



JSP 6강 – 내장객체, 액션태그, 자바빈

양 명 속

[now4ever7@gmail.com]



목차

- JSP 페이지의 내장객체와 영역
- 액션 태그
- 자바빈
- 자바빈과 <jsp:useBean>액션 태그의 연동



JSP 페이지의 내장객체와 영역



내장 객체

- JSP 내장객체
 - 별다른 선언과정과 객체 생성 없이 사용할 수 있는 9개의 객체들을 웹 컨테이너가 제공함
- 내장객체의 4가지 범주
 - JSP 페이지 입출력 관련 내장 객체
 - JSP 페이지 외부 환경 정보 제공 내장 객체
 - JSP 페이지 서블릿 관련 내장 객체
 - JSP 페이지 예외 관련 내장 객체



JSP 내장객체

내장 객체	리턴 타입	설명
request	javax.servlet.http.HttpServletRequest 또는 javax.servlet.ServletRequest	클라이언트의 요청 정보를 저장하고 있는 객체이다
response	javax.servlet.http.HttpServletResponse 또는 javax.servlet.ServletResponse	클라이언트의 요청에 대한 응답 정보를 저장하고 있는 객체
out	javax.servlet.jsp.JspWriter	JSP 페이지 출력할 내용을 가지고 있는 출력 스트림 객체
session	javax.servlet.http.HttpSession	세션 정보를 저장하고 있는 객체
application	javax.servlet.ServletContext	웹 어플리케이션 Context의 정보를 저장하고 있는 객체
pageContext	javax.servlet.jsp.PageContext	JSP 페이지에 대한 정보를 저장하고 있는 객체
page	javax.servlet.jsp.HttpJspPage	JSP 페이지를 구현한 자바 클래스 객체
config	javax.servlet.ServletConfig	JSP 페이지에 대한 설정정보를 저장하고 있는 객체
exception	java.lang.Throwable	JSP 페이지에서 예외 발생시에만 사용되는 객체

JSP 내장객체

```
request.setAttribute("msg", "성공!");  
String msg=(String)request.getAttribute("msg");  
  
session.setAttribute("userid", "hong");  
String id = (String)session.getAttribute("userid");
```

- request, session, application, pageContext 내장 객체
 - 속성(attribute)값을 저장하고 읽을 수 있는 메서드인 `setAttribute()`, `getAttribute()` 메서드를 제공함
 - 내장 객체의 속성을 저장하고 읽어내는 공통 메서드

메서드	리턴 타입	설명
<code>setAttribute(String key, Object value)</code>	void	주어진 key 속성의 값을 value로 지정한다
<code>getAttributeNames()</code>	java.util.Enumeration	모든 속성의 이름을 구한다
<code>getAttribute(String key)</code>	Object	주어진 key 속성의 값을 얻어낸다
<code>removeAttribute(String key)</code>	void	주어진 key 속성의 값을 제거한다.



request 내장객체

- request 객체

- 웹 브라우저에서 jsp 페이지로 전달되는 정보의 모임
- HTTP 헤더와 HTTP 바디로 구성되어 있음
- 웹 컨테이너는 요청된 HTTP 메시지를 통해 HttpServletRequest 객체를 얻어내서 그것으로부터 사용자의 요구사항을 얻어냄
- JSP 페이지에서는 HttpServletRequest 객체를 request 객체명으로 사용함



request 내장객체

- request 객체의 요청 파라미터 메서드

메서드	설명
String getParameter(name)	이름이 name인 파라미터에 할당된 값을 리턴하며, 지정된 파라미터 값이 없으면 null값을 리턴함
String[] getParameterValues(name)	이름이 name인 파라미터의 모든 값을 String 배열로 리턴함 Checkbox 에서 주로 사용됨
Enumeration getParameterNames()	요청에 사용된 모든 파라미터 이름을 java.util.Enumeration 타입으로 리턴함



HTML 폼과 요청 파라미터의 처리

- 웹 브라우저 폼에 입력한 값 처리
- 웹 브라우저는 폼에 입력한 정보를 파라미터로 전송함
 - request 기본 객체는 웹 브라우저가 전송한 파라미터를 읽어올 수 있는 메서드 제공

예제

폼에 데이터를 입력한 후 '전송' 버튼을 클릭하세요.

이름: 홍길동

주소: 마포구 서교동

좋아하는 동물: ☒ 강아지 ☒ 고양이 ☐ 돼지

```
<%@ page contentType = "text/html; charset=utf-8" %>
```

```
<html>
```

```
<head><title>폼 생성</title></head>
```

```
<body>
```

폼에 데이터를 입력한 후 '전송' 버튼을 클릭하세요.

```
<form action="/test/viewParameter.jsp" method="post">
```

```
이름: <input type="text" name="name" size="10"> <br>
```

```
주소: <input type="text" name="address" size="30"> <br>
```

```
좋아하는 동물:
```

```
    <input type="checkbox" name="pet" value="dog">강아지
```

```
    <input type="checkbox" name="pet" value="cat">고양이
```

```
    <input type="checkbox" name="pet" value="pig">돼지
```

```
<br>
```

```
<input type="submit" value="전송">
```

```
</form>
```

```
</body>
```

```
</html>
```



예제

```
<%@ page contentType="text/html; charset=utf-8" %>
<%@ page import="java.util.Enumeration" %>
<%@ page import="java.util.Map" %>
<%
    request.setCharacterEncoding("utf-8");
%>
<html>
<head><title>요청 파라미터 출력</title></head>
<body>
<b>request.getParameter() 메소드 사용</b><br>
name 파라미터 = <%= request.getParameter("name") %> <br>
address 파라미터 = <%= request.getParameter("address") %>
<p>
<b>request.getParameterValues() 메소드 사용</b><br>
<%
    String[] values = request.getParameterValues("pet");
    if (values != null) {
        for (int i = 0 ; i < values.length ; i++) {
%>
            <%= values[i] %>
<%
        }
    }
%>
```

request.getParameter() 메소드 사용

name 파라미터 = 홍길동

address 파라미터 = 마포구 서교동

request.getParameterValues() 메소드 사용

dog cat

request.getParameterNames() 메소드 사용

pet address name

request.getParameterMap() 메소드 사용

name = 홍길동



예제

<p>

request.getParameterNames() 메소드 사용

<%

```
Enumeration paramEnum = request.getParameterNames();
while(paramEnum.hasMoreElements()) {
    String name = (String)paramEnum.nextElement();
```

%>

<%= name %>

<%

}

%>

<p>

request.getParameterMap() 메소드 사용

<%

```
Map parameterMap = request.getParameterMap();
String[] nameParam = (String[])parameterMap.get("name");
if (nameParam != null) {
```

%>

name = <%= nameParam[0] %>

<%

}

%>

</body>

</html>



request 기본객체

- 웹 브라우저의 요청과 관련이 있음
- 클라이언트가 전송한 요청 정보를 제공하는 것이 request 기본 객체임
- request 기본객체가 제공하는 기능
 - 클라이언트와 관련된 정보 읽기 기능
 - 서버와 관련된 정보 읽기 기능
 - 클라이언트가 전송한 요청 파라미터 읽기 기능
 - 클라이언트가 전송한 요청 헤더 읽기 기능
 - 클라이언트가 전송한 쿠키 읽기 기능
 - 속성 처리 기능



request 기본객체

- 클라이언트 정보 및 서버 정보 읽기
 - 클라이언트가 전송한 정보 및 서버 정보를 구할 수 있는 메서드를 제공

메서드	설명
String <code>getMethod()</code>	요청방식(get, post)을 리턴함
String <code>getRequestURI()</code>	요청에 사용된 URL로부터 URI를 리턴
String <code>getQueryString()</code>	요청에 사용된 Query 문장을 리턴
String <code>getRemoteHost()</code>	클라이언트의 호스트 이름을 리턴
String <code>getRemoteAddr()</code>	클라이언트의 IP 주소를 리턴
String <code>getProtocol()</code>	사용중인 프로토콜을 리턴
String <code>getServerName()</code>	서버의 도메인 이름을 리턴
String <code>getServerPort()</code>	서버의 port 번호를 리턴
String <code>getHeader(name)</code>	HTTP 요청 헤더에 지정된 name 의 값을 리턴
String <code>getContextPath()</code>	해당 jsp 페이지가 속한 웹 어플리케이션의 컨텍스트 경로 를 리턴

예제

클라이언트IP = 127.0.0.1
요청정보길이 = -1
요청정보 인코딩 = null
요청정보 콘텐츠타입 = null
요청정보 프로토콜 = HTTP/1.1
요청정보 전송방식 = GET
요청 URI = /test/requestInfo.jsp
컨텍스트 경로 = /test
서버이름 = localhost
서버포트 = 9090

```
<%@ page contentType = "text/html; charset=utf-8" %>
<html>
<head><title>클라이언트 및 서버 정보</title></head>
<body>
```

```
클라이언트IP = <%= request.getRemoteAddr() %> <br>
요청정보길이 = <%= request.getContentLength() %> <br>
요청정보 인코딩 = <%= request.getCharacterEncoding() %> <br>
요청정보 콘텐츠타입 = <%= request.getContentType() %> <br>
요청정보 프로토콜 = <%= request.getProtocol() %> <br>
요청정보 전송방식 = <%= request.getMethod() %> <br>
요청 URI = <%= request.getRequestURI() %> <br>
컨텍스트 경로 = <%= request.getContextPath() %> <br>
서버이름 = <%= request.getServerName() %> <br>
서버포트 = <%= request.getServerPort() %> <br>
```

```
</body>
</html>
```



response 내장 객체

■ response 객체

- 웹 브라우저로 응답할 응답 정보를 가지고 있음
- 웹 브라우저에 보내는 응답 정보는 HttpServletResponse 객체로 jsp에서는 response 객체로 사용함
- 응답 정보와 관련하여 주로 헤더 정보 입력, 페이지 리다이렉트 등의 기능 제공
- response 내장 객체의 메서드

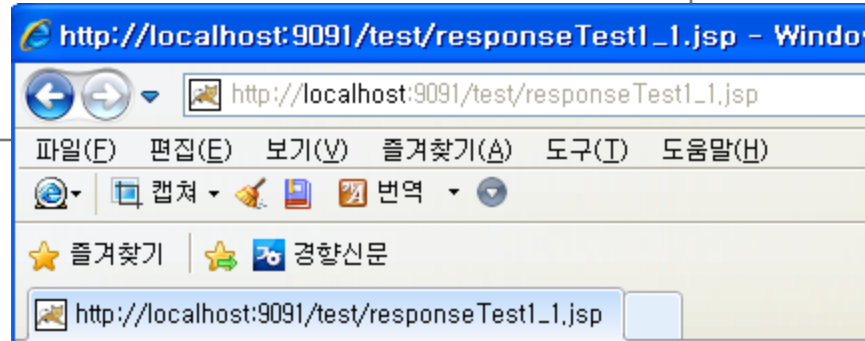
메서드	설명
void setHeader(name, value)	name에 해당하는 속성을 value값으로 설정한다.
void setContentType(type)	요청의 결과로 보여질 페이지의 contentType을 설정한다
void sendRedirect(url)	지정된 url로 페이지가 이동한다.

