2020년도 2학기

웹프로젝트실습 과목

- 7주차 -

장용미

H.P: 010-3309-4849

E-MAIL: changmi29@dongyang.ac.kr

eClass의 쪽지

지난 주의 내용

1. JSP 내장객체

2. JSP의 4개 영역(scope)

- **page 영역 :** 1개의 jsp파일 (pageContext내장객체)
- **request 영역 :** 하나의 request과 관련된 jsp파일 (request내장객체)
- **session 영역 :** 하나의 브라우저 영역 (session내장객체)
- **application 영역 :** 하나의 웹어플리케이션 영역 (application내장객체)

page영역 < reques영역 < session영역 < application영역

3. 한 화면의 모듈(module)화

- **액션 태그 include :** <jsp:include page="포함할 파일">
- **지시어 include :** <%@ include file="포함할 파일" %> 이번주

목차

01. JSP 내장객체 개요

원리를 알면 IT가 맛있다
IT COOKBOOK

JSP 내장객체란?

- **JSP 내장객체란 'JSP 내에서 선언하지 않고 사용할 수 있는 객체'라는 의미에서 붙여진 이름.**
- 구조적으로는 JSP가 서블릿 형태로 자동 변환된 코드 내에 포함되어 있는 멤버변수, 메서드 매개변수 등의 각종 참조 변수(객체)를 말함.

[표 6-1] JSP 내장객체

9개

참조 변수 이름(내장객체)	자바 클래스	주요 역할
request	javax.servlet.http.HttpServletRequest	HTML 폼 요소의 선택 값 등 사용자 입력 정보를 읽으려고 사용한다.
response	javax.servlet.http.HttpServletResponse	사용자 요청에 대한 응답을 처리하려고 사용한다.
pageContext	javax.servlet.jsp.PageContext	현재 JSP 실행에 대한 context 정보를 참조하려고 사용한다.
session	javax.servlet.http.HttpSession	클라이언트의 세션 정보를 처리하려고 사용한다.
application	javax.servlet.ServletContext	웹 서버의 애플리케이션 처리와 관련된 정보를 참조하려고 사용한다.
out	javax.servlet.jsp.JspWriter	사용자에게 전달하기 위한 output 스트림을 처리하려고 사용한다.
config	javax.servlet.ServletConfig	현재 JSP의 초기화 환경을 처리하려고 사용한다.
page	java.lang.Object	현재 JSP의 클래스 정보를 보려고 사용한다.
exception	java.lang.Throwable	예외 처리를 하려고 사용한다.

01. JSP 내장객체 개요

1. JSP 내장객체의 구조적 특징

- 서블릿으로 변경된 JSP 코드는 모두 _jspService() 메서드에 위치함.
- 메서드 매개변수인 request, response 를 비롯한 pageContext, session, application, page, config, out 등 메서드 내에서 참조할 수 있는 참조변수들이 내장객체가 됨.

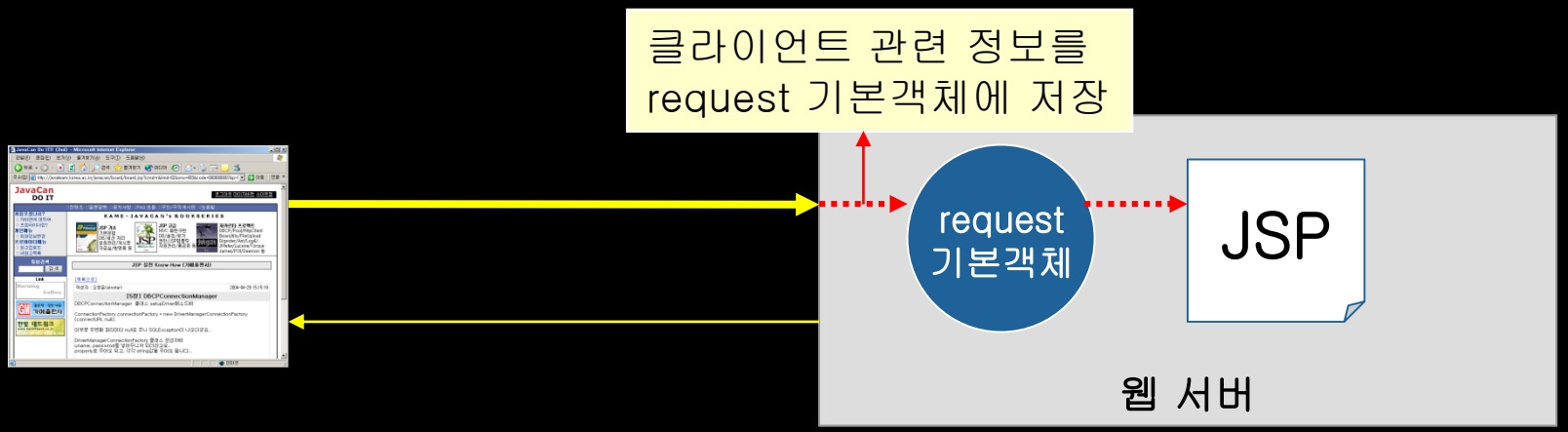
```
public void _jspService(HttpServletRequest request, HttpServlet Response response) throws java.io.IOException, ServletException {
    JspFactory _jspxFactory = null;
    javax.servlet.jsp.PageContext pageContext = null;
    HttpSession session = null;
    ServletConfig config = null;
    JspWriter out = null;
    Object page = this;
    JspWriter _jspx_out = null;
    try {
        pageContext = _jspxFactory.getPageContext(this, request, response, null, true, 8192, true);
        application = pageContext.getServletContext();
        config = pageContext.getServletConfig();
        session = pageContext.getSession();
        out = pageContext.getOut();
    ...
        사용자 작성 JSP 코드가 오는 위치....
}
```

2. 내장객체를 이용한 속성 관리 기법

- 내장객체가 단순히 특정한 기능을 제공하는 컨테이너 관리 객체라는 점 외에도 한 가지 특징이 있다. 바로 page, request, session, application 내장객체를 이용한 속성 관리 기법이다. 이들 내장객체는 각자 지정된 생명주기가 있으며 setAttribute( ), getAttribute( )라는 메서드를 통해 해당 생명주기 동안 자바 객체를 유지하는 기능을 제공한다.

- 속성 관리 기법은 '7절. JSP 내장객체와 속성 관리'에서 자세히 살펴볼 것이다.

request 기본 객체

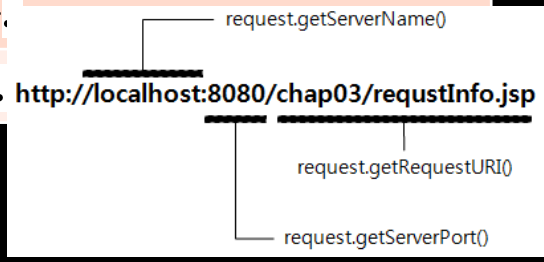


request 기본 객체가 제공하는 기능

- 클라이언트(웹 브라우저)와 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

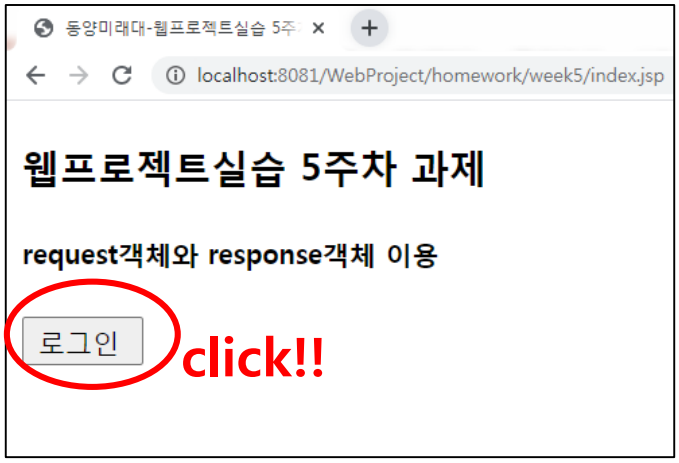
request 객체 주요메서드

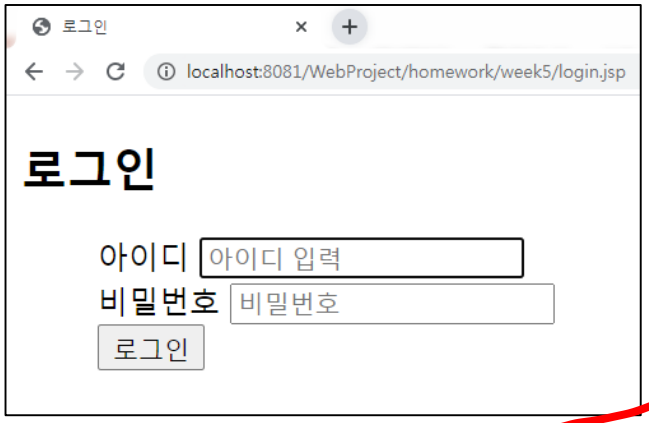
메서드	리턴타입	설명
getParmeter(String name)	String	이름이 name인 파라미터의 값을 구한다. 존재하지 않을 경우 null
getParameterValues(String name)	String[]	이름이 name인 모드 파라미터의 값을 배열로 구한다. 존재하지 않을 경우 null
getParameterNames()	java.util.Enumeration	웹 브라우저가 전송한 파라미터의 이름을 구한다
getParmeterMap()	java.util.Map	웹 브라우저가 전송한 파라미터의 맵 <파라미터 이름, 값> 세트로 구한다
getRemoteAddr()	**String**	웹 서버에 연결한 클라이언트의 **IP** 주소를 구한다**.** 게시판이나 방명록 등에서 글 작성자의 **IP** 주소가 자동으로 입력되기도 하는데, 이때 입력되는 **IP** 주소가 바로 이 메서드를 사용하여 구한 것이다**.**
getMethod()	**String**	웹 브라우저가 정보를 전송할 때 사용한 방식을 구한다**.**
getRequestURI()	**String**	웹 브라우저가 요청한 **URL**에서 경로를 구한다**.**
getContextPath()	**String**	**JSP** 페이지가 속한 웹 어플리케이션의 컨텍스트 경로를 구한다**.**
getServerName()	**String**	연결할 때 사용한 서버 이름을 구한다.
getServerPort()	**int**	서버가 실행중인 포트 번호를 구한다.



