Chapter 6 쿠키와 세션

쿠키와 세션은 웹프로그래밍에 있어서 없어서는 안될 존재가 되어있다. 그 이유는 http 프로토콜의 특성 때문인데 이 놈의 프로토콜이 한 번 요청을 한 후에는 단절이 된다는 것이다. 따라서 사용자 정보를 유지 할 수가 없다. 어떤 경우에 이러한 사용자 정보의 유지가 필요한 것일까? 일반적으로 회원제 웹사이트를 운영하고 있다면 사용자 ID와 비밀번호를 요구하게 된다. 따라서 사용자 정보를 유지할 길이 없다면 매번 페이지를 접근할 때 마다 ID와 비밀번호를 요구해야 한다. 쇼핑몰의 경우도 마찬가지이다. 쇼핑몰을 이리 저리 돌아다니면서 상품을 장바구니에 담는데 사용자 정보가 없다면 장바구니에 담는다는 개념 자체가 모호해질 것이다. 매번 장바구니에 다른 상품만이 존재하거나 장바구니가 비어져 있을 테니까… 물론 장바구니 기능을 데이터베이스로 이동 시킬 수도 있다. 데이터베이스로 기능을 이전시킨다면 문제가 없을 것이며 실제로 이런 식으로 구현하는 경우도 있지만 대규모의 하드웨어 장비를 갖추지 않은 곳이라면 서버에 많은 부담이 될 것이다. 또 하나 광고에 관련된 것이 있을 수 있는데 사용자들의 성향에 따라서 사용자마다 그 관심사에 따라 광고의 내용을 달리 하는 정책이 있을 수 있다. 이럴 때에 그런 사용자의 관심 정보를 쿠키에 넣어 놓고 진행할 수 있다. 필자가 개발하고 있는 eCRM(Ebusiness 상에서의 고객관계관리) 솔루션은 이러한 개념을 진일보 시켜서 좀더 분석적인 마케팅효과를 노린 개발이라고 할 수 있겠다. 하지만, 복잡하지 않은 어느정도의 eCRM의 개념을 쿠키등을 이용해서 적용할 수 있을 것이다. 그럼 먼저 쿠키에 대해서 알아보자.

Cookie

쿠키로 사용자 상태 유지를 하려고 할 때 가지는 단점을 살펴보자. 쿠키는 간단한 텍스트 형태의 파일로 클라이언트에게 보내진다. 따라서 어떤한 보안 매커니즘도 가지고 있지 않다. 이러한 보안의 문제 때문에 종종 사람들은 쿠키기능을 거부할 때도 있다. 쿠키 기능을 거부한다면 쿠키는 더 이상 사용하지 못하게 된다. 하지만, 쿠키에 대해서 그렇게 민감한 사람이 얼마나 있을까하는 의문도 든다. 또 하나 쿠키가 가지는 단점은 사이트당 20개의 쿠키, 모두 합쳐서 300개의 쿠기가 최대이며 각 쿠키는 4 킬로바이트를 넘을 수 없다. 이러한 단점에도 불구하고 쿠키는 널리 사용되고 있고 우리의 목적은 쿠키를 어떻게 사용할 수 있느냐는 것이므로 JSP에서는 어떻게 쿠키를 사용하는지를 보도록 하자.

Cookie API

쿠키는 javax.servlet.http 패키지에 들어 있는 클래스이다. 쿠키를 설정해서 보내는 일은 대충 3 단계의 순서로 이루어진다.

1. 쿠키 객체를 생성한다
2. 쿠키의 속성을 설정한다
3. 쿠키를 전송한다

쿠키 API 에 대해서 구체적으로 알아보자.
생성자는 Cookie(String name, String value)로 되어있다. name과 value를 한 쌍으로 쿠키를 셋팅한다.
속성의 설정은 대부분 setXxx 로 되어 있다. 반대로 속성을 살펴보려면 getXxx 메쏘드를 사용하면 된다.

public int getMaxAge() : 쿠키의 유효 시간을 얻어온다. 파기되기까지의 시간을 초단위로 리턴한다.
public void setMaxAge(int MaxAge) : 쿠키의 유효 시간을 설정한다. 물론 초단위이다. 디폴트 값은 –1 이며 이는 현재 세션이 살아있는 동안 계속 살아있다는 것을 의미한다.

public String getDomain() : 쿠키가 적용되는 도메인을 가져온다.
public void setDomain(String Domain) : 쿠키가 적용될 도메일을 설정한다. 브라우저는 쿠키를 서버쪽으로 보낼 때 원래 그 쿠키를 보낸 서버 호스트에게만 돌려준다. 따라서 setDomain 메쏘드를 사용해서 셋팅을 하면 그 도메인에 속해있는 다른 호스트도 이 쿠키를 접근할 수 있으므로 같은 도메인내에서 쿠키를 공유할 수가 있게 된다. 주의할 점은 . 을 2개이상 설정해야 한다는 것이다. 도메인의 이름을 .yahoo.com 과 같이 해야지 그냥 yahoo.com 으로 해서는 안된다.

public String getPath() : 쿠키가 적용될 수 있는 path를 가져온다.
public void setPath(String path) : 쿠키의 적용 path를 설정하게 되는데, 만약 이 설정을 하지 않으면 쿠키를 보낸 path의 하위 path에서는 접근이 가능하지만 다른 path에서는 접근이 불가능하다. 무슨 이야기인고 하니, jsp.boolpae.com/jsp/example.jsp 에서 쿠키를 보냈다면 jsp의 하위 폴더의 웹페이지에서는 이 쿠키에 접근할 수 있지만 jsp.boolpae.com/xml 이라는 폴더의 하위 폴더에서는 접근할 수 없다. 따라서 웹어플리케이션의 모든 곳에서 접근이 가능하도록 하려면 setPath("/") 와 같이 설정해 주면 된다. / 는 루트 디렉토리를 가리키므로…

public void setValue(String value) : 쿠키의 value(값)을 설정한다. 이미 생성되어 있는 쿠키이므로 값을 바꾸는 메쏘드로 이해하면 될 것이다.
public String getValue() : 쿠키의 값을 가져온다.

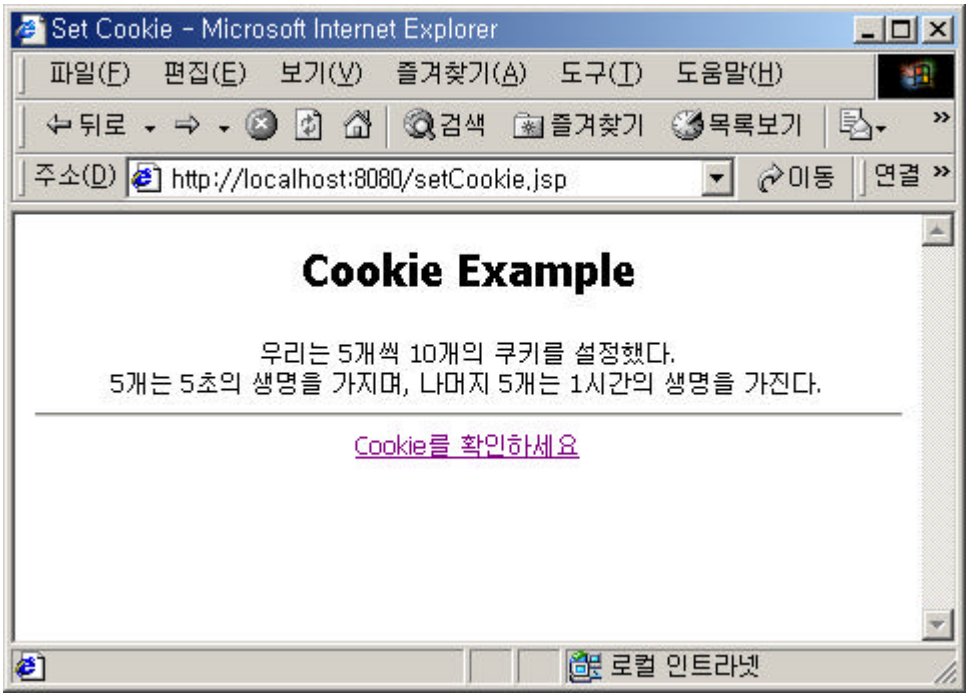
Example

쿠키를 셋팅하는 예제를 보자. 이 예제는 JSP 페이지에서 예를 들도록 하겠다.

