부터 1000가지 더한 결과는 ? ${RESULT}
        <BODY>
</HTML>
```

- 위에서 "RESULT"라는 속성을 “현재 페이지”부터 찾아서 그 결과를 출력한다.

- 이와 같이 EL은 속성을 찾기 위해서 속성의 범위를 지칭하는 객체를 별도로 가지고 있다. 즉, “페이지 범위”, “request 범위”, “session 범위”, "application 범위“를 의미하는 객체를 가지고 있다.

● EL이 가지고 있는 객체들I, 즉 내장 객체I

- 지금까지 살펴본 EL에서 사용할 수 내장 객체들을 정리하면 다음과 같다.

객체의 이름	의 미
pageScope	page에 설정된 속성들의 집합
requestScope	request에 설정된 속성들의 집합
sessionScope	session에 설정된 속성들의 집합
applicationScope	applicatoin에 설정된 속성들의 집합

- 예를 들어, request객체에 설정된 "NAME"의 값을 출력하기 위한 JSP(SetName.jsp)

```
<%
        request.setAttribute("NAME", "John");
        RequestDispatcher dispatcher = request.getRequestDispatcher("FindName.jsp");
        dispatcher.forward(request, response);
%>
```

- "NAME" 속성을 찾아서 그 값을 출력하는 JSP(FindName.jsp)

```
<%@ page contentType = "text/html; charset = euc-kr" %>
<HTML>
        <HEAD><TITLE>이름 찾기</TITLE></HEAD>
        <BODY>
                내 이름은  ? ${requestScope.NAME} 입니다.
        </BODY>
</HTML>
```

● EL이 가지고 있는 객체들II, 즉 내장 객체II

- 이외에도 EL은 다음의 객체를 가지고 있다.

객체의 이름	의 미
param	클라이언트의 웹 브라우저에 입력된 데이터의 집합
paramValues	클라이언트의 웹 브라우저에 입력된 데이터의 집합. 동일한 파라미터에 저장된 값이 여러개 일 경우에 사용
header	서버로 요청할 때 사용되는 메세지의 헤더 집합
headerValues	서버로 요청할 때 사용되는 메세지의 헤더 집합. 동일한 이름의 헤더가 여러개 일 경우에 사용
cookie	클라이언트의 웹 브라우저에서 전송된 쿠키의 집합
initParam	웹 어플리케이션을 초기화할 때 사용하는 초기화 파라미터들의 집합
pageContext	JSP 페이지의 환경 정보의 집합

- param의 사용 예

- 회원의 애완동물을 입력받는 JSP(InputPets.html)

```
<HTML>
    <HEAD>
        <META http-equiv="Content-Type" content="text/html;charset=euc-kr">
        <TITLE>아이 러브 펫</TITLE>
    </HEAD>
    <BODY>
        <FORM ACTION=PetsResult.jsp>
```

```
        아이디: <INPUT TYPE=TEXT NAME=ID><BR><BR>
        다음 중 회원님이 키우고 있는 애완 동물을 선택하십시오.<BR><BR>
          개<INPUT TYPE=CHECKBOX NAME=ANIMAL VALUE="dog">
          고양이<INPUT TYPE=CHECKBOX NAME=ANIMAL VALUE="cat">
          새<INPUT TYPE=CHECKBOX NAME=ANIMAL VALUE="bird"><BR><BR>
        <INPUT TYPE=RESET VALUE="취소">
        <INPUT TYPE=SUBMIT VALUE="확인">
      </FORM>
    </BODY>
</HTML>
```

- 입력받은 회원의 ID와 애완동물을 보여주는 JSP(PetsResult.jsp)