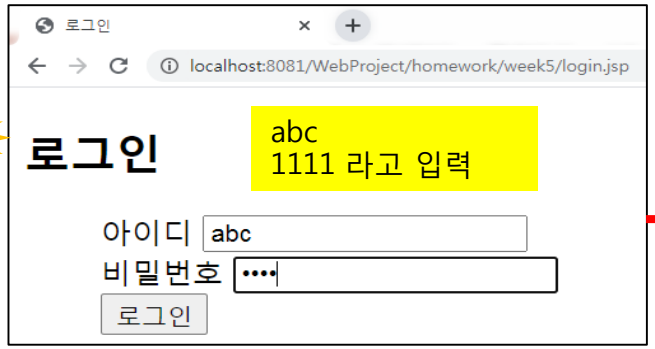
5주차 과제

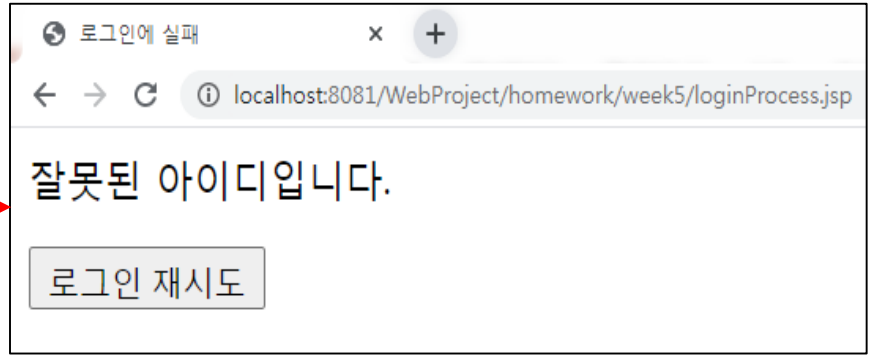
원리를 알면 IT가 맛있다
IT COOKBOOK



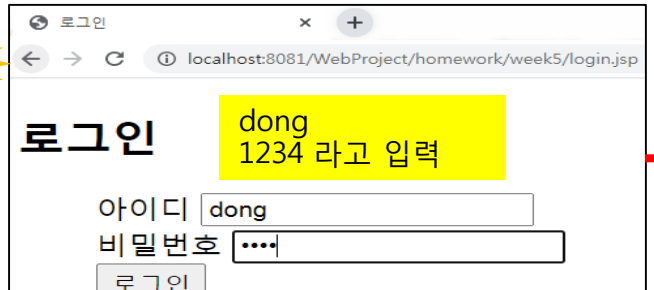


로그인 실패





로그인 성공

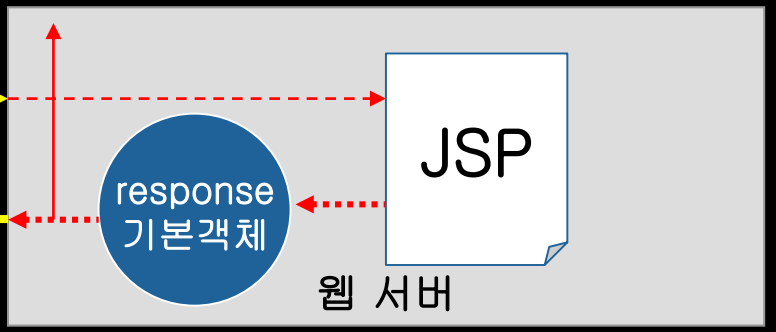


다시 첫화면 index.jsp로 이동!!!!!!

response 기본 객체

클라이언트에 전송할 정보를
response 기본객체에 저장

헤더, 쿠키 등 전송



response 기본 객체가 제공하는 기능

- 헤더 정보 전달
- 쿠키 전달
- 리다이렉트 지정

* 이외에 몇가지 기능이 더 있지만 거의 사용되지 않는다.

03. response

주요 소스코드 분석

forward 액션과 response.sendRedirect() 의 차이점

```
01 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
02 <% request.setCharacterEncoding("UTF-8"); %>
03     <jsp:forward page="page_control_end.jsp">
04         <jsp:param name="tel" value="000-000-0000" />
05 </jsp:forward>
```

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

- forward 액션은 최종적으로 전달되는 페이지에 파라미터(HTML 폼 입력값 등)를 함께 전달함.
- sendRedirect() 는 단순히 지정된 페이지로 최종 화면이 이동됨.
- forward 액션은 브라우저 URL에 최초 요청된 페이지 URL이 나타나고 sendRedirect() 는 최종 전달된 페이지의 URL이 표시됨.

response 기본 객체를 이용한 *redirect*

"웹 브라우저에게 다른 페이지로 이동하라고 지시하는 것을 리다이렉트(redirect)라고 한다"

1. a.jsp 요청 보냄

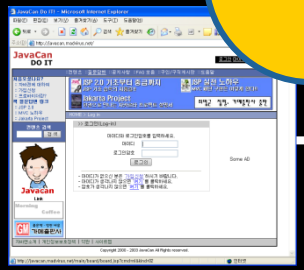
a.jsp

response.sendRedirect("b.jsp")

2. b.jsp로 리다이렉트 하라고 지정함

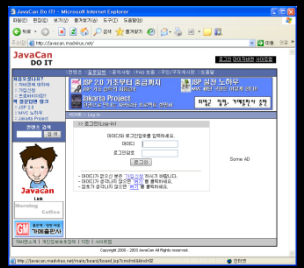
a.jsp

웹브라우저는 리다이렉트를 결과로 받은 경우 곧바로 지정한 페이지로 요청을 전송한다.



3. b.jsp를 요청함

b.jsp



4. b.jsp 응답

b.jsp

액션태그 forward : <jsp:forward>

1. a.jsp 요청 보냄

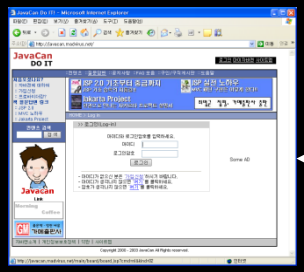
a.jsp

<jsp:forward page="b.jsp" />

2. b.jsp로 포워드 하라고 지정함

b.jsp

3. b.jsp 응답



out 기본 객체

"JSP 페이지가 생성하는 모든 내용은 out 기본 객체를 통해서 전송된다"

```
<html>
<head>
<title>제목</title>
</head>
<body>
안녕하세요?
```

```
<%= someValue %>
```

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

out 기본 객체

out 기본 객체를 통해서 클라이언트에 내용이 전달됨

04. out

out 내장객체

- **out은 출력 스트림으로써, 사용자 웹 브라우저로 출력하기 위한 내장 객체임.**
- 여러 예제에서 살펴본 것처럼 스크립트릿에서 브라우저로 텍스트를 출력하는 데 사용.
- out은 javax.servlet.jsp.JspWriter 객체의 참조 변수로, 버퍼 관련 메서드와 출력 관련 메서드로 구성되며 out을 이용해서 출력한 내용은 서버의 콘솔이 아닌 사용자에 전달된다.