List 6.1 setCookie.jsp

```
<HTML>
<HEAD>
<TITLE> Set Cookie</TITLE>
</HEAD>
<BODY>
<%
for(int i=0;i<5;i++){
Cookie c = new Cookie("First Cookie"+i, "Value"+i);
c.setMaxAge(5);
response.addCookie(c);

c = new Cookie("Second Cookie"+i, "Value"+i);
c.setMaxAge(60*60);
response.addCookie(c);
}%>
<center>
<h1> Cookie Example </h1>
우리는 5개씩 10개의 쿠키를 설정했다.<br>
5개는 5초의 생명을 가지며, 나머지 5개는 1시간의 생명을 가진다.
<hr>
<a href="showCookie.jsp">Cookie를 확인하세요</a>
</BODY>
</HTML>
```

쿠키를 총 10개 설정했으며 그 유효시간을 5개는 5초, 나머지 5개는 1시간으로 설정했다. 유효시간은 초단위로 설정하기 때문에 1시간이면 3600 초이다. 필자는 사칙연산에 약해서 60*60으로 했다. 1년이면 60*60*24*365 가 될 것 같다. 여러분은 머리가 뛰어날 것이라 예상되므로 모두 곱해서 넣어도 된다. List 6.1의 결과 화면은 아래와 같다.



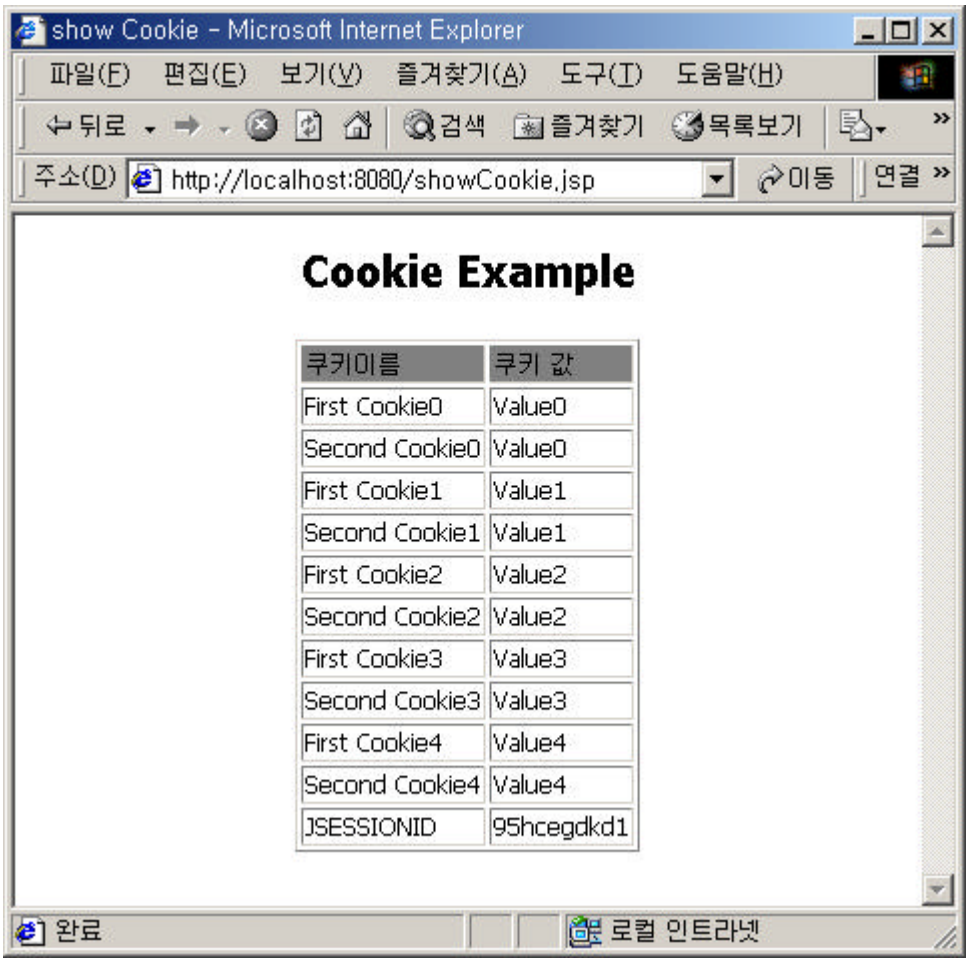
List 6.2 showCookie.jsp

```
<HTML>
<HEAD>
<TITLE> show Cookie </TITLE>
</HEAD>
<BODY>
<center>
<h1> Cookie Example </h1>
<table border=1>
<tr><td bgcolor="gray">쿠키이름</td>
<td bgcolor="gray">쿠키 값</td></tr>
<% Cookie[] c = request.getCookies();
for(int i=0;i<c.length;i++){
    Cookie cookie = c[i];
%>
<tr><td><%=cookie.getName()%></td><td><%=cookie.getValue()%></tr>
<% } %>
</table>
</BODY>
```

```
</HTML>
```

List 6.2는 쿠키를 직접확인하는 JSP 페이지이다. Cookie를 확인하기 위해서는 코드에서 보는 바와 같이 request.getCookies() 메쏘드를 이용해서 전체쿠키 객체 배열을 반환받는다.
배열을 루프를 돌면서 getName 과 getValue 메쏘드를 이용해서 화면에 출력시켰다. 결과 화면은 두가지인데 한가지는 쿠키를 셋팅한후 바로 쿠키를 확인한 결과 화면이며, 다른 한가지는 5초가 지난 후에 확인한 화면이다. 5개의 쿠키는 5초의 생명력을 가지므로 5초 이후에는 사라지고 없을 것이다.

[셋팅후 바로 확인한 결과]



[5초가 지난후 확인한 결과]

Second Cookie 만이 남아있는 것을 확인할 수 있다. 한가지, JSESSIONID 라는 것이 있는데 처음 요청이 들어왔을 때 세션ID를 먼저 셋팅함을 알 수 있다. 하지만, 이 세션 ID는 쿠키를 클라이언트측에 저장하는 것은 아니다.

CookieUtil

쿠키를 검색하기 위한 작업을 간편히 하기위해 클래스를 만들어 보자.

이 클래스는 쿠키 객체를 반환하는 것과 쿠키의 값을 반환하는 두가지 메쏘드들을 가질 것이다.

