프로젝트로 배우는

자바 웹 프로그래밍

Servlet, JSP, JDBC

황희정 지음

Chapter 06. JSP 내장객체

한빛아카데미 Hanbit Academy, Inc.

목차

04. out

out 내장객체

- out은 출력 스트림으로 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 메소드와 출력 관련 메소드로 구성되며 out을 이용해서 출력한 내용은 서버의 콘솔이 아닌 사용자에 전달된다.

[표 6-4] out 주요 메소드

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.

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

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 유사한 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.

calc4_jsp 서블릿 클래스 (revisted)

```
package org.apache.jsp.ch05.declaration;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;

public final class calc4_jsp extends
org.apache.jasper.runtime.HttpJspBase
    implements org.apache.jasper.runtime.JspSourceDependent
{

// 소속 변수 선언
 int num1, num2 = 0;
 String op = " ";

 // 연산자별 처리를 위한 소속 메소드 선언
public int calculator()  {
  int result = 0;

 if (op.equals("+"))  {
  result = num1 + num2;
 } else if (op.equals("-"))  {
  result = num1 - num2;
 } else if (op.equals("*"))  {
  result = num1 * num2;
 } else if (op.equals("/"))  {
  result = num1 / num2;
 }
 return result;
}
```

```
 public void _jspInit() {
 }

 public void _jspDestroy() {
 }

public void _jspService(final HttpServletRequest request, final
HttpServletResponse response) throws java.io.IOException,
javax.servlet.ServletException {

   final PageContext pageContext;
   HttpSession session = null;
   final ServletContext application;
   final ServletConfig config;
   JspWriter out = null;
   final Object page = this;
   JspWriter _jspx_out = null;
   PageContext _jspx_page_context = null;

   try {
      response.setContentType("text/html; charset=UTF-8");
      pageContext = _jspxFactory.getPageContext(this, request, response,
                                                null, true, 8192, true);
      _jspx_page_context = pageContext;
      application = pageContext.getServletContext();
      config = pageContext.getServletConfig();
      session = pageContext.getSession();
      out = pageContext.getOut();
      _jspx_out = out;
```

JSP 내장객체 사용시 유의할 점

- 스크립트릿 내부에서는 별도의 정의없이 자유롭게 사용 가능
- 선언부에서는 정의하지 않고 사용할 수 없음

- 선언부에서 사용하고 싶은 경우에는 메소드의 매개변수로 전달

- <%!
- public int calculator(HttpServletRequest request) { … }

- public int calculator(PageContext pageContext) {
- HttpServletRequest request = pageContext.getRequest();
- HttpSession session = pageContext.getSession();
- …
- }
- %>

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 유사한 기능을 한다.
  - ❶ forward() 메서드 사용 : <% pageContext.forward("HelloWorld.jsp"); %>
  - ❷ forward 액션 사용 : <jsp:forward page="HelloWorld.jsp" />

