표준 액션

강사 : 강병준

표준 액션이란?

- 액션의 종류는 크게 둘로 나뉘어 진다.
  - 표준 액션(standard action)은 JSP 페이지에서 바로 사용할 수 있다.
  - 커스텀 액션(custom action)은 별도의 라이브러리를 설치해야만 사용할 수 있다.
- 표준 액션과 커스텀 액션은 태그 안에 사용하는 접두어(prefix)가 다르기 때문에 외형상으로 쉽게 구분 할 수 있다. 표준 액션에는 모든 태그의 이름 앞에 jsp라는 접두어가 붙고, 커스텀 액션에는 그 밖의 접두어가 붙는다.

```
<jsp:include page= “sub.jsp ”/>
```

표준 액션임을 표시하는 접두어

```
<c:set var= “cnt ” value= “0 ”/>
```

커스텀 액션 중 하나임을 표시하는 접두어

JSP 페이지의 모듈화에 사용되는 표준 액션

- <jsp:include> 표준 액션의 사용 방법
  - <jsp:include>는 JSP 페이지에서 다른 웹 자원(JSP 페이지, HTML 문서 등)을 포함시키고자 할 때 사용하는 표준 액션이다.
  - 이 표준 액션에는 포함할 웹 자원의 URL을 지정하는 page 애트리뷰트를 써야 한다.

```
<jsp:include page= “Copyright.html ”/>
```

Copyright.html을 include하는 표준 액션

  - 액션 태그는 XML 문법을 따르므로 단독 태그일 경우에는 위와 같이 ‘/>’로 끝나도록 만들어야 한다.
  - JSP 페이지 안에 위와 같은 액션 태그가 있으면, 웹 컨테이너는 JSP 페이지를 처리할 때 이 태그의 위치에 Copryright.html의 내용을 대신 출력한다.

❖ <jsp:include> 표준 액션의 사용 방법

BookInfo.jsp

```
<%@page contentType= “text/html; charset=utf-8 ”%>
<HTML>
    <HEAD><TITLE>책 소개</TITLE></HEAD>
    <BODY>
        <H3>책 소개</H3>
            제목: Java 프로그래밍 <BR>
            저자: 홍길동 <BR>
            페이지수: 908 <BR><BR>
            <jsp:include page= “Copyright.html ”/>
    </BODY>
</HTML>
```

Copyright.html 문서를 include합니다.

Copyright.html

```
<font size=2>@홍길동과 춘향이는 어울리나 ?</font>
```

❖ <jsp:include> 표준 액션의 사용 방법

- 포함할 HTML 문서가 <jsp:include> 표준 액션이 속하는 JSP 페이지와 다른 디렉터리에 있으면 그에 해당하는 URL 경로명을 다음과 같이 쓰면 된다.

```
<jsp:include page= “common/Copyright.html ”/>
```

↑ 상대적인 URL 경로명입니다.

```
<jsp:include page= “/common/Copyright.html ”/>
```

↑ 슬래시(/)로 시작하는 값은 웹 애플리케이션 디렉터리를 기준으로 한 URL 경로명입니다

- <jsp:include> 표준 액션은 다른 JSP 페이지를 포함시키기 위해서도 사용될 수 있다.

```
<jsp:include page= “Date.jsp ”/>
```

← 같은 디렉터리에 있는 Date.jsp를 include하는 표준 액션

```
<jsp:include page= “/util/Date.jsp ”/>
```

← 웹 애플리케이션 디렉터리의 util 서브디렉터리에 있는 Date.jsp를 include하는 표준 액션

❖ <jsp:include> 표준 액션의 사용 방법

Winners.jsp

```
<%@page contentType= “text/html; charset=utf-8 ”%>
<HTML>
    <HEAD><TITLE>당첨자 명단</TITLE></HEAD>
    <BODY>
        <H3>당첨자 명단</H3>
        14553 연흥부 <BR>
        63563 심청 <BR>
        73992 이몽룡 <BR><BR>
        <jsp:include page= “Now.jsp ”/>
    </BODY>
</HTML>
```

Now.jsp를 include합니다

```
<%@page contentType=“text/html; charset=utf-8”%>
<%@page import=“java.util.*”%>
<%
    GregorianCalendar now = new GregorianCalendar();
    String date = String.format(“%TY년 %Tm월 %Td일” , now, now, now);
    String time = String.format(“%Tp %TR”, now, now);
%>
[현재 시각] <%= date %> <%= time %>
```

시스템 시계로부터 현재 시각을 가져다가
YY년 MM월 DD일 포맷의 날짜와 AM/PM hh:mm 포맷의 시각으로 편집합니다.

includeTestForm.jsp

```
<%@ page language="java" contentType="text/html; charset=utf-8 "
    pageEncoding="EUC-KR"%>

<html>
<head>
<title>include 액션태그</title>
</head>
<body>
  <h2>include 액션태그</h2>

    <form method="post" action="includeTest.jsp">
        이름 : <input type="text" name="name"><br>
        페이지이름 : <input type="text" name="pageName"
value="includedTest"><br>
        <input type="submit" value="입력완료">
    </form>
</body>
</html>
```

includeTest.jsp

```
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="EUC-KR"%>
<% request.setCharacterEncoding("euc-kr");%>
<%
   String pageName = request.getParameter("pageName");
   pageName += ".jsp";
%>
   포함하는 페이지 includeTest.jsp 입니다.<br>
   <hr>
   <jsp:include page="<%=pageName%>"  flush="false"/>
    includeTest.jsp의 나머지 내용입니다.
```

includedTest.jsp

```
   <%@ page language="java" contentType="text/html; charset=EUC-KR"
       pageEncoding="EUC-KR"%>
   <%
   String name = request.getParameter("name");
   %>
   포함되는 페이지 includedTest.jsp 입니다.<p>
   <b><%=name%></b> 님 오셨구려..
   <hr>
```

includeTest2.jsp

```
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="EUC-KR"%>
<% request.setCharacterEncoding("utf-8");%>
<%   String name = "Eunnogi";
   String pageName = "includedTest2.jsp";
%>
   포함하는 페이지 includeTest2.jsp 입니다.<br>
   포함되는 페이지에 파라미터 값을 전달합니다.<br>
   <hr>   <jsp:include page="<%=pageName%>"  flush="false">
        <jsp:param  name="name" value="<%=name %>"/>
        <jsp:param  name="pageName" value="<%=pageName %>"/>
   </jsp:include>
    includeTest2.jsp의 나머지 내용입니다.
```

includeTest2.jsp

```
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="utf-8"%>
<% String name = request.getParameter("name");
    String pageName = request.getParameter("pageName");
%>
    파라미터 값을 전달받아 실행되는 <br>
    포함되는 페이지 <%=pageName %> 입니다.<br>
    <b><%=name%></b> 님 오셨구려..     <hr>
```