예제

```
<%@ page contentType="text/html; charset=utf-8"%>
<h2>Response내장객체 예제</h2>
  현재 페이지는 responseTest1.jsp 페이지입니다.

<%
  response.sendRedirect("responseTest1_1.jsp");
%>
```

```
<%@ page contentType="text/html; charset=utf-8"%>
<html>
  <body>
    <h2>리다이렉트된 페이지</h2>
    지금 보시는 페이지는 responseTest1_1.jsp입니다.
  </body>
</html>
```



리다이렉트된 페이지

지금 보시는 페이지는 responseTest1_1.jsp입니다.



out 내장객체

■ out 객체

- jsp 페이지가 생성한 결과를 웹 브라우저에 전송해 주는 출력 스트림
- jsp 페이지가 웹 브라우저에게 보내는 모든 정보는 out 객체로 통해서 전송됨
- out 객체는 `javax.servlet.jsp.jspWriter` 클래스 타입으로 jsp에서는 out 객체로 사용됨
- `println()`메서드
 - 웹 브라우저에 출력을 하기 위한 메서드
 - 표현식 `<%=코드%>` 와 `<% out.println(코드)%>`는 동일



out 내장객체

■ out 객체의 메서드

메서드	설명
boolean isAutoFlush()	출력 버퍼가 다 찼을 때 처리 여부를 결정하는 것으로 자동으로 flush 할 경우에는 true를 리턴하고, 그렇지 않을 경우 false 리턴
int getBufferSize()	출력 버퍼의 전체 크기를 리턴
int getRemaining()	현재 남아 있는 출력 버퍼의 크기를 리턴
void clearBuffer()	현재 출력 버퍼에 저장되어 있는 내용을 웹 브라우저에 전송하지 않고 비운다
String println(str)	주어진 str값을 웹 브라우저에 출력함. 이 때 줄바꿈은 적용되지 않음
void flush()	현재 출력 버퍼에 저장되어 있는 내용을 웹 브라우저에 전송하고 비운다.
void close()	현재 출력 버퍼에 저장되어 있는 내용을 웹 브라우저에 전송하고 출력 스트림을 닫는다.



pageContext 내장 객체

- pageContext 내장 객체
 - 현재 jsp 페이지의 컨텍스트를 나타내며, 주로 다른 내장 객체를 얻어내거나, 페이지의 흐름제어, 에러 데이터를 얻어낼 때 사용됨
 - javax.servlet.jsp.pageContext 객체 타입으로, jsp에서는 pageContext객체로 사용됨
 - 예) pageContext 객체를 이용하여 out 객체를 얻어내는 방법
 - `jspWriter outObj = pageContext.getOut();`

pageContext 내장 객체

■ pageContext 내장 객체의 메서드

메서드	설명
ServletRequest getRequest()	페이지 요청 정보를 가지고 있는 request 내장 객체를 리턴함
ServletResponse getResponse()	페이지 요청에 대한 응답 정보를 가지고 있는 response 내장 객체를 리턴함
JspWriter getOut()	페이지 요청에 대한 응답 출력 스트림인 out 내장 객체를 리턴
Httpsession getSession()	요청한 클라이언트의 세션 정보를 담고 있는 session 내장 객체를 리턴
ServletContext getContext()	페이지에 대한 서블릿 실행 환경 정보를 담고 있는 application 내장 객체를 리턴
Object getPage()	page 내장 객체를 리턴
ServletConfig getConfig()	페이지의 서블릿 초기 설정 정보를 담고 있는 config 내장 객체를 리턴
Exception getException()	페이지 실행 중에 발생하는 에러 페이지에 대한 예외 정보를 갖고 있는 exception 내장 객체를 리턴



session 내장 객체

■ session 내장 객체

- 웹 브라우저의 요청시 요청한 웹 브라우저(세션)에 관한 정보를 저장하고 관리하는 내장 객체
- javax.servlet.http.HttpSession 객체 타입으로, jsp에서는 session 객체로 사용됨
- session 객체는 웹 브라우저(클라이언트)당 1개가 할당됨
- request 기본 객체가 하나의 요청을 처리하는 데 사용되는 jsp 페이지 사이에서 공유된다면, session 기본 객체는 웹 브라우저의 여러 요청을 처리하는 jsp 페이지 사이에서 공유됨

session 내장 객체

```
String id="hong";  
session.setAttribute("userid", id);
```

```
String id = (String)session.getAttribute("userid");
```

■ session 내장 객체의 메서드

메서드	설명
String getId()	해당 세션의 고유한 세션 ID를 리턴
long getCreationTime()	해당 세션이 생성된 시간을 리턴
long getLastAccessedTime()	웹 브라우저의 요청이 시도된 마지막 접근 시간을 리턴
void setMaxInactiveInterval(time)	해당 세션을 유지할 시간을 초단위로 설정
int getMaxInactiveInterval()	기본값은 30분, setMaxInactiveInterval(time)로 지정된 값을 리턴
boolean isNew()	새로 생성된 세션의 경우 true 값 리턴
void invalidate()	현재 설정된 세션의 속성 값을 모두 제거함 주로 세션을 종료시킬 때 사용

예제

```
<%@ page contentType="text/html; charset=utf-8"%>
```

```
<html>
```

```
<body>
```

```
<h1>session내장객체 예제</h1>
```

```
<FORM METHOD="post" ACTION="sessionTest1.jsp">
```

```
아이디 : <INPUT TYPE="text" NAME="id" style="ime-mode:inactive;"><p>
```

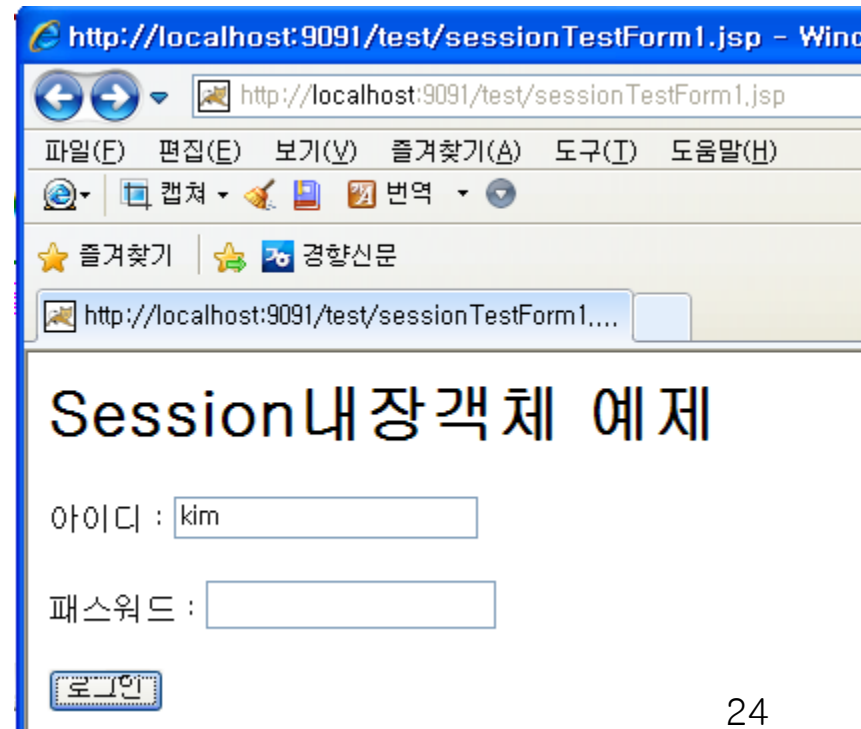
```
패스워드 : <INPUT TYPE="password" NAME="passwd" style="ime-mode:inactive;"><p>
```

```
<INPUT TYPE="submit" VALUE="로그인">
```

```
</FORM>
```

```
</body>
```

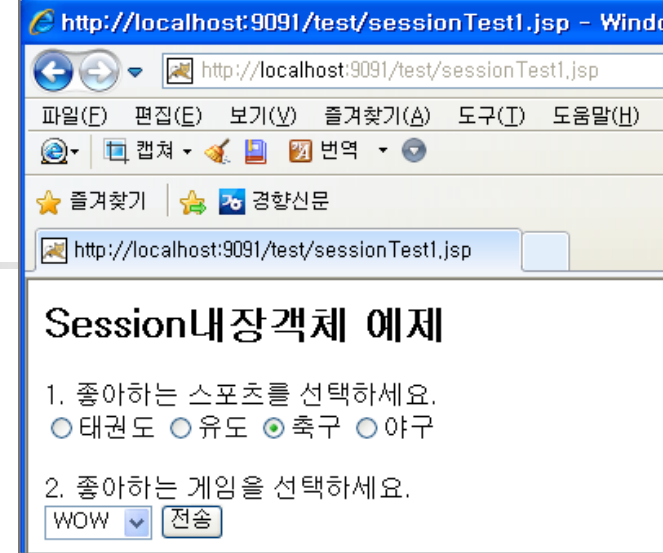
```
</html>
```



예제

```
<%@ page contentType="text/html; charset=utf-8"%>
<%//파라미터값의 한글처리
    request.setCharacterEncoding("utf-8");
%>
<% 파라미터값을 얻어냄
    String id = request.getParameter("id");
    String passwd = request.getParameter("pwd");
    //세션속성 설정, 속성명은 id이고 속성값은 id레퍼런스 변수가 가진 객체
    session.setAttribute("id",id);
    session.setMaxInactiveInterval(60*2);//세션유지시간 2분설정
%>
<h2>session내장객체 예제</h2>
<FORM METHOD="post" ACTION="sessionTest2.jsp">
    1. 좋아하는 스포츠를 선택하세요.<br>
    <INPUT TYPE="radio" NAME="sports" VALUE="태권도">태권도
    <INPUT TYPE="radio" NAME="sports" VALUE="유도">유도
    <INPUT TYPE="radio" NAME="sports" VALUE="축구">축구
    <INPUT TYPE="radio" NAME="sports" VALUE="야구">야구<p>

    2. 좋아하는 게임을 선택하세요.<br>
    <SELECT NAME="game">
        <OPTION VALUE="스타크">스타크</OPTION>
        <OPTION VALUE="WOW">WOW</OPTION>
        <OPTION VALUE="리니지">리니지</OPTION>
    </SELECT>
    <INPUT TYPE="submit" VALUE="전송">
</FORM>
```

