07강 표현언어와 JSTL

표현 언어로 표현을 단순화 하기

- 표현언어(EL:Expression Language)는 값을 웹 페이지에 표시하는데 사용하는 태그이다.

- 기존 표현식

```
<%=변수%>
```

- 표현언어

```
${변수}
```

- 표현을 위한 3가지 방법 : 예제1

표현 언어로 표현을 단순화 하기

- 표현 언어에서 사용 가능한 데이터 타입으로 문자열, 정수, 실수, 논리형, null이 있다
- 다만 null은 공백으로 출력한다. : 예제2
- 표현언어 내부에서 사용가능 한 연산자

종류	연산자
산술	+, -, *, /(div) ,% (mod)
관계형	==(eq), !=(ne), <(lt), >(gt), <=(le), >=(ge)
조건형	a ? b : c
논리형	&& (and), || (or), ! (not)
null 검사	empty

표현 언어로 표현을 단순화 하기

- 연산자는 기호와 텍스트 둘 다 사용 가능하다

```
${3==3}
```

```
${3eq3}
```

- empty 는 객체가 비었는지 확인할 때 사용=> 비어있다면 true 반환

```
${empty input}
```

- 예제 3

- 후에 나올 jstl와 함께 사용하면 보다 가독성 높은 코드를 작성할 수 있다

표현 언어로 요청 파라미터 처리하기

- 요청 처리는 request.getParameter()를 사용한다.
- 다만 표현언어에서는 param객체를 사용한다.

내장 객체	설명
param	JSP의 내장 객체인 request의 getParameter()와 동일한 역할인 파라미터의 값을 알려준다
paramValues	동일한 이름으로 전달되는 파라미터 값들을 배열 형태로 얻어오는 데 사용하는 request의 getParameterValues()와 동일한 역할을 한다.

- param객체는 .또는 []로 사용자의 입력값을 가져온다.

- 예제 4

표현 언어로 요청 파라미터 처리하기

- 표현 언어에서 null처리는 공백으로 사용된다.
- 또한 기존 Java에서 객체 비교를 위해선 equals메소드를 사용해야 하는 반면 표현언어로는 ==으로 객체에 저장된 값을 비교할 수 있다.
- 그리고 표현언어는 문자열을 숫자로 변환할 필요도 없다.

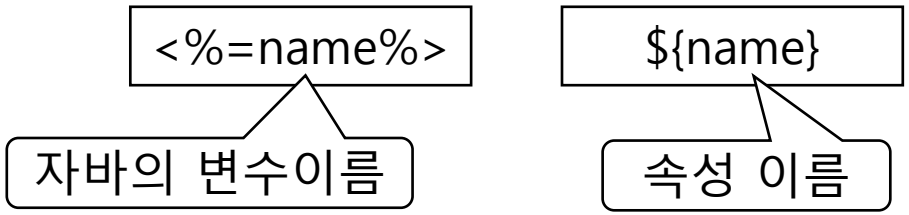
- 예제 5

표현 언어로 내장 객체에 접근 하기

- JSP에서 웹 애플리케이션을 구현하는 데 필요한 정보를 JSP의 내장 객체에 속성값으로 저장해서 사용했다.
- 각 속성에 저장된 값을 표현언어에서는 다음과 같은 형태로 사용할 수 있다.

카테고리	내장 객체	설명
범위	pageScope	page 기본 객체에 저장된 속성의<속성,값> 매핑을 저장한 Map객체
	requestScope	request기본 객체에 저장된 속성의<속성,값> 매핑을 저장한 Map객체
	sessionScope	session 기본 객체에 저장된 속성의<속성,값> 매핑을 저장한 Map객체
	applicationScope	application 기본 객체에 저장된 속성의<속성,값> 매핑을 저장한 Map객체

- 표현식에서 작성된 이름은 자바의 변수로 인식하지만 표현언어에서는 속성의 이름으로 인식



표현 언어로 내장 객체에 접근 하기

- JSP 내장 객체에 정보를 주고 받기 위해서는 다음 메소드를 사용한다.

