

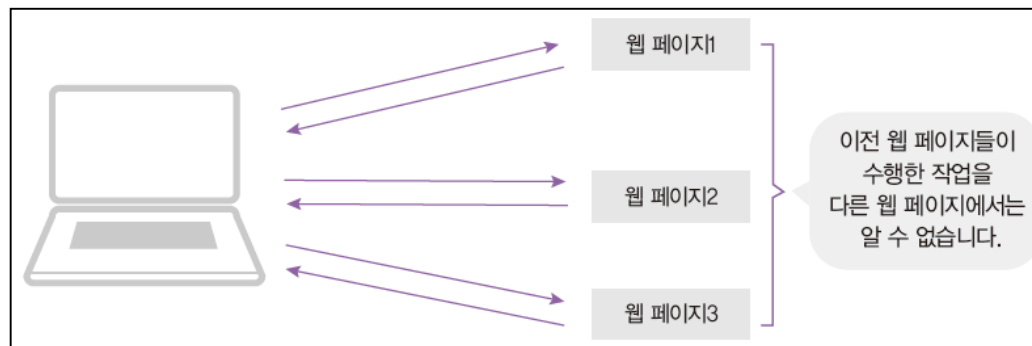
Chapter 07

쿠키와 세션

HTTP 프로토콜의 특성

■ HTTP 프로토콜의 특성

- HTTP 프로토콜은 비연결형이며 무상태 프로토콜임
 - ▶ 상태 정보를 저장하지 않음
- 로그인을 사용하는 사이트의 경우 각 페이지 사이의 상태나 정보가 공유되어야 함
 - ▶ 각 페이지는 이동할 때마다 로그인 된 상태 정보가 공유되어야 함
 - ▶ HTTP 프로토콜은 상태 정보를 저장하지 않으므로 페이지 사이에 정보가 공유되지 않음
 - 예를 들어, HTTP 프로토콜을 사용한다면 쇼핑몰 사이트에서 장바구니를 구현하기 어려움
- 상태 정보가 공유될 수 있도록 하기 위해서는 세션 트래킹을 구현해야 함
 - ▶ 세션 트래킹을 구현하는 대표적인 방법은 쿠키와 세션임



HTTP 프로토콜의 특성

- 웹 페이지 사이에서 정보를 공유하기 위해 세션 트래킹을 구현하는 방법

방법	설명
<hidden> 태그	HTML의 <hidden> 태그를 사용해 페이지 간 정보를 공유하는 방법
URL Rewriting	GET 방식으로 URL 뒤에 정보를 추가해 다른 페이지로 전송하는 방법
쿠키 (cookies)	클라이언트에 쿠키 정보를 저장한 후 페이지들이 공유하는 방법
세션 (session)	서버 메모리에 정보를 저장한 후 페이지들이 정보를 공유하는 방법

쿠키 [Cookies]

쿠키의 개요

■ 쿠키 (cookies)

• 변수와 값으로 구성되는 작은 파일

- ▶ 쿠키의 변수와 값은 웹 서버에 의해 생성
- ▶ 웹 서버에 의해 클라이언트 컴퓨터의 쿠키 저장 장소에 저장
- ▶ 4KB 이하의 크기
- ▶ 쿠키는 도메인을 기준으로 생성되며, 생성된 쿠키는 다른 도메인과 공유되지 않음
 - 하나의 도메인 당 20개 정도의 쿠키를 생성할 수 있음
- ▶ 클라이언트의 컴퓨터에는 최대 300개의 쿠키가 저장 가능
 - 300개 이상의 쿠키가 생성된다면 생성된 순서대로 먼저 생성된 쿠키부터 소멸

• 쿠키의 종류

속성	Persistence Cookies	Session Cookies
생성 위치	파일로 생성	브라우저 메모리에 생성
종료 시기	쿠키를 삭제하거나 쿠키 설정 값이 종료된 경우	브라우저를 종료한 경우
용 도	로그인 유무 또는 팝업 창을 제한할 때	사이트 접속 시 Session 인증 정보를 유지할 때

쿠키의 개요

■ 쿠키의 생성과 동작

• 쿠키 생성 단계

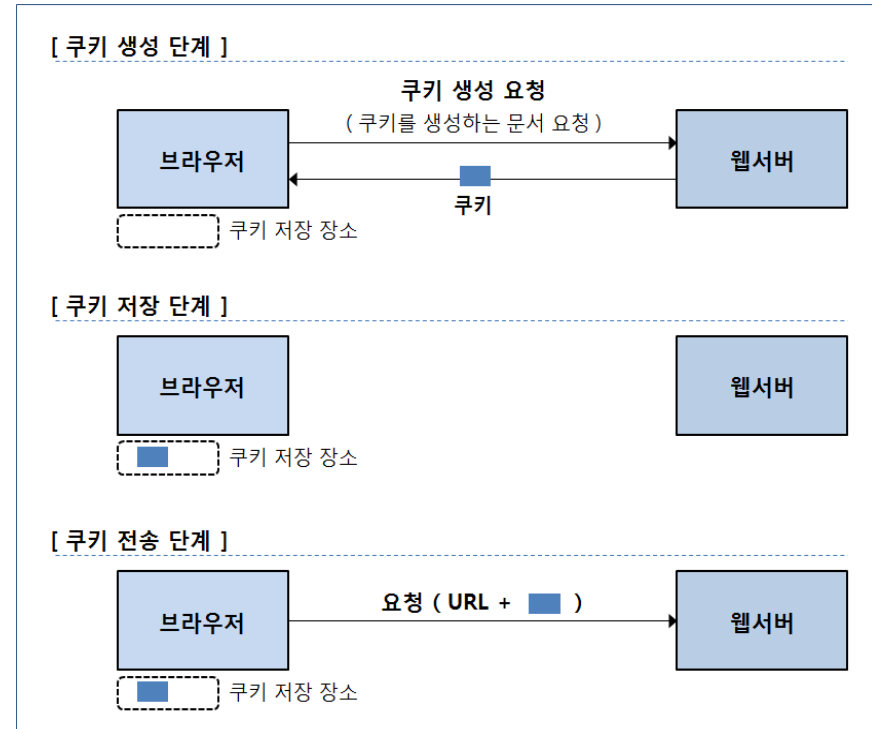
- ▶ 쿠키의 생성은 쿠키를 생성하도록 프로그래밍 되어 있는 JSP 문서에 의해 이루어짐
- ▶ 클라이언트가 웹 서버에 쿠키의 생성을 요청하면 웹 서버는 쿠키를 생성
- ▶ 생성된 쿠키는 웹 서버에 의해 클라이언트로 전달

• 쿠키 저장 단계

- ▶ 전달된 쿠키는 클라이언트 컴퓨터의 하드디스크에 있는 [쿠키 저장 장소]에 저장

• 쿠키의 전송 단계

- ▶ 저장된 쿠키는 브라우저가 웹 서버에 요청을 할 때마다 URL과 함께 전송
- ▶ 웹 서버는 전송된 쿠키로부터 필요한 정보를 추출한 다음 쿠키를 필요로 하는 작업들을 수행



쿠키의 개요

■ 쿠키의 구조와 유효 시간

• 쿠키의 구성

▶ 쿠키는 이름과 값(name & value)으로 구성됨

- 한 사이트는 하나 이상의 쿠키를 생성할 수 있으므로 쿠키의 이름은 유일 해야 함
- 쿠키의 이름은 쿠키를 구분하는 식별자로 사용되며 생성된 쿠키는 임의의 값을 가짐
- 쿠키의 이름과 값은 가장 중요한 요소로 쿠키를 생성할 때 지정

• 유효 시간(validate time)