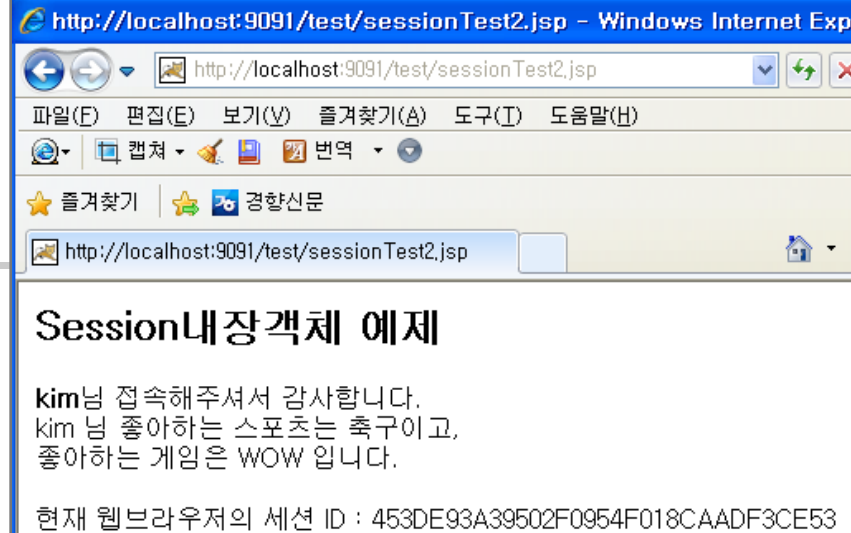


예제

```
<%@ page contentType="text/html; charset=utf-8"%>
<%
    request.setCharacterEncoding("utf-8");
%>
<h2>session내장객체 예제</h2>
<% //7~8라인:파라미터값을 얻어냄
    String sports = request.getParameter("sports");
    String game = request.getParameter("game");
    //10라인: 세션속성값을 얻어냄 => 세션 유지 여부를 판단가능
    String id = (String)session.getAttribute("id");
    String sessionId = session.getId();//웹 브라우저의 고유세션ID를 얻어냄

    if(id != null){//인증된 사용자 영역
%>
<b><%=id%></b>님 접속해주셔서 감사합니다. <br>
<%=id%> 님 좋아하는 스포츠는 <%=sports%>이고, <br>
좋아하는 게임은 <%=game%> 입니다.<p>
현재 웹브라우저의 세션 ID : <%=sessionId%><p>

<%
    session.invalidate();//세션 무효화
    }else{//인증되지 않은 사용자 영역
        out.println("로그인을 하시기 바랍니다.");
    }
%>
```





application 내장 객체

- application 내장 객체
 - 웹 어플리케이션의 설정 정보를 갖는 context와 관련이 있는 객체
 - 웹 어플리케이션과 연관이 있음
 - 웹 어플리케이션이 실행되는 서버의 설정 정보 및 자원에 대한 정보를 얻어내거나 어플리케이션이 실행되고 있는 동안에 발생할 수 있는 이벤트 로그 정보와 관련된 기능들을 제공함
 - application 내장 객체는 웹 어플리케이션당 1개의 객체가 생성됨
 - 하나의 웹 어플리케이션에서 공유하는 변수로 사용됨
 - 웹 사이트의 방문자 기록을 카운트할 때 주로 사용됨



application 내장 객체

```
String path = application.getRealPath("/");
```

■ application 내장 객체의 메서드

메서드	설명
String getServerInfo()	웹 컨테이너의 이름과 버전을 리턴
String getMimeType(fileName)	지정한 파일의 MIME 타입을 리턴
String getRealPath(path)	지정한 경로를 웹 어플리케이션 시스템 상의 경로로 변경하여 리턴함
void log(message)	로그 파일에 message 를 기록함

예제

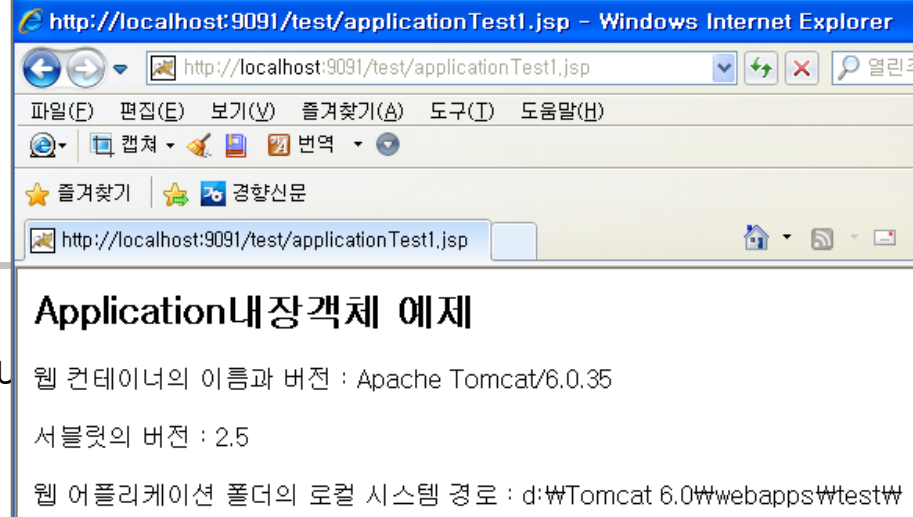
```
<%@ page contentType="text/html; charset=utf-8"%>
<%
    String info = application.getServerInfo();
    int major = application.getMajorVersion();
    int minor = application.getMinorVersion();
    String path = application.getRealPath("/");
%>
```

<h2>application내장객체 예제</h2>

웹 컨테이너의 이름과 버전 : <%=info%><p>

서블릿의 버전 : <%=major%>.<%=minor%><p>

웹 어플리케이션 폴더의 로컬 시스템 경로 : <%=path%>





config 내장 객체

- config 내장 객체
 - javax.servlet.ServletConfig 객체 타입으로
ServletConfig 객체는 서블릿이 초기화되는 동안 참조
해야 할 정보를 전달해 주는 역할을 함
 - 서블릿이 초기화될 때 참조해야 하는 정보를 가지고 있
다가 전달해줌
 - config 내장 객체는 컨테이너당 1개의 객체가 생성됨
 - 같은 컨테이너에서 서비스되는 모든 페이지는 같은 객
체를 공유함



config 내장 객체

```
String path = config.getServletContext().getRealPath("/");
```

■ config 내장 객체의 메서드

메서드	설명
Enumeration getInitParameterName()	모든 초기 파라미터 이름을 리턴
String getInitParameter(name)	이름이 name인 초기 파라미터의 값을 리턴
String getServletName()	서블릿의 이름을 리턴
ServletContext getServletContext()	실행하는 서블릿 ServletContext 객체를 리턴



page 내장 객체

- page 내장 객체
 - jsp 페이지 그 자체를 나타내는 객체
 - jsp 페이지 내에서 page 객체는 this 키워드로 자기 자신을 참조할 수 있다
 - javax.servlet.jsp.HttpJspPage 클래스 타입으로 제공되는 jsp 내장 객체
 - 거의 사용되지 않음

예제

```
<%@ page contentType="text/html;charset=utf-8"%>
```

```
<%@ page info = "Page 내장 객체 예제"%>
```

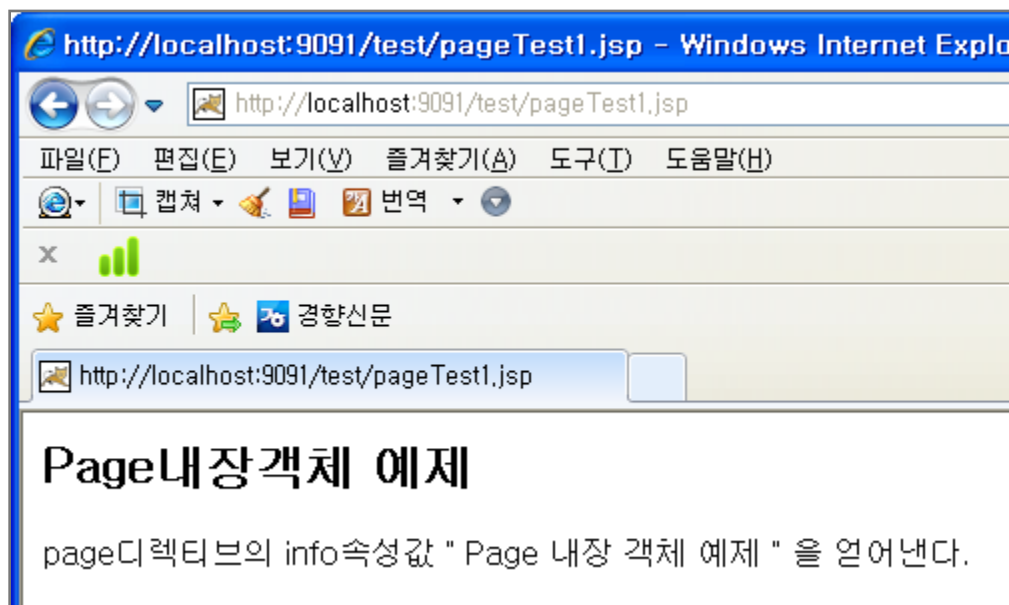
```
<%
```

```
    String info = this.getServletInfo();
```

```
%>
```

```
<h2>Page내장객체 예제</h2>
```

```
page디렉티브의 info속성값 " <%=info%> " 을 얻어낸다.
```





exception 내장 객체

- exception 내장 객체
 - jsp 페이지에서 예외가 발생하였을 경우 예외를 처리할 페이지를 지정하였을 때 예외 페이지에 전달되는 객체
 - page 디렉티브의 isErrorPage="true" 로 지정한 jsp 페이지에서만 사용 가능한 내장 객체
 - java.lang.Throwable 객체 타입



exception 내장 객체

- exception 내장 객체의 메서드

메서드	설명
String getMessage()	발생된 예외의 메시지를 리턴
String toString()	발생된 예외 클래스명과 메시지 리턴
String printStackTrace()	발생된 예외를 역추적하기 위해 표준 예외 스트림을 출력한다



내장 객체의 영역

■ 내장 객체의 영역

- 객체의 유효기간, 객체를 누구와 공유할 것인가를 나타냄
- 웹 어플리케이션은 page, request, session, application 4개의 영역을 가지고 있음

■ page 영역

- 한 번의 웹 브라우저(클라이언트)의 요청에 대해 **하나의 jsp 페이지**가 호출됨
- 웹 브라우저의 요청이 들어오면 이때 단 한 개의 페이지만 대응이 됨
- page 영역은 객체를 **하나의 페이지 내에서만 공유함**
- page 영역은 `pageContext` 내부 객체를 사용함



내장 객체의 영역

같은 request를 사용하는 페이지에서는 객체를 공유함

- request 영역
 - 한 번의 웹 브라우저의 요청에 대해 같은 요청을 공유하는 페이지가 대응됨
 - 웹 브라우저의 한 번의 요청에 단지 한 개의 페이지만 요청될 수 있고, 때에 따라 같은 request 영역이면 두 개의 페이지가 같은 요청을 공유할 수 있음
 - request 영역은 객체를 하나 또는 두 개의 페이지 내에서 공유할 수 있음
 - **include** 액션 태그, **forward** 액션 태그를 사용하면 request 내장 객체를 공유하게 되어서 같은 request 영역이 됨
 - 주로 페이지 모듈화에 사용됨
 - request 영역은 request 내장 객체를 사용함