메소드	설명
setAttribute(name,value)	주어진 이름(name)에 값(value)을 설정한다
getAttribute(name)	주어진 이름(name)에 설정된 값을 얻어온다
getAttributeNames()	현재 객체에 관련된 모든 속성의 이름을 얻어온다
removeAttribute(name)	주어진 이름(name)에 설정된 값(value)을 제거한다

- 서블릿에서 값을 저장해 전달하고 JSP에서 해당 값을 가져와서 출력하는 예제를 살펴 본다.
- 예제6

- JSP 만으로 웹 프로그래밍을 구현하는 방법을 모델1 이라고한다.
- 비즈니스 로직은 서블릿에서 전담하고 JSP는 결과 출력에 집중하는 방식을 모델 2라고 한다.

표현 언어로 내장 객체에 접근 하기

- JSP 각 내장 객체를 표현언어에서는 각각 어떻게 접근하는 지 표를 통해 알아보자

속성	JSP 내장객체	표현 언어의 내장객체	서블릿 클래스
page 속성	pageContext	pageScope	javax.servlet.jsp.jspContext 클래스
request 속성	request	requestScope	javax.servlet.ServletRequest인터페이스
session 속성	session	sessionScope	javax.servlet.http.HttpSession인터페이스
application 속성	application	applicationScope	javax.servlet.ServletContext인터페이스

표현 언어로 내장 객체에 접근 하기

- page 객체에 저장된 값을 얻어오는 법

자바 코드	표현 언어
pageContext.getAttribute("num1");	${pageScope.num1}

- request 객체에 저장된 값을 얻어오는 법

자바 코드	표현 언어
request.getAttribute("num1");	${requestScope.num1}

- session 객체에 저장된 값을 얻어오는 법

자바 코드	표현 언어
session.getAttribute("num1");	${sessionScope.num1}

- application 객체에 저장된 값을 얻어오는 법

자바 코드	표현 언어
application.getAttribute("num1");	${applicationScope.num1}

표현 언어로 내장 객체에 접근 하기

- 다만 표현언어에서 내장 객체에 데이터를 접근할 때 어느 객체에 접근할지 생략할 수있다.

${num1} 형태로 사용 가능하다.

- 이때 num1 속성의 값을 가져올 때 어느 내장 객체인지 알 수 없으므로 표현언어에서 사용할 때는 다음 순서로 자동으로 검색해서 해당 속성이 있을 때 가져온다.

```
pageScope -> requestScope -> sessionScope -> applicationScope
```

- 예제 7

표현 언어로 자바 빈에 접근 하기

- 자바 빈에 저장된 객체의 필드에 저장된 값을 사용하는 법

```
${자바빈즈.프로퍼티이름}
```

```
${자바빈즈["프로퍼티 이름"]}
```

- 예제 8

JSTL

- JSTL이란 JSP Standard Tag Library의 약어로 JSP에 사용가능한 표준 태그 라이브러리이다.
- 기존 JSP의 스크립트릿에서 코드를 보다 간결하고 가독성 높게 처리가 가능해진다.
- 예

```
--- 기존 스크립트릿 코드---
<%
if(request.getParameter("color").equals("1")){
%>
<span style="color:red;">빨강</span>
<%
}else if(request.getParameter("color").equals("2")){
%>
<span style="color:green;">초록</span>
<%
}else if(request.getParameter("color").equals("3")){
%>
<span style="color:blue;">파랑</span>
<%
}
%>
```

```
--- JSTL 적용 코드---
<c:if test="${param.color == 1}">
    <span style="color:red;">빨강</span>
</c:if>
<c:if test="${param.color == 2}">
    <span style="color:green;">초록</span>
</c:if>
<c:if test="${param.color == 3}">
    <span style="color:blue;">파랑</span>
</c:if>
```

JSTL

- JSP는 스크립트릿과 자바코드가 한데 어우러져 복잡한 구조로 되어있다. 이것을 보다 간결하게 사용하기 위해서 자신만의 태그를 추가 할 수 있는데 이런 태그를 커스텀 태그라고 한다.

- 이런 커스텀 태그를 모아서 배포하면 이를 커스텀 태그 라이브러리라고 한다

- 다만 이런 커스텀 태그 라이브러리는 말 그대로 개발자 개개인마다 다르기 때문에 이것을 표준화 한 것이 JSTL이다.