▶ 유효시간이란 쿠키가 유지되는 시간을 의미

- 쿠키를 생성할 때 유효 시간을 지정할 수 있음
- 필요에 따라 쿠키를 강제로 제거할 수도 있음

▶ 쿠키는 유효 시간이 끝나지 않거나 강제로 제거될 때까지 계속 유지됨

- 브라우저를 종료해도 쿠키는 소멸되지 않고 계속 존재할 수도 있음
- 브라우저를 닫을 때 자동으로 종료시킬 수도 있음

쿠키의 개요

■ 쿠키의 적용 범위

- 도메인(domain)
 - 쿠키가 사용될 수 있는 도메인을 의미
 - 쿠키에 특정 도메인을 지정하면 쿠키는 그 도메인 내에서만 유효
- 경로(path)
 - 특정 도메인 내에서 쿠키가 적용될 수 있는 경로
 - 쿠키는 특정 도메인 전체에서 유효하도록 지정 가능
 - 도메인 내의 일부 영역에서만 유효하도록 지정 가능

쿠키의 생성과 참조

■ 쿠키의 생성

```
Cookie 인스턴스이름 = new Cookie("쿠키이름", "쿠키값");  
response.addCookie(인스턴스이름);
```

- java.servlet.Http.Cookie 클래스와 response 객체의 addCookie()라는 메서드를 사용
 - ▶ 생성된 쿠키는 response 객체를 통해 클라이언트가 요구한 데이터와 함께 쿠키가 전송됨

[쿠키 생성의 예] - 'seoul'이라는 값을 가진 쿠키 'korea'를 생성

```
Cookie cookie1 = new Cookie("korea", "seoul");  
response.addCookie(cookie1)
```

쿠키의 생성과 참조

■ 쿠키의 참조

메서드	설명
getName()	지정한 쿠키의 이름을 반환
getValue()	지정한 쿠키의 값을 반환

쿠키의 생성과 참조

■ 쿠키의 생성과 참조 시 유의사항 (쿠키 값이 한글인 경우의 처리)

- 쿠키의 값이 한글인 경우 쿠키의 생성

- ▶ java.net.URLEncoder 클래스의 encode() 메서드를 사용해야 함

```
Cookie mycookie1 = new Cookie("name","홍길동");
```

//오류 발생

```
Cookie mycookie1 = new Cookie("name",URLEncoder.encode("홍길동","utf-8"));
```

//정상 출력

- 쿠키의 값이 한글인 경우 쿠키 값의 참조

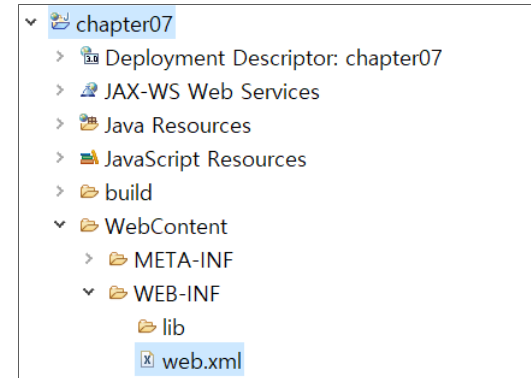
- ▶ java.net.URLDecoder 클래스의 decode() 메소드를 사용해야 함

```
<%=URLDecoder.decode(mycookie1.getValue(), "utf-8") %>
```

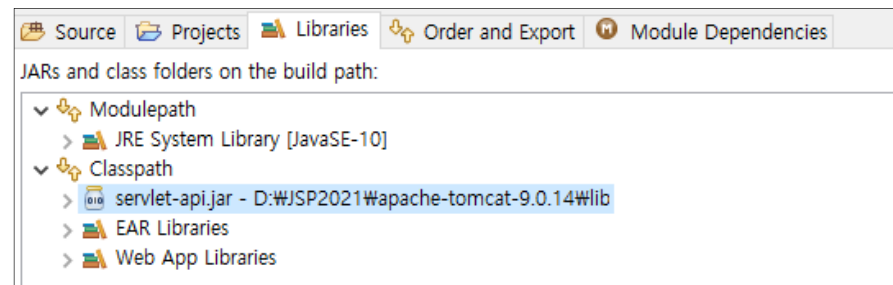
쿠키의 생성과 참조

■ 프로젝트 생성

- 프로젝트 이름 : chapter07
 - ▶ web.xml 파일은 반드시 생성해야 함

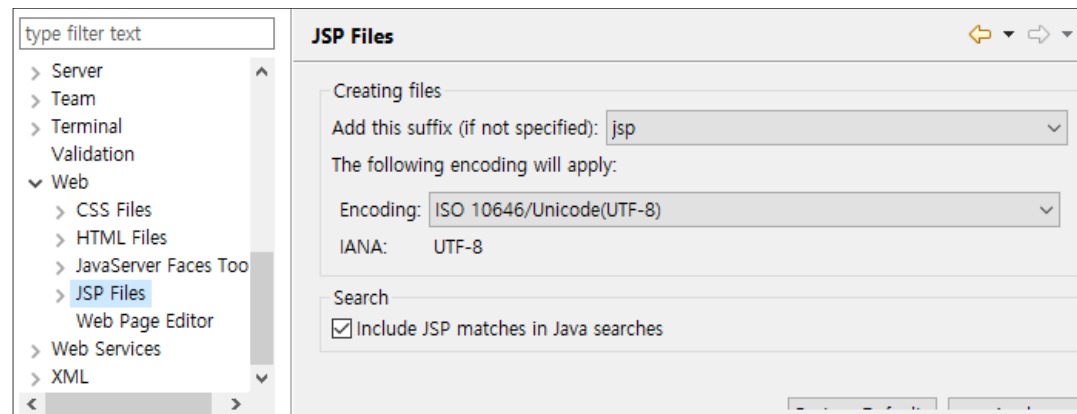


- servlet-api 라이브러리 추가

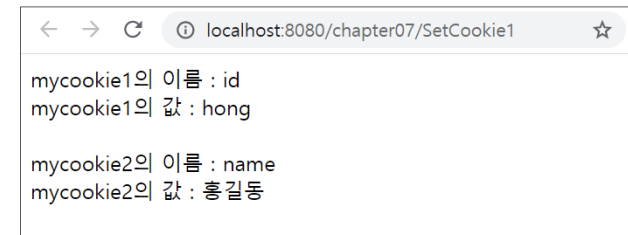


- JSP 문서 문자 집합 지정

- ▶ UTF-8로 지정



```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%@ page import="java.net.URLEncoder" %>
4 <%@ page import="java.net.URLDecoder" %>
5
6 <%
7
8   // 쿠키 생성
9   Cookie mycookie1 = new Cookie("id","hong");
10  response.addCookie(mycookie1);
11
12  Cookie mycookie2 = new Cookie("name",URLEncoder.encode("홍길동", "UTF-8"));
13  response.addCookie(mycookie2);
14
15  // 쿠키 정보 추출
16  String ck1_name = mycookie1.getName();
17  String ck1_value = mycookie1.getValue();
18
19  String ck2_name = mycookie2.getName();
20  String ck2_value = URLDecoder.decode(mycookie2.getValue(),"UTF-8") ;
21
22  // 쿠키 정보 출력
23  out.println("mycookie1의 이름 : " + ck1_name + "<br>");
24  out.println("mycookie1의 값 : " + ck1_value + "<br><br>");
25
26  out.println("mycookie2의 이름 : " + ck2_name + "<br>");
27  out.println("mycookie2의 값 : " + ck2_value);
28 %>
```