❖ <jsp:forward> 표준 액션의 사용 방법

- <jsp:forward>는 JSP 페이지에서 다른 JSP 페이지로 제어를 넘기고자 할 때 사용하는 표준 액션이다.
- <jsp:include>와 마찬가지로 page 애트리뷰트를 이용해서 해당 JSP 페이지의 URL을 지정해야 한다.

```
<jsp:forward page= “Next.jsp ”/>
```

Next.jsp로 실행의 제어를 넘기는 표준 액션

❖ <jsp:forward> 표준 액션의 사용 방법

Hundred.jsp

```
<%
    int sum = 0;
    for (int cnt = 1; cnt <= 100; cnt++)
        sum += cnt;
    request.setAttribute( “RESULT ”, new Integer(sum));
%>
<jsp:forward page= “HundredResult.jsp ” />
```

실행의 제어를 넘긴다.

```
<%@page contentType= “text/html; charset=utf-8 ”%>
<HTML>
    <HEAD><TITLE>1부터 100까지의 합</TITLE></HEAD>
    <BODY>
        1부터 100까지 더한 결과는? ${RESULT}
    </BODY>
</HTML>
```

앞 장에서 배운 익스프레션 언어를 이용해서 결과를 출력한다

forwardTestForm.jsp

```
<%@ page language="java" contentType="text/html; charset=utf-8 "
pageEncoding="utf-8"%>
<html>
<head><title>forward 액션태그</title></head>
<body>
   <h2>forward 액션태그</h2>
    <form method="post" action="forwardTest.jsp">
        아이디 : <input type="text" name="id"><br>
        취미 :
        <select name="hobby">
            <option value="잠자기">잠자기</option>
            <option value="만화보기">만화보기</option>
            <option value="TV시청">TV시청</option>
        </select><br>
        <input type="submit" value="입력완료">
    </form>
</body>
</html>
```

forwardTest.jsp

```
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="utf-8"%>
<% request.setCharacterEncoding("utf-8");%>
  포워딩하는  페이지 forwardTest.jsp로 절대 표시되지 않습니다.<br>
  <jsp:forward page="forwardToTest.jsp" />
   forwardTest.jsp페이지의 나머지 부분으로 표시도 실행도 되지 않습니다.
```

forwardToTest.jsp

```
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="utf-8"%>
<%
   String id = request.getParameter("id");
   String hobby = request.getParameter("hobby");
%>
포워딩되는 페이지 forwardToTest.jsp 입니다.<br>
<b><%=id%></b> 님의 <br>
취미는 <b><%=hobby %></b> 입니다.
```

forwardTest2.jsp

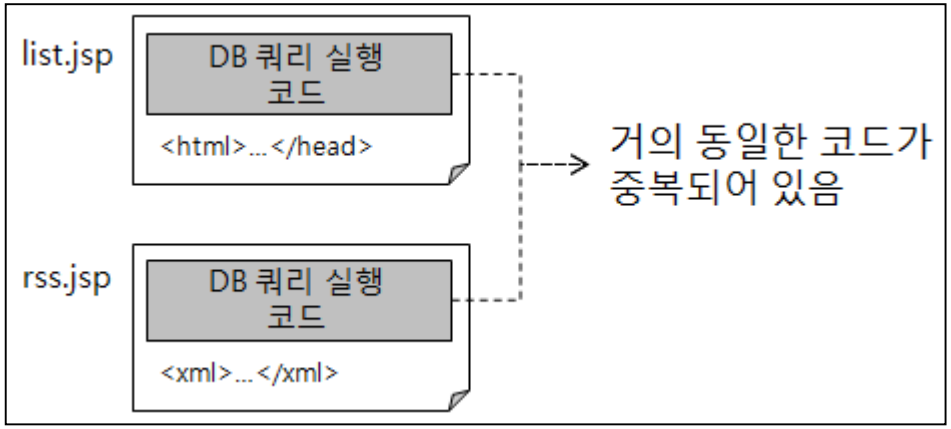
```
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="utf-8"%>
<% request.setCharacterEncoding("utf-8");%>
<%
   String id = "Eunnogi";
   String hobby = "TV시청";
%>
   포워딩하는  페이지 forwardTest2.jsp입니다.<br>
   <jsp:forward page="forwardToTest2.jsp">
       <jsp:param  name="id" value="<%=id%>"/>
        <jsp:param  name="hobby" value="<%=hobby%>"/>
   </jsp:forward>
```

forwardToTest2.jsp

```
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="utf-8"%>
<%
   String id = request.getParameter("id");
   String hobby = request.getParameter("hobby");
%>
포워딩되는 페이지 forwardToTest2.jsp 입니다.<br>
<b><%=id%></b> 님의 <br>
취미는 <b><%=hobby %></b> 입니다.
```

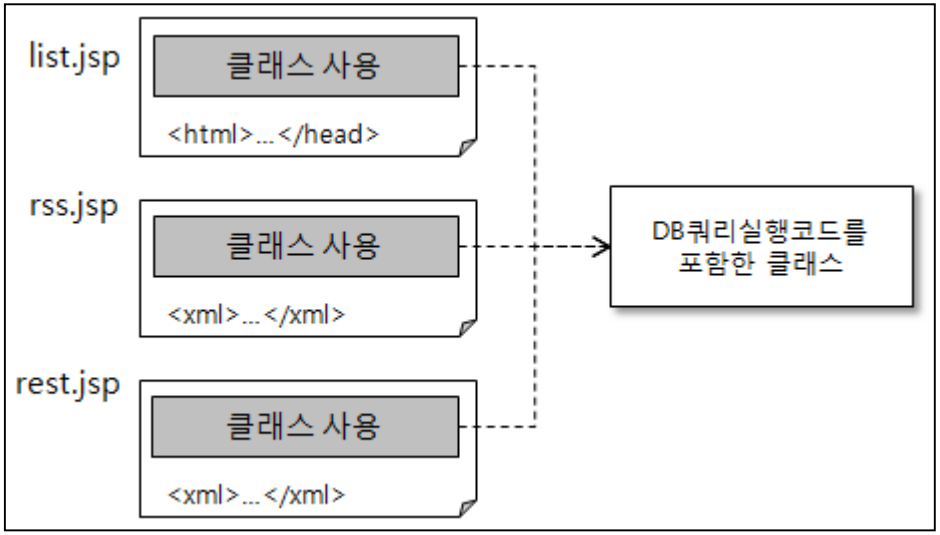
WebApplication 구조

- JSP로만 구성된 웹 애플리케이션



- 동일한 로직을 수행하는 코드가 중복될 가능성이 높음
- 문제점
  - 기능 변경 발생 시 여러 코드에 동일한 수정 반영해 주어야 함
  - 누락될 가능성 발생 → 버그 발생 가능성 높음