JSTL

- JSTL에서 제공하는 기능
  - 간단한 프로그램 로직의 구현가능(변수 선언, 조건(if),반복(for)등에 해당하는 로직)
  - 다른 JSP 페이지 호출
  - 날짜, 시간, 숫자의 포맷
  - JSP 페이지 하나를 가지고 여러 가지 언어의 웹 페이지 생성
  - 데이터 베이스로의 입력, 수정, 삭제, 조회
  - XML 문서의 처리
  - 문자열을 처리하는 함수 호출

- JSTL은 크게 core, format, xml, sql, functions 5가지 커스텀 태그로 나누어서 제공한다.

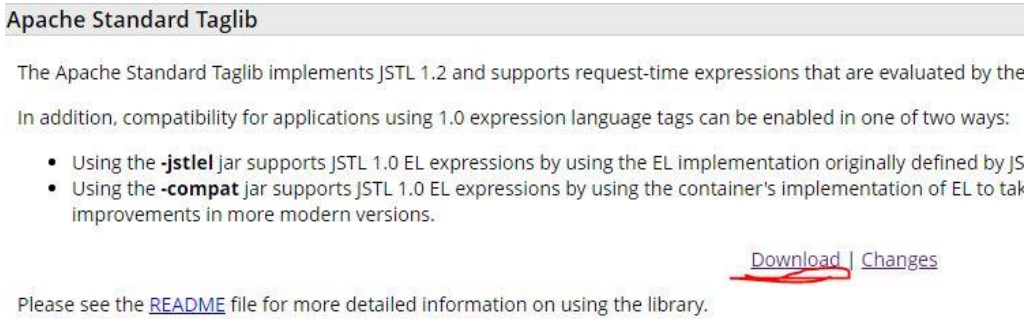
커스텀 태그	설명
기본 기능(core)	일반적인 프로그램이 제공하는 것과 유사한 기능을 제공
형식화(format)	숫자, 날짜, 시간을 포맷팅하는 기능
데이터베이스(sql)	데이터 베이스의 데이터를 입력, 수정, 삭제, 조회하는 기능
XML 처리(xml)	XML 문서를 처리할 때 필요한 기능
함수 처리(functions)	문자열을 처리하는 함수를 제공

JSTL

- JSTL을 설치하기
- 아파치 톰캣 사이트에 들어간다. (https://tomcat.apache.org/)

- 사이드 탭에서 taglib를 클릭한다.

- 그후 화면 중앙에 download 를 클릭한다.



JSTL

- 하단의 파일 3개를 받아서 저장한다.

Jar Files

- Binary README
- Impl:
  - taglibs-standard-impl-1.2.5.jar (pgp, sha512)
- Spec:
  - taglibs-standard-spec-1.2.5.jar (pgp, sha512)
- EL:
  - taglibs-standard-jstlel-1.2.5.jar (pgp, sha512)
- Compat:
  - taglibs-standard-compat-1.2.5.jar (pgp, sha512)

- 웹 애플리케이션 프로젝트에 해당 위치에 복사한다.

JSTL

- 사용할 때는 아래에 해당 지시자를 상단에 붙이고 사용한다..

```
CORE LIBRARY
  <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

XML LIBRARY
  <%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>

FMT LIBRARY
  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>

SQL LIBRARY
  <%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql" %>

FUNCTIONS LIBRARY
  <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
```

- 여기서 prefix가 태그에 사용할 접두어이다. 예제 9

- mavan repository에도 있어서 메이븐 프로젝트시 다운이 가능하다.

JSTL - core태그

- 5가지의 태그 분류중에서 가장 많이 사용되는 태그는 core태그이다.
- 기본 접두어는 c를 사용한다.

태그	설명
<c:set>	변수에 값을 설정한다.
<c:remove>	변수에 설정된 값을 제거한다.
<c:if>	조건에 따라 처리를 달리 할 때 사용한다.
<c:choose>	여러 조건에 따라 처리를 달리 할 때 사용한다.
<c:forEach>	반복 처리를 위해서 사용한다.
<c:forTokens>	구분자로 분리된 각각의 토큰을 처리할 때 사용한다.

JSTL - core태그

- 5가지의 태그 분류중에서 가장 많이 사용되는 태그는 core태그이다.
- 기본 접두어는 c를 사용한다.