쿠키의 생성과 참조

■ 클라이언트에서 서버로의 쿠키 전달

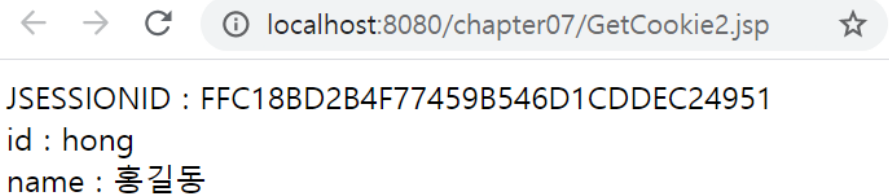
- 클라이언트 컴퓨터에 하나 이상의 쿠키가 저장되어 있을 경우 모든 쿠키들이 함께 전달됨
 - 서버는 클라이언트로부터 전달된 모든 쿠키를 추출하거나 필요한 쿠키 만을 추출해 사용
 - 서버는 전달된 쿠키들을 추출할 수 있는 방법이 필요
- 전달된 모든 쿠키는 `getCookies()` 메서드를 통해 추출할 수 있음

```
Cookie[] 배열 인스턴스 = request.getCookies();
```

- `getCookies()` 메서드는 전달된 모든 쿠키를 내용으로 하는 `cookie` 배열을 반환
- 전달된 쿠키가 존재하지 않는다면 `null`을 반환

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%@ page import="java.net.URLEncoder" %>
4
5 <%
6
7   // 쿠키 생성
8   Cookie mycookie1 = new Cookie("id","hong");
9   response.addCookie(mycookie1);
10
11   Cookie mycookie2 = new Cookie("name",URLEncoder.encode("홍길동", "UTF-8"));
12   response.addCookie(mycookie2);
13
14   // 쿠키 정보 추출 페이지 호출
15   response.sendRedirect("GetCookie2.jsp");
16
17 %>
```

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%@ page import="java.net.URLDecoder" %>
4 <%
5
6   // 모든 쿠키를 쿠키 배열 cookies에 저장
7   Cookie[] cookies = request.getCookies();
8
9   // cookies 배열의 모든 쿠키를 출력
10  if( (cookies != null) && (cookies.length >0) ) {
11      for( int i=0; i<cookies.length; i++) {
12          out.println( cookies[i].getName() + " : " );
13          out.println( URLDecoder.decode(cookies[i].getValue(), "utf-8") + "<BR>" );
14      }
15  } else {
16      out.println("No Cookies!");
17  }
18
19 %>
```



← → ↺ ⓘ localhost:8080/chapter07/GetCookie2.jsp ☆

JSESSIONID : FFC18BD2B4F77459B546D1CDDEC24951
id : hong
name : 홍길동

쿠키의 변경

■ 쿠키의 값을 변경하는 방법

- 쿠키를 변경하는 별도의 메서드는 없음
- `addCookie()` 메서드를 사용해 새로운 값으로 재지정하여 변경
 - 기존 쿠키 값은 새로운 쿠키 값으로 overwrite됨

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2     pageEncoding="UTF-8"%>
3 <%@ page import="java.net.URLEncoder" %>
4
5 <%
6
7     // 쿠키 생성
8     Cookie mycookie1 = new Cookie("id","hong");
9     response.addCookie(mycookie1);
10
11     Cookie mycookie2 = new Cookie("name",URLEncoder.encode("홍길동", "UTF-8"));
12     response.addCookie(mycookie2);
13
14     // 쿠키 정보 추출 페이지 호출
15     response.sendRedirect("MdfyCookie3.jsp");
16
17 %>
```

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%@ page import="java.net.URLEncoder" %>
4
5 <%
6
7   Cookie[] cookies = request.getCookies();
8
9   if( (cookies != null) && (cookies.length >0) ) {
10
11     // 이전의 홍길동을 찾아 심청으로 변경
12     for( int i=0; i<cookies.length; i++) {
13       if( cookies[i].getName().equals("name")) {
14         Cookie mycookie2 = new Cookie("name",URLEncoder.encode("심청", "utf-8"));
15         response.addCookie(mycookie2);
16       }
17     }
18
19   }
20
21   // 쿠키 정보 추출 페이지 호출 (이전에 작성한 GetCookie2.java 호출 )
22   response.sendRedirect("GetCookie2.jsp");
23
24 %>
```

← → ↻ ⓘ localhost:8080/chapter07/GetCookie2.jsp ☆

JSESSIONID : FFC18BD2B4F77459B546D1CDDEC24951
id : hong
name : 심청

쿠키의 유효시간 지정

■ 쿠키의 유효 시간 지정

쿠키. SetMaxAge(유효시간)

- ▶ 쿠키의 유효 시간을 초단위로 지정
- ▶ addCookie() 메서드 실행 전에 지정해야 함

• Persistence Cookie 생성

- ▶ 지정된 시간 동안 유지
- ▶ 양의 정수(초)로 지정

```
Cookie mycookie1 = new Cookie("id","hong");  
mycookie1.setMaxAge(20);           // 20초로 지정  
response.addCookie(mycookie1);
```

• Session Cookie의 생성

- ▶ 브라우저가 종료될 때까지만 유지
- ▶ SetMaxAge()를 지정하지 않거나, 인자를 -1로 지정

```
Cookie mycookie1 = new Cookie("id","hong");  
mycookie1.setMaxAge(-1);  
response.addCookie(mycookie1);
```

• SetMaxAge() 메서드의 인자를 0으로 지정할 경우

- ▶ 지정된 쿠키는 바로 제거됨 (로그아웃 기능을 구현할 때 사용될 수 있음)

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%@ page import="java.net.URLEncoder" %>
4
5 <%
6
7   // id 쿠키 생성
8   Cookie mycookie1 = new Cookie("id","hong");
9
10  // 유효시간을 5초로 지정
11  mycookie1.setMaxAge(5);
12  response.addCookie(mycookie1);
13
14  // name 쿠키 생성
15  Cookie mycookie2 = new Cookie("name",URLEncoder.encode("홍길동", "UTF-8"));
16  response.addCookie(mycookie2);
17
18  // 쿠키 정보 추출 페이지 호출
19  response.sendRedirect("GetCookie2.jsp");
20
21 %>
```

← → ↻ ⓘ localhost:8080/chapter07/GetCookie2.jsp ☆

JSESSIONID : D58469FFAD50A5315609FCE97890CFCD
id : hong
name : 홍길동

↓ 5초 후 reload 수행

← → ↻ ⓘ localhost:8080/chapter07/GetCookie2.jsp ☆

JSESSIONID : D58469FFAD50A5315609FCE97890CFCD
name : 홍길동

쿠키 제거

■ 쿠키 제거

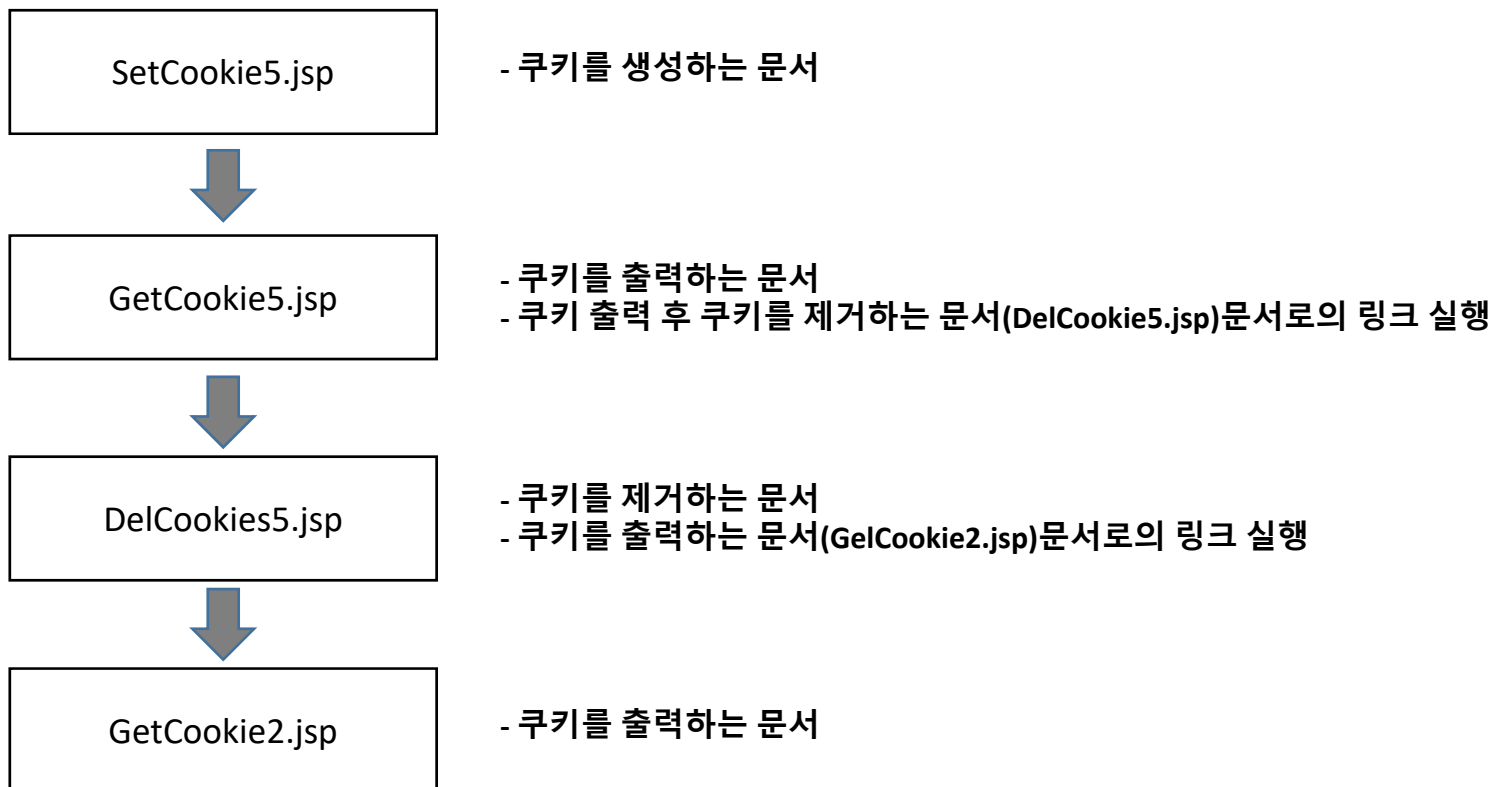
- 쿠키를 제거하는 별도의 메서드는 없음
- `setMaxAge()` 메서드의 인자를 0으로 재지정