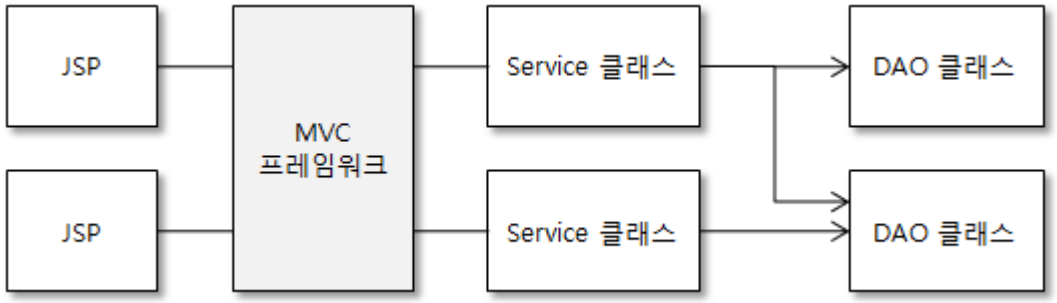
❖ 클래스를 이용한 중복 코드 제거



❖ 클래스를 이용해서 중복된 코드를 한 곳으로 분리

❖ 화면 요청 처리하는 JSP와 실제 로직을 수행하는 클래스로 분리하는 것이 일반적인 구성

- 일반적인 웹 애플리케이션 구조



- Service 클래스 - 사용자의 요청을 처리하는 기능을 제공하는 클래스로 DAO 클래스를 이용해서 DB 연동을 처리
- DAO 클래스 - DB와 관련된 CRUD 작업을 처리(SQL 쿼리를 실행)
- JSP(뷰) - Service 클래스가 실행한 결과를 화면에 출력해주거나 Service가 기능을 수행하는 데 필요한 데이터를 전달
- MVC 프레임워크 - 사용자의 요청을 Service에 전달하고 Service의 실행 결과를 JSP와 같은 뷰에 전달을 수행하는 것으로 스프링 MVC나 스트러츠와 같은 프레임워크가 MVC 프레임워크에 해당하며 서블릿으로 직접 구현 가능
- Service 클래스나 Dao 클래스의 경우 특별한 경우가 아니면 동시에 여러 개의 객체를 생성해서 사용 하는 경우가 없기 때문에 Singleton 패턴을 이용하는 것이 좋습니다.

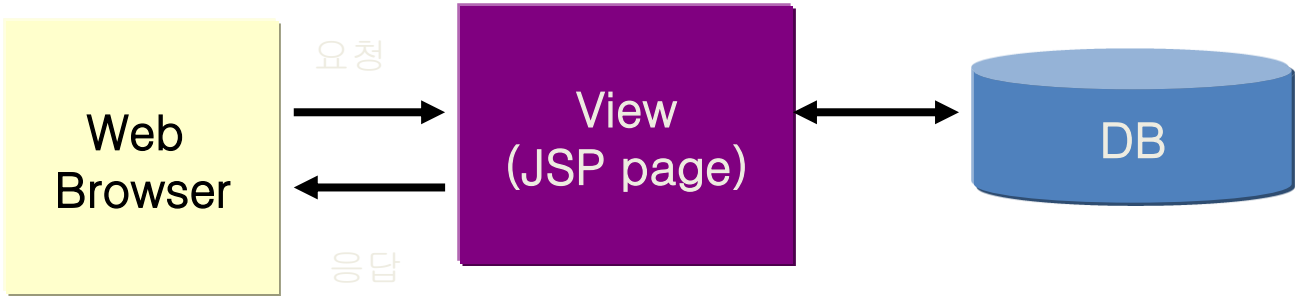
자바빈(JavaBean)의 개요

- 현재까지 작성해온 JSP페이지의 문제점
  - 첫 번째로 JSP 페이지에 화면표출부분과 로직들이 혼재함으로 해서 JSP 페이지를 이해하기 어려워진다는 점
    - 디자이너와 협업이 어려워짐
  - 두 번째로는 JSP 페이지에 화면표출부분과 로직들이 혼재한 형태의 코드는 재사용하기가 어려움
    - 반복적인 일을 피하기 위해서는 JSP 페이지 내에 있는 반복적인 코드를 따로 작성하여 재사용할 필요가 있음
  - JSP페이지와 로직의 분리해서 로직을 모듈화하는 작업이 필요 => 자바빈의 사용이 필요

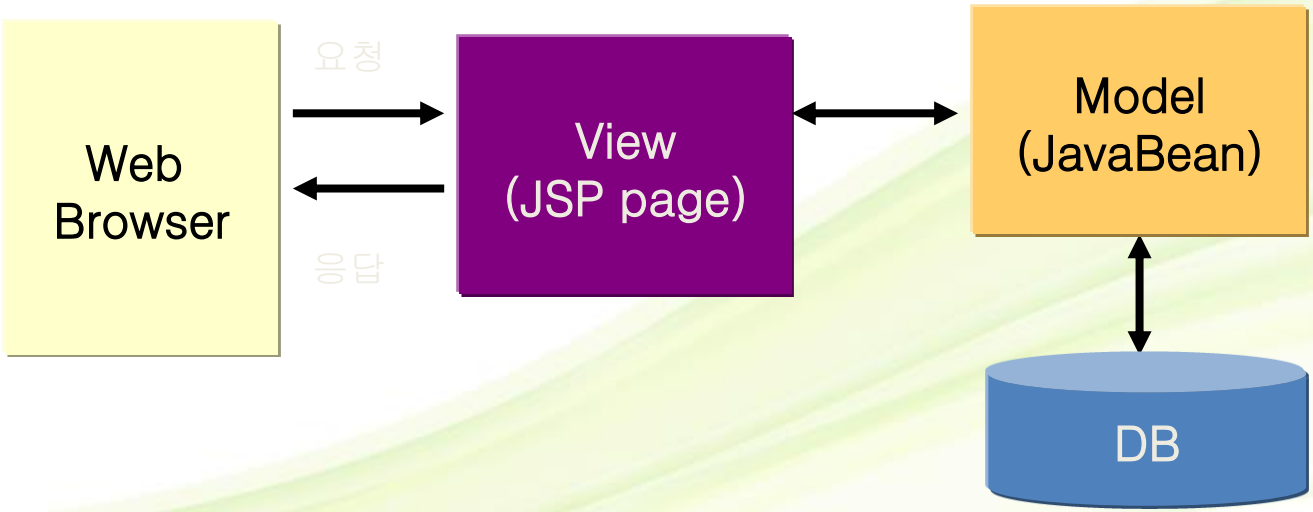
- 자바빈 사용하는 목적
  - JSP 페이지의 로직 부분을 분리해서 코드를 재사용함으로 프로그램의 효율을 높이는 것
  - 현재 모든 프로그래밍에서 모듈화(컴포넌트(component)화)가 대세
    - 프로그램의 모듈화는 한 번 잘 작성해 놓은 코드를 재사용하므로 프로그램의 작성기간이 단축되고, 이미 실시스템에 올렸던 코드를 사용하므로 코드의 안정성이 보장되어 유지 보수에도 좋음