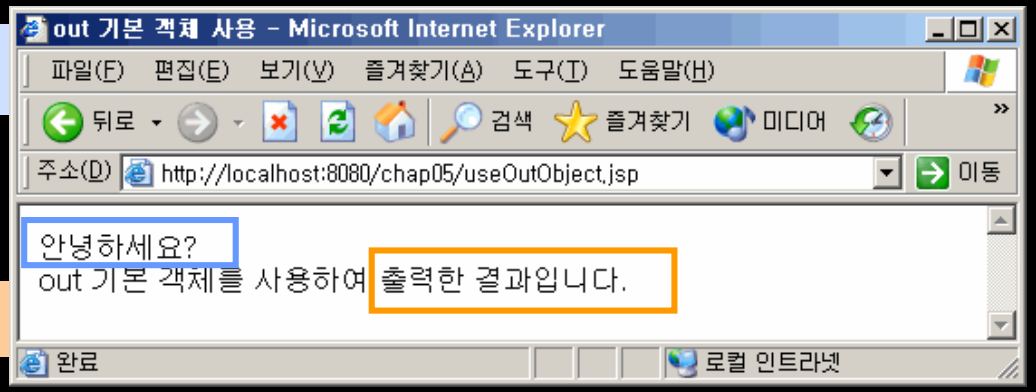
[표 6-4] out 주요 메서드

메서드	설명
getBufferSize()	output buffer의 크기를 바이트로 알려준다.
getRemaining()	남아 있는 버퍼의 크기 중 사용 가능한 비율을 알려준다.
clearBuffer()	버퍼에 있는 콘텐츠를 모두 지운다.
flush()	버퍼를 비우고 output stream도 비운다.
close()	output stream을 닫고 버퍼를 비운다.
println(content)	content의 내용을 newline과 함께 출력한다.
print(content)	content의 내용을 출력한다.

out 기본 객체의 출력 메소드

Example

```
<%@ page contentType = "text/html; charset=euc-kr" %>
<html>
<head><title>out 기본 객체 사용</title></head>
<body>
<%
    out.println("안녕하세요?");
%>
<br>
out 기본 객체를 사용하여
<%
    out.println("출력한 결과입니다.");
%>
</body></html>
```



언제 사용?

```
<% if(grade>10) { %>
<%= gradeStringA %>
<%  } else if (grade>5)  { %>
<%= gradeStringB %>
<% } %>
```

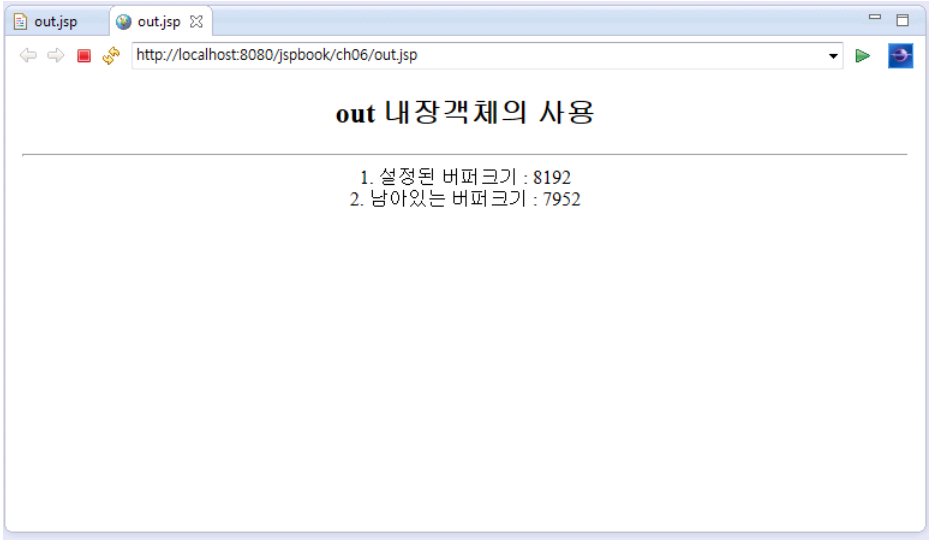
→

```
<%
    if(grade>10) {
       out.pringln(gradeStringA);
    } else if (grade>5)  {
       out.pringln(gradeStringB);
    }
 %>
```

04. out

[실습] out 참고 변수 메서드(out.jsp)

- 교재 p.210 ~ 211 참고



[그림 6-6] 실행 화면

04. out

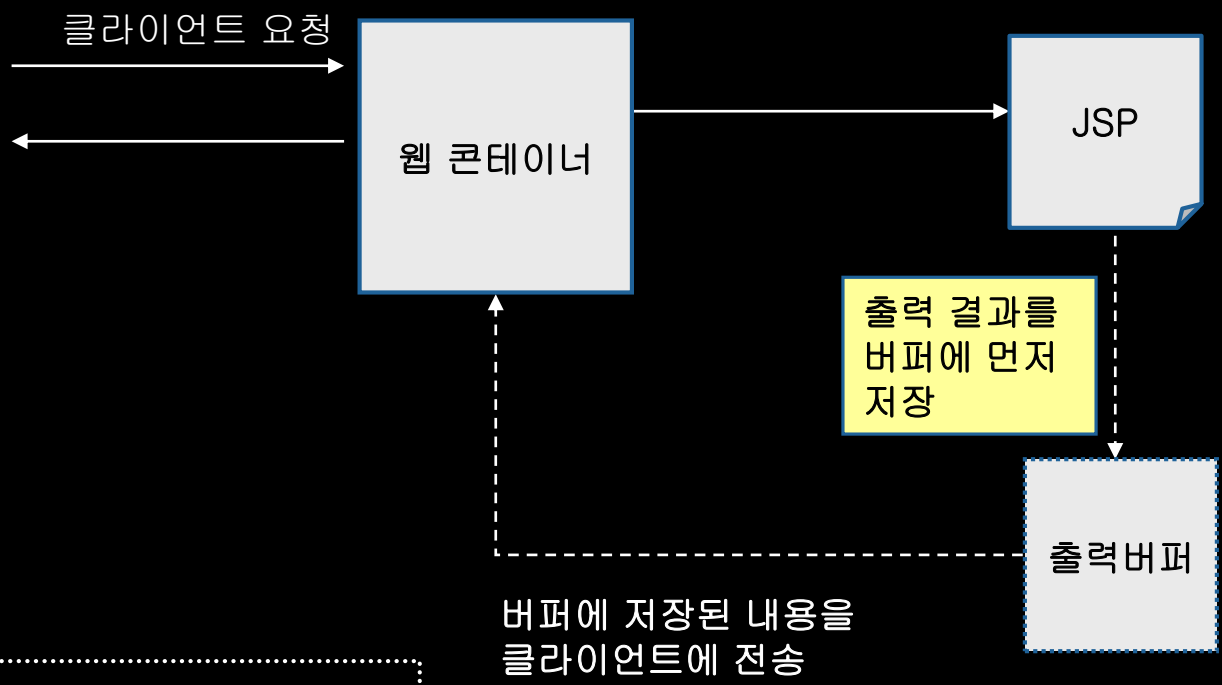
주요 소스 코드 분석

- println 을 제외한 버퍼관련 메서드들은 JSP 컨텐츠를 서버에서 클라이언트로 전달 할 때 원활한 출력 스트림 활용을 위해 제공되는 것으로 일반적으로 사용하는 경우는 많지 않음.

```
09 1. 설정된 버퍼 크기 : <%=out.getBufferSize() %> <BR>
10 2. 남아 있는 버퍼 크기 : <%=out.getRemaining() %> <BR>
11 <% out.flush(); %>
12 3. flush 후 남아 있는 버퍼 크기 : <%=out.getRemaining() %> <BR>
13 <% out.clear(); %>
14 4. clear 후 남아 있는 버퍼 크기 : <%=out.getRemaining() %> <BR>
15 <% out.close(): %>
16 5. close 후 남아 있는 버퍼 크기 : <%=out.getRemaining() %> <BR>
```

- 기본 설정된 버퍼 및 남아 있는 버퍼 크기를 구할 수 있음.
- flush() 메서드는 버퍼를 비우는 메서드로 버퍼 내용을 클라이언트로 전달하기 때문에 여기서 앞의 1, 2가 출력 된다.
- 3번의 경우 바로 뒤에 나오는 clear() 메서드에 의해 내용이 지워져 출력되지 않는다.
- 4번과 5번은 15라인의 close() 메서드로 인해 출력 스트림이 닫혀 내용이 출력되지 않는다.

출력 버퍼



버퍼의 장점

- 출력 성능의 향상
- 중간에 버퍼를 비우고 새로운 내용을 출력할 수 있음
- 버퍼가 다 차기 전까지 헤더를 변경할 수 있다.

05. session

session 내장객체

- HTTP 프로토콜이 비연결형 프로토콜이기 때문에 한 페이지가 출력된 다음에는 클라이언트와 서버의 연결이 끊어진다. 따라서 한번 로그인한 사용자가 로그아웃할 때까지 페이지를 이동해도 보관해야 할 정보가 있다면 이에 대한 처리가 매우 곤란해진다.
- 이러한 HTTP 프로토콜 문제점을 해결하려고 나온 것이 쿠키와 세션이다.
- session 은 javax.servlet.http.HttpSession 인터페이스의 참조 변수 이다.
- session 은 접속하는 사용자 별로 따로 생성되며 일정시간 유지되고 소멸된다.
- 이러한 세션의 특징을 이용해 setAttribute() 메서드를 이용해 임의의 값을 저장해 놓고 활용할 수 있음.
- 세션이 주로 사용되는 경우는 다음과 같다.

  ❶ 사용자 로그인 후 세션을 설정하고 일정 시간이 지난 경우 다시 사용자 인증을 요구 할 때.

  ❷ 쇼핑몰에서 장바구니 기능을 구현할 때.

  ❸사용자의 페이지 이동 동선 등 웹 페이지 트래킹 분석 기능 등을 구현할 때.

05. session

- session 내장객체의 주요 메서드는 다음과 같다.

