프로젝트로 배우는

자바 웹 프로그래밍

Servlet, JSP, JDBC

황희정 지음

Chapter 06. JSP 내장객체

한빛아카데미 Hanbit Academy, Inc.

목차

JSP 내장객체의 구조적 특징 (p. 197)

```
public void _jspService(final javax.servlet.http.HttpServletRequest request,
                        final javax.servlet.http.HttpServletResponse response)
        throws java.io.IOException, javax.servlet.ServletException {
    final javax.servlet.jsp.PageContext pageContext;
    javax.servlet.http.HttpSession session = null;
    final javax.servlet.ServletContext application;
    final javax.servlet.ServletConfig config;
    javax.servlet.jsp.JspWriter out = null;
    final java.lang.Object page = this;
    javax.servlet.jsp.JspWriter _jspx_out = null;
    javax.servlet.jsp.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;

        ...

        // 사용자 작성 JSP 코드가 오는 위치
}
```

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 세션**이다.
- session 은 javax.servlet.http.HttpSession 인터페이스의 참조 변수이다.
- session 은 접속하는 사용자 별로 따로 생성되며 일정시간 유지되고 소멸된다.
- 이러한 세션의 특징을 이용해 setAttribute() 메서드를 이용해 임의의 값을 저장해 놓고 활용할 수 있음.
  - setAttribute(...), getAttribute()
- 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.

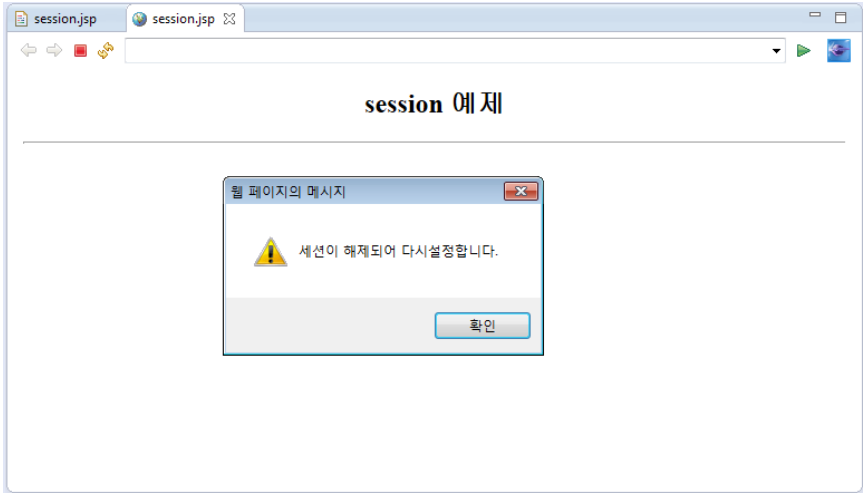
[표 6-5] session 내장객체 메서드

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

[예제 6-8] session.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>

<HTML>
<HEAD><TITLE>ch06 :session.jsp</TITLE></HEAD>
<BODY>
<div align="center">
<H2>session 예제 </H2>
<HR>
<%
    // isNew()  메서드를 이용해 최초세션설정을 확인하고 있다.
    if(session.isNew()) {
        out.println("<script> alert('세션이 해제되어 다시설정합니다.') </script>");
        session.setAttribute("login","홍길동");
    }
%>
# <%= session.getAttribute("login") %> 님 환영 합니다.!!!!<BR>
1. 세션 ID : <%= session.getId() %> <BR>
2. 세션 유지시간 : <%= session.getMaxInactiveInterval() %> <BR>
</div>
</BODY>
</HTML>
```

JSP 내장객체의 구조적 특징 (p. 197)

```
public void _jspService(final javax.servlet.http.HttpServletRequest request,
                        final javax.servlet.http.HttpServletResponse response)
        throws java.io.IOException, javax.servlet.ServletException {
    final javax.servlet.jsp.PageContext pageContext;
    javax.servlet.http.HttpSession session = null;
    final javax.servlet.ServletContext application;
    final javax.servlet.ServletConfig config;
    javax.servlet.jsp.JspWriter out = null;
    final java.lang.Object page = this;
    javax.servlet.jsp.JspWriter _jspx_out = null;
    javax.servlet.jsp.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;

        ...