자바빈(JavaBean)의 개요)

❖ 우리가 지금까지 작성한 JSP페이지의 형태



❖ 이제부터 우리가 작성할 형태



❖ 자바빈 작성

- **자바빈은 자바의 클래스이므로 자바의 클래스를 만드는 것과 같은 규칙을 갖는다**

```
순서
1. package 패키지명; //없으면 생략가능
2. import 패키지명을 포함한 클래스의 풀네임; //없으면 생략가능
3. class 클래스명{
   }
```

- **자바의 클래스를 만들 때는 포함될 패키지명(package문), 해당 클래스를 생성할 때 필요한 패키지명을 포함한 클래스명(import문) 이 필요. 생략이 가능하다. 그러나 class문은 생략할 수 없다. 자바파일에는 1개이상의 클래스를 포함해야함**

자바빈(JavaBean) 만들기

- 자바빈의 클래스를 선언
  - 접근제어자 [키워드] class 클래스명{ }
    - 접근 제어자는 public, private, default(접근제어자가 없는 형태)가 올 수 있는데, 자바 빈을 작성할 때는 접근제어의 강도가 가장 약한 public을 주로 사용
    - 웹에서는 불특정 다수의 접근을 허용해야 하기 때문에 누구나 접근할 수 있는 public을 사용. 키워드는 자바빈에서는 사용안함.
  - 자바에서 클래스명은 첫 글자는 대문자로 시작하고 나머지는 소문자를 사용한다. 또한 단어가 여러개 합쳐질 때는 시작되는 단어의 시작은 대문자로 시작한다.
    - public class UtilClass{ }

- 자바빈의 프로퍼티(멤버변수)를 선언
  - 프로퍼티(property)는 값을 저장하기 위한 필드로 접근제어자를 private로 선언해서 작성
    - private String userid;
- 자바빈의 메소드의 선언
  - 프로퍼티에 접근하기위한 getXxx(), setXxx()메소드는 접근제어자를 public로 선언해서 작성

```
public void setId(String id){       this. id=id;   }
public String getId(){       return id;    }
```

- 데이터 저장소의 역할을 하는 프로퍼티에 값을 저장할때
  - setXxx()메소드를 사용
- 저장된 값을 사용할때
  - getXxx()메소드를 사용
- 이때 Xxx는 프로퍼티명으로 첫글자는 대문자로 작성. 하나의 프로퍼티당 하나의 setXxx()메소드와 getXxx()메소드가 존재.

- **setXxx()메소드 작성방법**
  - 프로퍼티에 값을 저장해야 하므로 파라미터로부터 값을 받아오는 setId(String id)와 같은 형태
  - 값을 저장만하는 메소드이므로 리턴타입이 void.
  - this.id=id;
    - 넘어온 id파라미터의 값을 id프로퍼티에 저장하는 부분으로 this.id가 프로퍼티이다. 프로퍼티명과 파라미터명이 같으면 프로그램에 혼란을 주기 때문에 프로퍼티명앞에는 this가 붙는다.
      - this는 자기 자신의 클래스를 가리키는 레퍼런스

```
public void setId(String id){
    this.id=id;
}
```

```
public void getId(){
    return id;
}
```

- **getXxx()메소드 작성방법**
  - 저장된 프로퍼티명을 사용하는 메소드이므로 getId()와 같이 파라미터가 필요없다.
  - 저장된 값을 사용해야 하기 때문에 반드시 리턴타입을 기술.
    - String getId()는 리턴타입이 String이라는 것.
  - void이외의 리턴타입이 기술되면 해당 메소드의 마지막에 return문을 반드시 기술해야 한다. 기술안하면 에러가 발생.

자바빈과 <jsp:useBean>액션태그의 연동

- **자바빈을 사용하기 위한 3가지의 액션태그**
  - 자바빈 객체를 생성하기 위한 <jsp:useBean>액션태그
  - 자바빈 객체의 프로퍼티 값을 저장하기 위해 사용되는 <jsp:setProperty>액션태그
  - 자바빈 객체에서 저장된 프로퍼티값을 사용하기 위해 사용되는 <jsp:getProperty>액션태그

자바 빈 관련 액션태그	내용
<jsp:useBean id="..." class="..." scope="..."/>	자바빈 객체를 생성.
<jsp:setProperty name="..." property="..." value="..."/>	생성된 자바빈 객체에 프로퍼티 값을 저장.
<jsp:getProperty name="..." property="..." />	생성된 자바빈 객체에서 자징된 프로퍼티값을 가져옴.(사용함)

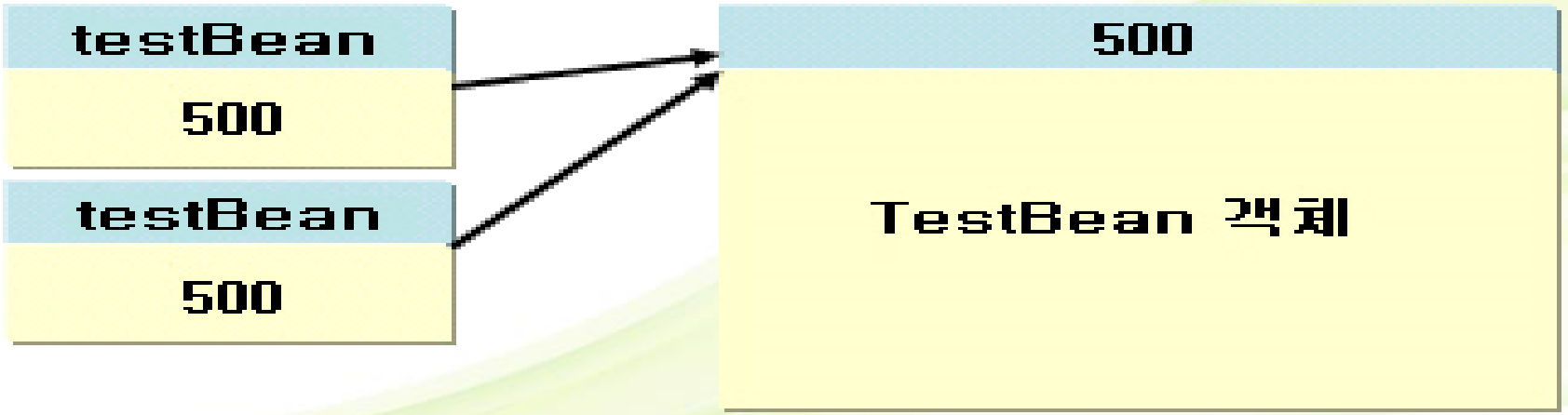
- <jsp:useBean>액션태그
  - <jsp:useBean>액션태그는 자바빈 객체를 생성
  - 사용하는 방법
    - <jsp:useBean id= "빈 이름" class="자바빈 클래스 이름" scope="범위" />
      - id속성은 생성될 자바빈 객체(인스턴스)의 이름을 명시하는 곳.
      - 필수 속성으로 생략이 불가능. class속성은 객체가 생성될 자바빈 클래스명을 기술하는 곳. 패키지명을 포함한 자바클래스의 풀네임을 기술. 필수 속성으로 생략이 불가능.
      - scope속성은 자바빈 객체의 유효범위로 자바빈 객체가 공유되는 범위를 지정. scope속성값으로는 page, request, session, application을 가지며 scope속성 생략시 기본값은 page.
  - <jsp:useBean id= "testBean" class="ch10.bean.TestBean" scope="page" />
    - TestBean클래스의 자바빈 객체를 생성하는데 이때 생성되는 인스턴스명(레퍼런스명)이 testBean 이다. 향후 TestBean 클래스의 멤버변수(프로퍼티)나 메소드에 접근하려면 testBean 레퍼런스를 사용.
  - <jsp:useBean>액션태그에서 id속성값에 지정한 이름이 이미 존재하는 경우
    - 자바빈 객체를 새로 생성하는 것이 아니라 기존에 생성된 객체를 그대로 사용. 이때 id속성값, class속성값, scope속성값이 모두 같아야 같은 객체