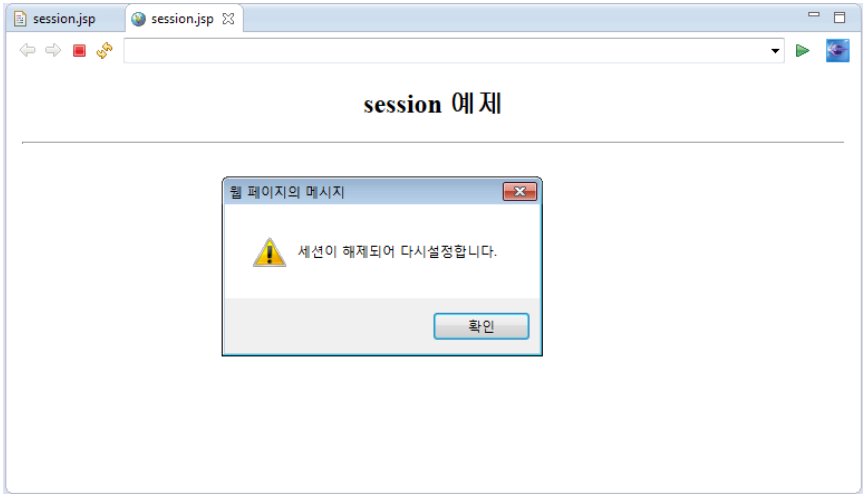
[표 6-5] session 내장객체 메서드

메서드	설명
getId()	각 접속에 대한 세션 고유의 ID를 문자열 형태로 반환한다.
getCreatingTime()	세션 생성 시간을 January 1, 1970 GMT.부터 long형 밀리세컨드 값으로 반환한다.
getLastAccessedTime()	현재 세션으로 마지막 작업한 시간을 long형 밀리세컨드 값으로 반환한다.
getMaxInactiveInterval()	세션의 유지 시간을 초로 반환한다. 이를 통해 세션의 유효 시간을 알 수 있다.
setMaxInactiveInterval(t)	세션의 유효 시간을 t에 설정된 초 값으로 설정한다.
invalidate()	현재 세션을 종료한다. 세션과 관련된 값들은 모두 지워진다.
getAttribute(attr)	문자열 attr로 설정된 세션 값을 java.lang.Object 형태로 반환한다.
setAttribute(name,attr)	문자열 name으로 java.lang.Object attr을 설정한다.

05. session

[실습] session 내장객체 활용(session.jsp)

- 교재 p.213 ~ 214 참고



[그림 6-7] 최초 세션 설정

[그림 6-8] 설정된 세션 확인

05. session

주요 소스 코드 분석(session.jsp)

- 세션 설정 여부 확인

```
11 if(session.isNew()) {
12      out.println("<script> alert('세션이 해제되어 다시 설정합니다.') </script>");
13      session.setAttribute("login", "홍길동");
14 }
```

  - 세션은 브라우저 실행 후 서버 접속 시 생성되어 일정시간 유지됨.
  - 여기서는 생성된 세션이 없는 경우 세션에 login 이라는 key 값으로 "홍길동" 을 저장

- 세션 설정값 가져오기

```
16 # <%= session.getAttribute("login") %>님 환영합니다.!!<BR>
```

  - 세션에 설정된 값은 getAttribute() 메서드를 이용해 key 값으로 가져올 수 있음.

- 세션 유지 시간
  - 기본 세션 유지 시간은 1,800초(약 30분) 이며 setMaxInactiveInterval() 을 이용해 변경할 수 있음.

```
18 2. 세션 유지시간 : <%= session.getMaxInactiveInterval() %>  <BR>
```

06. 그 밖의 내장객체

1. config

- 서블릿이 최초로 메모리에 적재될 때 컨테이너는 서블릿 초기화와 관련된 정보를 읽고 javax.servlet.ServletConfig 객체에 저장한다.
- config는 바로 ServletConfig 클래스에 대한 참조 변수로 web.xml 에 설정된 초기화 파라미터를 참조하기 위한 용도로 사용할 수 있다.

[표 6-6] config 주요 메서드

메서드	설명
getInitParameterNames()	초기 매개변수 값들의 설정 이름을 열거 객체로 반환한다.
getInitParameter(name)	문자열 name에 해당하는 초기화 매개변수 값을 반환한다.

06. 그 밖의 내장객체

2. application

- application은 웹 애플리케이션(컨텍스트) 전체를 관리하는 객체로 application 객체를 통해 각 서블릿이나 JSP에서 공유하려고 하는 각종 정보를 설정하고 참조할 수 있다.
- application은 javax.servlet.ServletContext 객체에 대한 참조 변수로써, config 객체를 통해 생성한다. ServletContext 객체는 컨테이너와 관련된 여러 정보를 제공하며, application 참조 변수를 통해서 서블릿이 실행되는 환경이나 서버 자원과 관련한 정보 를 얻거나 로그 파일을 기록하는 작업 등을 수행한다.
- application 내장객체는 일반적으로 톰캣의 시작과 종료 라이프사이클을 가진다.
- 유형별로 많은 메서드를 제공하므로 주로 관리 기능의 웹 애플리케이션 개발에 유용하다.

06. 그 밖의 내장객체

[표 6-7] 개발자를 위한 서버 정보

메서드	설명
getServerInfo()	JSP/서블릿 컨테이너의 이름과 버전을 반환한다.
getMajorVersion()	컨테이너가 지원하는 서블릿 API의 주 버전 정보를 반환한다.
getMinorVersion()	컨테이너가 지원하는 서블릿 API의 하위 버전 정보를 반환한다.

[표 6-8] 서버 자원 정보

메서드	설명
getMimeType(filename)	문자열 filename에 지정된 파일에 대한 MIME Type을 반환한다.
getResource(path)	문자열 path에 지정된 자원을 URL 객체로 반환한다.

메서드	설명
getResourceAsStream(path)	문자열 path에 지정된 자원을 InputStream 객체로 반환한다.
getRealPath(path)	문자열 path에 지정된 자원을 파일 시스템의 실제 경로로 반환한다.
getContext(path)	문자열 path에 지정된 자원의 컨텍스트 정보를 반환한다.
getRequestDispatcher(path)	문자열 path에 지정된 자원을 위한 request dispatcher를 생성한다.

06. 그 밖의 내장객체

[표 6-9] 로그 관련 정보

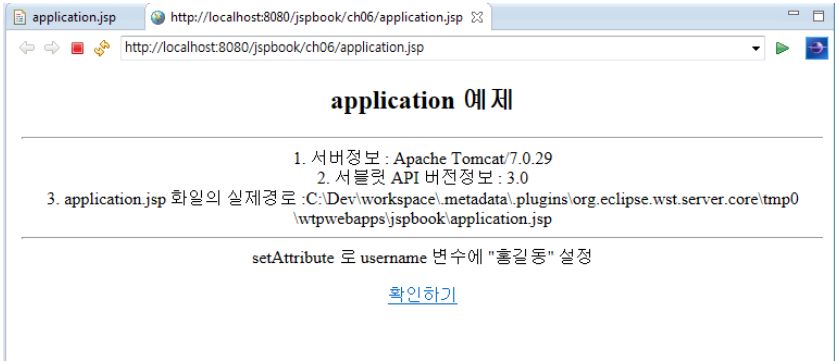
메서드	설명
log(message)	문자열 message의 내용을 로그 파일에 기록한다. 로그 파일의 위치는 컨테이너에 따라 다르다.
log(message,exception)	예외 상황에 대한 정보를 포함하여 로그 파일에 기록한다.

[표 6-10] 속성 관련 정보

메서드	설명
getAttribute(String name)	문자열 name에 해당하는 속성 값이 있다면 Object 형태로 가져온다. 따라서 반환 값에 대한 적절한 형변환이 필요하다.
getAttributeNames()	현재 application 객체에 저장된 속성들의 이름을 열거 형태로 가져온다.
setAttribute(String name, Object value)	문자열 name 이름으로 Object형 데이터를 저장한다. Object형이므로 자바 클래스 형태로도 저장할 수 있다.
removeAttribute(String name)	문자열 name에 해당하는 속성을 삭제한다.