List 6.3 CookieUtil.java

```
package com.boolpae.jsp;

import javax.servlet.http.Cookie;

public class CookieUtil{
    public static String getValue(Cookie[] cookies, String name){
        for(int i=0;i<cookies.length;i++){
            if(name.equals(cookies[i].getName()))
```

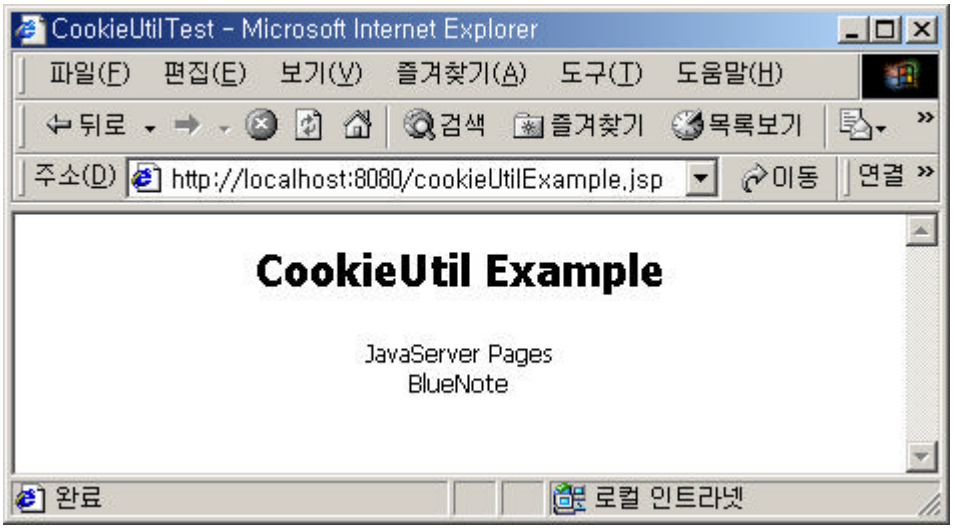
```
                return cookies[i].getValue();
        }
        return null;
    }

    public static Cookie getCookie(Cookie[] cookies, String name){
        for(int i=0;i<cookies.length;i++){
            if(name.equals(cookies[i].getName()))
                return cookies[i];
        }
        return null;
    }
}
```

두 메쏘드 모두 아규먼트로 Cookie 배열과 이름을 받아들이고 이름과 일치하는 Cookie가 있으면 그 값(value) 또는 직접 쿠키를 반환한다. 둘 다 static 메쏘드이므로 클래스의 초기화 없이 사용가능하다. 그럼 이 클래스를 사용한 JSP 페이지를 보도록 하자.

List 6.4 CookitUtilExample.jsp

```
<%@ page import="com.boolpae.jsp.CookieUtil" %>
<% Cookie c = new Cookie("Book", "JavaServer Pages");
        response.addCookie(c);
        c = new Cookie("Jazz","BlueNote");
        response.addCookie(c);
%>
<HTML>
<HEAD>
<TITLE> CookieUtilTest </TITLE>
</HEAD>
<BODY>
<center>
<h1>CookieUtil Example</h1>
<%= CookieUtil.getValue(request.getCookies() , "Book") %><br>
<%= CookieUtil.getValue(request.getCookies() , "Jazz") %>
</BODY>
</HTML>
```

쿠키를 두개 넣고 확인하는 간단한 예이다. 처음에는 둘다 null 로 출력될 것이다. 최초에는 Cookie가 없기 때문이다. 쿠키는 response Header를 통해서 보내는 것이기 때문이다. 그다음 refresh 버튼을 클릭해보면 아래와 같은 화면이 출력될 것이다.



이번에 만들어 볼 것은 쿠키클래스를 상속해서 우리가 쿠키 관련 설정을 할 때 편하도록 하게 해주려고 한다. 소스 코드를 바로 보도록 하자.

List 6.5 BlueCookie.java

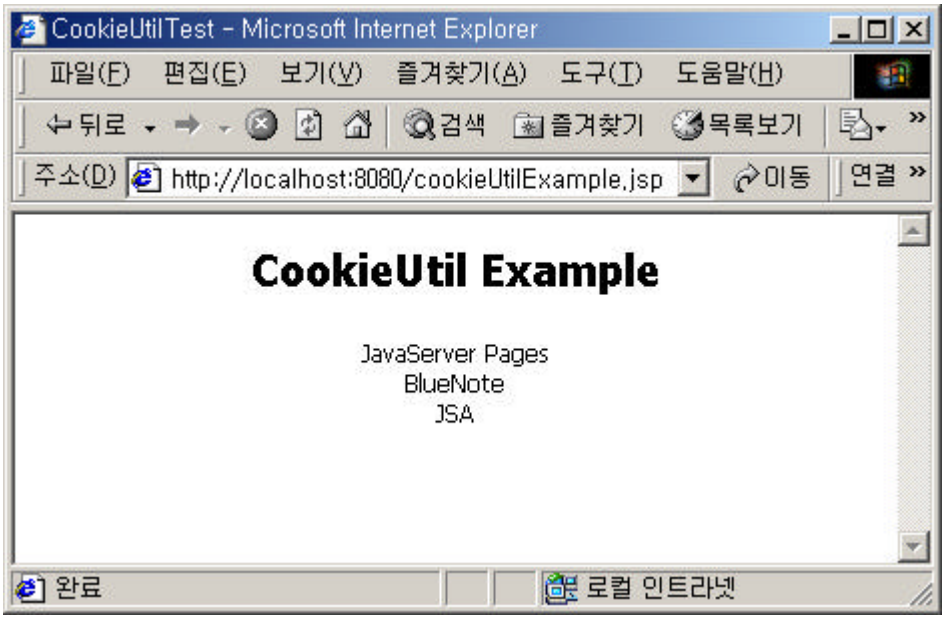
```
package com.boolpae.jsp;

import javax.servlet.http.Cookie;

public class BlueCookie extends Cookie{

    public static final String path = "/";

    public BlueCookie(String name, String value, int maxAge){
        super(name, value);
        setMaxAge(maxAge);
        setPath(path);
    }
}
```

maxAge 즉, 쿠키의 유효시간은 아규먼트로 받아들여서 정하고 path는 항상 모든 웹페이지에서 접근이 가능하도록 / 를 사용했다. CookieUtilExample.jsp 페이지를 아래와 같이 수정하자.

List 6.6 CookieUtilExample.jsp

```
<%@ page import="com.boolpae.jsp.*"%>
<% Cookie c = new Cookie("Book", "JavaServer Pages");
        response.addCookie(c);
        c = new Cookie("Jazz","BlueNote");
        response.addCookie(c);
        c = new BlueCookie("Movie","JSA", 5);
        response.addCookie(c);
%>
<HTML>
<HEAD>
<TITLE> CookieUtilTest </TITLE>
</HEAD>
<BODY>
<center>
<h1>CookieUtil Example</h1>
<%= CookieUtil.getValue(request.getCookies() , "Book") %><br>
<%= CookieUtil.getValue(request.getCookies() , "Jazz") %><br>
<%= CookieUtil.getValue(request.getCookies() , "Movie") %>
</BODY>
</HTML>
```

결과화면은 여러분이 예상하듯이 아래와 같다. 역시나 5초 지나고 refresh를 하면 Movie는 null 로 출력될 것이다.