❖ <jsp:useBean>액션태그

```
<jsp:useBean id= "testBean" class="ch10.bean.TestBean"
scope="page" />

…
중략
..

<jsp:useBean id= "testBean" class="ch10.bean.TestBean"
scope="page" />
```



<jsp:setProperty>액션태그

- <jsp:setProperty>액션태그는 자바빈 객체의 프로퍼티 값을 저장하기 위해 사용된다.
- 사용하는 방법은 다음과 같다.

  - <jsp:setProperty>액션태그의 속성에 대한 설명은 다음과 같다.
  - name속성은 자바빈 객체의 이름을 명시하는 곳이다. 필수 속성으로 생략이 불가능하다.

```
<jsp:setProperty  name= "빈 이름"  property="프로퍼티 이름" value="프로퍼티에 저장할 값 " />
```

  - property속성은 프로퍼티명을 기술하는 곳이다. 필수 속성으로 생략이 불가능하다.
  - value속성은 프로퍼티에 저장할 값을 기술하는 곳이다. 생략 가능하다.

<jsp:getProperty>액션태그

- <jsp:getProperty>액션태그는 자바빈 객체에서 저장된 프로퍼티값을 사용하기 위해 사용된다.
- 사용하는 방법은 다음과 같다.

  - name속성은 자바빈 객체의 이름을 명시하는 곳이다. 필수 속성으로 생략이 불가능하다.
  - property속성은 프로퍼티명을 기술하는 곳이다. 필수 속성으로 생략이 불가능하다.

```
<jsp:getProperty  name= "빈 이름"  property="프로퍼티 이름"  />
```

자바빈의 호출에 사용되는 표준 액션

PersonalInfo.java

```
package mall;
public class PersonalInfo {
        private String name; // 이름
        private char gender; // 성별
        private int age;          // 나이
        public void setName(String name)  {          this.name = name;       }
        public String getName()                    {         return name;               }
        public void setGender(char gender) {         this.gender = gender;   }
        public char getGender() {             return gender;       }
        public void setAge(int age) {
            this.age = age;
        }
        public int getAge() {
            return age;
        }
}
```

- 이 클래스에 있는 get-메서드와 set-메서드는 자바빈의 내부 데이터를 읽고 쓰는 기능을 제공한다. 이런 메서드를 통해 읽고 쓸 수 있는 데이터를 자바빈의 프로퍼티(property)라고 한다.

❖ 자바빈 관련 표준 액션의 기초 사용 방법

- 자바 프로그램에서 클래스를 사용하기 위해서는 클래스의 객체를 만들어야 하며, JavaBean 클래스의 경우도 마찬가지이다.

```
ProductInfo obj = new ProductInfo();
```

변수 이름 ↑ — ↑ 클래스 이름

- <jsp:useBean> 표준 액션을 이용하면 자바 코드를 작성하지 않고도 자바빈 객체를 만들 수 있다.
- <jsp:useBean>에는 기본적으로 class와 id라는 두 개의 애트리뷰트를 써야 한다.

```
<jsp:useBean id= “obj ” class= “mall.PersonalInfo ” />
```

변수 이름 ↑ — ↑ 클래스 이름

❖ 자바빈 관련 표준 액션의 기초 사용 방법

- 자바빈 객체를 만든 다음에는 set-메서드를 이용해서 객체의 프로퍼티 값을 설정할 수 있다.

```
obj.setAge( "27 ");
```

변수 이름 메서드 이름 프로퍼티 값

- <jsp:setProperty> 표준 액션을 이용하면 set-메서드를 직접 호출하지 않고도 프로퍼티 값을 설정할 수 있다.
- 이 액션에는 name, property, value라는 세 개의 애트리뷰트를 써야 한다. 이 중 name 애트리뷰트에는 자바빈 객체가 들어 있는 변수 이름을 지정해야 하고, property와 value 애트리뷰트에는 각각 프로퍼티의 이름과 값을 지정해야 한다.

```
<jsp:setProperty name= "obj " property= "age " value= "27 " />
```

변수 이름 프로퍼티 이름 프로퍼티 값

❖ 자바빈 관련 표준 액션의 기초 사용 방법

- 자바빈 객체의 프로퍼티 값을 가져올려면 다음과 같이 get-메서드를 이용하면 된다.

```
int age = obj.getAge();
```

변수 이름 — obj

메서드 이름 — getAge()

- <jsp:getProperty> 표준 액션을 사용하면 get-메서드를 직접 호출하지 않고도 프로퍼티 값을 가져올할 수 있다.
- 이 액션에는 name과 property라는 두 개의 애트리뷰트를 써야 한다. name 애트리뷰트에는 자바빈 객체가 들어 있는 변수의 이름을 써야 하고, property 애트리뷰트에는 프로퍼티의 이름을 써야 한다.

```
<jsp:getProperty name= “obj ” property= “age ” />
```

변수 이름 — obj

프로퍼티 이름 — age

❖ 자바빈 관련 표준 액션의 기초 사용 방법

CustomerInfo.jsp

```
<%@page contentType= “text/html; charset=utf-8 ”%>
<HTML>
    <HEAD><TITLE>회원 정보</TITLE></HEAD>
    <BODY>
        <jsp:useBean class= “mall.PersonalInfo ” id= “pinfo ” />
        <jsp:setProperty name= “pinfo ” property= “name ” value= “김연희 ” />
        <jsp:setProperty name= “pinfo ” property= “gender ” value= “여 ” />
        <jsp:setProperty name= “pinfo ” property= “age ” value= “29 ” />
        이름: <jsp:getProperty name= “pinfo ” property= “name ” /> <BR>
        성별: <jsp:getProperty name= “pinfo ” property= “gender ” /> <BR>
        나이: <jsp:getProperty name= “pinfo ” property= “age ” />
    </BODY>
</HTML>
```

자바빈 객체를 생성합니다.