내장 객체의 영역

■ session 영역

- 하나의 웹 브라우저당 1개의 session 객체가 생성됨
- 같은 웹 브라우저 내에서 요청되는 페이지들은 같은 객체를 공유하게 됨
- 주로 회원관리에서 회원인증에 사용되며, session 내장 객체를 사용함

■ application 영역

- 하나의 웹 어플리케이션당 1개의 applicatoin 객체가 생성됨
- 같은 웹 어플리케이션에 요청되는 페이지들은 같은 객체를 공유함
- /mystudy 웹 어플리케이션에서는 같은 application 객체를 공유함
- application 영역은 application 내장 객체를 사용함



JSP 페이지의 액션 태그

- 액션 태그의 개요
- jsp 페이지의 모듈화
- jsp 페이지의 흐름 제어



액션 태그

- jsp 페이지에서 페이지의 모듈화와 흐름 제어를 위해
 - include, forward 액션 태그를 제공함
- 자바빈의 사용을 위해
 - useBean, setProperty, getProperty 액션 태그를 제공



액션 태그의 개요

■ 액션 태그

- 스크립트, 주석, 디렉티브와 함께 jsp 페이지를 이루고 있는 요소
- 페이지와 페이지 사이의 제어를 이동시킬 수도 있고
- 다른 페이지의 실행결과를 현재의 페이지에 포함시킬 수 있으며
- 자바빈도 jsp 페이지에서 사용할 수 있는 기능을 제공함
- 웹 브라우저에서 자바 애플릿을 실행시킬 수 있도록 지원하는 기능도 있음
- 액션 태그는 XML 문법을 따름 - 단독 태그의 경우도 반드시 종료 태그를 포함해야 함



액션 태그의 개요

- jsp에서 제공하는 액션 태그
 - `<jsp:include>`
 - 페이지를 모듈화할 때 `<jsp:include>` 액션 태그가 사용됨
 - `<jsp:forward>`
 - 페이지의 흐름을 제어할 때 사용
 - `<jsp:useBean>`
 - 자바빈을 사용할 때(자바 빈 객체 생성시)
 - `<jsp:setProperty>`, `<jsp:getProperty>`
 - 자바빈의 속성값을 저장하고 읽어올 때 사용



jsp 페이지의 모듈화

- 다른 페이지를 현재 페이지에 포함시킬 수 있는 기능
 - include 액션 태그, include 디렉티브
- include 액션 태그
 - <jsp:include>
 - 다른 페이지의 실행결과를 현재 페이지에 포함시킬 때 사용됨
 - 페이지의 처리 결과를 포함시킴
 - 포함되는 페이지 - html, jsp, servlet 등 모두 가능
 - 페이지를 모듈화할 때 사용됨
 - 즉, 템플릿 페이지를 작성할 때 사용됨
- include 디렉티브
 - <%@ include %>
 - 단순히 소스의 내용이 텍스트로 포함됨
 - 주로 조각 코드를 삽입할 때 사용됨



include 액션 태그

```
<jsp:include page="includeTagTop1.jsp" />
```

■ [1] 사용법

```
<jsp:include page="포함될 페이지" flush="false"/>
```

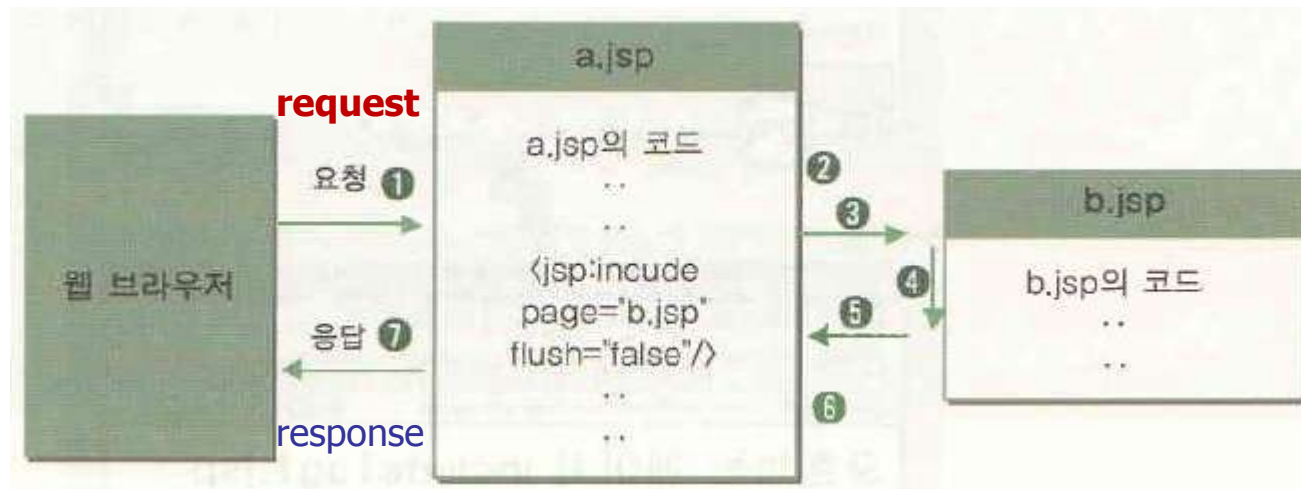
- 포함될 페이지 - 상대경로나 절대 경로 사용
- page 속성의 값은 **표현식 사용가능**
 - String content=request.getParameter("name");

```
<jsp:include page="<%=content%>" flush="false"/>
```

- flush 속성 - 포함될 페이지로 제어가 이동될 때 현재 포함하는 페이지가 지금까지 출력버퍼에 저장한 결과를 처리하는 방법을 결정하는 것
 - true 로 지정-포함될 페이지로 제어가 이동될 때 현재 페이지가 지금까지 버퍼에 저장한 내용을 웹 브라우저에 출력하고 버퍼를 비움
 - false로 지정하는 것이 일반적임

include 액션 태그

- include 액션 태그의 처리 과정



a.jsp와 **b.jsp**는 같은 **request** 를 공유함



include 액션 태그

- include 액션 태그의 처리 과정
 - 웹 브라우저가 a.jsp 페이지를 웹 서버에 요청함
 - 서버는 요청받은 a.jsp 페이지를 처리하는데, a.jsp 페이지 내에서 출력 내용을 출력 버퍼에 저장하는 작업을 처리함
 - `<jsp:include page="b.jsp" flush="false"/>` 문장을 만나면 하던 작업을 멈추고 **프로그램 제어를 b.jsp 페이지로 이동시킴**
 - b.jsp 페이지는 페이지 내에 출력 내용을 출력 버퍼에 저장하는 등의 작업을 처리함
 - b.jsp 페이지의 처리가 끝나면 다시 a.jsp 페이지로 프로그램의 제어가 이동함
 - 이동 위치는 `<jsp:include page="b.jsp" flush="false"/>` 문장 이후
 - a.jsp 페이지의 나머지 부분을 처리함, 출력할 내용이 있으면 출력 버퍼에 저장함
 - 출력 버퍼의 내용을 웹 브라우저로 응답함
- **`<jsp:include>` 액션 태그는 같은 request 내부 객체를 공유하므로, a.jsp와 b.jsp 는 같은 request 내부 객체를 공유함**

예제-includeTag1Form.jsp

```
<%@ page contentType="text/html;charset=utf-8"%>
```

includeTag1Form.jsp

```
<h1>include 액션태그 사용법 예제</h1>
```

```
<FORM METHOD=POST ACTION="includeTag1.jsp">
```

```
이름 : <INPUT TYPE="text" NAME="name"><p>
```

```
<INPUT TYPE="submit" VALUE="보내기">
```

```
</FORM>
```

```
<%@ page contentType="text/html;charset=utf-8"%>
```

```
<h1>includeTag1.jsp 입니다.</h1>
```

```
<%
```

```
    request.setCharacterEncoding("utf-8");
```

```
%>
```

```
<hr>
```

```
<jsp:include page="includeTagTop1.jsp" flush="false"/>
```

```
<hr>
```

includeTag1.jsp의 나머지 내용입니다.

includeTag1.jsp

예제-includeTagTop1.jsp

```
<%@ page contentType="text/html; charset=utf-8"%>
```

```
<%//includeTag1Form.jsp페이지의 파라미터 변수 name의 값을 얻어냄
```

```
//같은 request객체를 공유하기 때문에 사용가능
```

```
String name = request.getParameter("name");
```

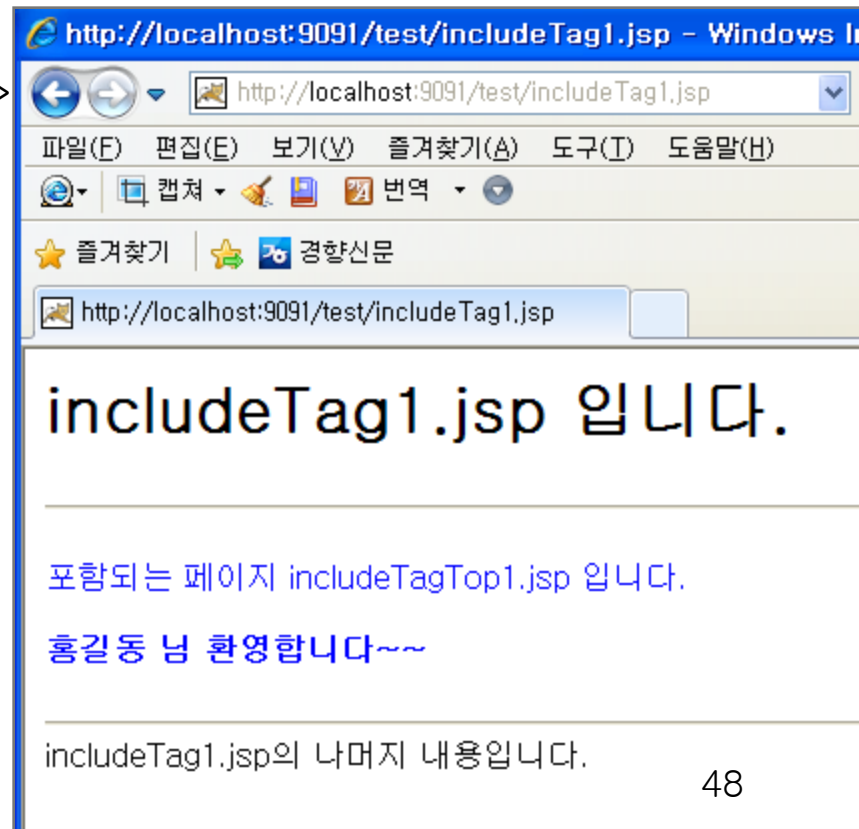
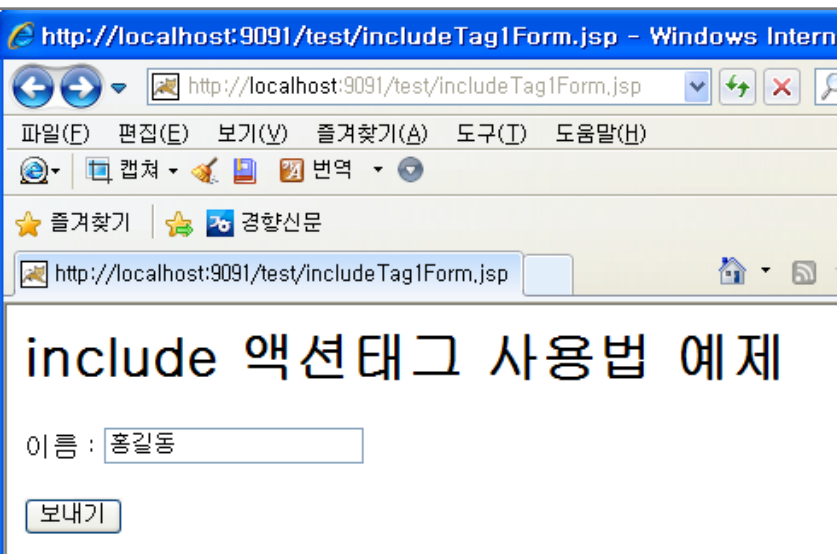
```
%>
```

```
<p style="color:blue">
```

```
포함되는 페이지 includeTagTop1.jsp 입니다.<br>
```

```
<b><%=name%> 님 환영합니다~~</b>
```

```
</p>
```