```
<%  
    Cookie[] cookies = request.getCookies();  
  
    if( (cookies != null) && (cookies.length >0) ) {  
  
        for( int i=0; i<cookies.length; i++ ) {  
            cookies[i].setMaxAge(0);  
            response.addCookie(cookies[i]);  
        }  
    }  
%>
```

쿠키 제거

■ 쿠키 제거 실습

- 동일한 브라우저에서 다음의 순서대로 실행



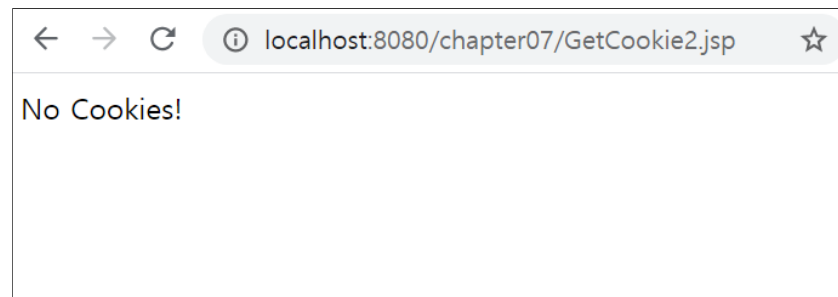
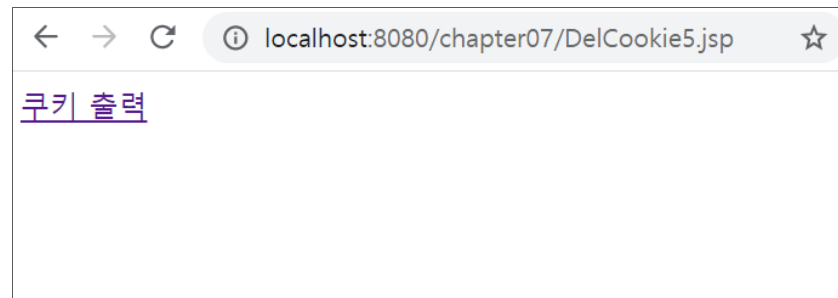
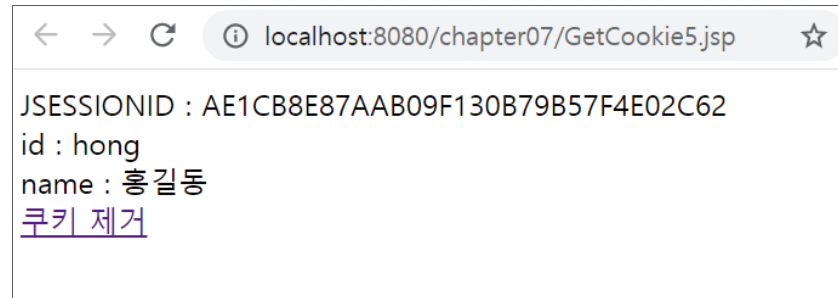
```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%@ page import="java.net.URLEncoder" %>
4
5 <%
6
7   // 쿠키 생성
8   Cookie mycookie1 = new Cookie("id","hong");
9   response.addCookie(mycookie1);
10
11   Cookie mycookie2 = new Cookie("name",URLEncoder.encode("홍길동", "UTF-8"));
12   response.addCookie(mycookie2);
13
14   // 쿠키 정보 추출 페이지 호출
15   response.sendRedirect("GetCookie5.jsp");
16
17 %>
```



```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2     pageEncoding="UTF-8"%>
3 <%@ page import="java.net.URLDecoder" %>
4
5 <%
6
7     Cookie[] cookies = request.getCookies();
8
9     if( (cookies != null) && (cookies.length > 0) ) {
10         for( int i=0; i<cookies.length; i++) {
11             out.println( cookies[i].getName() + " : " );
12             out.println( URLDecoder.decode(cookies[i].getValue(), "utf-8") + "<BR>" );
13         }
14     } else {
15         out.println("No Cookies!");
16     }
17
18     out.println("<a href='DelCookie5.jsp'>쿠키 제거</a>");
19
20 %>
```

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3
4 <%
5
6   Cookie[] cookies = request.getCookies();
7
8   if( (cookies != null) && (cookies.length >0) ) {
9       for( int i=0; i<cookies.length; i++) {
10           cookies[i].setMaxAge(0);
11           response.addCookie(cookies[i]);
12       }
13   } else {
14       out.println("No Cookies!");
15   }
16
17   out.println("<a href='GetCookie2.jsp'>쿠키 출력</a>");
18
19 %>
```

쿠키 제거



쿠키 도메인과 경로

■ 쿠키 도메인 지정

- 원칙적으로 쿠키는 그 쿠키를 생성한 서버로만 전달
 - ▶ 다른 서버로 쿠키가 전송된다면 보안에 큰 문제를 발생시킬 수 있음
- 생성되어 있는 쿠키를 같은 도메인을 가지는 서버에서 공유해야 하는 경우가 있음
 - ▶ 예를 들어, `www.abc.co.kr`, `mail.abc.co.kr`와 같이 같은 도메인을 가지는 서버에 쿠키를 공유하고자 할 경우

```
mycookie.setDoamin(".abc.co.kr ");
```

- dot로 시작해야 함에 유의

- 서버의 이름을 정확히 지정한 경우, 해당 서버에만 쿠키가 유효함

```
mycookie.setDoamin("mail.abc.co.kr");
```