지금까지 예제에서 보았듯이 쿠키는 객체를 생성만 했다고 끝나는 것이 아니라 response 헤더에 보내야 한다는 것을 명심하자. 따라서 addCookie 메쏘드를 호출하지 않는다면 쿠키는 클라이언트측에 저장되지 않는다. 마지막으로 쿠키를 삭제하려면 쿠키의 maxAge를 0으로 해주고 다시 addCookie해주어야 한다. 아래의 간단한 예를 보자.

```
Cookie c = new Cookie(“xxx”, “yyy”);
response.addCookie(c);
c = new Cookie(“xxx”, “yyy”);
c.setMaxAge(0);
response.addCookie(c);
```

쿠키를 삭제하기 위해서는 위와 같은 편법(?)을 써서 maxAge가 0인 쿠키와 바꿔치기를 해야하는 것이다.

Session

쿠키를 설명할 때 Http 프로토콜은 지속성이 없다고 이미 이야기 하였다. 이런 것을 해결하기 위해 우리는 앞서서 Cookie를 살펴보았고 쿠키의 한계도 알아보았다. 그럼 쿠키이외에 이런 문제를 해결할 수 있는 방법은 무엇이 있을까? URL-rewriting 과 Hidden Form 그리고 우리가 이제 알아보려고 하는 session 이 있다. 간단히 설명하면 URL-rewriting 은 주소창에 sessionID를 달고 다니는 것이며, Hidden Form 은 HTML form 태그 내부에 <input type=”hidden” name=”name” value=”value”> 이런 식으로 계속 GET 또는 POST 방식의 파라미터로 넘기는 것이다. 이들은 약점이 많아 특별한 경우가 아니면 사용

되지 않을 것이다. 우리의 관심사는 Session에 있다.

Session API

Session의 정확한 명칭은 javax.servlet.http.HttpSession 인터페이스이다. HttpSession 인터페이스의 메쏘드들을 살펴보자.

public Object getAttribute(String name)
name을 가지고 value를 가지고 온다. 쿠키에서는 반환 값이 String인 반면 session에서는 Object임을 항상기억하자. 서블릿API 2.1 버전까지는 이 메쏘드가 없었다. 이 때는 getValue 메쏘드가 이 역할을 하였는데, 2.2 버전에서는 getAttribute 메쏘드가 같은 일을 하게 되었다. getValue 메쏘드는 추천되지 않는 사장될 메쏘드로 지정되었다.(AIP문서에서 deprecated라고 쓰여있는 메쏘드들이 그러하며 호환성에 문제가 될 수도 있다)
pubic void setAttribute(String name, Object value)
세션에 값을 저장하는 메쏘드이다. 여기서도 두번째 아규먼트가 Object 형이라는 것을 눈여겨 보아두길 바란다. 역시 putValue라는 메쏘드가 있으나 getValue의 경우와 같이 deprecated 된 메쏘드이다.

public void removeAttribute(String name)
속성을 삭제하는 메쏘드이다. 이 메쏘드역시 removeValue 메쏘드가 있으며 deprecated 되었다. 어쨌든 가장 최근에 추천되는 메쏘드를 사용하면 별 문제가 없을 것이다.

public Enumeration getAttributeNames()
세션의 name 전체를 Enumeration 타입으로 반환한다.

public String getId()
세션 ID 를 반환한다. 덧붙이자면 사용자를 식별해야 하므로 세션 ID는 유일하다.

public long getCreationTime()
세션이 만들어진 시간을 long 형으로 반환한다. 단위는 milliSecond이다. 이 시간이 의미하는 바는 1970년 1월 1일을 기준으로 몇 milliSecond가 지났는지를 말하는 것이다.

public void invalidate()
세션을 무효화 시킨다. 세션을 제거할 때 사용된다.

public boolean isNew()
세션이 새로 생성되었는지 여부를 boolean 형으로 반환한다. invalidate 메쏘드를 사용한

후 isNew를 호출하면 true가 반환될 것이다.

Example

그럼 허덥하기 그지없는 쇼핑카트를 만들어 보도록 하자. 만들어 볼 쇼핑카트는 책을 팔고 있는 쇼핑몰이다. 우습게도 이 온라인 서점에서는 단 두 권의 책만을 판다. 파란공책이 쓴 JSP랑 째진공책이 쓴 JSP책이다. 먼저 클래스들을 만들어 볼 것인데, 온라인 서점에서 필요한 것들이 무엇인지를 가만히 생각해보자.
먼저 책들의 목록을 가지고 있는 '목록'이 있을 것이고 책의 제목, 가격등을 가지는 '책'이라는 것이 있을 것이다. 그다음 주문의 내역이 표시되는 '주문'이라는 것이 있을테고 이러한 주문상태를 담아놓는 '장바구니'가 있을 것이다. 이렇게 대체로 명사로 표현할 수 있는 것들은 클래스로 만들어야 한다고 생각하자. 각각의 클래스가 가지고 있어야 할 기능을 생각해보면 그것들이 클래스들의 메쏘드가 될 것이며, 고유의 상태정보는 필드(클래스 변수 혹은 인스턴스 변수)가 될 것이다.
그리고 프리젠테이션 부분은 JSP가 담당할 것이기 때문에 화면에 책의 리스트를 보여주고 장바구니에 담을 수 있도록 해주는 Books 라는 이름의 JSP페이지를 만들 것이다. 다음으로 장바구니의 내역을 보여주는 CartPage라는 JSP페이지를 만들 것이다. CartPage.jsp 페이지는 장바구니에 담는 비니지스 로직을 구현하는 부분이다. 사실 여기서는 이 CartPage.jsp 페이지에 로직처리과 프리젠테이션이 어느정도 섞여 있다. 이 부분을 확연히 찢는(?)다면 Request 요청을 받아서 처리하는 부분은 서블릿이 맡을 것이며 처리가 끝나면 장바구니를 보여주는 JSP 페이지로 넘겨서 처리하게 될 것이다. 우리는 현재 주문을 Cartpage.jsp 내에서 처리하고 보여 주도록 하고 있으므로 그 것을 중심으로 보자. 이런 방식이 무조건 나쁜 것이 아니다. 때로는 이런 방식이 필요할 때도 있다고 언급한 적이 있다. 이 예제는 훗날 쇼핑몰 프로젝트의 토대가 될 것이며 그때 가서 이 CartPage.jsp 페이지도 찢도록 하겠다. 우선 클래스들을 만들어 보자. 그 첫번째는 책을 표시하는 Book 클래스이다.

List 6.7 Book.java

```
package com.boolpae.jsp;

public class Book{
    private String isbn;
    private String title;
    private int price;

    public Book(String isbn, String title, int price){
```

```
        this.isbn = isbn;
        this.title = title;
        this.price = price;
    }

    public String getIsbn(){
        return isbn;
    }

    public String getTitle(){
        return title;
    }

    public int getPrice(){
        return price;
    }
}
```

보시다시피 특별히 볼 부분은 없다. 생성자에서 isbn, 타이틀, 가격(단가) 변수를 셋팅하고 그 변수들을 반환하는 메쏘드뿐이다. 이 클래스는 책 자체의 정보들만 가지고 있는 것이다. 다음은 BookList 클래스인데, BookList 클래스는 전체 책의 목록을 가지고 있다.