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

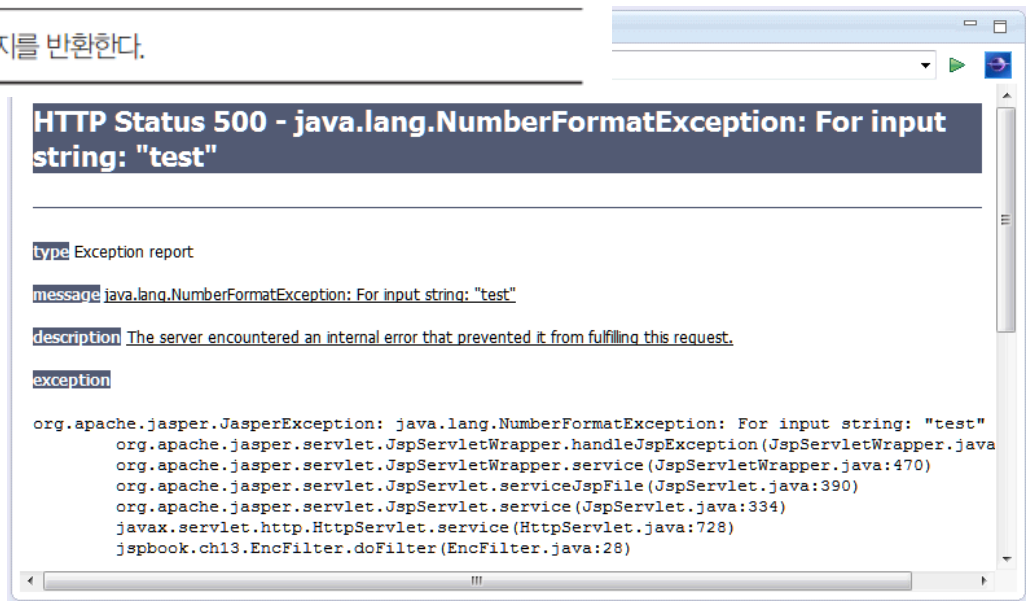
06. 그 밖의 내장객체

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



Tomcat 오류 출력화면

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

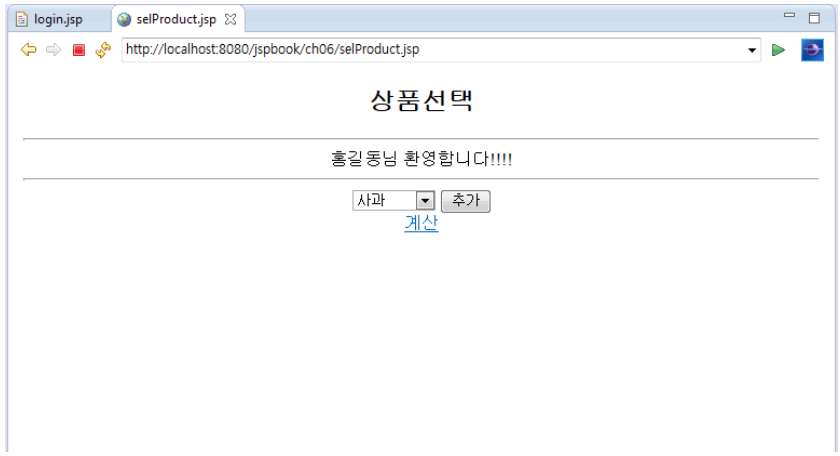
파일 이름	역할
login.jsp	로그인하는 화면으로, 비밀번호 입력은 없으며 사용자 이름을 입력하는 양식만 제공한다.
selProduct.jsp	상품을 선택하는 화면으로, 리스트에서 원하는 상품을 선택하고 〈추가〉 버튼을 눌러 상품을 추가한다.
add.jsp	selProduct.jsp에서 선택한 상품을 세션에 넣는다. 선택된 데이터를 모두 저장해야 하므로 ArrayList를 이용한다. 상품이 추가되었다는 메시지를 보여주고 다시 selProduct.jsp로 되돌아간다.
checkOut.jsp	세션이 살아 있고 하나 이상의 상품을 선택한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

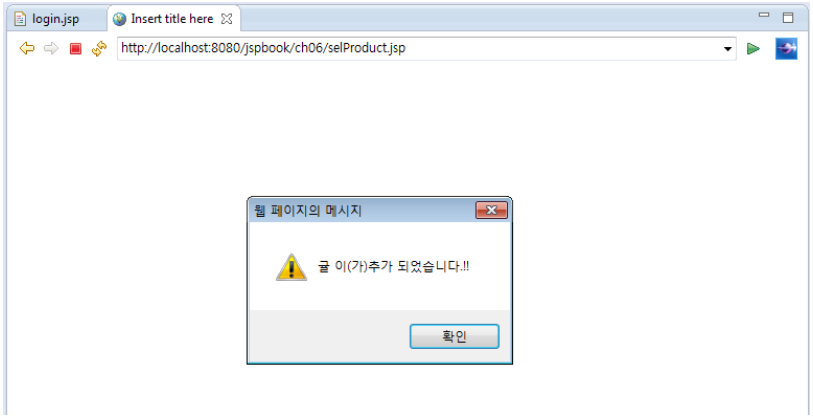
[실습] 상품 선택 화면(selProduct.jsp) – 교재 p.231 ~ 232 참고