태그	설명
<c:import>	외부의 자원을 url을 지정하여 가져다 사용한다.
<c:redirect>	지정한 경로로 이동한다.
<c:url>	url을 재 작성한다.
<c:out>	데이터를 출력할 때 사용하는 태그로 표현식인<%= %>을 대체할 수 있다
<c:catch>	예외 처리에 사용한다.

JSTL - core태그

- core 태그를 사용하기 위해서 지시자를 등록해야 한다.

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
```

JSTL - core태그 – 변수 제어

<c:set> : 변수에 값을 저장할 때 사용

속성	설명
var	변수 이름을 String형으로 지정한다.
value	변수에 저장할 값을 지정한다.
scope	변수가 효력을 발위할 영역으로 생략될 경우 기본 값은 page이다.

```
<c:set var="msg2" value="Hello2" scope="request" />
```

변수 이름 / 저장할 값 / 저장 영역

```
request.setAttribute("msg2","Hello2")
```

다음과 같은 형태도 가능하다.

```
<c:set var="msg2" scope="request" >
        Hello2
</c:set>
```

JSTL - core태그 – 변수 제어

<c:set> : 변수에 값을 저장할 때 사용

- 액션태그의 setProperty도 대체가 가능하다.

```
<jsp:setProperty name="자바빈 객체" property="프로퍼티이름" value="저장할 값" />
```

```
<c:set target="${자바빈 객체}"  property="프로퍼티 이름" value="저장할 값" />
```

- 다음과 같이 변수 선언후 산술연산이나 비교연산등도 가능하다.

```
<c:set var="변수" value="${10>5}" />
```

JSTL - core태그 – 변수 제어

<c:remove> : 변수를 삭제할 때 사용

```
<c:remove var="변수이름" scope="범위"/>
```

- 예제 10

JSTL - core태그 – 흐름 제어용

기존 JSP에서 조건,반복등을 사용하면 전체적인 코드가 복잡해서 가독성이 매우 떨어진다.
그래서 이런 불편함을 줄이고자 if, choose, forEach 태그들이 나왔다

<c:if> : 조건문

- c:if문은 if문과 비슷한 기능을 제공한다. 단, if~else문은 제공하지 않는다.

```
<c:if test="조건식">
   조건이 참일 경우 실행할 영역
</c:if>
```

- 예제 11

JSTL - core태그 – 흐름 제어용

<c:choose> : 조건문

- c:if문의 경우 else기능을 제공하지 않는다.

그러므로 else 기능을 하기 위해서는 c:if를 여러 개 나열할 수밖에 없는데

이때 choose를 사용하면 여러 조건을 보다 간결하게 처리할 수 있다

```
<c:choose>
    <c:when test="조건식1">  조건식1이 참일 경우 실행할 영역 </c:when>
    <c:when test="조건식2">  조건식2이 참일 경우 실행할 영역 </c:when>
    <c:when test="조건식3">  조건식3이 참일 경우 실행할 영역 </c:when>
    <c:otherwise> 모든 조건이 만족하지 않을 때 실행 영역 </c:otherwise>
</c:choose>
```

- 예제 12

JSTL - core태그 – 흐름 제어용

<c:forEach> : 반복문

- c: forEach 문의 경우 배열, 컬렉션등의 집합체에 저장된 값을 순차적으로 처리할 때 사용하는 반복문이다. (향상된 for문과 유사)

```
<c:forEach  var="원소 하나를 저장할 변수" items="반복처리할 집합체">
    반복할 코드
</c: forEach >
```

- 추가적인 프로퍼티

프로퍼티	설명
varStatus	각 항목의 Index를 사용해야 할 때 반복 상태등을 저장하는 변수
index	items에 지정한 집합체의 현재 반복중인 항목의 index를 알려준다.(0부터 시작)
count	반복을 할 때 몇 번째 반복 중인지 알려준다 (1부터 시작)
first	현재 반복이 처음인지 여부를 알려준다(boolean타입)
last	현재의 반복이 마지막인지 여부를 알려준다.(boolean타입)

- 예제 13

JSTL - core태그 – 흐름 제어용

<c:forEach> : 반복문

- c: forEach 문의 경우 일반적인 사용법은 집합체에 저장된 값을 순차적으로 꺼내 올 때 사용하지만 단독으로 숫자를 이용해서 횟수 반복으로도 사용가능하다.

이때 사용하는 프로퍼티는 다음과 같다.

프로퍼티	설명
begin	반복에 사용할 것으로 첫번째 항목의 Index값
end	반복에 사용할 것으로 마지막 항목의 Index값
step	증가 값

- 예제 14

JSTL - core태그 – 흐름 제어용

<c:forTokens> : 반복문

- c: forTokens 문의 경우 문자열을 구분자로 쪼개고 각각 쪼개진 문자열 하나하나의 집합체로서 순차적으로 반복해서 사용하는 반복문이다.