List 6.8 BookList.java

```
package com.boolpae.jsp;

import java.util.Vector;

public class BookList{

    private static Vector books= new Vector();

    static{
        books.addElement(new Book("A010101", "JavaServer Pages by 파란 공책",
100 ));
        books.addElement(new Book("A020202", "JavaServer Pages by 째진 공책",
40 ));
    }

    public static Vector getBooks(){
        return books;
    }
```

```
    public static Book getBook(String isbn){
        if(isbn == null) return null;
        for(int i=0;i<books.size();i++){
            Book book = (Book)books.elementAt(i);
            if(isbn.equals(book.getIsbn())){
                return book;
            }
        }
        return null;
    }
}
```

BookList 클래스는 전체 책의 목록을 가지고 있다. 이미 우리는 Book 클래스를 만들었고 이 Book 클래스를 담고 있는 것이다. 이 목록은 다른 페이지에서도 바로 가져다 쓸 가능성이 많고 매번 객체를 생성하는 것보다 효율적일 것이라는 생각에 static 변수로 선언했다. 그리고 static 초기화 블록을 사용해서 books 변수를 초기화 했다. 혹시 이런 초기화 블록을 보지 못하신 분들은 static 변수의 초기화는 static{ … } 으로 사용할 수 있다는 것을 알아 두면 된다. 그리고 두가지 메쏘드가 있는데 하나는 책목록을 Vector 로 반환하는 것이며 다른 하나는 특정 ISBN을 가지고 그 ISBN에 해당하는 Book 객체를 반환하는 메쏘드이다. 다음으로 볼 클래스는 '주문' 클래스이다.

List 6.9 Order.java

```
package com.boolpae.jsp;

public class Order{
    private Book book;
    private int count = 1;

    public Order(Book book){
        this.book = book;
    }

    public Book getBook(){
        return book;
    }

    public void plusCount(){
        count++;
    }

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

    public int getCount(){
        return count;
    }
```

```
    public int getTotalPrice(){
        return book.getPrice()*count;
    }
}
```

Order 클래스는 주문한 책과 책의 수량에 관한 정보를 가지고 있는 클래스이다. 따라서 Book 과 수량(count) 두가지의 변수를 가지고 이 두 변수를 변경하고 가져오는 메쏘드들을 가지며 가격과 수량을 곱한 총합계 금액을 계산하는 메쏘드를 가지고 있다. 첫 주문 수량을 1로 초기화 시키며 수량을 1증가시키는 plusCount 메쏘드와 한꺼번에 변경할 수 있는 setCount 메쏘드가 있음을 주목하라. 다음은 장바구니(Cart)클래스이다.

List 6.10 Cart.java

```
package com.boolpae.jsp;

import java.util.Vector;

public class Cart{
    private Vector ordered;

    public Cart(){
        ordered = new Vector();
    }

    public Vector getOrdered(){
        return ordered;
    }

    public void addBook(String isbn){
        for(int i=0;i<ordered.size();i++){
        Order order = (Order)ordered.elementAt(i);
            if(isbn.equals(order.getBook().getIsbn())){
                order.plusCount();
                return;
            }
        }
        Order newOrder = new Order(BookList.getBook(isbn));
        ordered.addElement(newOrder);
    }

    public void setCount(String isbn, int count){
        Order order;
        for(int i=0;i<ordered.size();i++){
            order = (Order)ordered.elementAt(i);
            if(isbn.equals(order.getBook().getIsbn())){
                if(count<=0){
                    ordered.removeElementAt(i);
                }else{
                    order.setCount(count);
                }
                return;
```

```
            }
        }
        Order newOrder = new Order(BookList.getBook(isbn));
        ordered.addElement(newOrder);
    }
}
```

이 장바구니 클래스는 조금 복잡해 보이는데 실체는 별게 없다는 것을 알게 될 것이다. Vector 형의 ordered 변수는 Order 를 저장하는 Vector 변수이다. 주문서를 하나씩 장바구니에 담아 둔다고 생각하면 된다. 책을 한권 더 사려고 할때는 addBook 메쏘드를 호출해서 주문서를 수정할 것이며, 왕창 더사거나 취소할때는 setCount 메쏘드를 사용해서 주문서를 수정할 것이다. 사실 한 권 더 사려면 주문서에서 한 권을 더 수정하면 되는데, 장바구니에 담아놓고 또 이리저리 쇼핑을 하러 돌아다니다가 장바구니의 주문서를 안뒤져 보고 그냥 무턱대고 한 권 더 사게 될 때 addBook 메쏘드가 호출되는 것이다. 똑 같은 책의 주문서를 두장 담아 두지 않게 하기 위해서이다. 별게 없지 않은가?
addBook 메쏘드를 조금 더 살펴보면 현재 주문서에서 같은 ISBN의 책이 있는지 확인하고 같은 책이 있으면 수량만 1을 증가시키고(주문서-Order 클래스의 plusCount 메쏘드) 만약, 새로운 책이라면 새 주문내역을 주문서에 한 줄 더 추가하는 것이다.(ordered 변수에 새로운 Book 객체를 addElement 한다는 소리다)
setCount 메쏘드는 주문서의 주문 수량을 조절한다. 수량을 조절할 책의 ISBN과 수량을 가지고 주문서의 해당 ISBN을 찾아서 수량을 업데이트한다.(주문서-Order 클래스의 setCount 메쏘드) 만약 조절하려는 수량이 0이거나 0보다 작으면 주문을 삭제 처리한다.
만약 장바구니에 없는 책을 수량조절을 한다면 새로운 책을 장바구니에 담게된다. 사실 장바구니에서 조회도 안되는 책을 수량조절한다는 게 말이 안된다. 근데, 미리 말하지만 이 장바구니는 세션의 유효시간동안에만 저장이 된다. 따라서 세션이 끊어진 후에 다시 수량조절을 하려는 경우에는 새로운 책을 담게 된다는 것이다. 또는 URL 주소창에 변수값을 바로 치고 들어오는 성격이 약간 모난 사람의 경우도 있을 것이다. 이랬거나 저랬거나 우리가 원하는 모든 클래스를 만들었다. 이제 이것들을 가지고 사용자에게 보여주면 된다. 이제 JSP 페이지를 만들어보자. 일단 우리 서점을 찾아주시는 손님들에게 책의 목록을 보여주어야 할것이다. 책은 두권 밖에 없지만…