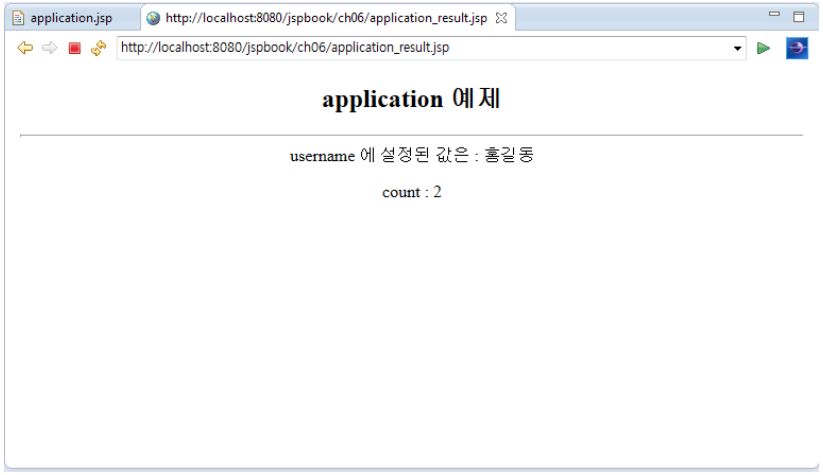
06. 그 밖의 내장객체

[실습] application 내장객체의 활용(application.jsp) - 교재 p.218 ~ 219 참고

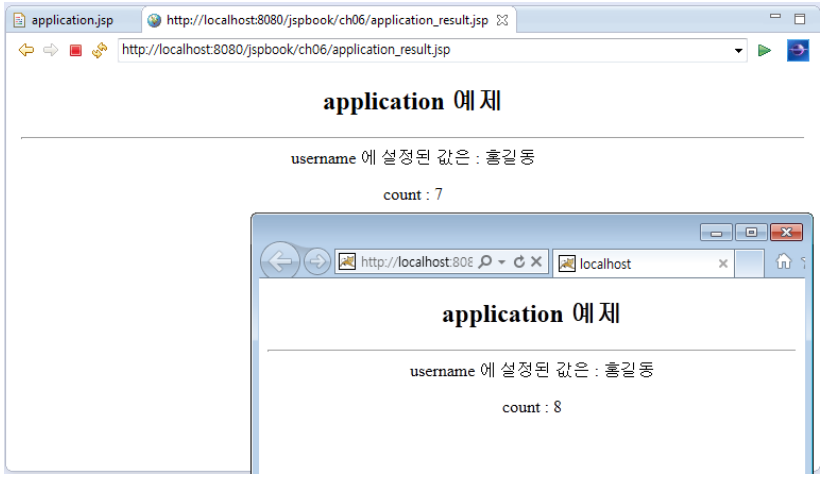


[그림 6-9] application.jsp 실행 화면

[실습] application 내장객체의 활용 결과(application_result.jsp) - 교재 p.220 참고



[그림 6-10] application_result.jsp 실행 화면



[그림 6-11] 카운트 확인

06. 그 밖의 내장객체

3. page

- page는 JSP 컨테이너에서 생성된 서블릿 인스턴스 객체를 참조하는 참조 변수며, JSP에서 자기 자신을 참조할 때 사용된다.
- JSP 스크립트 언어가 자바가 아니라면 유용하게 사용할 수 있지만, 자바인 경우 page 참조 변수를 통하지 않고도 생성된 서블릿 클래스의 멤버변수나 메서드에 직접 접근할 수 있다.
- 따라서 page 참조 변수는 거의 사용하지 않는다.

06. 그 밖의 내장객체

4. pageContext

- pageContext는 javax.servlet.jsp.PageContext 인스턴스에 대한 참조 변수로, 다른 모든 내장객체에 대한 프로그램적인 접근 방법을 제공한다.
- 많이 사용하는 형태는 HTTP 요청을 처리하는 제어권을 다른 페이지로 넘길 때 사용하는 것으로 forward 액션과 동일한 기능을 제공한다.(forward 액션의 내부 구현 코드로 이해할 수 있다.)

[표 6-11] 내장객체 참조 관련 메서드

메서드	설명
getPage()	현재 페이지에서 생성된 서블릿 인스턴스인 page 내장객체를 반환한다.
getRequest()	현재 페이지의 클라이언트 요청 정보가 있는 request 내장객체를 반환한다.
getResponse()	현재 페이지의 클라이언트 응답 정보가 있는 response 내장객체를 반환한다.
getOut()	현재 페이지의 output stream인 out 내장객체를 반환한다.
getSession()	현재 페이지의 session 내장객체를 반환한다.
getServletConfig()	현재 페이지의 config 내장객체를 반환한다.
getServletContext()	현재 페이지의 서블릿 컨텍스트(application 내장객체)를 반환한다.
getException()	오류 페이지, 즉 page 지시어에서 errorPage 속성을 지정한 페이지에서 오류가 발생할 때, 발생한 예외 정보가 있는 exception 내장객체를 반환한다.

06. 그 밖의 내장객체

[표 6-12] 페이지 전달 관련 메서드

메서드	설명
forward(path)	문자열 path에 지정된 페이지로 전달한다.
include(path)	문자열 path에 지정된 페이지를 포함시킨다.

- forward() 메서드는 forward 액션과 동일한 기능을 한다.
  - ❶ forward() 메서드 사용 : pageContext.forward("HelloWorld.jsp")
  - ❷ forward 액션 사용 : <jsp:forward page="HelloWorld.jsp" />

- include() 메서드는 include 액션과 동일한 기능을 한다.
  - ❶ include() 메서드 사용 : <%
    out.flush();
    pageContext.include("HelloWorld.jsp");
    %>
  - ❷ include 액션 사용 : <jsp:include page="HelloWorld.jsp" flush=true />

06. 그 밖의 내장객체

5. exception

- exception은 page 지시어에서 오류 페이지로 지정된 JSP 페이지에서 예외가 발생할 때 전달되는 java.lang.Throwable의 인스턴스에 대한 참조 변수다.
- 이를 통해 현재 페이지를 처리하다 발생하는 예외상황에 대한 정보를 가져올 수 있다.
- 일반적으로 오류 페이지를 별도로 구성하거나 문제 발생할 경우, 로깅을 위한 추가적인 정보를 획득하기 위해 사용한다.

[표 6-13] 예외 관련 메서드

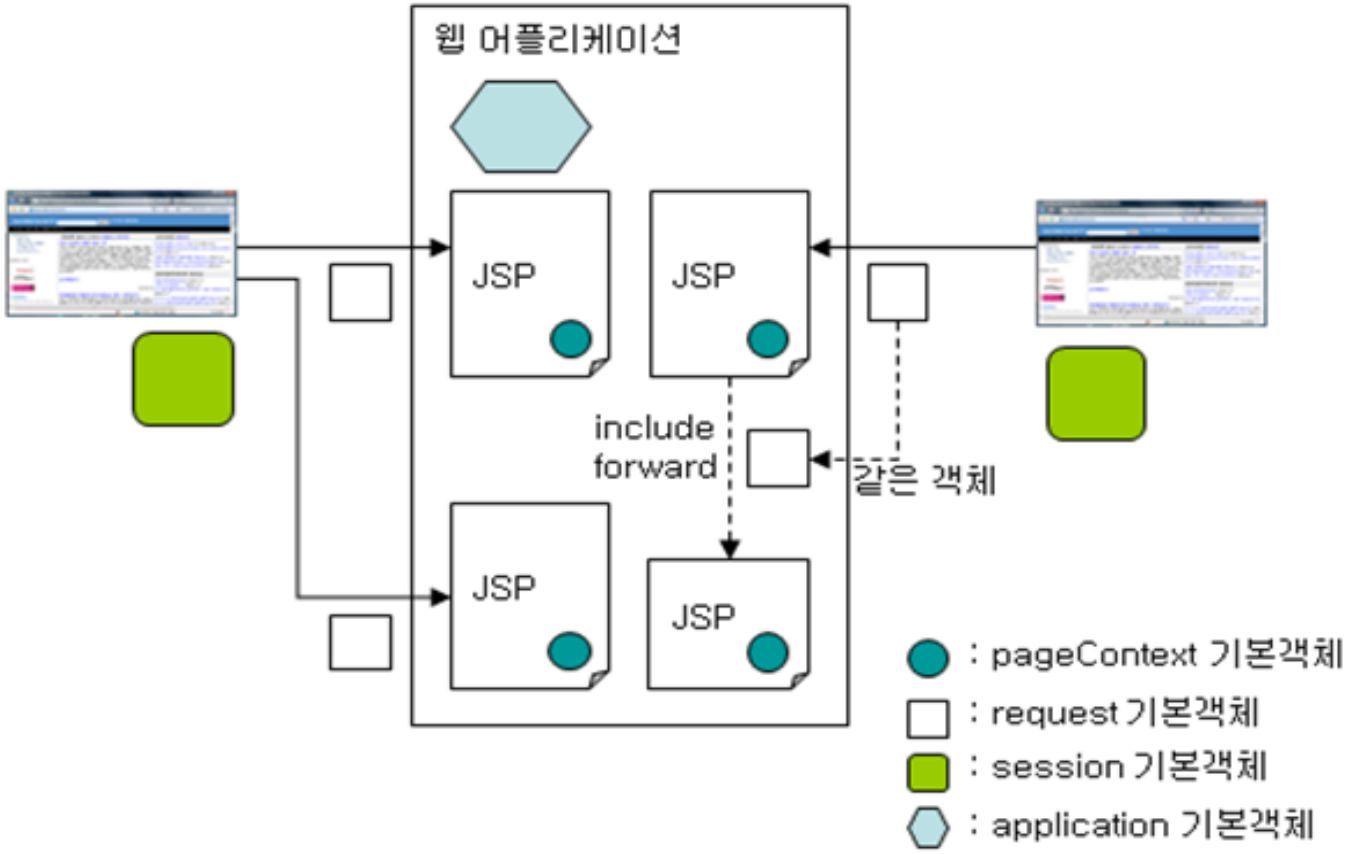
메서드	설명
getMessage()	문자열로 된 오류 메시지를 반환한다.
printStackTrace()	표준 출력 스트림으로써, 스택 추적 정보를 출력한다.
toString()	예외 클래스 이름과 함께 오류 메시지를 반환한다.

JSP의 4개 영역 (scope)

4가지 영역

- **Page 영역:** 하나의 JSP 페이지를 처리할 때 사용 되는 영역
- **Request 영역:** 하나의 HTTP 요청을 처리할 때 사용 되는 영역
- **Session 영역:** 하나의 웹브라우저와 관련된 영역
- **Application 영역:** 하나의 웹어플리케이션과 관련된 영역

그림)



JSP 기본 객체의 속성

- 기본 객체 pageContext, request, session, application은 속성을 갖는다.
- 속성은 각각의 기본 객체가 존재하는 동안에 사용 가능
- JSP 페이지 사이에서 정보를 주고 받거나 공유하기 위한 목적으로 사용
- 속성은 <속성이름, 값>의 형태를 가짐
- 서로 다른 이름을 가진 속성을 여러 개 포함할 수 있다.