```
<c:forTokens  var="토큰을 저장할 변수" items="토큰으로 나눌 문자열" delims="구분자">
    반복할 코드
</c:forTokens>
```

- 예제 15

JSTL - core태그 – 페이지 제어

- 다른 페이지를 포함하거나 이동할 때 사용하는 태그

 - c:import 문의 경우 지정된 페이지를 불러와서 변수에 저장해 두고 해당 변수를 호출할 때 해당 페이지에서 가져온 결과를 출력한다.

```
<c:import  var="저장할 변수" url="URL" scope="변수를 저장할 영역" charEncoding="UTF-8">
</c:import>
```

- 예제 16

JSTL - core태그 – 페이지 제어

- c:url 문의 경우 URL을 생성해서 적절한 위치에 사용할 수 있다

```
<c:url  var="저장할 변수" value="URL" scope="변수를 저장할 영역>
</c:url>
```

- 예제 17

JSTL - core태그 – 페이지 제어

- c:redirect 문의 경우 지정한 페이지로 이동할 때 사용한다.
- response.sendRedirect와 같다

```
<c:redirect url="URL" / >
```

- 예제 18

JSTL - core태그 – 기타

- c:out 문은 출력을 위한 태그이다.
- 표현식이나 표현언어와 동일한 역할을 하기 때문에 표현언어보다 자주 사용되지 않는다.

```
< c:out value="출력할 값" [defalut="기본값"] / >
```

JSTL - core태그 – 기타

- c:catch 문은 예외 처리을 위한 태그이다.
- 예외가 발생하면 잡아내는 역할을 한다.

```
<c:catch var="발생한 예외가 저장될 변수" >
        예외가 발생할 가능성이 있는 코드
</c:catch>
```

-예제 19

JSTL - fmt태그

- fmt태그는 포맷팅에 관련된 태그 모음이다.
- 주로 숫자, 날짜, 시간의 형식을 다루는데 사용되며 다양한 언어를 지원한다.

기능	태그	설명
숫자 날짜 형식	<fmt:formatNumber>	숫자를 양식에 맞춰서 출력한다.
	<fmt:formatDate>	날짜 정보를 담고 있는 객체를 포맷팅하여 출력할 때 사용한다.
	<fmt:parseDate>	문자열을 날짜로 파싱한다.
	<fmt:parseNumber>	문자열을 수치로 파싱한다.
	<fmt:setTimeZone>	시간대 별로 시간을 처리할 수 있는 기능을 제공한다.
	<fmt:timeZone>	시간대 별로 시간을 처리할 수 있는 기능을 제공한다.
로케일 지정	<fmt:setLocale>	국제화 태그들이 사용할 로케일을 지정한다.
	<fmt:requestEncoding>	요청 파라미터의 인코딩을 지정한다.
메시지 처리	<fmt:bundle>	태그 몸체에서 사용할 리소스 번들을 지정한다.
	<fmt:message(param)>	메시지를 출력한다.
	<fmt:setBundle>	특정 리소스 번들을 사용할 수 있도록 로딩한다.

JSTL - fmt태그

- fmt 태그를 사용하기 위해서 다음 지시자를 등록한다.