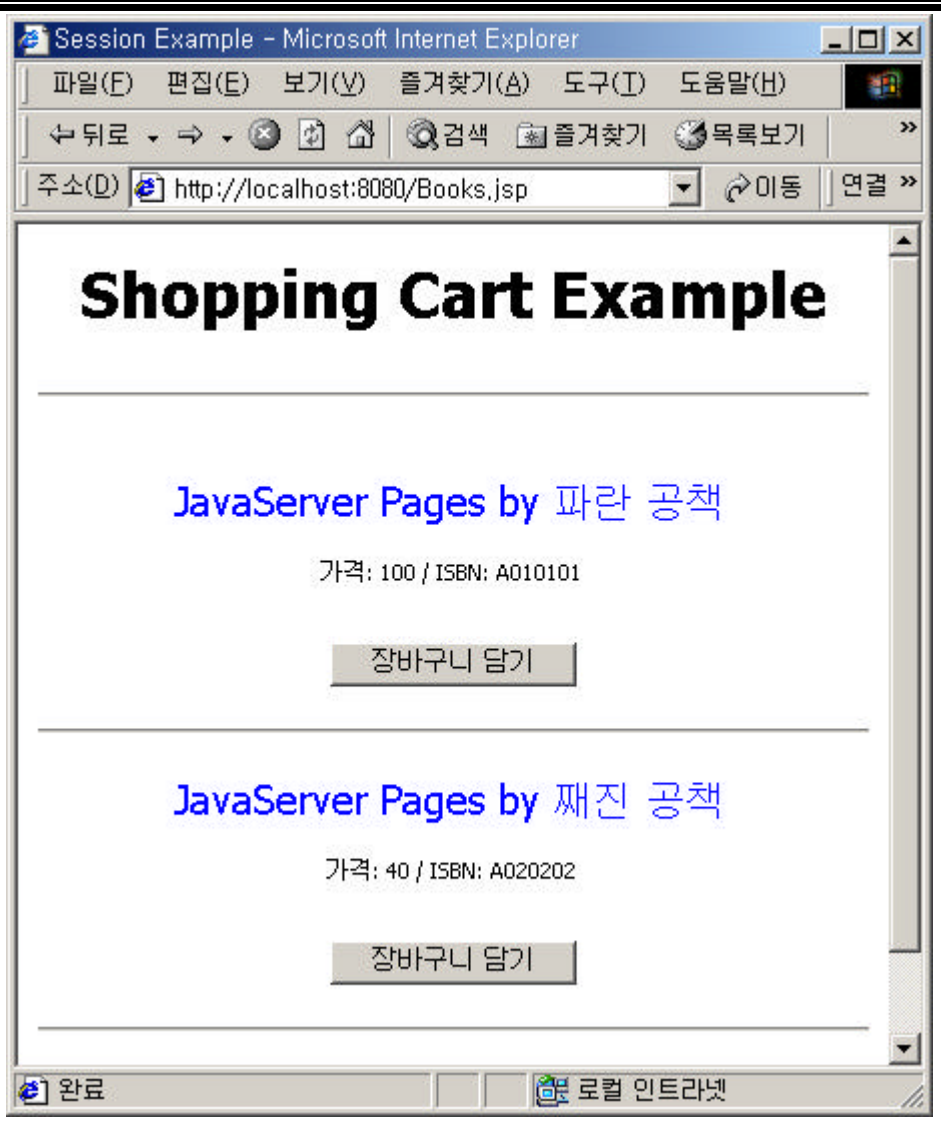
List 6.11 Books.jsp

```
<%@ page import="java.util.Vector" contentType="text/html;charset=EUC-KR" %>
<HTML>
<HEAD>
<TITLE> Session Example </TITLE>
<style>
BODY, PRE {font-family:Tahoma, 돋움; font-size:8pt}
.blue{color:Blue; font-size:15pt}
</style>
```

```
</HEAD>
<BODY>
<center>
<h1> Shopping Cart Example </h1>
<hr>
<pre>
<%
Vector books = com.boolpae.jsp.BookList.getBooks();
for(int i=0;i<books.size();i++){
      com.boolpae.jsp.Book book = (com.boolpae.jsp.Book)books.elementAt(i);
%>
<font class= blue><%= book.getTitle()%></font><p>
가격: <%=book.getPrice()%> / ISBN: <%=book.getIsbn()%>
<form method='get' action='CartPage.jsp'>
<input type='hidden' name="isbn" value='<%=book.getIsbn()%>'>
<input type='submit' value='장바구니 담기'>
</form>
<hr>
<% } %>
</BODY>
</HTML>
```

역시 JSP 페이지도 별게 없다. 책 목록에서 책을 가져와서 마음껏 2권을 뿌려주는 것이다. <form> 의 내용을 받아서 처리할 곳은 CartPage.jsp 페이지이며 이 페이지가 장바구니의 내용을 보여주는 역할까지 하게 된다고 미리 말했었다. Form에서 넘기는 것은 보시다시피 hidden 타입으로 ISBN만들 파라미터로 넘긴다. Book 클래스를 패키지 이름까지 포함해서 전체이름으로 써준 이유는 그냥(!)이다. 여러분은 page 지시자의 import 속성을 사용해도 된다. 이런 것들을 가지고 괜히 고민하시는 분들이 가끔 계신 것 같아서 그냥 이야기 해보았다. List 6.11의 실행화면은 아래와 같다.

[Books.jsp]



예상했던 대로 파란공책과 째진공책이 쓴 두권의 JSP 책만 파는 허덥한 서점이다. 이제 장바구니에 담아보자. ISBN을 파라미터로 받아서 처리하는 CartPage.jsp 를 보자.

List 6.12 CartPage.jsp

```
<%@ page import="java.util.Vector" contentType="text/html;charset=EUC-KR" %>
<jsp:useBean id="cart" class="com.boolpae.jsp.Cart" scope="session" />
<HTML>
<HEAD>
<TITLE> add Cart </TITLE>
</HEAD>
<BODY>
<center>
<h1>장바구니 Example</h1>
<% String requestIsbn = request.getParameter("isbn");
   String requestCount = request.getParameter("count");
```

```
    if(requestIsbn != null){
        if(requestCount == null){
            cart.addBook(requestIsbn);
        }else{
            int parseCount;
            try{
                parseCount = Integer.parseInt(requestCount);
            }catch(NumberFormatException e){
                parseCount = 1;
            }
            cart.setCount(requestIsbn, parseCount);
        }
    }

    Vector ordered = cart.getOrdered();

    if(ordered.size() ==  0){
        out.print("<h1> 장바구니가 비어 있습니다 </h1>");
    }else{
        out.print("<table  border=1><tr><td>ISBN</td><td>제목</td><td>단가
</td><td>수량</td><td>총 금액</td>");
        for(int i=0;i<ordered.size();i++){
            com.boolpae.jsp.Order order =
                                (com.boolpae.jsp.Order)ordered.elementAt(i);
            int count = order.getCount();

            com.boolpae.jsp.Book book = order.getBook();
            String isbn = order.getBook().getIsbn();
            String title = order.getBook().getTitle();
            int price = book.getPrice();
            int totalPrice = order.getTotalPrice();
%>
<tr>
    <td><%=isbn%></td>
    <td><%=title%></td>
    <td><%=price%></td>
    <td><form method="get" action="CartPage.jsp">
    <input type="hidden" name="isbn" value="<%=isbn%>">
    <input type="text" size="5" maxlength="2"
                        name="count" value="<%=count%>">
    <input type="submit" value="수정">
    </form>
    </td>
    <td><font color="red"><%=totalPrice%></font></td>
</tr>
<% } //end of for %>
</table>
주의 : 수량에 문자를 입력시 1개로 셋팅됩니다.
<% } //end of if %>
</BODY>
</HTML>
```

눈여겨 보아야 할 부분은 <jsp:useBean> 태그이다. 사실 Session 을 소개하는 곳에서 Session을 사용한 곳은 없었다.
<jsp:useBean id="cart" class="com.boolpae.jsp.Cart" scope="session"/> 이라고 되어 있지 않은가? 이 것이 의미하는 것은 이 Cart 객체를 세션에 담는 것과 똑같다. 따라서
<% com.boolpae.jsp.Cart cart = new com.boolpae.jsp.Cart();
session.setAttribute("cart", cart); %>
와 같이 해주는 것과 동일하다. 이랬거나 저랬거나 여러분이 편하실대로 쓰면 된다. 사용자와 관련된 정보들은 이런 식으로 하나의 클래스를 작성하여 이 클래스에 몰아 넣고 이 객체를 세션에 담아두는 것이 여러면에서 편리하고 유용하며 깔끔하다. 로직을 처리하는 부분을 좀 살펴보도록 하자. 아래의 코드에 들어있는 것이 로직인데…