[네 기본 객체의 속성 관련 메소드]

메소드	리턴타입	설명
setAttribute(String name, Object value)	void	이름이 name인 속성의 값을 value로 지정한다.
getAttribute(String name)	Object	이름이 name인 속성의 값을 구한다. 지정한 이름의 속성이 존재하지 않을 경우 null을 리턴한다.
removeAttribute(String name)	void	이름이 name인 속성을 삭제한다.
getAttributeNames()	java.util.Enumeration	속성의 이름 목록을 구한다. (pageContext 기본 객체는 이 메소드를 제공하지 않는다.)

☞ 속성의 값 타입은 Object이므로 값을 사용할 때에는 알맞은 형변환이 필요

속성의 활용

기본 객체	영역	쓰임새
pageContext	PAGE	(한번의 요청을 처리하는) 하나의 JSP 페이지 내에서 공유될 값을 저장한다. 주로 커스텀 태그에서 새로운 변수를 추가할 때 사용된다.
request	REQUEST	한번의 요청을 처리하는 데 사용되는 모든 JSP 페이지에서 공유될 값을 저장한다. 주로 하나의 요청을 처리하는 데 사용되는 JSP 페이지 사이에서 정보를 전달하기 위해 사용된다.
session	SESSION	한 사용자와 관련된 정보를 JSP 들이 공유하기 위해서 사용된다. 사용자의 로그인 정보와 같은 것들을 저장한다.
application	APPLICATION	모든 사용자와 관련해서 공유할 정보를 저장한다. 임시 디렉토리 경로와 같은 웹 어플리케이션의 설정 정보를 주로 저장한다.

모듈화 Module

1. 교재 5장

   1) 액션태그 include : <jsp:include page="포함할 파일">

   2) 지시어 include : <%@ include file="포함할 파일" %>

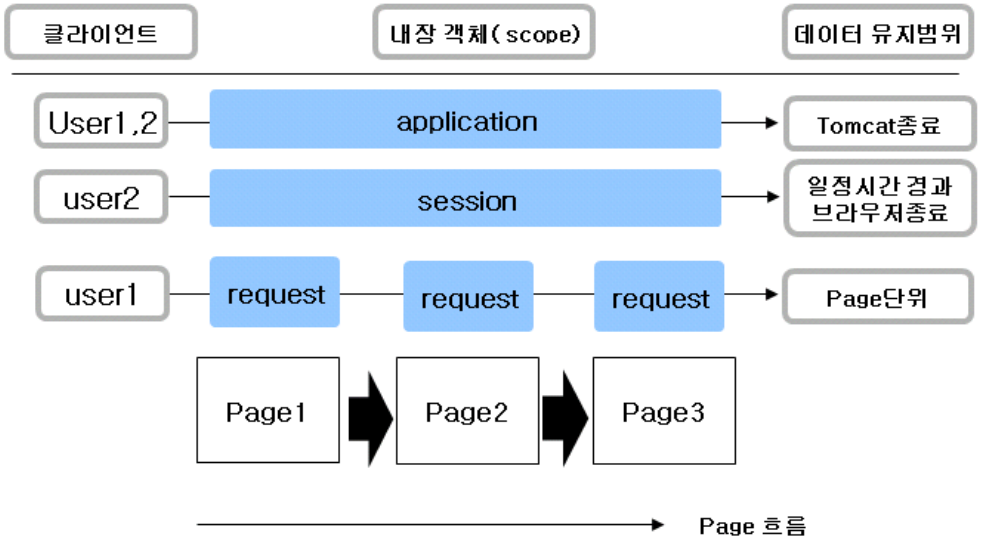
2. 6장_보조자료_모듈화_include.pdf 파일 참조

07. JSP 내장객체와 속성 관리

1. HTTP 프로토콜 특징과 내장객체 속성 관리

- JSP는 HTTP 프로토콜의 사용하는 웹 환경에서 구동되는 프로그램 이다.
- HTTP는 비연결형으로 사용자가 서버에 특정 페이지를 요청하고 요청결과를 응답받으면 서버와의 연결이 끊기는 형태임.
- 예를 들어 게시판에 글을 작성하는 페이지에서 작성한 내용은 다른 jsp에서 처리해야 하고 서버는 방금 글을 작성한 사람이 누구인지 모를 수 있음.
- 또 다른 예로 쇼핑몰에서 여러 상품 페이지를 이동하면서 장바구니에 물건을 담아 두고 한꺼번에 구매하고자 할 때 접속된 사용자별로 선택된 상품을 처리하는 경우 지금까지 배운 JSP 문법만 가지고는 이를 처리하기 어려움.
- **JSP 에서는 page, request, session, application 내장객체를 통해 서로 다른 페이지에서 처리된 값을 저장하고 공유하기 위한 방법을 제공함**.
- 이는 컨테이너 기반 프로그램의 특징 중 하나로 실제 프로그램 구현 시 매우 중요한 기법임.

07. JSP 내장객체와 속성 관리



[그림 6-12] JSP 내장객체 속성 관리

❶ application은 모든 사용자가 공유하는 데이터를 저장할 수 있으며 톰캣이 종료될 때 까지 데이터를 유지할 수 있다(맨 위의 user1, user2 해당).

❷ session의 경우 사용자마다 분리된 저장 영역이 있으며 Page1, Page2, Page3 모두에서 공유되는 정보를 관리할 수 있다. 물론 이 데이터는 각자 공유 영역에서 관리되며 사용자 간에는 공유되지 않는다.

❸ 페이지 흐름이 Page1, Page2, Page3순으로 진행된다고 할 때, 한 페이지에서 다른 페이지로 데이터를 전달하려면 request 내장객체를 이용해야 한다(맨 아래의 user1에 해당한다). page 마다 생성됨.

07. JSP 내장객체와 속성 관리

- request, session, application 은 각각 생성 시점과 소멸시점이 다르며 이를 잘 이해하고 적절한 내장객체를 이용해야 한다.
- 각각의 내장객체는 모두 getAttribute(), setAttribute() 메서드를 통해 속성을 저장하거나 가져올 수 있다.

[표 6-14] **주요 내장객체의 생성 시점과 소멸 시점**

내장객체	생성 시점	소멸 시점
request	해당 페이지 요청 시점	해당 페이지 로딩 완료 시점
session	해당 컨텍스트 내 특정 파일 요청 시점 (사용자 최초 접속 시점)	• 웹 브라우저 종료 시점 • 일정 시간 경과 시점
application	웹 애플리케이션 시작 시점	웹 애플리케이션 종료 시점

07. JSP 내장객체와 속성 관리

2. request, session, application을 이용한 속성 관리

- request, session, application은 맵 형태의 속성 관리 기능을 제공 한다.
- 속성을 저장하기 위해서는 setAttribute(String name, Object value) 형태를 취한다.
- 반대로 속성에 저장된 값을 가져오는 getAttribute(String name) 메서드는 name에 해당하는 Object 를 리턴 한다.
- 리턴되는 타입이 Object 이므로 속성을 가지고 올 때에는 적절한 형 변환이 필요하다.
- 예를 들어 page1에서 session.setAttribute("name,"홍길동")으로 문자열 객체를 저장한다면 page3에서는 session.getAttribute("name")으로 저장된 값을 참조할 수 있다.

07. JSP 내장객체와 속성 관리

3. 컨테이너 기반 프로그램의 특징과 JSP 내장객체

- 최신의 프로그램 아키텍처의 특징 중 하나는 컨테이너를 기반으로 한 구조임.
- 프로그램 관점에서 컨테이너란 프로그램 실행에 관여하면서 모듈화된 프로그램을 실행할 수 있게 하고 프로그램 간의 원활한 데이터(객체) 교환을 지원하는 소프트웨어를 말한다.
- JSP에서 내장객체를 이용한 속성 관리가 가능한 것은 JSP와 빈즈 객체들이 톰캣이라고 하는 컨테이너에 의해 관리되고 실행되기 때문이다.
- 웹 프로그램도 컨테이너 기반이며 대표적인 프레임워크인 스프링 역시 컨테이너 기반의 프로그램 모델이 된다. 이러한 컨테이너 기반 프로그램의 장점은 다음과 같다.

❶ 프로그램의 모듈화가 용이하다.
❷ 독립적으로 실행되는 모듈 간의 데이터 교환이 용이하다.
❸ 개별 프로그램에서 화면/상태 전환 시 데이터를 유지/관리하기 용이하다.
❹ JSP에서는 컨테이너에 의해 관리되는 내장객체를 통해 임의의 객체를 각각의 생명주기 시점에 따라 공유할 수 있다.

4. MVC 패턴과 JSP 내장객체

- MVC 패턴은 프로그램을 Model, View, Controller 세가지 역할로 구분해 구현하는 소프트웨어 디자인 패턴을 말함.
- MVC 패턴에 따르면 JSP는 뷰의 역할만 수행해야 함. 즉 화면에 데이터를 출력하는 기능만 제공해야 한다는 것인데, 문제는 컨트롤러에서 처리한 데이터(예를 들면 데이터베이스로부터 가져온)를 어떻게 JSP로 전달해야 하는지에 대한 것이다.
- 이처럼 MVC 패턴을 사용하게 되면 JSP는 별도로 데이터를 가지고 오는 로직 없이 데이터를 출력할 수 있어야 하는데 이때 JSP 내장객체를 이용한 속성 관리가 사용된다.
- 예를 들어, 컨트롤러에서 처리한 데이터는 request.setAttribute( )를 이용해서 저장하고 화면에 보여질 JSP로 포워딩 하면 해당 JSP에서는 request 내장객체를 통해 데이터를 참조할 수 있으므로 완전한 MVC 패턴의 구현을 구현할 수 있다.
- 뷰를 효과적으로 구성하는 방법은 <jsp:useBean>, <jsp:getProperty>, 표현식을 사용하는 것이나 표현언어(Expression Language)와 JSTL을 이용할 경우 더욱 편리하게 뷰를 구현할 수 있다.