```
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
```

- 주로 숫자나 날짜 형식 지정에 사용된다.

JSTL - fmt태그- formatNumber

<fmt:formatNumber>

- 태그의 속성

속성	표현식	타입	설명
value	true	String 또는 Number	형식화할 수치 데이터
type	true	String	숫자,통화, 퍼센트중 어느 형식으로 표시할 지 지정
pattern	true	String	사용자가 지정한 형식 패턴
currencySymbol	true	String	통화 기호, 통화형식(type="currency")일 때만 적용
groupingUsed	true	boolean	콤마와 같이 단위를 구분할 때 사용하는 기호를 표시 할지의 여부를 결정한다. true이면 구분기호 사용,false면 구분기호 미사용 (기본값 true)
var	false	String	형식 출력 결과 문자열을 담는 scope에 해당하는 변수 이름
scope	false	String	var 소성에 지정한 변수가 효력을 발휘할 수 있는 영역에 지정

JSTL - fmt태그- formatNumber

<fmt:formatNumber>

- <fmt:formatNumber value="1234567.89" /> => 1,234,567.89
- <fmt:formatNumber value="1234567.89" groupingUsed="false" /> => 1234567.89
- <fmt:formatNumber value="0.5" type="percent" /> => 50%
- <fmt:formatNumber value="10000" type="currency" /> => ₩10,000
- <fmt:formatNumber value="10000" type="currency" currencySymbol="$"/> => $10,000

- 패턴지정 pattern #,0, .으로 표기
  - 0은 빈자리를 0으로 채워 표기
  - #은 빈자리를 공백으로 표기
- <fmt:formatNumber value="1234567.8912345" pattern="#,#00.0#" /> => 1,234,567.89
- <fmt:formatNumber value="1234567.8" pattern="#,#00.0#" /> => 1,234,567.8
- <fmt:formatNumber value="1234567.89" pattern=".000" /> => 1,234,567.890

JSTL - fmt태그 - formatDate

<fmt:formatDate>

- 태그의 속성

속성	표현식	타입	설명
value	true	java.util.Date	형식화될 Date와 time
type	true	String	형식화할 데이터가 시간(time), 날짜(date), 모두(both) 셋중 하나를 지정
dateStyle	true	String	미리 정의된 날짜 형식으로 default, short, medium, long, full 넷 중에 하나를 지정
timeStyle	true	String	미리 정의된 시간 형식으로 short, medium, long, full 넷중 하나 지정
pattern	true	String	사용자 지정 형식 스타일
timeZone	true	String 또는 java.util.TimeZone	형식화 시간에 나타날 타임존
var	false	String	형식 출력 결과 문자열을 담는 scope에 해당하는 변수 이름
scope	false	String	var의 scope

JSTL - fmt태그 - formatDate

<fmt:formatDate>

- <c:set vat="now" value="<%=new java.util.Date()%>" /> => 날짜 객체 생성
- <fmt:formatDate value="${now}" /> => 2020.12.31
- <fmt:formatDate value="${now}" type="time" /> => 오전 8:12:45
- <fmt:formatDate value="${now}" type="both" /> => 2020.12.31 오전 8:12:45

- <fmt:formatDate value="${now}" pattern="yyyy년 MM월 dd일 hh시 mm분 ss초" />

=> 2020년 12월 31일 8시 12시 45초

- 예제 20

JSTL - fmt태그 – setTimeZone, timeZone

<fmt: setTimeZone >

- 특정 지역 타임존을 설정하는 태그

```
<fmt: setTimeZone value="타임존" />
```

<fmt: timeZone >

- 타임존을 부분 적용하는 태그

```
<fmt:timeZone value="타임존" />
    타임존 적용 영역
</fmt:timeZone>
```

- 예제 21

JSTL - fmt태그 – setLocale

<fmt: setLocale >

- 나라마다 화폐의 종류가 다르고 날짜 표기법이 다르다. 만약 다국적 페이지를 만들고자 한다면 일일이 모두 바꿔주어야 하는데 이때 로케일의 값만 바꿔주면 자동으로 적용된다.

```
<fmt:setLocale value="언어코드_국가코드" />
```

- 한글의 경우 ko_KR을 지정하는데 이때 ko는 한글 언어코드이고, KR은 한국 국가 코드이다.

- 예제 22

JSTL - fmt태그 – requestEncoding

<fmt: requestEncoding >

- POST방식으로 넘어온 데이터의 글자가 깨지지 않도록 처리하기 위한 태그이다.
- 기존 request.setCharacterEncoding()메소드와 같은 역할을 한다.

```
<fmt:requestEncoding value="UTF-8" />
```

- 예제 23