- `mail.abc.co.kr` 서버에서만 유효

쿠키 도메인과 경로

■ 쿠키 경로 지정

- 특정 도메인 내에서도 전달할 경로를 지정할 수 있음
 - ▶ 예를 들어, 쿠키를 특정 폴더의 하위 폴더에서만 사용할 수 있도록 지정할 수 있음
- 쿠키 경로 지정은 `setPath()` 메서드를 사용
- [경로 지정의 예] mycookie2라는 쿠키를 생성한 `setcookie.jsp` 문서의 경로가 다음과 같다고 가정

```
mycookie.setDoamin("www.abc.co.kr ");
```

- ▶ `mycookie2.setPath("/internal/part1");`
 - mycookie2가 `www.abc.co.kr` 서버에 있는 `/internal/part1` 폴더와 그 하위 폴더에서만 사용 가능
- ▶ `mycookie2.setPath("/")`
 - mycookie2가 `www.abc.co.kr` 서버에 있는 전체 폴더에서 사용 가능
- 일반적으로 도메인 전체에서 사용하도록 지정

세션 [session]

세션의 개요

■ 세션(session)

- 세션은 쿠키와 유사하게 클라이언트와 서버를 연결시켜주는 효과를 제공
 - ▶ 쿠키와 같이 세션도 이름과 값으로 구성된 세션 속성을 유지하여 사용 가능
- 세션은 브라우저를 통해 JSP 문서를 요청했을 때 응답이 이루어지는 순간 생성
 - ▶ 로그아웃 처리와 같이 명시적으로 세션을 종료시킬 수 있음
 - ▶ 타임아웃을 지정하여 일정 시간이 지나면 자동으로 종료되게 할 수도 있음
 - ▶ 브라우저가 종료되면 세션은 자동으로 종료됨
- 세션과 쿠키의 차이점
 - ▶ 쿠키는 모든 정보를 클라이언트의 컴퓨터에 보관
 - ▶ 세션을 사용해 생성된 모든 정보들은 클라이언트의 컴퓨터가 아닌 서버에 저장
 - 사용자를 구분하기 위한 세션 ID는 쿠키 형태로 클라이언트 컴퓨터에 저장됨

세션의 개요

■ 세션ID

- **실제 호스팅이 이루어지면 서버는 수많은 사용자의 요청을 처리하게 됨**
 - ▶ 특정 사용자가 브라우저를 통해 처음으로 문서를 요청하는 순간 하나의 세션이 시작
 - 사용자가 브라우저를 닫는 순간 세션은 종료
 - ▶ 접속한 사용자의 수만큼 즉, 해당 어플리케이션에 접속한 브라우저의 수만큼 세션이 생성됨
 - 서버는 많은 사용자를 구분할 수 있는 방법이 필요
- **서버가 클라이언트를 구분하는 방법**
 - ▶ 세션이 시작되면 서버는 다른 사용자의 세션과 구분할 수 있는 쿠키를 만들어 해당 사용자의 컴퓨터에 전송해 저장
 - ▶ 이 쿠키의 이름이 바로 세션ID 임
 - ▶ 클라이언트가 서버에 요청할 때마다 URL과 세션 ID가 함께 전송
 - 서버는 세션 ID를 통해 사용자를 구분
 - 세션 ID는 유일한 값이어야 하므로 사용자마다 다른 값을 가지게 됨
- **사용자가 브라우저를 종료하면 세션 역시 종료되고 JSESSIONID 역시 제거됨**
 - ▶ 사용자가 다시 브라우저를 열어 문서를 요청하면 기존의 세션이 아닌 다른 세션이 생성
 - 새로운 JSESSIONID가 클라이언트 컴퓨터에 전송되어 저장

세션의 개요

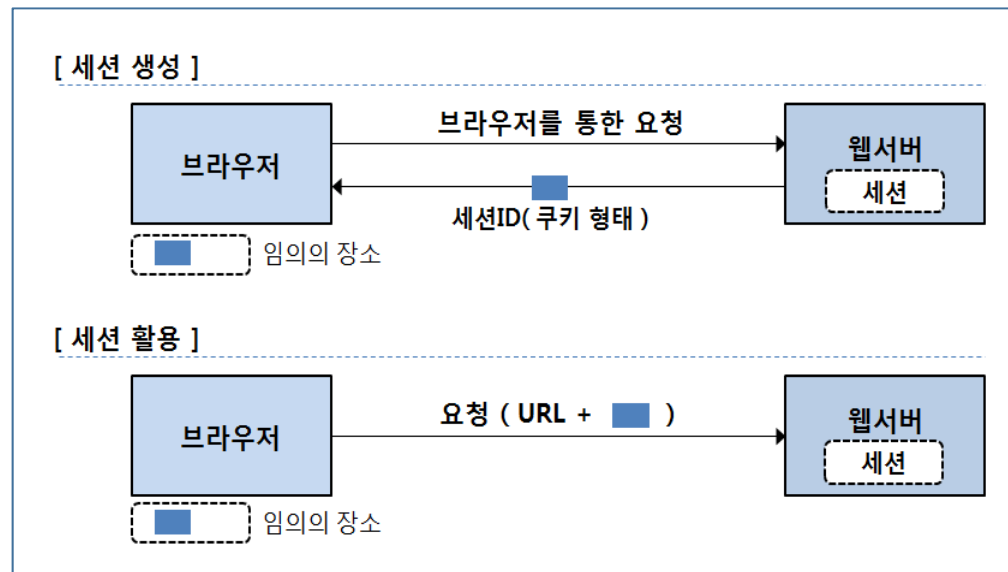
■ 세션의 생성 과정

• 세션 생성

- ▶ 브라우저를 통한 요청에 대한 응답이 발생되면 서버에 세션이 생성
- ▶ 서버는 쿠키 형태의 유일한 값인 세션ID를 생성해 클라이언트로 전송
 - 클라이언트 컴퓨터의 임의의 공간에 저장 (JSESSIONID가 세션ID임)

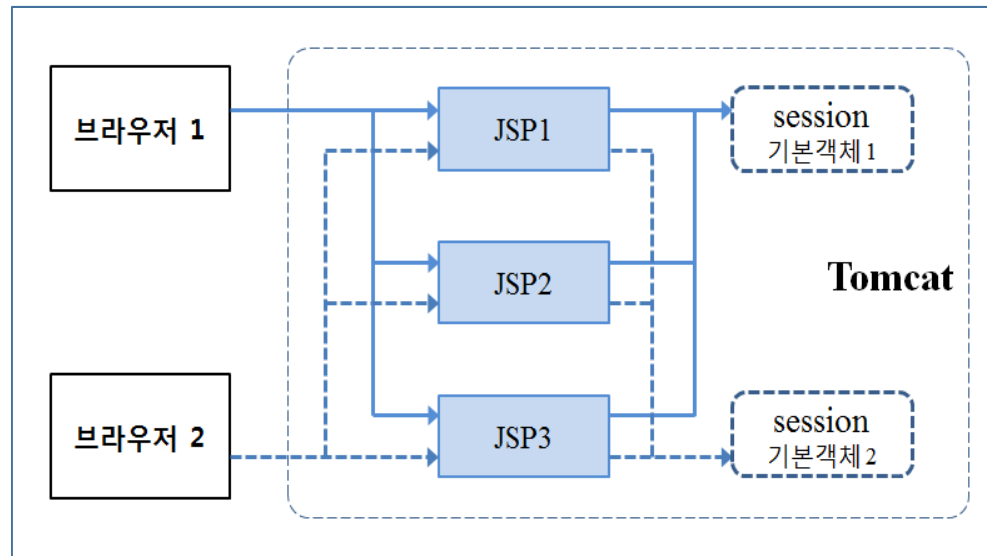
• 세션 활용

- ▶ 세션이 생성된 후 클라이언트에서 서버로의 요청이 발생할 경우 세션ID가 함께 전송
- ▶ 이를 통해 서버는 사용자를 구분



세션의 개요

■ session 객체와 브라우저와의 관계



- ▶ [브라우저1]을 통해 JSP1을 요청하고 그 응답이 이루어지면 세션이 생성
 - [session 객체1]이 생성됨
 - 동일한 브라우저를 통해 JSP2와 JSP3을 실행하면, JSP1, JSP2, JSP3 문서는 [session 객체1]에 속함
- ▶ [브라우저2]을 통해 JSP1을 요청하고 그 응답이 이루어지면 세션이 생성
 - [session 객체2]이 생성됨
 - 동일한 브라우저를 통해 JSP2와 JSP3을 실행하면, JSP1, JSP2, JSP3 문서는 [session 객체2]에 속함