자바빈 객체에 프로퍼티 값을 설정합니다

자바빈 객체로부터 프로퍼티 값을 가져다가 출력합니다

❖ 자바빈 관련 표준 액션의 기초 사용 방법

CustomerInfoSaver.jsp

```
<%@page contentType= “text/html; charset=utf-8 ”%>
<jsp:useBean class= “mall.PersonalInfo ” id= “pinfo ” scope= “request ” />
<jsp:setProperty name= “pinfo ” property= “name ” value= “김현수 ”  />
<jsp:setProperty name= “pinfo ” property= “gender ” value= “남 ”  />
<jsp:setProperty name= “pinfo ” property= “age ” value= “23 ” />
<jsp:forward page= “CustomerInfoViewer.jsp ” />
```

호출

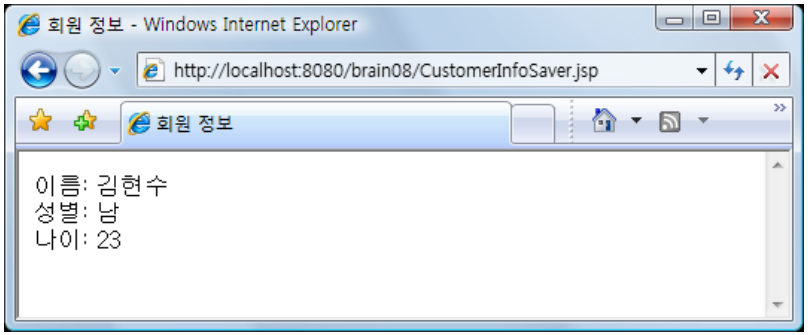
request 영역으로부터 자바빈 객체를 가져옵니다

CustomerInfoViewer.jsp

```
<%@page contentType= “text/html; charset=utf-8 ”%>
<HTML>
    <HEAD><TITLE>회원 정보</TITLE></HEAD>
    <BODY>
        <jsp:useBean class= “mall.PersonalInfo ” id= “pinfo ” scope= “request ”
/>
        이름: <jsp:getProperty name= “pinfo ” property= “name ” /> <BR>
        성별: <jsp:getProperty name= “pinfo ” property= “gender ” /> <BR>
        나이: <jsp:getProperty name= “pinfo ” property= “age ” />
    </BODY>
</HTML>
```

❖ 자바빈 관련 표준 액션의 기초 사용 방법

- 앞 페이지 예제의 두 JSP 페이지를 톰캣의 webapps/brain08 디렉터리에 각각 CustomerInfoSaver.jsp와 CustomerInfoViewer.jsp라는 이름으로 저장한다.



[그림 8-6] 예제 8-6의 실행 결과

- 서블릿 클래스에서 forward 메서드를 이용해서 JSP 페이지를 호출하면서 자바빈 객체를 넘겨주려면 doGet, doPost 메서드의 첫 번째 파라미터에 대해 setAttribute 메서드를 호출하면 된다.
- 호출된 JSP 페이지 안에서 넘겨받은 자바빈 객체를 가져오려면 <jsp:useBean> 표준 액션의 scope 애트리뷰트 값을 request로 지정하면 된다.

PersonalInfoServlet.java

```
package mall;   import javax.servlet.http.*;  import javax.servlet.*;    import java.io.*;
public class PersonalInfoServlet extends HttpServlet {
        public void doGet(HttpServletRequest request, HttpServletResponse response)
                                throws IOException, ServletException {
            PersonalInfo obj = new PersonalInfo();
            obj.setName( "이정호" );        obj.setGender( '남' );
            obj.setAge(24);              request.setAttribute( "pinfo ", obj);
            RequestDispatcher dispatcher =
request.getRequestDispatcher( "CustomerInfoViewer.jsp ");
            dispatcher.forward(request, response);
        }
}
```

자바빈 객체를 request 영역에 저장합니다

JSP 페이지를 호출합니다.

CustomerInfoViewer.jsp

```
<%@page contentType= "text/html; charset=utf-8 "%>
<HTML>
   <HEAD><TITLE>회원 정보</TITLE></HEAD>
   <BODY>
        <jsp:useBean class= "mall.PersonalInfo " id= "pinfo " scope= "request " />
        이름: <jsp:getProperty name= "pinfo " property= "name " /> <BR>
        성별: <jsp:getProperty name= "pinfo " property= "gender " /> <BR>
        나이: <jsp:getProperty name= "pinfo " property= "age " />
   </BODY>
</HTML>
```

request 영역으로부터 자바빈 객체를 가져옵니다

❖ 웹 브라우저로부터 입력된 데이터를 자바빈 프로퍼티로 설정하는 방법

- **<jsp:getProperty> 표준 액션의 value 애트리뷰트 값을 익스프레션 문법을 이용해서 지정하려면 우선 해당 프로퍼티의 타입에 맞게 변환을 해야 한다.**

```
<%
      String str = request.getParameter( “AGE ”);
      int num = Integer.parseInt(str);
%>
<jsp:setProperty name= “pinfo ” property= “age ” value=
“<%= num %> ” />
```

AGE라는 이름의 입력 데이터를 가져와서 타입 변환을 합니다

변환된 값을 자바빈 프로퍼티 값으로 설정합니다

- **익스프레션 언어의 param 내장 객체를 사용하면 입력 데이터를 바로 가져옴과 동시에 타입 변환도 자동으로 이루어져서 편리하다.**

```
<jsp:setProperty name= “pinfo ” property= “age ” value= “${param.AGE} ” />
```

AGE라는 이름의 입력 데이터를 가져와서 자바빈 프로퍼티 값으로 설정합니다

Person.html

```
<HTML>
   <HEAD>
      <META http-equiv="Content-Type" content="text/html;
   charset=utf-8 ">
      <TITLE>회원 가입</TITLE>
   </HEAD>
   <BODY>
      개인 정보를 입력하십시오.
      <FORM ACTION=NewerCustomerInfo.jsp
   METHOD=POST>
          이름: <INPUT TYPE=TEXT NAME=name><BR>
         성별 : <INPUT TYPE=TEXT NAME=gender><BR>
          나이: <INPUT TYPE=TEXT NAME=age><BR>
          <INPUT TYPE=SUBMIT VALUE='확인'>
      </FORM>
   </BODY>
</HTML>
```

❖ 웹 브라우저로부터 입력된 데이터를 자바빈 프로퍼티로 설정하는 방법

- **<jsp:setProperty> 액션에 value 애트리뷰트 대신 param이라는 애트리뷰트를 사용하면 입력 데이터의 이름을 직접 지정할 수 있다.**

```
<jsp:setProperty name= “pinfo ” property= “age ” param= “AGE ” />
```