include 액션 태그

- [2]<jsp:include> 액션 태그에서 포함되는 페이지에 값 전달하기
 - 포함되는 jsp 페이지에 값 전달은 요청 파라미터를 추가적으로 지정해서 사용함
 - <jsp:include> 액션 태그의 body 안에 <jsp:param> 액션 태그를 사용하여 전달

```
<jsp:include page="포함되는 페이지" flush="false">  
    <jsp:param name="paramName1" value="var1"/>  
    <jsp:param name="포함되는 jsp 페이지에 전달할 파라미터의 이름"  
        value="전달할 파라미터의 값"/>  
</jsp:include>
```

- value 속성의 값으로 표현식을 사용할 수 있음

b.jsp?no=7

```
<jsp:include page="b.jsp" flush="false">  
    <jsp:param name="no" value="<%=var%>" />  
</jsp:include>
```



예제

```
<%@ page contentType="text/html; charset=utf-8"%>
```

includeTag2Form.jsp

```
<h2>include 액션 태그에서 포함되는 페이지에 값 전달하기</h2>
<FORM METHOD=POST ACTION="includeTag2.jsp">
사이트 명 : <INPUT TYPE="text" NAME="siteName1"><p>
<INPUT TYPE="submit" VALUE="보내기">
</FORM>
```

```
<%@ page contentType="text/html; charset=utf-8"%>
```

```
<h2>includeTag2.jsp 페이지 입니다</h2>
```

```
<%
```

includeTag2.jsp

```
    request.setCharacterEncoding("utf-8");
```

```
    String siteName1 = request.getParameter("siteName1");
```

```
%>
```

```
<hr>
```

```
<jsp:include page="includeTagTop2.jsp">
```

```
    <jsp:param name="siteName" value="<%=siteName1%>" />
```

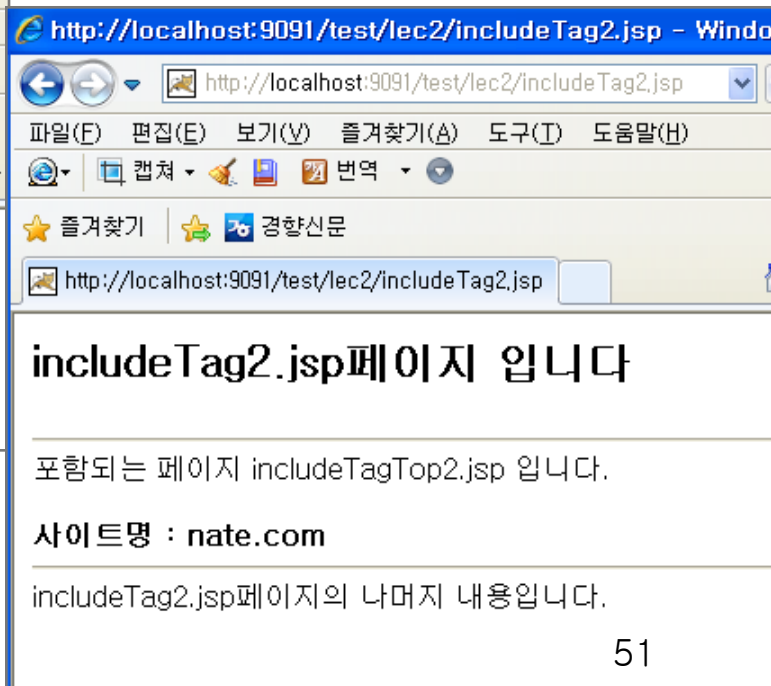
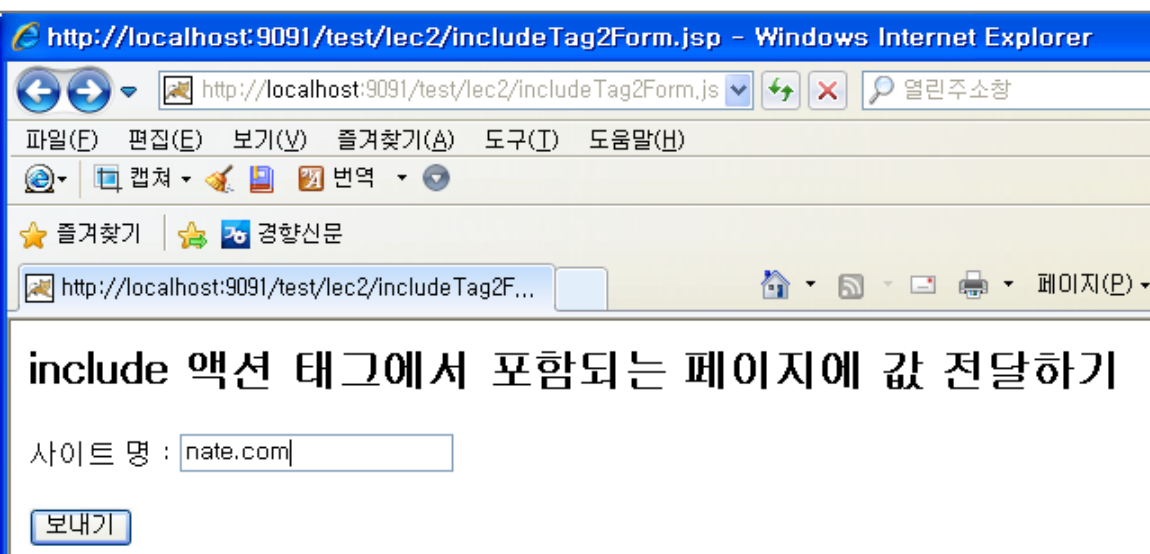
```
</jsp:include>
```

```
<hr>
```

```
includeTag2.jsp 페이지의 나머지 내용입니다.<p>
```

예제-includeTagTop2.jsp

```
<%@ page contentType="text/html; charset=utf-8"%>
<%
    String siteName = request.getParameter("siteName");
%>
포함되는 페이지 includeTagTop2.jsp 입니다.<p>
<b>사이트명 : <%=siteName%></b>
```



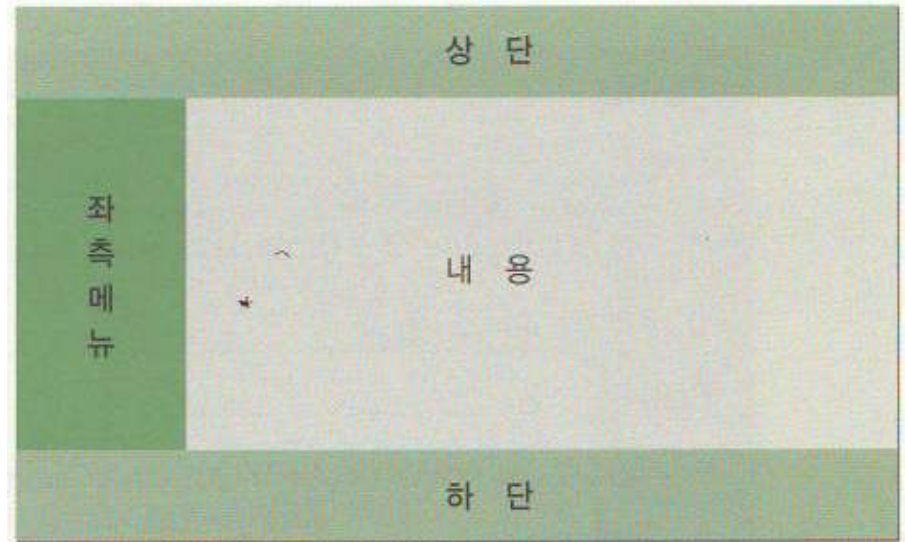


include 액션 태그

- [3] jsp 페이지의 중복 영역 처리: jsp 페이지의 모듈화
 - 여러 명이 작업 했을 때 페이지의 통일성을 어떻게 유지할 것인가
 - => 템플릿 페이지를 작성해서 활용
 - 중복되는 페이지를 어떻게 처리할 것인가
 - => 중복되는 페이지를 모듈화해서 사용
- 페이지를 모듈화하는 이유
 - 사이트 유지보수가 쉽기 때문
 - 페이지를 모듈화하면 중복되는 페이지만 수정해도 모든 페이지가 수정된 것과 같은 효과를 줌

웹 페이지의 구조

```
<TABLE>
<TR>
  <TD colspan="2">상단</TD>
</TR>
<TR>
  <TD>좌측 메뉴</TD>
  <TD>중앙의 내용</TD>
</TR>
<TR>
  <TD colspan="2">하단</TD>
</TR>
</TABLE>
```