세션 관련 메서드

■ 세션 관련 메서드

메서드	설명
isNew()	세션이 처음으로 시작되었다면 true를 반환(이미 생성되어 있다면 false를 반환)
getId()	세션의 식별자인 세션ID를 반환
getCreationTime()	세션이 시작된 시각을 반환
getLastAccessedTime()	브라우저가 마지막으로 세션에 접근한 시각을 반환
getMaxInactiveInterval()	현재 지정된 세션의 유효 시간을 반환
setMaxInactiveInterval(time)	세션의 유효 시간을 지정한 시간(초)로 지정
getSession()	현재의 세션을 반환(request 객체의 속성)
invalidate()	현재의 세션을 종료
setAttribute(name, value)	세션 속성의 이름을 name으로 지정하고, 값을 value로 지정
getAttribute(name)	이름이 name인 세션 속성의 값을 반환

세션 관련 메서드

■ isNew() 메소드

```
session.isNew()
```

- 세션이 새로 시작된 경우 true를 반환
 - 이미 생성되어 있는 세션인 경우 false를 반환
 - 세션이 생성되었는지 조사하기 위해 주로 조건식으로 사용
 - 세션이 새로 생성된 경우에만 수행해야 할 작업들을 지정하기 위해 사용

■ getID() 메소드

```
session.getID()
```

- 세션ID를 문자열 타입으로 반환

세션 관련 메서드

■ getCreationTime() 메서드

```
session. getCreationTime()
```

- 세션이 생성된 시각을 반환

- ▶ 시각은 1970년 1월 1일을 기준으로 1/1,000초 단위인 타임스탬프(timestamp) 형태로 반환

■ getMaxInactiveInterval() 메서드

```
session. getMaxInactiveInterval()
```

- 세션의 유효 시간을 초단위로 반환

세션 관련 메서드

■ getLastAccessedTime() 메소드

```
session. getLastAccessTime()
```

- 브라우저가 가장 마지막으로 세션에 접근한 시각을 반환
 - ▶ 동일한 세션 내에서 특정 JSP 문서에 마지막으로 접근한 시각을 반환함
 - ▶ 세션이 시작되면 어플리케이션 내의 JSP 문서에 접근할 때마다 마지막 접근 시각이 갱신됨
- 반환되는 시각은 1970년 1월 1일을 기준으로 1/1,000초 단위로 반환

세션 관련 메서드

■ getSession() 메서드

```
HttpSession session = request.getSession( [true] | [false] )
```

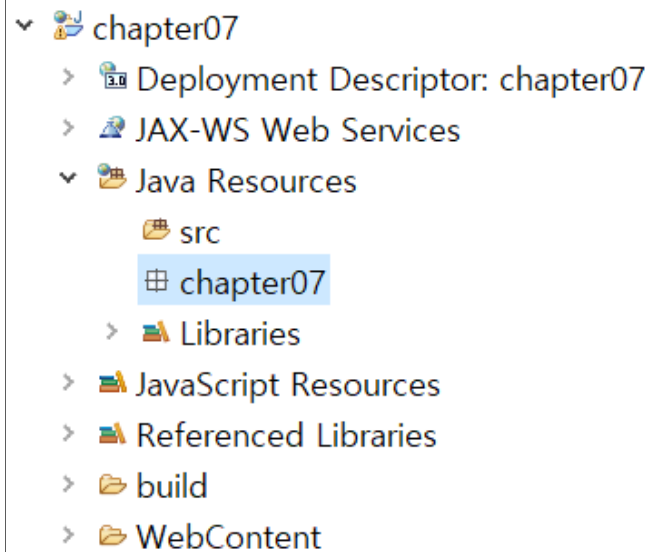
- request 객체에 적용하는 메서드로 세션이 존재하는지 조사하는 메서드
 - ▶ 반환 값은 session임
 - ▶ isNew() 메서드와 유사한 기능을 수행
- 인자가 존재하지 않거나 true인 경우
 - ▶ 현재의 세션을 반환
 - 만약 현재 세션이 존재하지 않으면 새로운 세션을 생성하여 반환
- 인자가 false인 경우
 - ▶ 현재의 세션을 반환
 - 만약 현재 세션이 존재하지 않으면 null을 반환

쿠키의 생성과 참조

■ 패키지과 서블릿 생성

- 패키지 생성

- ▶ 패키지 이름 : chapter07



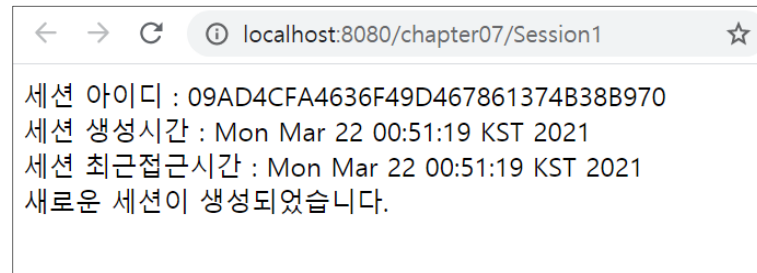
The screenshot shows a project tree in an IDE. The 'chapter07' package is expanded, showing its contents. The 'src' folder is expanded, and a new package named 'chapter07' is being created, highlighted with a blue selection box. The tree structure is as follows:

- chapter07
 - Deployment Descriptor: chapter07
 - JAX-WS Web Services
 - Java Resources
 - src
 - chapter07
 - Libraries
 - JavaScript Resources
 - Referenced Libraries
 - build
 - WebContent

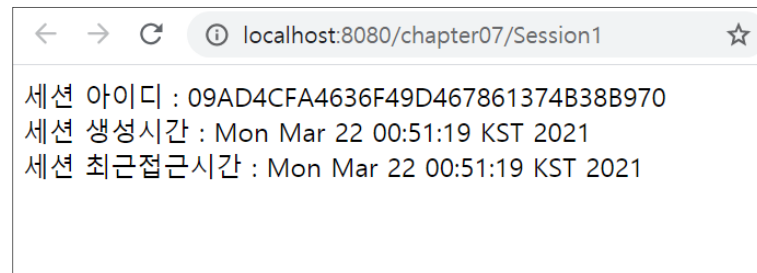

```
1 package chapter07;
2
3 import java.io.IOException;
4 import java.io.PrintWriter;
5 import java.util.Date;
6
7 import javax.servlet.ServletException;
8 import javax.servlet.annotation.WebServlet;
9 import javax.servlet.http.HttpServlet;
10 import javax.servlet.http.HttpServletRequest;
11 import javax.servlet.http.HttpServletResponse;
12 import javax.servlet.http.HttpSession;
13
14 @WebServlet("/Session1")
15 public class Session1 extends HttpServlet {
16     private static final long serialVersionUID = 1L;
17
18     protected void doGet(HttpServletRequest request, HttpServletResponse response)
19         throws ServletException, IOException {
20
21         response.setContentType("text/html; charset=utf-8");
22         PrintWriter out = response.getWriter();
23         HttpSession session = request.getSession();
24
25         out.println("세션 아이디 : " + session.getId() + "<br>");
26         out.println("세션 생성시간 : " + new Date(session.getCreationTime()) + "<br>");
27         out.println("세션 최근접근시간 : " + new Date(session.getLastAccessedTime()) + "<br>");
```

```

28
29     if(session.isNew()) {
30         out.println("새로운 세션이 생성되었습니다.");
31     }
32 }
33
34 protected void doPost(HttpServletRequest request, HttpServletResponse response)
35     throws ServletException, IOException {
36     doGet(request, response);
37 }
38
39 }
    
```