```
<%@page contentType="text/html; charset=euc-kr"%>
<HTML>
    <HEAD><TITLE>아이 러브 펫</TITLE></HEAD>
    <BODY>
        아이디: ${param.ID} <BR>
        선택한 동물: ${paramValues.ANIMAL[0]} ${paramValues.ANIMAL[1]} ${paramValues.ANIMAL[2]}
    </BODY>
</HTML>
```

- 회원의 ID를 저장하는 파라미터는 "ID"이며, 회원의 ID에 해당하는 파라미터 값을 읽기위해 사용하는 EL의 내장객체는 param이다.
- 애완동물을 저장하는 파라미터는 "ANIMAL"인데, “개”, “고양이”, “새”등의 세 가지 값을 모두 저장할 수 있다. 이런 경우에 사용하는 EL의 내장 객체는 paramValues이며 배열형태로 복수개의 값을 저장한다.

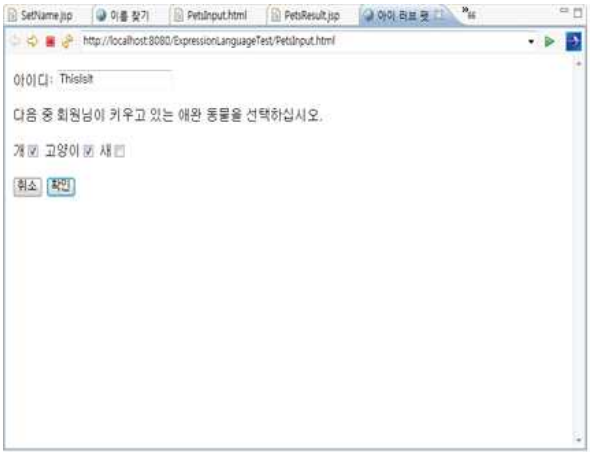
● 도전 과제 : EL을 사용해서 회원 정보를 읽어서 출력하는 JSP를 작성하세요.

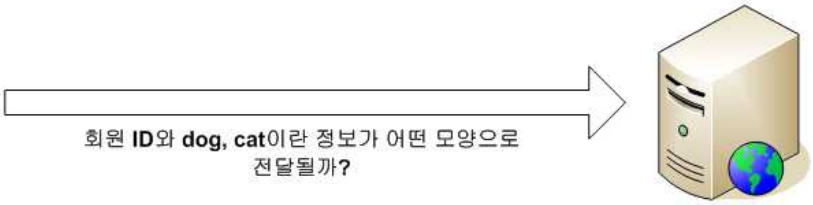
회원정보는 다음과 같다.

- 아이디, 이름, 주소, 성별(2가지 : 둘 중의 하나만 선택가능), 전화번호, 취미(5가지 : 복수 선택가능), 관심분야(5가지 : 복수 선택가능)
- 회원 정보 중 성별은 라디오 버튼, 취미와 관심분야는 체크 버튼으로 구현하세요.

● 여기서 잠깐!

- InputPest.html에서 “확인” 버튼을 클릭하면, 회원의 ID와 회원이 선택한 애완동물 데이터가 서버로 전송되는데, 전송될 때는 메세지의 형태로 전송이 된다.





- 전달되는 형태는 자세히 알필요는 없지만, 대충 어떤 내용이 어떤 모양으로 전송되는지 정도는 알아둘 필요가 있다.
- 클라이언트에서 선택한 데이터는 서버로 일정한 메세지 형태로 전송이 되며, 다음과 같은 형태를 갖는다.

① GET 방식으로 전송하는 경우(별도의 지정이 없으면 GET방식이 된다.)

```
GET /프로젝트이름/PetsResult.jsp?ID=ThisIsIt&ANIMAL=dog&ANIMAL=cat HTTP/X.X
Host:www.xxx.xxx
User-Agent:xxxxx
Accept: image/gif,image/jpeg,xxxx
Accept-Language: ko-KR, en-US;q=0.5
Content-Type: application/x-www-form-urlencoded
Content-Length: x
Connection: Keep-Alive
Cookie: NAME=John; Gender=Male; Age=15
……
```

요청 헤더

몸체 없승

② POST 방식으로 전송하는 경우

```
POST /프로젝트이름/PetsResult.jsp HTTP/X.X
Host:www.xxx.xxx
User-Agent:xxxxx
Accept: image/gif,image/jpeg,xxxx
Accept-Language: ko-KR, en-US;q=0.5
Content-Type: application/x-www-form-urlencoded
Content-Length: x
Connection: Keep-Alive
Cookie: NAME=John; Gender=Male; Age=15
……