웹 페이지의 구조

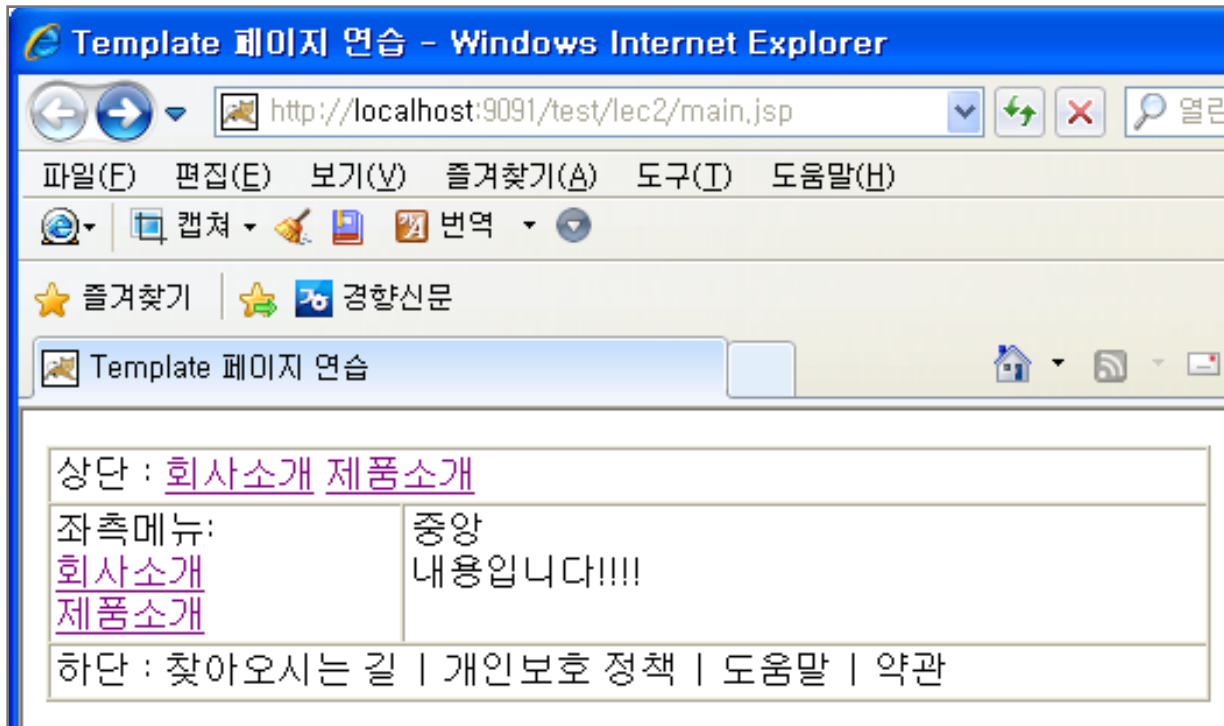
```
<TABLE>
<TR>
  <TD colspan="2"><jsp:include page="top.jsp" flush="false"/></TD>
</TR>
<TR>
  <TD><jsp:include page="left.jsp" flush="false"/></TD>
  <TD><jsp:include page="<%=content%>" flush="false"/></TD>
</TR>
<TR>
  <TD colspan="2"><jsp:include page="bottom.jsp" flush="false"/></TD>
</TR>
</TABLE>
```

상단, 좌측, 하단은 같은 페이지를 유지하고, 중앙의 내용만 바뀌는 이것은 <jsp:include>액션태그를 이용

=> 중앙의 내용은 매 페이지마다 바뀌므로 **page**의 속성값을 표현식을 사용해서 그때 그때 표시되는 페이지를 변동시킬 수 있음

예제-main.jsp

```
<%@ page contentType = "text/html; charset=utf-8" %>
<jsp:forward page="templateTest.jsp" >
  <jsp:param name="CONTENTPAGE" value="content.jsp"/>
</jsp:forward>
```



main.jsp의 실행결과로 templateTest.jsp가 표시



예제-templateTest.jsp

```
<%@ page contentType = "text/html; charset=utf-8" %>
<%
    String contentPage = request.getParameter("CONTENTPAGE");
%>
<html>
<head><title>Template 페이지 연습</title></head>
<body>

<table width="500" border="1" cellpadding="2" cellspacing="0">
<tr>
    <td colspan="2">
        <jsp:include page="top.jsp" flush="false" />
    </td>
</tr>
<tr>
    <td width="150" valign="top">
        <jsp:include page="left.jsp" flush="false" />
    </td>
    <td width="350" valign="top">
        <jsp:include page="<%= contentPage %>" flush="false" />
    </td>
</tr>
<tr>
    <td colspan="2">
        <jsp:include page="bottom.jsp" flush="false" />
    </td>
</tr>
</body></html>
```




예제

```
<%@ page contentType = "text/html; charset=utf-8" %>
```

top.jsp

상단 :

```
<a href="">회사소개</a>
```

```
<a href="">제품소개</a>
```

```
<%@ page contentType = "text/html; charset=utf-8" %>
```

left.jsp

좌측메뉴:


```
<a href="">회사소개</a><br>
```

```
<a href="">제품소개</a>
```

```
<%@ page contentType = "text/html; charset=utf-8" %>
```

bottom.jsp

하단 : 찾아오시는 길 | 개인정보 보호 정책 | 도움말 | 약관

```
<%@ page contentType="text/html; charset=utf-8"%>
```

centent.jsp

중앙

내용입니다!!!!



include 디렉티브

- `<%@include%>` 디렉티브
 - 중복되는 부분을 포함시켜 사용함
 - `<jsp:include>` 액션 태그는 결과를 포함시키고,
 - include 디렉티브는 코드를 복사해서 함께 서블릿으로 변환함
 - include 디렉티브는 조각코드를 삽입할 때 사용함
 - include 디렉티브는 코드 차원에서 포함되므로 주로 공용변수, 저작권 표시와 같은 중복 문장에 사용됨



include 디렉티브

```
<!--color.jsp-->
<% //공용변수들
    String bodyback_c="#e0ffff";
    String back_c="#8fb8f";
    String title_c="#5f9ea0";
    String value_c="#b0e0e6";
    String bar="#778899";
%>
```

- 공용변수를 저장해서 필요시에 사용함
 - <%@include file="color.jsp" %>



jsp 페이지의 흐름 제어

- forward 액션 태그 (<jsp:forward> 액션 태그)
 - 페이지의 제어 흐름을 현재 페이지에서 다른 페이지로 이동시킬 때 사용됨
 - 다른 페이지로 프로그램의 제어를 이동할 때 사용됨
 - jsp 페이지 내에 <jsp:forward> 액션 태그를 만나게 되면, 그 전까지 출력 버퍼에 저장되어 있던 내용을 제거하고 <jsp:forward> 액션 태그가 지정하는 페이지로 이동함
 - 사용자가 입력한 값에 따라 여러 페이지로 이동해야 할 경우 사용



forward 액션 태그

■ [1] 사용법

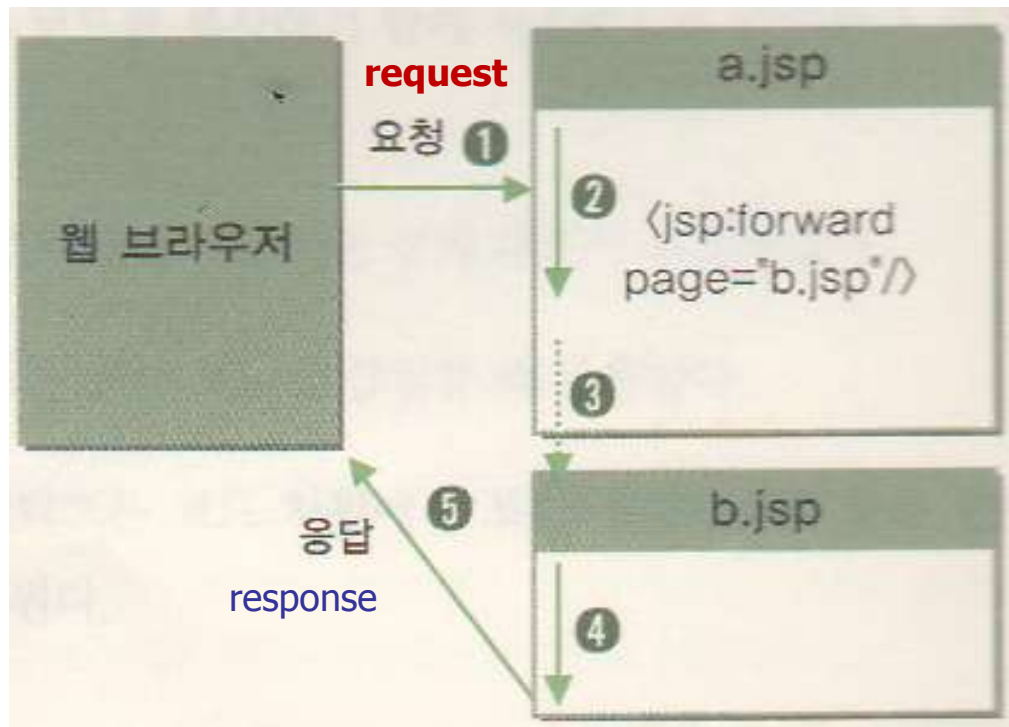
```
<jsp:forward page="이동할 페이지명"/>  
<jsp:forward page="이동할 페이지명"></jsp:forward>  
<jsp:forward page='<%=expression + ".jsp" %>' />
```

■ page 속성 - 이동할 페이지 명 기술

- 웹 어플리케이션 상대경로나 절대 경로로 지정, 표현식도 사용 가능

forward 액션 태그

- <jsp:forward> 액션 태그의 처리 과정



a.jsp와 b.jsp는 같은 request 를 공유함



forward 액션 태그

- `<jsp:forward>` 액션 태그의 처리 과정
 - 웹 브라우저에서 웹 서버로 a.jsp 페이지를 요청
 - 요청된 a.jsp 페이지를 수행함
 - a.jsp 페이지를 수행하다가 `<jsp:forward>` 액션 태그를 만나면 지금까지 저장되어 있는 출력 버퍼의 내용을 제거하고 page 속성에서 지정한 b.jsp로 프로그램 제어를 이동함
 - b.jsp 페이지를 수행함
 - b.jsp 페이지를 수행한 결과를 웹 브라우저에게 응답함
- 요청된 페이지는 a.jsp 이지만 웹 브라우저에 표시되는 내용은 b.jsp 임
- 이 방법을 활용해서 모듈화된 jsp 페이지의 흐름을 제어함

예제

http://localhost:9091/test/lec2/forwardTag1Form.jsp -

http://localhost:9091/test/lec2/forwardTag1Form.jsp

파일(F) 편집(E) 보기(V) 즐겨찾기(A) 도구(T) 도움말(H)

캡처 번역

즐거찾기 | 경향신문

http://localhost:9091/test/lec2/forwardTag1F...

Forward 사용법 예제

아이디 :

패스워드 :

http://localhost:9091/test/lec2/forwardTagFrom1.jsp - Windo

http://localhost:9091/test/lec2/forwardTagFrom1.jsp

파일(F) 편집(E) 보기(V) 즐겨찾기(A) 도구(T) 도움말(H)

캡처 번역

즐거찾기 | 경향신문

http://localhost:9091/test/lec2/forwardTagFr...

포워딩되는 페이지: forwardTagTo1.jsp

hong님의

패스워드는 1234입니다.