AGE라는 이름의 입력 데이터를 가져와서 자바빈 프로퍼티 값으로 설정합니다

NewerCustomerInfo.jsp

```
<%@page contentType= “text/html; charset=utf-8 ”%>
<% request.setCharacterEncoding("utf-8");%>
<HTML>
    <HEAD><TITLE>회원 정보</TITLE></HEAD>
    <BODY>
        <jsp:useBean class= “mall.PersonalInfo” id= “pinfo ” />
        <jsp:setProperty name= “pinfo” property= “name” param= “name” />
        <jsp:setProperty name= “pinfo” property= “gender” param= “gender” />
        <jsp:setProperty name= “pinfo” property= “age ” param= “age” />
        이름: <jsp:getProperty name= “pinfo” property= “name” /> <BR>
        성별: <jsp:getProperty name= “pinfo” property= “gender” /> <BR>
        나이: <jsp:getProperty name= “pinfo” property= “age” />
    </BODY>
</HTML>
```

웹 브라우저로부터 입력된 데이터를 프로퍼티 값으로 설정합니다

❖ 웹 브라우저로부터 입력된 데이터를 자바빈 프로퍼티로 설정하는 방법

- **<jsp:setProperty> 액션에 param이나 value 애트리뷰트를 둘 다 쓰지 않고, property 애트리뷰 값을 '*'로 지정하면, 웹 브라우저로부터 입력된 모든 데이터가 똑같은 이름을 갖는 자바빈 프로퍼티에 각각 설정된다.**

```
<jsp:setProperty name= "pinfo " property= "* " />
```

모든 입력 데이터를 가져와서 그와 동일한 이름의 프로퍼티 값으로 설정

NewerCustomerInfo.jsp)

```
<%@page contentType= "text/html; charset=utf-8 "%>
<% request.setCharacterEncoding("utf-8");%>
<HTML>
    <HEAD><TITLE>회원 정보</TITLE></HEAD>
    <BODY>
        <jsp:useBean class= "mall.PersonalInfo " id= "pinfo " />
        <jsp:setProperty name= "pinfo " property= "* " />
        이름: <jsp:getProperty name= "pinfo " property= "name " /> <BR>
        성별: <jsp:getProperty name= "pinfo " property= "gender " /> <BR>
        나이: <jsp:getProperty name= "pinfo " property= "age " />
    </BODY>
</HTML>
```

입력된 모든 데이터를 그와 동일한 이름의 프로퍼티 값으로 설정합니다

❖ 자바빈의 다형성을 활용하는 방법

ProductInfo.java

```
package mall;
public class ProductInfo {
    private String code; // 제품코드
    private String name; // 제품명
    private int price; // 가격
    public void setCode(String code) {
        this.code = code;    }
    public void setName(String name) {
        this.name = name;    }
    public void setPrice(int price) {
        this.price = price;    }
    public String getCode() {
        return code;    }
    public String getName() {
       return name;    }
    public int getPrice() {
        return price;    }
}
```

상속

```
package mall;
public class BookInfo extends ProductInfo {
    private short page; // 페이지수
    private String writer; // 저자
    public void setPage(short page) {
        this.page = page;    }
    public void setWriter(String writer) {
        this.writer = writer;    }
    public short getPage() {
        return page;    }
    public String getWriter() {
        return writer;    }
}
```

상속

```
package mall;
public class ClothingInfo extends
ProductInfo {
    private char size; // 사이즈(L/M/S)
    private String color; // 색상
    public void setSize(char size) {
        this.size = size;    }
    public void setColor(String color) {
        this.color = color;    }
    public char getSize() {
        return size;    }
    public String getColor() {
        return color;    }
}
```

❖ 자바빈의 다형성을 활용하는 방법

- 클래스들이 서로 상속 관계를 맺고 있으면, 서브클래스 타입의 객체를 슈퍼클래스 타입의 변수에 대입해서 사용할 수 있다.

```
ProductInfo pinfo = new BookInfo();
ProductInfo pinfo = new ClothingInfo();
```

서브클래스의 객체를 슈퍼클래스 타입의 변수에 대입할 수 있습니다

- 서브클래스 타입 객체가 대입된 변수에 대해 슈퍼클래스의 메서드를 호출하면 변수에 실제로 어떤 객체가 대입되어 있든 상관없이 그에 해당하는 메서드가 호출되어서 실행된다.

```
int price = pinfo.getPrice();
```

pinfo 변수의 값에 상관없이 슈퍼클래스의 메서드를 호출할 수 있습니다.

BookInfoSaver.jsp

```
<%@page contentType= “text/html; charset=utf-8 ”%>
<jsp:useBean class= “mall.BookInfo ” id= “pinfo ” scope= “request ”/>
<jsp:setProperty name= “pinfo ” property= “code ” value= “50001 ” />
<jsp:setProperty name= “pinfo ” property= “name ” value= “의뢰인 ” />
<jsp:setProperty name= “pinfo ” property= “price ” value= “9000 ” />
<jsp:setProperty name= “pinfo ” property= “writer ” value= “존 그리샴 ” />
<jsp:setProperty name= “pinfo ” property= “page ” value= “704 ” />
<HTML>  <HEAD><TITLE>책 정보 관리</TITLE></HEAD>
  <BODY>  책 정보가 저장되었습니다. <BR>
    ----------<BR>
    <H3>제품 개략 정보</H3>
    <jsp:include page= “ProductInfo.jsp ” />
  </BODY></HTML>
```

포함

ProductInfo.jsp

```
<%@page contentType=“text/html; charset=utf-8”%>
<jsp:useBean class=“mall.ProductInfo” id=“pinfo”
scope=“request” />
코드: <jsp:getProperty name=“pinfo” property=“code” /> <BR>
제품명:<jsp:getProperty name=“pinfo” property=“name” />
<BR>
가격: <jsp:getProperty name=“pinfo” property=“price” /> <BR>
```