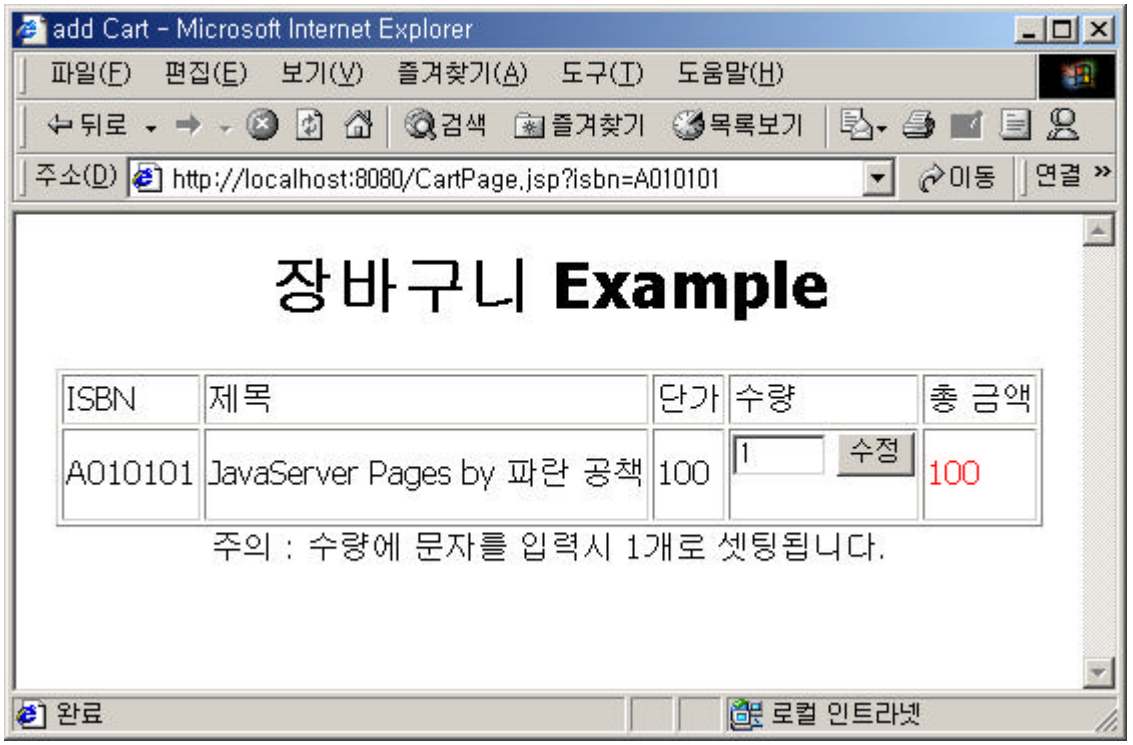
```
String requestIsbn = request.getParameter("isbn");
String requestCount = request.getParameter("count");

if(requestIsbn != null){
    if(requestCount == null){
        cart.addItem(requestIsbn);
    }else{
        int parseCount;
        try{
            parseCount = Integer.parseInt(requestCount);
        }catch(NumberFormatException e){
            parseCount = 1;
        }
        cart.setCount(requestIsbn, parseCount);
    }
}
```

isbn 과 count 두개의 파라미터를 받는다. Books.jsp 페이지에서는 ISBN 밖에 넘기지 않는다. 그럴 경우에는 cart.addBook 메쏘드를 사용하여 장바구니에 한 개를 담는다. 변수 requestCount 가 null 이란 말은 Books.jsp 로부터 요청이 온 것이기 때문이다. 수량의 조절이 있다면 count 가 null 로 넘어오지 않을 것이다. null로 넘어오지 않으면 cart.setCount 메쏘드가 수행될 것이다. 이때 아규먼트로 requestIsbn 과 parseCount 가 넘어가는데 setCount의 두번째 아규먼트가 int형이기 때문에 형변환을 해주어야 한다.
request 요청으로 넘어오는 파라미터의 값들은 모두 String 이며 이 값을 int 형으로 바꾸어 주기 위해서 우리는 Integer.parseInt 메쏘드를 사용하였다. 만약 적절한 int 이외의 값(문자 또는 int의 범위를 벗어나는 값)으로 형변환을 하려하면 NumberFormatException 일 발생할 것이다. 이럴 경우에는 무조건 Count를 1로 설정해 버렸다. 이런 결과의 parseCount 변수값을 가지고 setCount 메쏘드를 호출하여 수량 조절을 하게 되는 것이다.

주문량이 0이면 "장바구니가 비워져 있습니다." 라는 메시지가 나오도록 했으며 주문서에 적혀있는 책들의 리스트를 보여준다. 보여주면서 <form> 태그 내에서 text 필드를 주어서

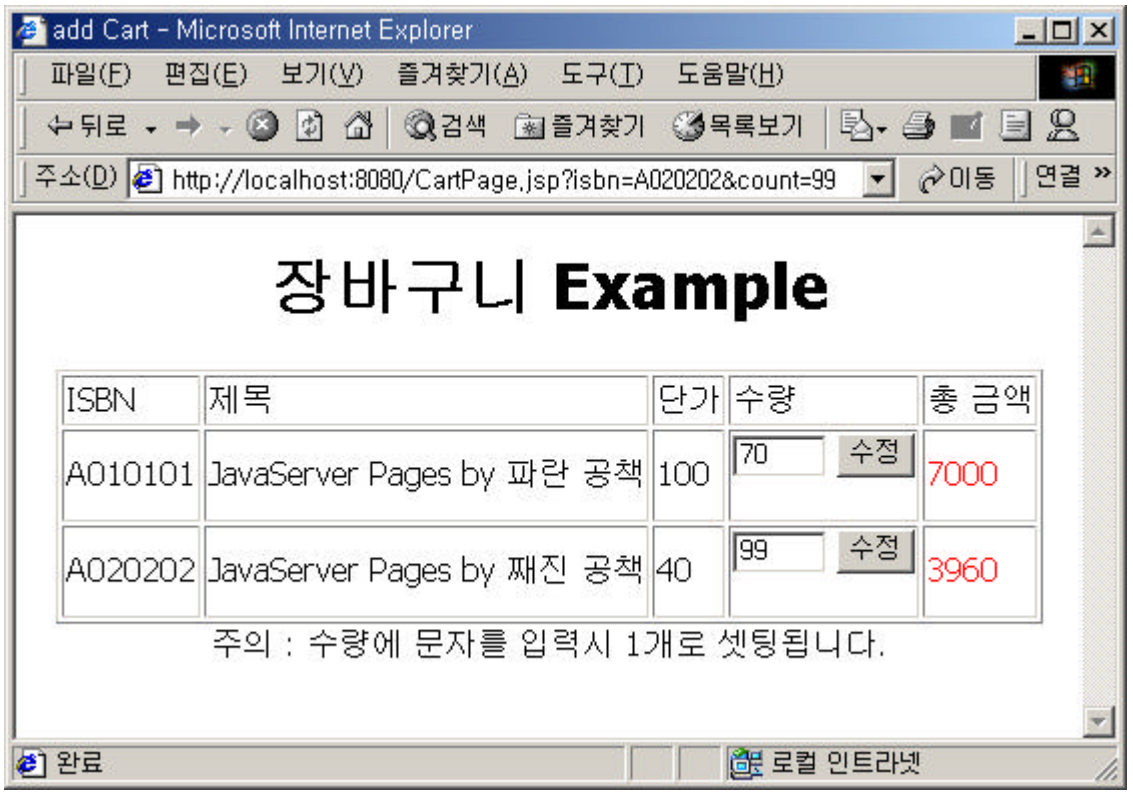
수량을 조절할 수 있도록 하였고 역시 이 폼의 처리는 자기자신이다. 이 form에서 수정하려는 수량(count)와 ISBN을 모두 넘기게 되는 것이다. 그 외 화면에 보여줄 정보를 Book 클래스와 Order 클래스의 메쏘드를 이용해서 가져왔다. 파란공책의 JavaServer Pages 를 장바구니에 담은 화면은 아래와 같다.