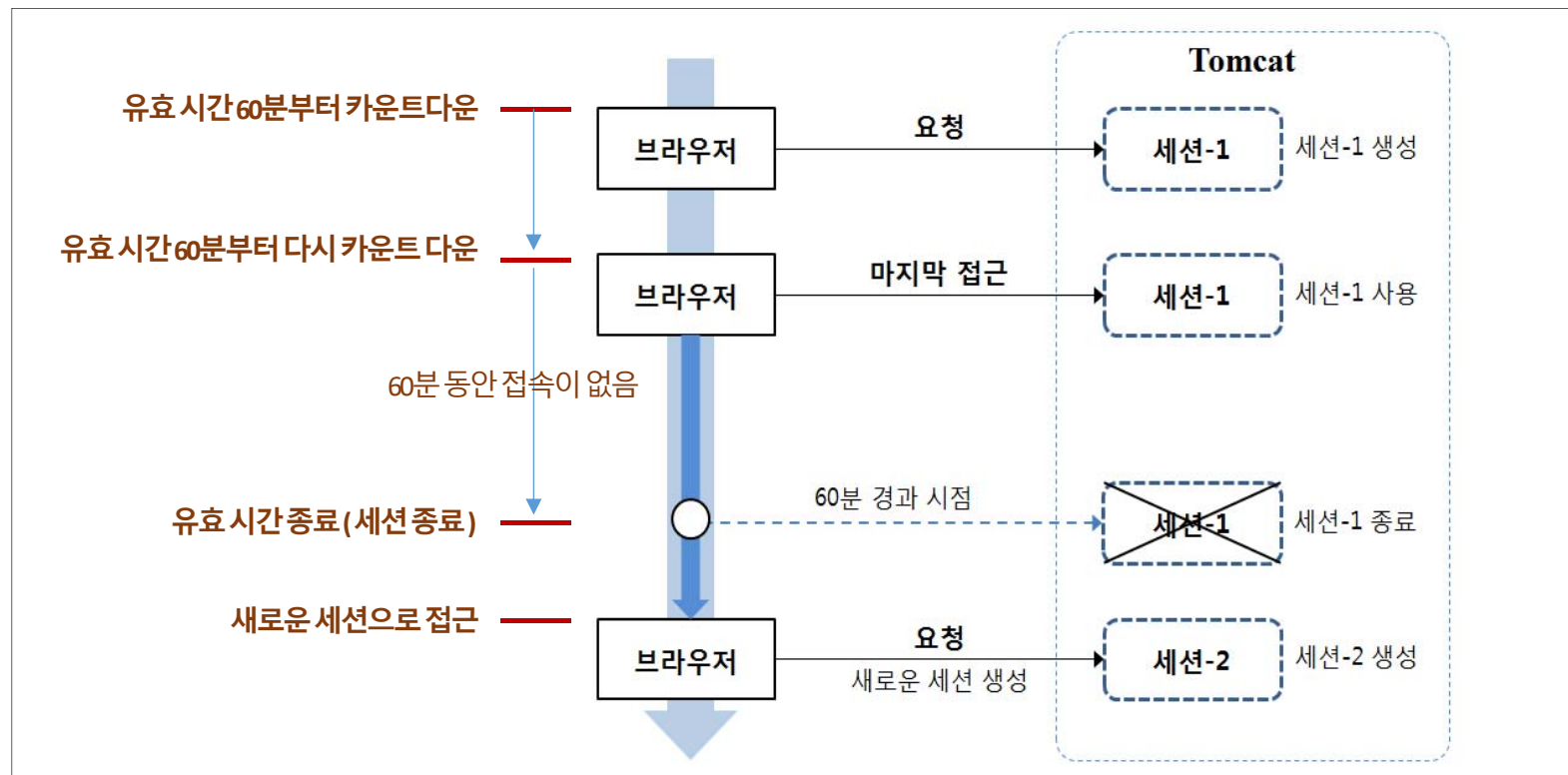
↓ 브라우저를 닫지 않고 reload 수행



세션의 유효 시간

■ 세션의 유효 시간

- 세션은 생성 시점이 아닌 세션의 마지막 접근 시간을 기준으로 함
 - ▶ 유효 시간 내에 세션에 접근에 다시 접근하면(세션 내의 임의의 문서에 접근하면) 유효 시간은 다시 시작됨
 - ▶ [예] 세션의 유효 시간을 60분으로 지정한 경우의 예



세션의 유효 시간

■ 세션의 유효 시간 지정 방법

- `setMaxInactiveInterval()` 메서드를 사용해 지정하는 방법
 - ▶ 세션의 유효 시간을 초단위로 지정
 - ▶ 일반적으로 사용하는 방법
- 어플리케이션의 `WEB-INF/web.xml` 문서에 지정하는 방법
 - ▶ 서버에 존재하는 어플리케이션마다 서로 다른 유효 시간을 지정하고자 하는 경우
- Tomcat 서버의 `web.xml`에 지정하는 방법
 - ▶ 서버에 존재하는 모든 어플리케이션에 같은 유효 시간을 지정하고자 하는 경우

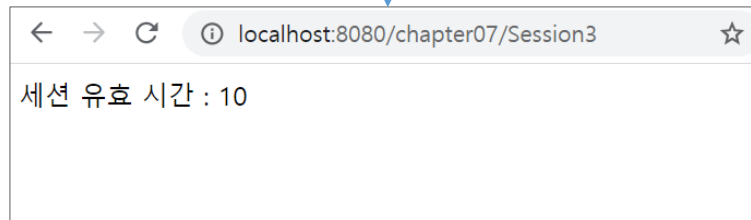
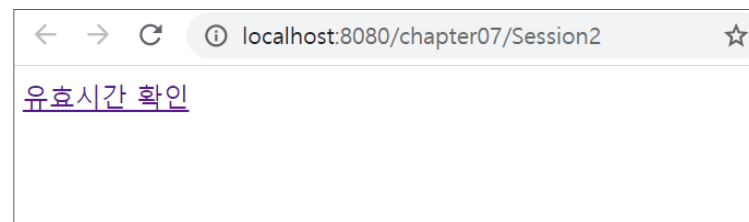
```
1 package chapter07;
2
3 import java.io.IOException;
4 import java.io.PrintWriter;
5
6 import javax.servlet.ServletException;
7 import javax.servlet.annotation.WebServlet;
8 import javax.servlet.http.HttpServlet;
9 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11 import javax.servlet.http.HttpSession;
12
13 @WebServlet("/Session2")
14 public class Session2 extends HttpServlet {
15     private static final long serialVersionUID = 1L;
16
17     protected void doGet(HttpServletRequest request, HttpServletResponse response)
18         throws ServletException, IOException {
19
20         response.setContentType("text/html; charset=utf-8");
21         PrintWriter out = response.getWriter();
22
23         HttpSession session = request.getSession();
24
25         if(session != null) {
26             session.setMaxInactiveInterval(10);
27             out.print("<a href='Session3'>유효시간 확인</a>");
28         }
29     }
```

```
30
31  protected void doPost(HttpServletRequest request, HttpServletResponse response)
32      throws ServletException, IOException {
33      doGet(request, response);
34  }
35 }
```