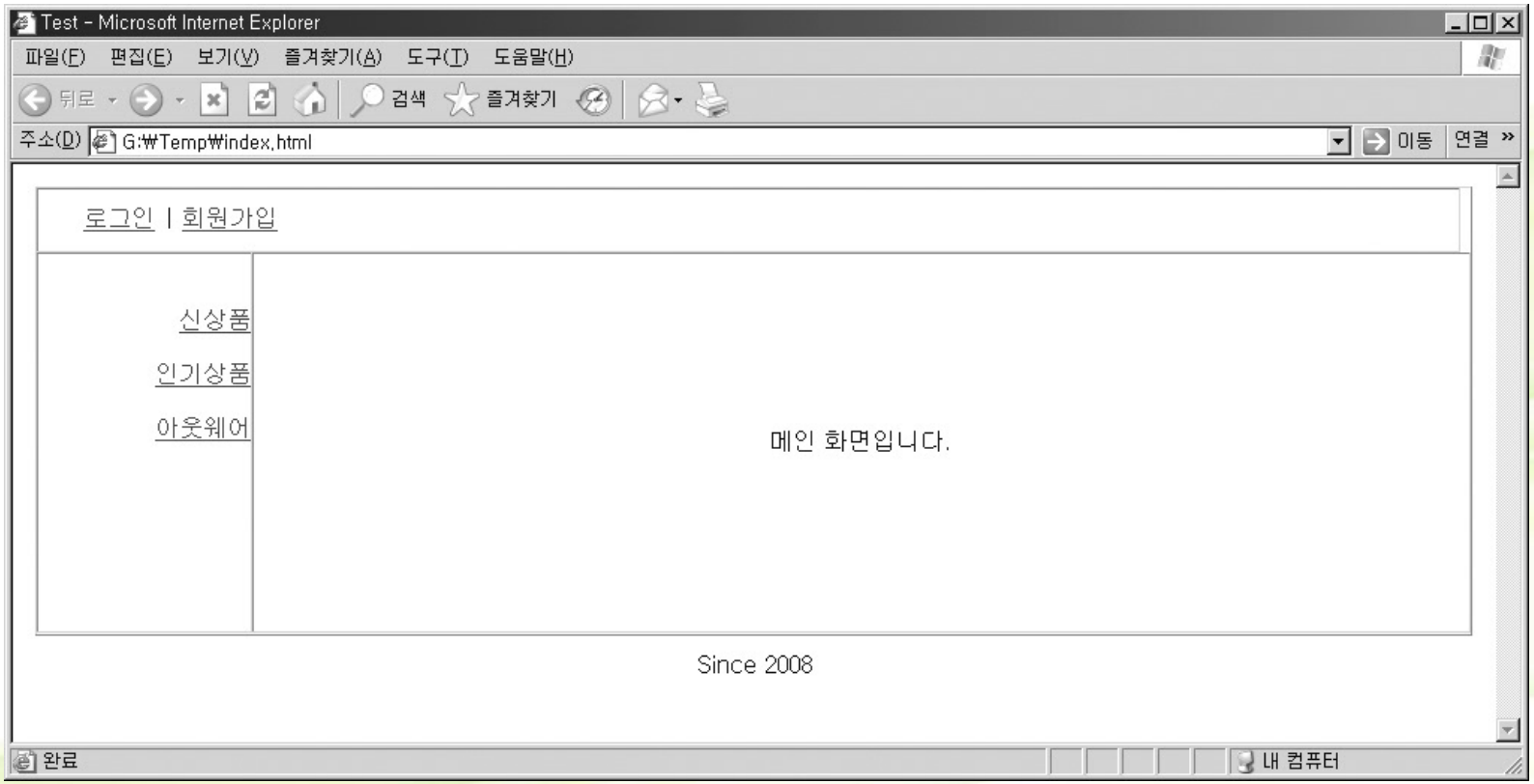
포함

```
<%@page contentType= “text/html; charset=utf-8 ”%>
<jsp:useBean class= “mall.ClothingInfo ” id= “pinfo ” scope= “request ”/>
<jsp:setProperty name= “pinfo ” property= “code ” value= “70002 ” />
<jsp:setProperty name= “pinfo ” property= “name ” value= “반팔 티셔츠 ” />
<jsp:setProperty name= “pinfo ” property= “price ” value= “15000 ” />
<jsp:setProperty name= “pinfo ” property= “size ” value= “M ” />
<jsp:setProperty name= “pinfo ” property= “color ” value= “베이지 ” />
<HTML>  <HEAD><TITLE>의류 정보 관리</TITLE></HEAD>
  <BODY>
    의류 정보가 저장되었습니다. <BR>
    ----------<BR>
    <H3>제품 개략 정보</H3>
    <jsp:include page= “ProductInfo.jsp ” />
  </BODY></HTML>
```

ClothingInfoSaver.jsp

액션태그를 활용한 템플릿페이지작성

- 템플릿페이지를 사용하는 이유
  - 반복되는 페이지의 사이트를 만들때 고정된 페이지를 템플릿 페이지로 만들면 유지보수가 용이
  - 템플릿 페이지를 사용하지 않을 경우 나중에 레이아웃을변경할경우작성된 페이지를 모두 새로 작성해야 하기 때문
- 템플릿페이지의 설계



액션태그를이용한템플릿페이지의작성

❖ 템플릿페이지구성

파일 이름	설명
top.jsp	상단에 표시될 메뉴 파일 이름이다.
bottom.jsp	하단에 표시될 파일 이름이다.
left.jsp	좌측에 표시될 메뉴 파일 이름이다.
newitem.jsp	신상품 페이지 파일 이름이다.
bestitem.jsp	인기상품 페이지 파일 이름이다.
template.jsp	템플릿 페이지(레이아웃) 파일 이름이다.

top.jsp

```
<%@ page language="java" contentType="text/html;charset=utf-8" pageEncoding="utf-8"%>
<a href="login.jsp">Login</a> |
<a href="join.jsp">Join</a>
```

bottom.jsp

```
<%@ page language="java" contentType="text/html; charset=utf-8" pageEncoding="utf-8"%>
<center>Since 2008</center>
```

left.jsp

```
<%@ page language="java" contentType="text/html; charset=utf-8" pageEncoding="utf-8"%>
<center><a href="./template.jsp?page=newitem">신상품</a><br><br>
<a href="./template.jsp?page=bestitem">인기상품</a><br><br></center>
```

newitem.jsp

```
<%@ page language="java" contentType="text/html; charset=utf-8" pageEncoding="utf-8"%>
<b>신상품 목록입니다.</b>
```

bestitem.jsp

```
<%@ page language="java" contentType="text/html; charset=utf-8" pageEncoding="utf-8"%>
<b>인기상품 목록입니다.</b>
```

template.jsp

```
<%@ page language="java" contentType="text/html; charset=utf-8" pageEncoding="utf-8"%>
<%      String pagefile=request.getParameter("page");
        if (pagefile==null){pagefile="newitem";}
%>
<html><head><title>Template Test</title></head>
<body>
<table width="960" border="1" color="gray" align="center">
        <tr>    <td height="43" colspan=3 align=left>
                        <jsp:include page="top.jsp"/>
                </td>   </tr>
        <tr>    <td width="15%" align=right valign=top><br>
                        <jsp:include page="left.jsp"/>
                </td>
                <td colspan=2 align=center>
                        <jsp:include page='<%=pagefile+".jsp" %>'/>
                </td>
        </tr>
        <tr>
                <td width="100%" height="40" colspan="3">
                        <jsp:include page="bottom.jsp"/>
                </td>
        </tr>
</table>
</body></html>
```