        // 사용자 작성 JSP 코드가 오는 위치
}
```

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
String getServerInfo()	JSP/서블릿 컨테이너의 이름과 버전을 반환한다.
int getMajorVersion()	컨테이너가 지원하는 서블릿 API의 주 버전 정보를 반환한다.
int 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

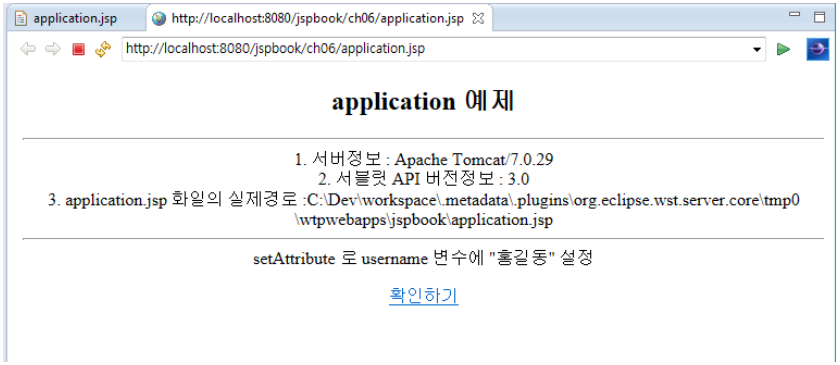
메서드	설명
log(message)	문자열 message의 내용을 로그 파일에 기록한다. 로그 파일의 위치는 컨테이너에 따라 다르다.
log(message,exception)	예외 상황에 대한 정보를 포함하여 로그 파일에 기록한다.

[표 6-10] 속성 관련 정보

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

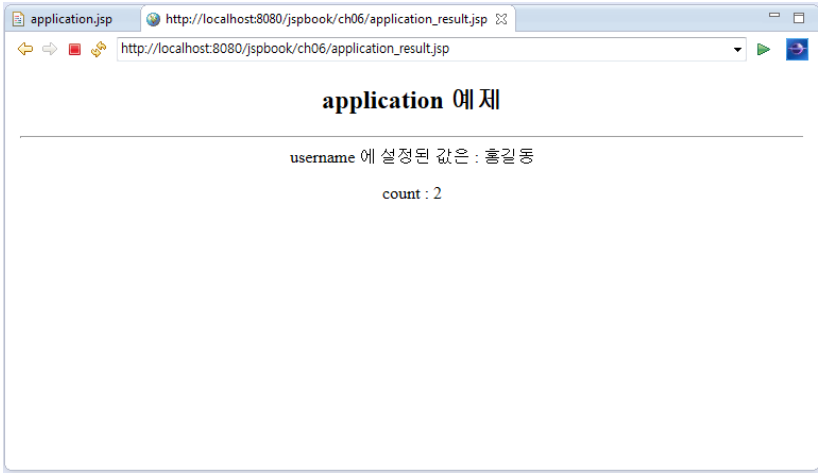
06. 그 밖의 내장객체

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

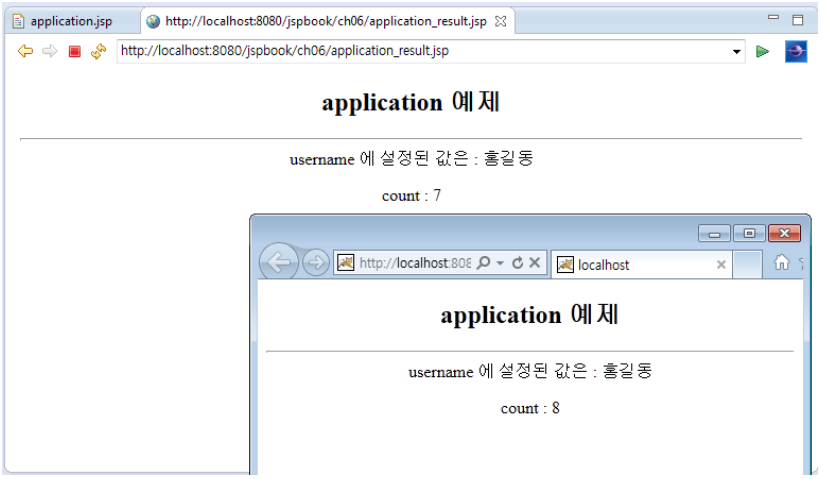


[그림 6-9] application.jsp 실행 화면

[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

Auto-boxing

- 포장 클래스(wrapper class)
  - int, double과 같은 기본 자료유형(primitive type)마다 매핑되는 포장 클래스가 존재
  - 기본형을 객체로 다루어야 할 경우에 유용
- boxing : 기본 자료형 → 참조 자료형
- unboxing : 참조 자료형 → 기본 자료형
- auto-boxing
  - JDK 1.5부터는 추가된 자동 boxing/unboxing 기능
  - auto-boxing은 대응되는 자료형 사이에만 일어난다는 점에 유의

- **before JDK 5.0**