세션을 이용한 로그인 실습

1. 세션을 이용한 로그인 실습

--→ <과제> 위의 로그인 실습예제에 “아이디/비번 저장하기“ 기능 추가

2. 모듈화되어 있는 페이지에서의 로그인

08. [기본실습]JSP 내장객체: 세션을 이용한 장바구니 구현

실습 개요

- 쇼핑몰 사이트에서 많이 활용되는 장바구니 기능의 구현을 통해 JSP 내장객체를 이용한 속성관리에 대한 이해를 높이고 특히 세션에 대한 실제 활용 사례를 익힌다.
- 쇼핑몰의 기본 흐름은 다음과 같다.

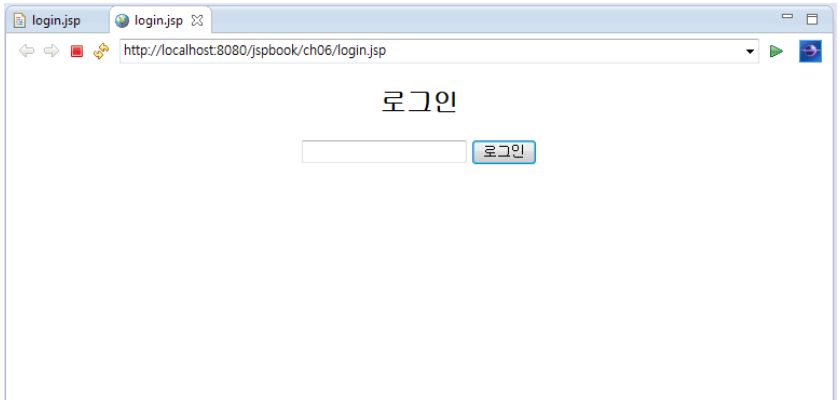
❶ 사용자가 로그인한다. → ❷ 원하는 만큼 상품을 선택한다. →
❸ <주문> 버튼을 클릭하면 지금까지 선택했던 상품이 모두 나타난다.

[표 6-15] 프로그램 소스 목록

파일 이름	역할
login.jsp	로그인하는 화면으로, 비밀번호 입력은 없으며 사용자 이름을 입력하는 양식만 제공한다.
selProduct.jsp	상품을 선택하는 화면으로, 리스트에서 원하는 상품을 선택하고 〈추가〉 버튼을 눌러 상품을 추가한다.
add.jsp	selProduct.jsp에서 선택한 상품을 세션에 넣는다. 선택된 데이터를 모두 저장해야 하므로 ArrayList를 이용한다. 상품이 추가되었다는 메시지를 보여주고 다시 selProduct.jsp로 되돌아간다.
checkOut.jsp	세션이 살아 있고 하나 이상의 상품을 선택한 상태라면 선택한 상품의 목록을 보여준다.

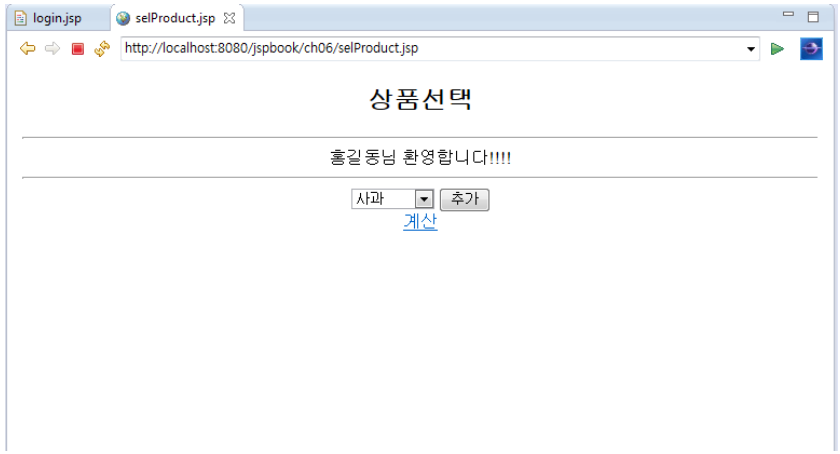
08. [기본실습] JSP 내장객체: 세션을 이용한 장바구니 구현

[실습] 로그인 화면(login.jsp) – 교재 p.230 참고



[그림 6-13] login.jsp 실행 결과

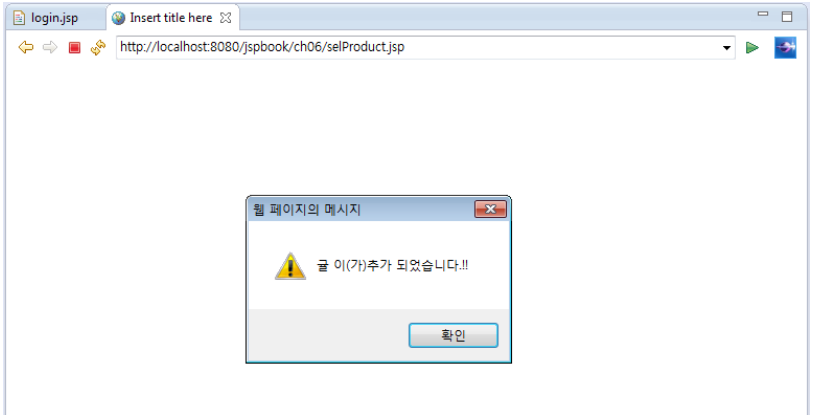
[실습] 상품 선택 화면(selProduct.jsp) – 교재 p.231 ~ 232 참고



[그림 6-14] selProduct.jsp 실행 화면

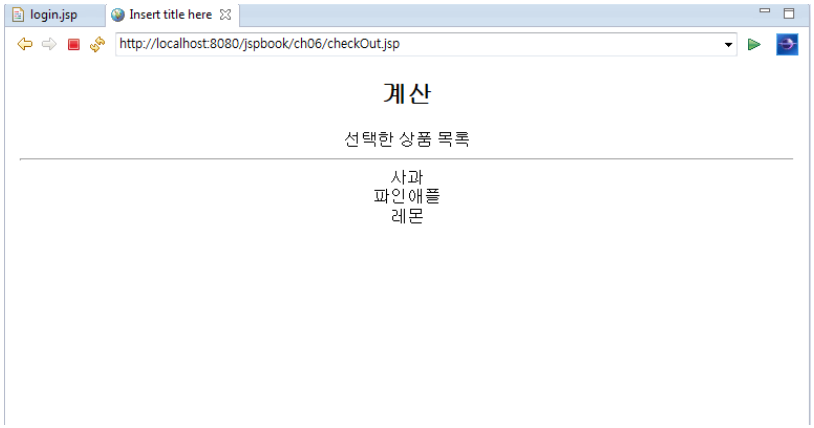
08. [기본실습] JSP 내장객체: 세션을 이용한 장바구니 구현

[실습] 상품 추가 화면(add.jsp) – 교재 p.232 ~ 233 참고



[그림 6-15] add.jsp로 상품 이름 전달

[실습] 선택 상품 목록 화면(checkOut.jsp) – 교재 p.234 참고



[그림 6-16] checkOut.jsp 실행 화면

08. [기본실습] JSP 내장객체: 세션을 이용한 장바구니 구현

주요 소스코드 분석 (add.jsp)

- 세션에 상품 정보 추가

```
11    String productname = request.getparameter("product");
12    ArrayList list = (ArrayList)session.getAttribute("productlist");
13    if(list == null) {
14        list = new ArrayList();
15    }
16 list.add(productname);
```

  - request.getParameter를 통해 선택된 상품 이름을 가져옴.
  - 세션으로 부터 "productlist" 라는 이름의 ArrayList 객체를 가져오고 만일 null이라면 새로운 객체를 생성.
  - list.add() 메서드를 통해 선택된 상품을 ArrayList 에 저장함.

- 상품 추가 후 자바스크립트 메시지 발생 및 이전 페이지로 돌아가기

```
19 <script>
20 alert("<%=productname %>이(가) 추가되었습니다!!");
21 history.go(-1);
```

08. [기본실습] JSP 내장객체: 세션을 이용한 장바구니 구현

주요 소스코드 분석 (checkOut.jsp)

- 세션에 저장된 상품 정보 출력

```
14 ArrayList list = (ArrayList)session.getAttribute("productlist");
15 if(list == null) {
16     out.println("선택한 상품이 없습니다.!!!");
17 }
18 else {
19     for(Object productname:list) {
20         out.println(productname+"<BR>");
21     }
22 }
```

- 세션으로부터 "productlist"라는 이름의 ArrayList 객체를 가져오고 만일 null 이라면 선택한 상품이 없다고 출력함.
- 데이터가 있는 경우 for 루프를 돌며 저장된 상품 이름을 모두 출력.

08. [응용실습] JSP 내장객체: 트위터 구현

실습 개요

- 단순한 문법 예제가 아닌 실제 사용할 수 있는 프로그램의 형태를 가진 웹 애플리케이션 개발을 통해 지금까지 배운 내용을 종합하고 내장객체에 대한 다양한 활용을 익힘.
- 예제의 주요 특징은 다음과 같다.