[그림 6-14] selProduct.jsp 실행 화면

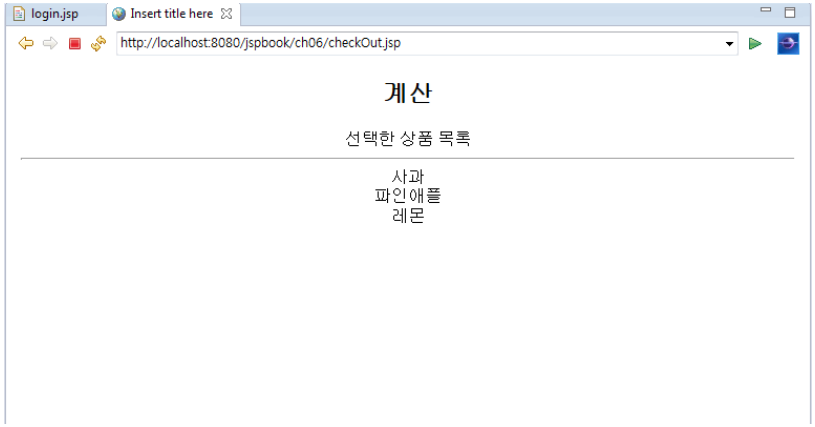
08. [기본실습] JSP 내장객체: 세션을 이용한 장바구니 구현

[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

상품선택 화면 (selProduct.jsp)

selProduct.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"
         import="java.util.ArrayList" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<HTML>
<HEAD>
<title>ch06 : selProduct.jsp</title>
</head>
<%
    request.setCharacterEncoding("UTF-8");

    String user = request.getParameter("username");
    if (user != null)  {
        session.setAttribute("username", user);
    }
%>
<body>
<div align="center">
    <H2>상품선택</H2>
    <HR>
    <%= session.getAttribute("username") %>님 환영합니다!!!!
    <HR>
    <form name="form1" method="POST" action="add.jsp">
        <SELECT name="product">
            <option>사과</option>
            <option>귤</option>
            <option>파인애플</option>
            <option>자몽</option>
            <option>레몬</option>
        </SELECT>
        <input type="submit" value="추가"/>
    </form>
    <a href="checkOut.jsp">계산</a>
</div>
</body>
</html>
```

상품추가 화면 (add.jsp)

add.jsp checkOut.jsp selProduct.jsp login.jsp ch06 : checkOut.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" import="java.util.*"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>ch06 : add.jsp</title>
</HEAD>
<body>
<%
    request.setCharacterEncoding("UTF-8");
    String productname = request.getParameter("product");

    ArrayList<String> list = (ArrayList<String>)session.getAttribute("productlist");

    if(list == null) {
        list = new ArrayList<String>();
    }

    list.add(productname);
    session.setAttribute("productlist",list);

%>
    <script>
        alert("<%=productname %> 이(가)추가 되었습니다.!!");
        history.go(-1);
    </script>
</body>
</html>
```

선택상품 목록 화면 (checkOut.jsp)

add.jsp | checkOut.jsp | selProduct.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" import="java.util.*"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<HTML>
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>ch06 : checkOut.jsp</title>
</HEAD>
<body>
<div align="center">
<H2>계산</H2>
선택한 상품 목록
<HR>
<%
    ArrayList<String> list = (ArrayList<String>)session.getAttribute("productlist");