웹 브라우저의 주소에는 forwardTagFrom1.jsp 페이지가 표시되지만 웹 브라우저에 표시되는 내용은 forwardTagTo1.jsp 페이지임



예제-forwardTag1Form.jsp

```
<%@ page contentType="text/html;charset=utf-8"%>
<h1>Forward 사용법 예제</h1>
<FORM METHOD=POST ACTION="forwardTagFrom1.jsp">
아이디 : <INPUT TYPE="text" NAME="id"><p>
패스워드 : <INPUT TYPE="password" NAME="pwd"><p>
<INPUT TYPE="submit" VALUE="보내기">
</FORM>
```

forwardTag1Form.jsp

```
<%@ page contentType="text/html;charset=utf-8"%>
<h2>포워딩하는 페이지: forwardTagFrom1.jsp</h2>
<%
    request.setCharacterEncoding("utf-8");
%>
<html>
<body>
    forwardTagFrom1.jsp의 내용 입니다.<br>
    화면에 절대 표시 안됩니다.
    <jsp:forward page="forwardTagTo1.jsp"/>
</body>
</html>
```

forwardTagFrom1.jsp



예제

```
<%@ page contentType="text/html;charset=utf-8"%>
<h2>포워딩되는 페이지: forwardTagTo1.jsp</h2>
<%
    String id = request.getParameter("id");
    String password = request.getParameter("pwd");
%>
<b><%=id%></b>님의<p>
패스워드는 <b><%=password%></b>입니다.
```

forwardTagTo1.jsp



forward 액션 태그

- [2]<jsp:forward> 액션 태그에서 포워딩되는 페이지에 값 전달하기
 - <jsp:forward> 액션 태그에서 <jsp:param> 태그로 프로그램의 제어가 이동할 페이지에 파라미터 값을 전달할 때 사용
 - <jsp:include> 액션 태그에서와 동일하게 사용

```
<jsp:forward page="이동할 페이지명">  
    <jsp:param name="paramName1" value="var1"/>  
    <jsp:param name="paramName2" value="var2"/>  
</jsp:forward>
```

b.jsp?no=7&name=홍길동



자바빈



자바빈의 개요

■ 자바빈

- 프로그램을 어떠한 단위 별로 작성해서 레고 블럭처럼 필요시에 필요한 모듈을 끼워서 사용하고, 표준화된 모듈을 여러 가지 형태로 가지고 있으면 어떤 프로그램이든지 쉽게 작성할 수 있음
- 자바빈 - 웹 프로그래밍에서 모듈을 작성해서 재 사용이 가능하도록 해 주는 것
- 로직을 담고 있는 자바 클래스, 자바로 작성되어진 컴포넌트들

■ 자바빈을 사용하는 목적

- jsp 페이지가 화면 표출 부분과 로직들이 혼재함으로 인한 복잡한 구성을 가능한 피하고
- jsp 페이지의 로직 부분을 분리해서 코드를 재 사용함으로 프로그램의 효율을 높이는 것



자바빈의 개요

- MVC

- 뷰 - jsp 페이지
- 모델 - 자바빈
 - 프로그램 로직을 가지고 있고, DB와 연동을 해서 작업을 처리하는 역할을 함
- 컨트롤러 - 서블릿



자바빈 작성

- 자바빈은 자바의 클래스이므로 자바의 클래스를 만드는 것과 같은 규칙을 가짐

[순서]

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

- 자바빈의 클래스 선언은 접근 제어자로 `public`을 사용
- 멤버 변수는 접근 제어자로 `private`을 사용
 - 자바빈에서는 멤버변수를 프로퍼티(property)라고 부름
- 프로퍼티 - 값을 저장하기 위한 필드
 - jsp 페이지의 내용을 DB에 저장하거나 DB에 저장된 내용을 jsp 페이지에 표출할 때 중간 데이터 저장소로 사용됨



자바빈 작성 예제-SimpleBean.java

```
package study.javaBean;

public class SimpleBean {
    //프로퍼티
    private String msg ;

    //getter
    public String getMsg() {
        return msg;
    }

    //setter
    public void setMsg(String msg) {
        this.msg = msg;
    }
}
```




자바빈과 <jsp:useBean>액션 태그 의 연동



자바빈 관련 액션 태그

자바빈 관련 액션 태그	내용
<code><jsp:useBean id="" class="" scope="" /></code>	자바빈 객체를 생성
<code><jsp:setProperty name="" property="" value="" /></code>	생성된 자바빈 객체에 프로퍼티 값을 저장
<code><jsp:getProperty name="" property="" /></code>	생성된 자바빈 객체에서 저장된 프로퍼티 값을 가져옴



<jsp:useBean>액션 태그

■ 자바빈 객체를 생성

```
<jsp:useBean id="빈 이름" class="자바빈 클래스 이름" scope="범위" />
```

- id : 생성될 자바빈 객체의 이름을 명시
- class : 객체가 생성될 자바빈 클래스명을 기술하는 곳
 - 패키지명을 포함한 자바 클래스의 풀네임을 기술
- scope : 자바빈 객체의 유효 범위로 자바빈 객체가 공유되는 범위를 지정
 - 속성값 : page, request, session, application
 - 생략시 기본값은 page



<jsp:useBean>액션 태그

```
<jsp:useBean id="sb" class="study.javaBean.SimpleBean"
  scope="page" />
```

- SimpleBean 클래스의 자바빈 객체를 생성
 - 생성되는 인스턴스명(레퍼런스명)이 sb
 - SimpleBean 클래스의 멤버변수나 메소드에 접근하려면 sb 레퍼런스를 사용해야 함
 - 자바에서 객체를 생성하는 다음 문장과 동일

```
SimpleBean sb = new SimpleBean();
```



<jsp:useBean>액션 태그

- <jsp:useBean>액션 태그에서 id 속성값에 지정한 이름이 이미 존재하는 경우 - 자바빈 객체를 새로 생성하는 것이 아니라 기존에 생성된 객체를 그대로 사용함
 - id, class, scope 속성값이 모두 동일해야 같은 객체가 됨

```
<jsp:useBean id="sb" class="study.javaBean.SimpleBean"
            scope="page" />
```

.....

```
<jsp:useBean id="sb" class="study.javaBean.SimpleBean"
            scope="page" />
```



<jsp:setProperty>액션 태그

- <jsp:setProperty>액션 태그 - 자바빈 객체의 프로퍼티 값을 저장하기 위해 사용됨

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

- name : 자바빈 객체의 이름을 명시하는 곳
- property : 프로퍼티명을 기술하는 곳
- value : 프로퍼티에 저장할 값을 기술하는 곳

```
<jsp:useBean id="sb" class="study.javaBean.SimpleBean"
    scope="page">
    <jsp:setProperty name="sb" property="msg" />
</jsp:useBean>
```



<jsp:setProperty>액션 태그

- <jsp:setProperty name="sb" property="msg" /> 액션 태그
 - 자바빈 클래스의 setMsg() 메서드와 자동 연동됨
 - 프로퍼티명 msg 는 자바빈 클래스의 프로퍼티 msg를 의미함

```
public void setMsg(String msg) {  
    this.msg = msg;  
}
```

- 프로퍼티가 많을 경우

```
<jsp:useBean id="sb" class="study.javaBean.SimpleBean"  
    scope="page">  
    <jsp:setProperty name="sb" property="*" />  
</jsp:useBean>
```

- **property** 속성값을 * 를 주면 모든 프로퍼티 값이 세팅됨
- <= 폼으로부터 넘어오는 파라미터의 이름이 프로퍼티의 이름과 일치해야 함



<jsp:setProperty>액션 태그

■ 사용자 입력 폼

```
사용자 ID : <input type="text" name="id" size="10" maxlength="10">  
<input type="button" value="ID중복확인" onclick="openConfirmid(this.form)">
```

■ 자바빈을 사용하는 JSP 페이지

```
<jsp:useBean id="sb" class="study.javaBean.SimpleBean"  
    scope="page">  
    <jsp:setProperty name="sb" property="id" />  
</jsp:useBean>
```

■ 자바빈 클래스

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




<jsp:setProperty>액션 태그

■ 사용자 입력 폼

```
사용자 ID : <input type="text" name="userid" size="10" maxlength="10">  
<input type="button" value="ID중복확인" onclick="openConfirmid(this.form)">
```

■ 자바빈을 사용하는 JSP 페이지

```
<jsp:useBean id="sb" class="study.javaBean.SimpleBean"  
    scope="page">  
    <jsp:setProperty name="sb" property="id" param="userid" />  
</jsp:useBean>
```

■ 자바빈 클래스

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

폼으로부터 넘어온 파라미터명과 자바빈의 프로퍼티가 일치하지 않는 경우
⇒<jsp:setProperty> 액션 태그에 **param** 속성을 기술해야 함
⇒**param** 속성값에는 폼으로부터 넘어온 파라미터명을 기술함



<jsp:getProperty>액션 태그

- <jsp:getProperty>액션 태그 – 자바빈 객체에서 저장된 프로퍼티 값을 사용하기 위해 사용됨

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

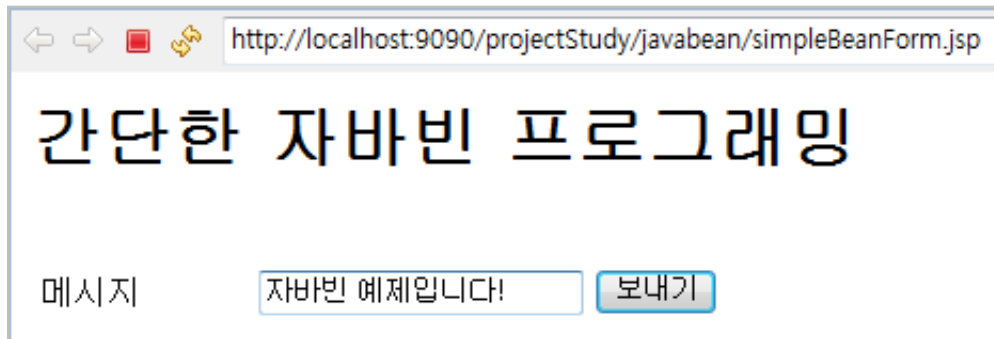
- name : 자바빈 객체의 이름을 명시하는 곳
- property : 프로퍼티명을 기술하는 곳

```
<jsp:useBean id="sb" class="study.javaBean.SimpleBean"
    scope="page">
    <jsp:getProperty name="sb" property="msg" />
</jsp:useBean>
```

```
public String getMsg() {
    return msg;
}
```

자바빈 객체를 생성해서 사용하는 JSP 페이지 작성 예제

- simpleBeanForm.jsp 페이지의 메시지에 입력된 값이 파라미터로 simpleBean_ok.jsp 페이지로 넘어옴
- simpleBean_ok.jsp 페이지는 SimpleBean 클래스의 객체를 <jsp:useBean>액션 태그를 사용해서 생성한 후, <jsp:setProperty>액션 태그를 사용해서 프로퍼티 값을 저장한 후 <jsp:getProperty>액션 태그를 사용해서 저장된 프로퍼티 값을 화면에 표출