❶ 데이터베이스를 사용하지 않고 application 내장객체를 이용해 공용 저장소로 활용 한다.
❷ 톰캣이 종료되면 저장된 데이터도 초기화된다.
❸ 다중 사용자 접속을 지원하며 개별 사용자 id를 유지한다.

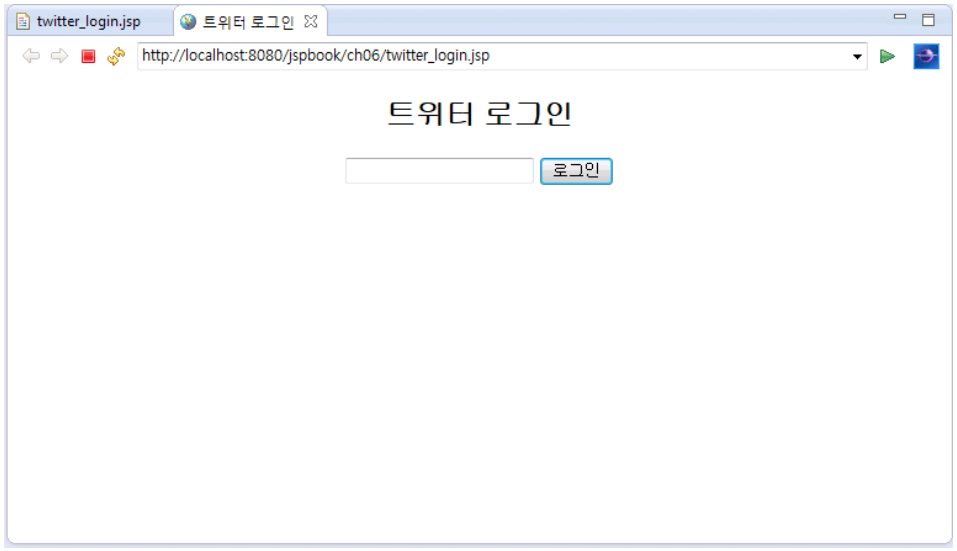
[표 6-16] **프로그램 소스 목록**

파일 이름	역할
twitter_login.jsp	로그인하는 화면으로, 비밀번호 입력은 없으며 사용자 이름을 입력하는 양식만 제공한다.
twitter_list.jsp	트위터 메인 화면으로 등록된 글의 목록이 나타난다. 작성자 아이디와 내용 시간이 출력된다.
tweet.jsp	현재 로그인한 사용자 아이디와 작성된 메시지를 저장한다. 여기서는 application 내장객체를 이용해 모든 사용자가 공유할 수 있는 임시 저장소로 활용한다.

08. [기본실습] JSP 내장객체: 세션을 이용한 장바구니 구현

1. 로그인 화면 구현

- [실습] 로그인 화면(twitter_login.jsp) – 교재 p.237 참고



[그림 6-17] 로그인 화면

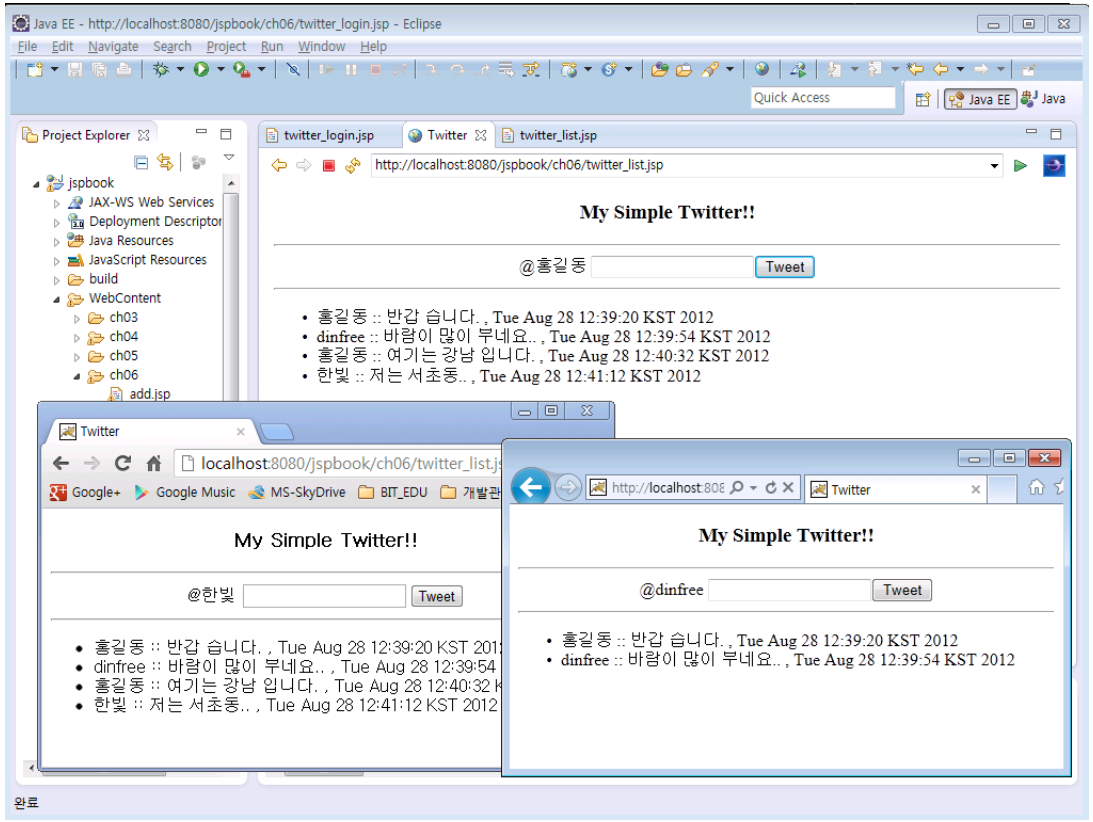
2. 트위터 메인 화면 구현

- [실습] 트위터 메인화면(twitter_list.jsp) – 교재 p.238 ~ 239 참고

08. [기본실습] JSP 내장객체: 세션을 이용한 장바구니 구현

3. 게시글 등록 구현

- **[실습] 게시글 등록 화면(tweet.jsp)** – 교재 p.240 ~ 241 참고



[그림 6-18] 최종 실행 화면

08. [기본실습] JSP 내장객체: 세션을 이용한 장바구니 구현

주요 소스코드 분석 (twitter_list.jsp)

- 저장된 게시글 출력

```
31 ArrayList<String>msgs=(ArrayList<String>)application.getAttribute("msgs")
32
33 // msgs가 null이 아닌 경우에만 목록 출력
34 if(msgs != null) {
35     for(String msg : msgs) {
36         out.println("<LI>"+msg+"</LI>");
37     }
38 }
```

- application 내장객체를 통해 ArrayList에 저장된 게시글을 가져옴.
- ArrayList가 null이 아닌 경우 즉, 저장된 데이터가 있는 경우에만 화면 출력.
- 데이터가 있을 경우 for 문을 이용해 데이터 출력, 각각의 데이터는 <UL><LI></LI></UL> 로 목록 형태로 구성함.

08. [기본실습] JSP 내장객체: 세션을 이용한 장바구니 구현

주요 소스코드 분석 (tweet.jsp)

- 게시글 등록

```
13 // 메시지 저장을 위해 application에서 msgs로 저장된 ArrayList 가지고 옴
14 ArrayList<String> msgs=(ArrayList<String>)application.getAttribute("msgs")
15
16 // null인 경우 새로운 ArrayList 객체를 생성
17 if(msgs == null) {
18     msgs = new ArrayList<String();
       // application에 ArrayList 저장
       application.setAttribute("msgs", msgs);
19 }
20
21 // 사용자 이름, 메시지, 날짜 정보를 포함하여 ArrayList에 추가
22 msgs.add(username+" :: "+msg+" , "+new java.util.Date());
```

- application 내장객체를 통해 ArrayList에 저장된 게시글을 가져옴.
- ArrayList가 null인 경우 새로운 ArrayList 객체를 생성하고 application에 저장.
- 이름, 메시지, 날짜 정보를 포함하여 ArrayList에 추가
- 추가된 데이터는 application에 저장되어 톰캣 종료 전까지 모든 사용자에게 공유됨.

08. [기본실습]JSP 내장객체: 세션을 이용한 장바구니 구현

주요 소스코드 분석 (tweet.jsp)

- 게시글 등록 후 메인 화면으로 이동

```
30 // 목록 화면으로 리다이렉팅
31 response.sendRedirect("twitter_list.jsp");
```

  - 게시글 등록 후 목록 화면으로 되돌아 가야 하므로 response 내장객체의 sendRedirect() 메서드를 이용해 twitter_list.jsp로 이동함.

실행 및 테스트

- 가급적 여러 컴퓨터에서 실행 컴퓨터의 ip 주소를 이용해 접속한 다음 다른 사용자로 로그인해 게시글을 남겨 확인 함.
- 접속 URL은 http://localhost:8080/jspbook/ch06/twitter_login.jsp 이고 localhost를 ip로 바꾸면 다른 컴퓨터에서도 접속이 가능함.
- 저장된 데이터는 톰캣을 종료하기 전까지는 유지되고 톰캣을 다시 시작하면 초기화됨.