[CartPage.jsp - 1]



다른 책이라고 해봐야 째진 공책이 쓴 책밖에 없지만, 그 책도 사고 수량도 여러권 조절한 화면을 아래와 같을 것이다. 째진 공책의 가격은 40원이어서 99권을 구입해서 파란공책이 쓴 책보다 훨 싸다는 것을 알 수 있다. 하지만 이 책은 어디에서도 팔지 않는다… 쩝쩝

[CartPage.jsp –2]



이번 장도 막바지를 향해서 다다르고 있다. 한가지 예제를 더 보도록 하자. 이번 예제는 흔히 우리가 현재 접속하고 있는 사람이 누구인지 리스트화 해서 보여주는데 쓸 수 있을 것이다.

List 6.13 UserList.java

```
package com.boolpae.jsp;

import java.util.*;

public class UserList{

  private static Vector userVector = new Vector();

  public static void addUser(String userName){
     String temp;
     for(int i=0;i<userVector.size();i++){
       temp = (String)userVector.elementAt(i);
       if(userName.equals(temp)){
          return;
       }
     }
     userVector.addElement(userName);
  }

  public  static void removeUser(String user){
```

```
    String temp;
    for(int i=0;i<userVector.size();i++){
        temp = (String)userVector.elementAt(i);
      if(user.equals(temp)){
        userVector.removeElementAt(i);
        return;
      }
    }
  }

  public  Vector getUser(){
    return userVector;
  }
}
```

List 6.13은 현재 접속 중인 사람들의 목록을 가지고 있다. 사용자를 Vector에 담고, 빼는 메쏘드를 가지고 있으며 그 Vector를 반환해주는 메쏘드가 있다. 정작 중요한 부분은 다음에 보여줄 List 6.14이다.

List 6.14 User.java

```
package com.boolpae.jsp;

import javax.servlet.http.*;

public class User implements HttpSessionBindingListener{

  private String userName;

  public void setUserName(String userName){
   this.userName = userName;
   UserList.addUser(userName);
  }

  public void valueBound(HttpSessionBindingEvent event){

  //   UserListBean.addUser(userName);
  }

  public void valueUnbound(HttpSessionBindingEvent event){

    UserList.removeUser(userName);
  }
}
```

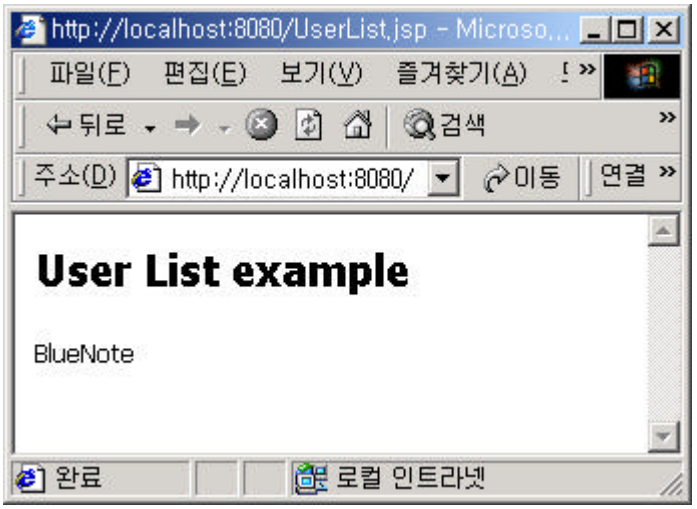
이 클래스는 javax.servlet.http.HttpSessionBindingListener 인터페이스를 구현하고 있다. 이 인터페이스에는 valueBound 와 valueUnbound 메쏘드를 정의하고 있다. 이 인터페이스는 사용자가 세션에 접속되는 순간과 세션이 끊어지는 순간의 이벤트를 탐지해서 이 두가지 메쏘드를 호출한다. 우리는 세션에 접속될때는 임의의 값을 Vector에 담을 것이고 세션

이 끊어지는 순간에는 자동적으로 그 사용자를 Vector에서 제거하게 될 것이다. valueUnbound 메쏘드의 내부에서 그 일을 하고 있는 것이다. 우리가 테스트해 볼 jsp페이지는 다음과 같다.

List 6.15 UserList.jsp

```
<HTML>
<BODY>
<H1>User List example</H1>
<jsp:useBean id="userList" class="com.boolpae.jsp.UserList" scope="application"
/>
<jsp:useBean id="user" class="com.boolpae.jsp.User" scope="session" />
  <jsp:setProperty name="user" property="userName" value="BlueNote" />

<% java.util.Vector userVector = userList.getUser();
for(int i=0;i<userVector.size();i++){
  out.print(userVector.elementAt(i));
}
%>
</BODY>
</HTML>
```

결과 화면은 아래와 같다.



테스트를 위해서 List 6.15 UserList.jsp 페이지의 <jsp:setProperty>의 value 속성의 값을 적당히 여러분의 별명으로 바꾸고서 저장해보자.

그리고 다시 접속해보자. 그렇다면 BlueNote와 여러분의 별명이 화면에 보일 것이다. 이제 한번만더 value 속성을 여러분 마음대로 변경하자. 그리고 1분을 기다린 다음 refresh 해보면 여러분의 별명은 사라지고 없을 것이다. 그 이전에 세션 타임아웃 시간을 1분으로 지정해야 할 것이다. 여러분이 테스트하고 있는 jsp페이지가 있는 웹어플리케이션의 web.xml 파일의 세션 타임아웃 시간을 1분으로 정하자. List 6.16은 web.xml 파일을 보

여주고 있다.

List 6.16 web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
    "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">

<web-app>
<session-config>
        <session-timeout>
            1
        </session-timeout>
</session-config>
</web-app>
```

참고로 웹어플리케이션의 web.xml 파일을 수정하라는 말은 WEB-INF 아래의 web.xml 파일을 수정하라는 말이다. JSP에서 하나의 웹 어플리케이션은 루트 디렉토리가 있고 그 아래에 WEB-INF, META-INF를 가지고 있다. 따라서 WEB-INF 폴더 아래의 web.xml 파일에 설정되어 있는 사항들은 이 웹 어플리케이션 전체에 영향을 미치는 것이다.
위 예제는 jsp:setProperty 의 value 값을 여러분의 용도에 따라 동적으로 수정하고 어딘가 어설퍼 보이는 부분에 약간의 손을 본다면 여러 용도로 사용할 수 있을 것이다.