← → [Icons] http://localhost:9090/projectStudy/javabean/simpleBeanForm.jsp

간단한 자바빈 프로그래밍

메시지



메시지를 입력하는 simpleBeanForm.jsp

```
<%@ page contentType="text/html; charset=utf-8" %>
```

```
<html>
```

```
<body>
```

```
<h1>간단한 자바빈 프로그래밍</h1>
```

```
<br>
```

```
<form action="simpleBean_ok.jsp" method="post" >
```

```
    메시지 <input type="text" name="msg" size="20" maxlength="30">
```

```
    <input type="submit" name="send" value="보내기">
```

```
</form>
```



자바빈 객체를 생성하고 프로퍼티 값을 저장 및 사용하는 simpleBean_ok.jsp

```
<%@ page contentType="text/html; charset=utf-8" %>
```

```
<% request.setCharacterEncoding("utf-8");%>
```

```
<jsp:useBean id="sb" class="study.javaBean.SimpleBean" />
```

```
<jsp:setProperty name="sb" property="msg" />
```

```
<html>
```

```
<body>
```

property 속성에 기술한 **msg** 프로퍼티의 값은 **simpleBeanForm.jsp**의 **input** 태그에서 지정한 **msg** 변수가 파라미터로 넘어오면서 넘겨줌

이때 넘어오는 파라미터인 **msg**와 **SimpleBean** 클래스의 **setMsg()** 메소드가 자동 연동되면서 자바빈의 프로퍼티 값이 저장되게 됨

```
<h1>간단한 자바빈 프로그래밍</h1>
```

```
<br>
```

```
메시지: <jsp:getProperty name="sb" property="msg" />
```

```
</body>
```

```
</html>
```



자바빈 사용

- 자바빈을 jsp 페이지에서 사용하려면
 - 자바빈 클래스를 생성
 - 데이터를 저장하는 자바빈
 - 데이터베이스와 연동하는 빈
 - 자바빈을 사용할 jsp 페이지를 작성



예제

```
package com.study;
```

```
public class SimpleBean {  
    private String userid;  
    private String name;  
    private int no;  
  
    public String getUserid() {  
        return userid;  
    }  
    public void setUserid(String userid) {  
        this.userid = userid  
    }  
    public String getName() {  
        return name;  
    }  
    public void setName(String name) {  
        this.name = name;  
    }  
    ...  
}
```



예제-simplebeanTest.jsp

```
<%@ page language="java" contentType="text/html; charset=utf-8"    pageEncoding="utf-8"%>
<!DOCTYPE html >
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>simplebeanTest1.jsp</title>
</head>
<body>
<form name="frm1" method="post" action="simplebeanOk.jsp">
    아이디 : <input type="text" name="userid"><br>
    이름 : <input type="text" name="name"><br>
    번호 : <input type="text" name="no"><br>
    <input type="submit" value="전송">
</form>
</body>
</html>
```



```
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="utf-8"%>
<%
    request.setCharacterEncoding("utf-8");
%>
<jsp:useBean id="sb" class="com.study.SimpleBean" scope="page"></jsp:useBean>
<%-- <jsp:setProperty property="userid" name="sb"/> --%>
<jsp:setProperty property="*" name="sb"/>
<%
    /*
    String userid = request.getParameter("userid");
    SimpleBean sbean = new SimpleBean();
    sbean.setUserId(userid);

    //sbean.setName("김연아");
    */
%>

<h2>자바빈과 액션태그 연동하기</h2>
<hr>
아이디는 <jsp:getProperty property="userid" name="sb"/> 입니다.<br>
이름은 <jsp:getProperty property="name" name="sb"/> 입니다.<br>
번호는 <jsp:getProperty property="no" name="sb"/> 입니다.<br>
<%=sbean.getUserId()%>
```



예제2

```
package com.study;

public class CounterBean {
    private int count;

    public int getCount() {
        return count;
    }

    public void setCount(int count) {
        this.count = count;
    }
}
```



예제2-Page 영역

```
<%@ page language="java" contentType="text/html; charset=utf-8"    pageEncoding="utf-8"%>
<h2>page scope 테스트</h2>
<form name="frm1" method="post" action="scope1.jsp">
    카운트 : <input type="text" name="count"><br>
    <input type="submit" value="전송">
</form>
```

```
<%@ page language="java" contentType="text/html; charset=utf-8"    pageEncoding="utf-8"%>
<jsp:useBean id="countbean" class="com.study.CounterBean" scope="page"></jsp:useBean>
<jsp:setProperty property="count" name="countbean"/>
<h2>scope1.jsp</h2>
count : <jsp:getProperty property="count" name="countbean"/>
<br>
<a href="result1.jsp">result1.jsp</a>
```

예제2

```
<%@ page language="java" contentType="text/html; charset=utf-8"    pageEncoding="utf-8"%>
<jsp:useBean id="countbean" class="com.study.CounterBean" scope="page"></jsp:useBean>
<h2>result1.jsp</h2>
<hr>

count : <jsp:getProperty property="count" name="countbean"/>
<br>
<a href="scope1.jsp">scope1.jsp</a>
```

page scope 테스트

카운트 : 10



scope1.jsp

count : 10
[result1.jsp](#)

result1.jsp

count : 0
[scope1.jsp](#)



Request 영역

```
<%@ page language="java" contentType="text/html; charset=utf-8"    pageEncoding="utf-8"%>
<h2>request scope 테스트</h2>
<form name="frm1" method="post" action="scope2.jsp">
    카운트 : <input type="text" name="count"><br>
    <input type="submit" value="전송">
</form>
```

```
<%@ page language="java" contentType="text/html; charset=utf-8"    pageEncoding="utf-8"%>
<jsp:useBean id="countbean" class="com.study.CounterBean" scope="request"></jsp:useBean>
<jsp:setProperty property="count" name="countbean"/>
<h2>scope2.jsp</h2>
count : <jsp:getProperty property="count" name="countbean"/>
<br>
<!-- <a href="result2.jsp">result2.jsp</a> -->
<jsp:forward page="result2.jsp"></jsp:forward>
```

Request 영역

```
<%@ page language="java" contentType="text/html; charset=utf-8" pageEncoding="utf-8"%>
<jsp:useBean id="countbean" class="com.study.CounterBean" scope="request"></jsp:useBean>
<h2>result2.jsp</h2>
<hr>
```

```
count : <jsp:getProperty property="count" name="countbean"/>
<br>
<a href="scope2.jsp">scope2.jsp</a>
```

request scope 테스트

카운트 :



result2.jsp

count : 10
[scope2.jsp](#)

session 영역

```
<%@ page language="java" contentType="text/html; charset=utf-8" pageEncoding="utf-8"%>
<jsp:useBean id="countbean" class="com.study.CounterBean" scope="session"></jsp:useBean>
<jsp:setProperty property="count" name="countbean"/>
<h2>scope3.jsp</h2>
count : <jsp:getProperty property="count" name="countbean"/>
<br>
<a href="result3.jsp">result3.jsp</a>
<br><br>
세션 아이디 : <%=session.getId() %>
```

session scope 테스트

카운트 :



scope3.jsp

count : 10

[result3.jsp](#)

세션 아이디 : B4159DD980939FC66B5A2A43010B8F33

result3.jsp

count : 10

[scope3.jsp](#)

세션 아이디 : B4159DD980939FC66B5A2A43010B8F33



session 영역

```
<%@ page language="java" contentType="text/html; charset=utf-8"    pageEncoding="utf-8"%>
<jsp:useBean id="countbean" class="com.study.CounterBean" scope="session"></jsp:useBean>
<h2>result3.jsp</h2>
<hr>
```

```
count : <jsp:getProperty property="count" name="countbean"/>
```

```
<br>
```

```
<a href="scope3.jsp">scope3.jsp</a>
```

```
<br><br>
```

```
세션 아이디 : <%=session.getId() %>
```


application 영역

scope4.jsp

count : 10
[result4.jsp](#)

세션 아이디 : F72080B1A4ED620766CDDF4E2E6D7EC9

```
<%@ page language="java" contentType="text/html; charset=utf-8" pageEncoding="utf-8"%>
<jsp:useBean id="countbean" class="com.study.CounterBean" scope="application">
</jsp:useBean>
<jsp:setProperty property="count" name="countbean"/>
<h2>scope4.jsp</h2>
count : <jsp:getProperty property="count" name="countbean"/>
<br>
<a href="result4.jsp">result4.jsp</a>
<br><br>
세션 아이디 : <%=session.getId() %>
```

scope4.jsp

count : 10
[result4.jsp](#)

세션 아이디 : CF41BADE26D57421C575FDC5C2053BDC

result4.jsp

count : 10
[scope4.jsp](#)

세션 아이디 : CF41BADE26D57421C575FDC5C2053BDC

application scope 테스트

카운트 :



application 영역

```
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="utf-8"%>
<jsp:useBean id="countbean" class="com.study.CounterBean" scope="application">
</jsp:useBean>
<h2>result4.jsp</h2>
<hr>

count : <jsp:getProperty property="count" name="countbean"/>
<br>
<a href="scope4.jsp">scope4.jsp</a>
<br><br>
세션 아이디 : <%=session.getId() %>
```

예제

```
<%@ page contentType="text/html;charset=utf-8"%>
<%
    String name = request.getParameter("name");
    String selectedColor = request.getParameter("selectedColor");
%>
<h2>포워딩되는 페이지 - <%=selectedColor+".jsp"%></h2>
<b><%=name%></b>님의 좋아하는 색은 "<%=selectedColor%>"이고
기분의 안정과 온화함을 상징하는 색입니다.<br>
" border="0" width="70" height="30">
```

green.jsp

```
<%@ page contentType="text/html;charset=utf-8"%>
<%
    String name = request.getParameter("name");
    String selectedColor = request.getParameter("selectedColor");
%>
<h2>포워딩되는 페이지 - <%=selectedColor+".jsp"%></h2>
<b><%=name%></b>님의 좋아하는 색은 "<%=selectedColor%>"이고
생명을 상징하는 색입니다.
" border="0" width="70" height="30">
```

red.jsp

예제

```
<%@ page contentType="text/html;charset=utf-8"%>
```

yellow.jsp

```
<%
```

```
    String name = request.getParameter("name");
```

```
    String selectedColor = request.getParameter("selectedColor");
```

```
%>
```

```
<h2>포워딩되는 페이지 - <%=selectedColor+".jsp"%></h2>
```

```
<b><%=name%></b>님의 좋아하는 색은 "<%=selectedColor%>"이고  
빛의 밝음과 따뜻함을 상징하는 색입니다.<br>
```

```
" border="0" width="70" height="30">
```

```
<%@ page contentType="text/html;charset=utf-8"%>
```

blue.jsp

```
<%
```

```
    String name = request.getParameter("name");
```

```
    String selectedColor = request.getParameter("selectedColor");
```

```
%>
```

```
<h2>포워딩되는 페이지 - <%=selectedColor+".jsp"%></h2>
```

```
<b><%=name%></b>님의 좋아하는 색은 "<%=selectedColor%>"이고  
자기탐구와 내적성장을 상징하는 색입니다.<br>
```

```
" border="0" width="70" height="30">
```