ID=ThisIsIt&ANIMAL=dog&ANIMAL=cat
```

요청 헤더

몸체 있승

● EL의 헤더(header) 내장 객체는 위의 요청 헤더 값을 가져올 때 사용하는 객체이다.
- 요청헤더 내의 여러가지 데이터들을 사용하기 위해서 header나 headerValues 객체를 사용한다.
- header객체로 부터 값을 읽어오는 EL의 예
① ${header.Host} : 요청 헤더 중 Host의 값을 읽어온다.
② ${header["Host"]} : 위의 기능과 동일한 다른 표현이다
- 위의 예와 같이 ①② 모두 사용가능하다. 하지만, ①처럼 사용하기 위해서는 요청 헤더의 이름이 자바 식별자의 이름규칙에 벗어나는 경우에는 사용할 수 없다.
${header.User-Agent} : **잘못된 사용**　　${header["User-Agent"]} : 올바른 사용
- 요청 헤더가 둘 이상 있을 경우에는 headerValues를 사용한다.
${headerValues.Accept[0]} 혹은 ${headerValues["Accept"][0]}

● EL의 cookie 내장 객체
- 요청헤더에 있는 쿠키 데이터를 사용하기 위해서 cookie내장 객체를 사용한다.
- "NAME"이라는 쿠키의 값을 읽어오기 위해서는 다음과 같이 한다.
${cookie.NAME.value}　혹은 ${cookie["NAME"]["value"]} ← **여기서의 value의 의미는 "값을 가져와라"란 뜻**
${cookie.NAME["value"]} 혹은 ${cookie["NAME"].value} ← **여기서의 value의 의미는 "값을 가져와라"란 뜻**

● 쿠키 값을 읽어서 출력하는 EL의 예제
- 쿠키 값을 설정하는 JSP(**CookieDataWriter.jsp**)

```
<%@page contentType="text/html; charset=euc-kr" %>
<%
    Cookie cookie = new Cookie("NAME", "John");
    response.addCookie(cookie);
%>
<HTML>
    <HEAD><TITLE>쿠키 데이터 저장 프로그램</TITLE></HEAD>
    <BODY>
        쿠키 값이 설정되었습니다.
    </BODY>
</HTML>
```

- 쿠키 값을 읽는 JSP(**CookieDataReader.jsp**)

```
<%@page contentType="text/html; charset=euc-kr" %>
<HTML>
    <HEAD><TITLE>쿠키 데이터 출력 프로그램</TITLE></HEAD>
    <BODY>
        NAME 쿠키 데이터의 값은? ${cookie.NAME.value}
    </BODY>
</HTML>
```

● EL의 initParam 객체
- initParam 객체는 웹 어플리케이션 전체의 초기화 파라미터 값을 출력할 때 사용한다.

- 웹 어플리케이션의 초기화 파라미터 값이 란 개발자가 작성한 웹 어플리케이션 전체에 적용되는 값을 의미한다.
- 예를 들면, 해당 웹 어플리케이션의 개발자 이메일 주소, 개발자의 전화번호 같은 정보들은 웹 전체에 모두 해당되는 값이 되며, 이러한 값들을 웹의 초기화 파라미터에 등록하여 반복적으로 사용할 수 있다.
- 웹 어플리케이션의 초기화 파라미터들은 /WEB-INF/web.xml에 생성해서 사용한다.
· DB의 이름을 초기화 파라미터로 다음과 같이 설정할 수 있다.

```
<context-param>
        <param-name>DB_NAME</param-name>
        <param-value>orcl</param-value>
</context-param>
```

· 개발자의 이메일 정보를 초기화 파라미터로 다음과 같이 설정할 수 있다.

```
<context-param>
        <param-name>Admin Email</param-name>
        <param-value>thisisit@octopus.com</param-value>
</context-param>
```

- 웹 어플리케이션의 초기화 파라미터를 읽어서 출력하는 JSP(ReadInitParam.jsp)

```
<%@page contentType="text/html; charset=euc-kr" %>
<HTML>
    <HEAD><TITLE>애플리케이션 초기화 파라미터 예제</TITLE></HEAD>
    <BODY>
        DB_NAME 초기화 파라미터의 값은? ${initParam.DB_NAME}<br>
         개발자의 이메일 주소 ? ${initParam["Admin Email"] }<br>
    </BODY>
</HTML>
```

● 도전과제 : web.xml을 이용하여 개발자의 이메일과 사무실 주소, 전화번호를 출력하는 JSP를 작성하세요.