```
int ia = 123;
Integer iA = new Integer(ia);  // boxing
Integer iB = new Integer(456);
int ib = iA.intValue(); // unboxing
int ic = iA.intValue() + iB.intValue();
// 사칙연산
```

- **after JDK 5.0**

```
int ia = 123;
Integer iA = 123;  // auto-boxing
Integer iB = new Integer(456);
int ib = iA;    // auto-unboxing
int ic = iA + iB;   // 사칙연산
Integer iC = iA + iB;  // 사칙연산
```

[예제 6-9] application.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<html>
<head><TITLE> </TITLE></HEAD>
<BODY>
<div align="center">
<H2>ch06 :application 테스트</H2>
<HR>
1. 서버정보 : <%= application.getServerInfo() %> <BR>
2. 서블릿 API 버전정보 : <%= application.getMajorVersion() +"."+application.getMinorVersion() %> <BR>
3. application.jsp 화일의 실제경로 :<%= application.getRealPath("application.jsp") %> <BR>

<HR>
setAttribute 로 username 변수에 "홍길동" 설정<P>
<% application.setAttribute("username","홍길동");
      application.log("username=홍길동");
      application.setAttribute("count",1) ;
%>

<a href="application_result.jsp">확인하기</a>

</div>
</BODY>
</HTML>
```

auto-boxing이 적용된 코드

초기 자바 컴파일러 코드는 다음과 같다.

```
application.setAttribute("count", new Integer(1));
```

[예제 6-10] application_result.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>

<HTML>
<HEAD><TITLE> </TITLE></HEAD>
<BODY>
<div align="center">
<H2>application 예제</H2>
<HR>
username 에 설정된 값은 : <%= application.getAttribute("username") %> <P>
<%
    Integer count = (Integer)application.getAttribute("count");
    int cnt = count.intValue()+1;
    application.setAttribute("count",cnt);
%>
count : <%= cnt %>
</div>
</BODY>
</HTML>
```

auto-boxing이 적용되는 자바 8.0 이상에서는 다음과 같은 코드로 대치 가능

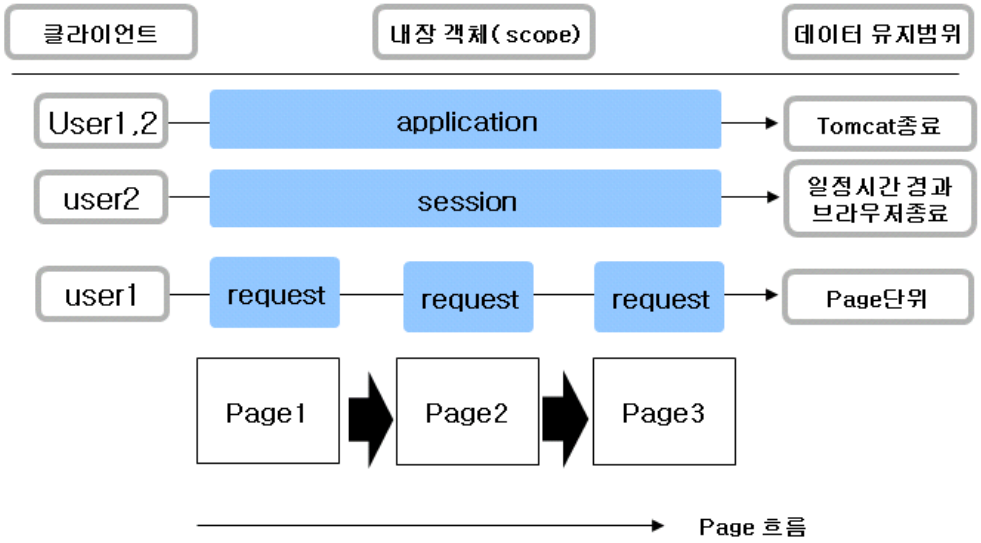
```
count = count + 1;
application.setAttribute("count", count);
out.println("count : <%= count %>
```

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 내장객체**를 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). page마다 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.
- **void setAttribute(String name, Object value)**
  - 속성을 내장 객체 내부에 저장하는 메소드
- **Object getAttribute(String name)**
  - 내장 객체에 저장된 값을 가져오는 메서드
  - 리턴되는 타입이 Object 이므로 속성을 **가지고 올 때에는 적절한 형 변환이 필요**하다.
- 예를 들어 page1에서 session.setAttribute("name,"홍길동")으로 문자열 객체를 저장한다면 page3에서는 session.getAttribute("name")으로 저장된 값을 참조할 수 있다.

프로젝트로 배우는

자바 웹 프로그래밍

Servlet, JSP, JDBC

황희정 지음