    if(list == null) {
        out.println("선택한 상품이 없습니다.!!!");
    }
    else {

        for(String productname:list) {

            out.println(productname+"<BR>");
        }
    }
%>
</div>
</body>
</html>

```

09. [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.

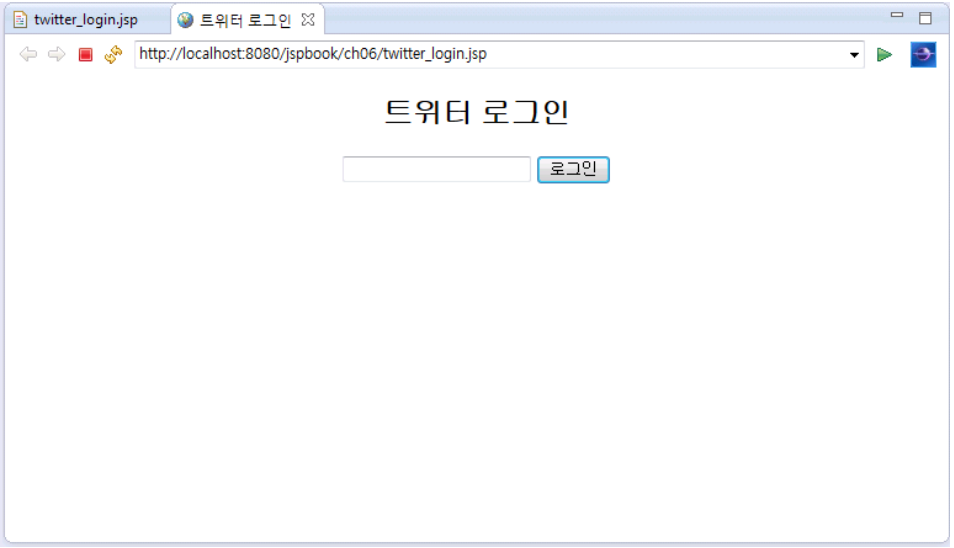
[표 6-16] 프로그램 소스 목록

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 applicati 든 사용자가 공유할 수 있는 임시 저장소로 활용한다.

09. [응용실습] JSP 내장객체: 트위터 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

09. [응용실습] JSP 내장객체: 트위터 구현

- 주요 소스코드 분석 (twitter_login.jsp)

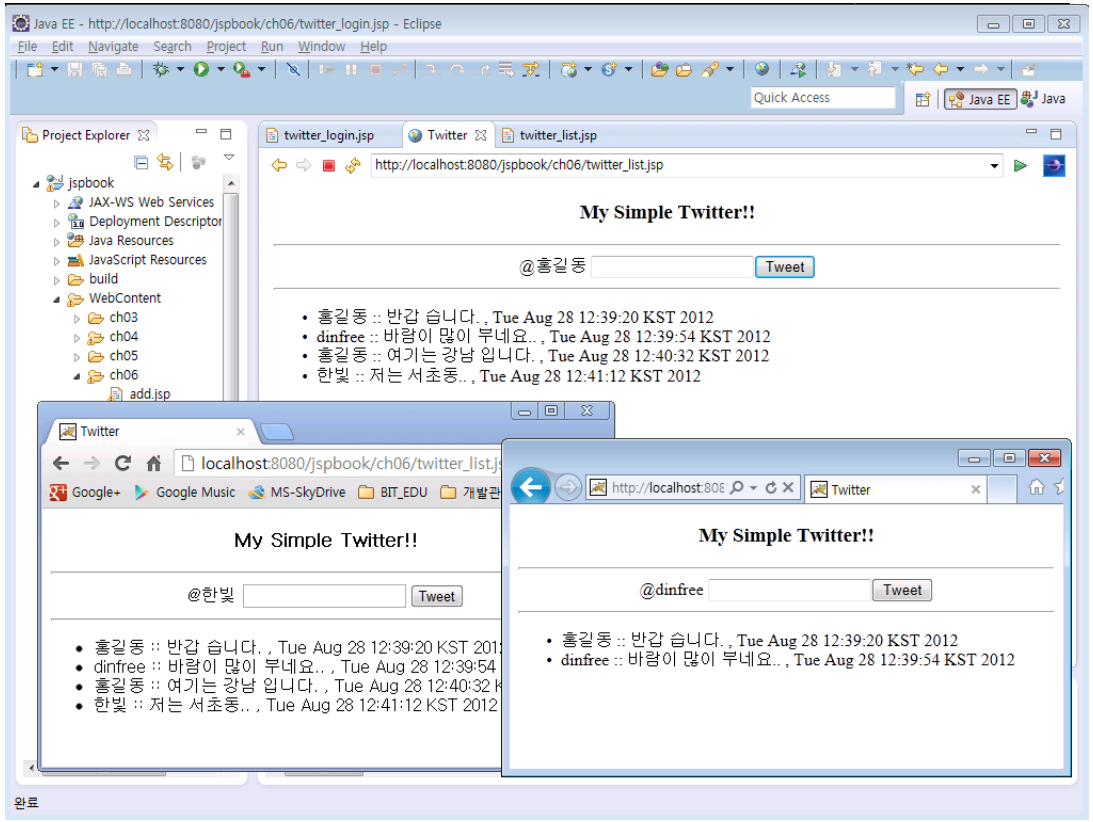
request1.jsp | calc.jsp | request2.jsp | twitter_login.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/T
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>ch06 : twitter_login.jsp</title>
</head>
<body>
<div align="center">
    <H2>트위터 로그인</H2>
    <form name="form1" method="POST" action="twitter_list.jsp">
        <input type="text" name="username"/>
        <input type="submit" value="로그인"/>
    </form>
</div>
</body>
</html>
```

09. [응용실습] JSP 내장객체: 트위터 구현

3. 게시글 등록 구현

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



[그림 6-18] 최종 실행 화면

09. [응용실습] JSP 내장객체: 트위터 구현

- 주요 소스코드 분석 (twitter_list.jsp)

notice.jsp error.jsp index.jsp Calendar.jsp twitter_list.jsp 23

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" import="java.util.ArrayList"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<%
    // 한글 캐릭터셋 변환
    request.setCharacterEncoding("UTF-8");
    // HTML 폼에서 username으로 전달된 값을 가지고 옴
    String username = request.getParameter("username");

    // username이 null 이 아닌 경우 세션에 값을 저장
    if(username != null) {
        session.setAttribute("user",username);
    }
%>
<html>
<head>
<title>ch06 : twitter_list.jsp</title>
</head>
<body>
<div align=center>
<H3>My Simple Twitter!!</H3>
<HR>
<form action="tweet.jsp" method="POST">
    <!-- 세션에 저장된 이름 출력 -->
    @<%= session.getAttribute("user") %> <input type="text" name="msg"><input type="submit" value="Tweet">
</form>
<HR>
```

09. [응용실습] JSP 내장객체: 트위터 구현

- 주요 소스코드 분석 (twitter_list.jsp)

```
27  <HR>
28  <div align="left">
29  <UL>
30  <%
31      // application 내장객체를 통해 msgs 이름으로 저장된 ArrayList를 가지고 옴
32      ArrayList<String>msgs = (ArrayList<String>)application.getAttribute("msgs");
33
34      //msgs가 null 이 아닌 경우에만 목록 출력
35      if(msgs != null) {
36          for(String msg : msgs) {
37              out.println("<LI>"+msg+"</LI>");
38          }
39      }
40  %>
41  </UL>
42  </div>
43  </div>
44  </body>
45  </html>
```

09. [응용실습] JSP 내장객체: 트위터 구현 (tweet.jsp)

notice.jsp error.jsp index.jsp Calendar.jsp twitter_list.jsp tweet.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ page import="java.util.*" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<%
    // 한글 캐릭터셋 변환
    request.setCharacterEncoding("UTF-8");

    // HTML 폼에서 전달된 msg 값을 가지고 옴
    String msg = request.getParameter("msg");

    // 세션에 저장된 로그인 사용자 이름을 가지고 옴
    String username = (String)session.getAttribute("user");

    // 메시지 저장을 위해 application 에서 msgs 로 저장된 ArrayList 가지고 옴
    ArrayList<String> msgs = (ArrayList<String>)application.getAttribute("msgs");

    // null 인 경우 새로운 ArrayList 객체를 생성
    if(msgs == null) {
        msgs = new ArrayList<String>();
    }

    // 사용자 이름, 메시지, 날짜 정보를 포함하여 ArrayList에 추가
    msgs.add(username+" :: "+msg+" , "+ new Date());
    // application 에 ArrayList 저장
    application.setAttribute("msgs",msgs);

    // 톰캣 콘솔을 통한 로깅
    application.log(msg+"추가됨");

    // 목록 화면으로 리다이렉팅
    response.sendRedirect("twitter_list.jsp");
%>
```

프로젝트로 배우는

자바 웹 프로그래밍

Servlet, JSP, JDBC

황희정 지음