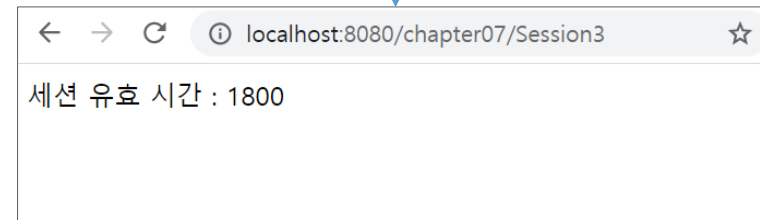
```
1 package chapter07;
2
3 import java.io.IOException;
4 import java.io.PrintWriter;
5
6 import javax.servlet.ServletException;
7 import javax.servlet.annotation.WebServlet;
8 import javax.servlet.http.HttpServlet;
9 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11 import javax.servlet.http.HttpSession;
12
13 @WebServlet("/Session3")
14 public class Session3 extends HttpServlet {
15     private static final long serialVersionUID = 1L;
16
17     protected void doGet(HttpServletRequest request, HttpServletResponse response)
18         throws ServletException, IOException {
19
20         response.setContentType("text/html; charset=utf-8");
21         PrintWriter out = response.getWriter();
22
23         HttpSession session = request.getSession();
24
25         if(session != null) {
26             out.println("세션 유효 시간 : " + session.getMaxInactiveInterval());
27         }
28     }
```

```
29  
30 protected void doPost(HttpServletRequest request, HttpServletResponse response)  
31     throws ServletException, IOException {  
32     doGet(request, response);  
33 }  
34 }
```

유효 시간을 10초로 지정



10초로 지정 이내에 '유효 시간 확인'을 클릭한 경우



10초로 지정 이후 '유효 시간 확인'을 클릭한 경우

세션의 유효 시간

■ 어플리케이션의 WEB-INF/web.xml 문서에 지정

- 현재 어플리케이션 전체에 유효 시간이 적용됨
 - <session-config>의 자식 태그로 <session-timeout>를 사용해 지속 시간을 지정
 - 시간 단위는 초(second)가 아니라 분(minute)임
- 세션 지속시간을 20분으로 지정하는 예

```
<web-app .....>
  <display-name>jspbook</display-name>
  <welcome-file-list>
    .....
  </welcome-file-list>

  <session-config>
    <session-timeout>20</session-timeout>
  </session-config>

</web-app>
```

세션의 유효 시간

■ Tomcat 서버의 web.xml에 지정

- 현재 서버에 존재하는 모든 어플리케이션에 세션에 적용
 - 지정하는 방법은 웹 어플리케이션의 WEB-INF/web.xml에서 지정하는 방법과 동일
 - 기본값은 30분임

```
<!-- ===== Default Session Configuration ===== -->
<!-- You can set the default session timeout (in minutes) for all newly -->
<!-- created sessions by modifying the value below. -->

<session-config>
  <session-timeout>30</session-timeout>
</session-config>
```

세션의 속성

■ 세션의 속성

- 쿠키 변수와 같이 세션을 사용해 문서 사이에서 정보를 공유하기 위해 세션 속성을 사용
 - 지정된 속성을 동일한 세션을 사용하는 모든 문서들에서 사용될 수 있음

- 세션 속성의 지정

```
session.setAttribute( name, value )
```

- 세션 속성의 이름을 name으로 지정하고, 값을 value로 지정

- 세션 속성의 추출

```
session.getAttribute( name )
```

- 이름이 name인 세션 속성의 값을 반환

세션의 종료

■ 세션의 종료

```
session.invalidate()
```

- 현재의 세션을 강제로 종료함
 - 세션 객체 내의 모든 세션 속성도 제거됨
- 주로 로그아웃을 구현에 사용됨

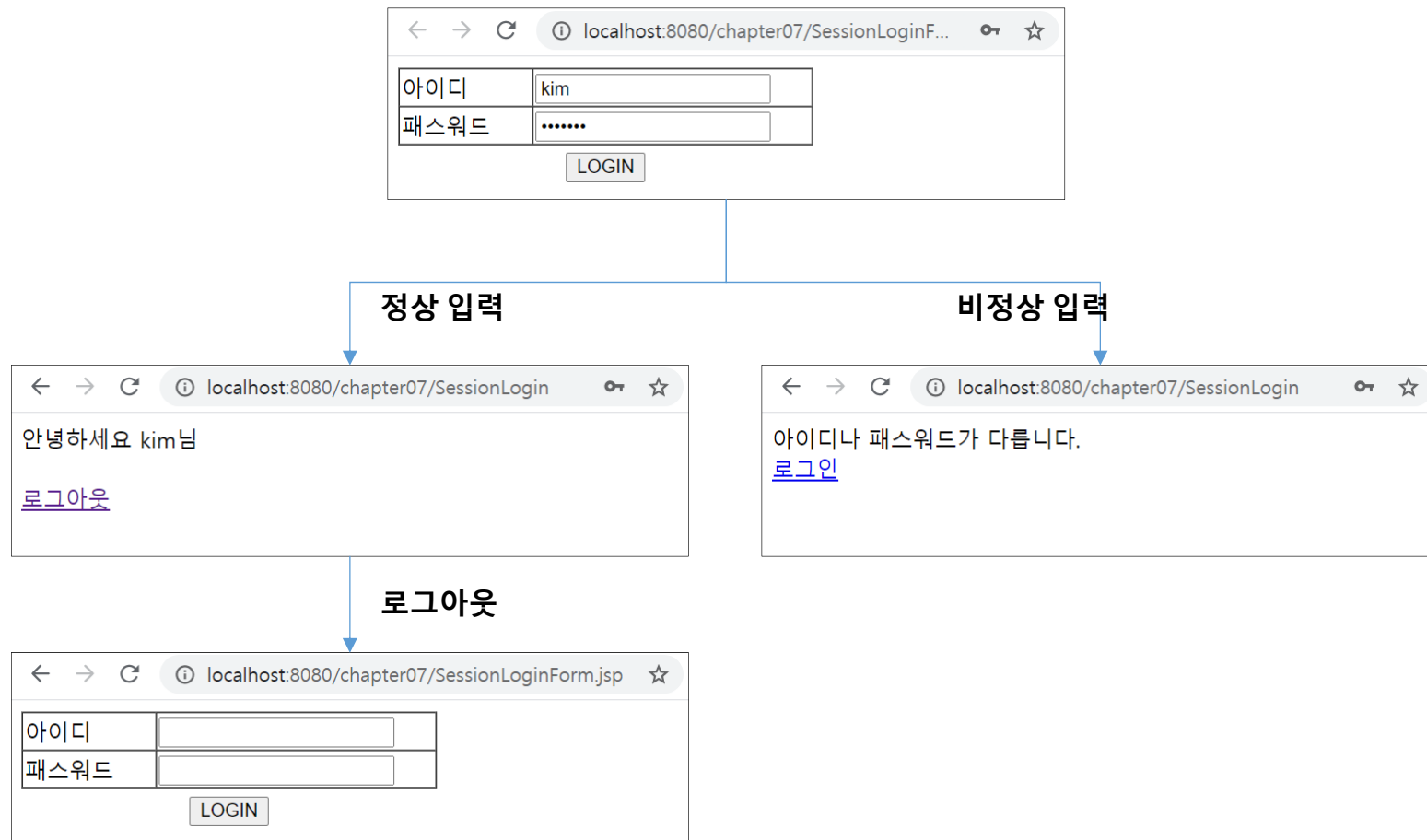
```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6   <meta charset="UTF-8">
7 </head>
8 <body>
9   <form name="user" method=post action="SessionLogin">
10    <table border=1 width=300 cellpadding=0>
11      <tr>
12        <td width=100>아이디</td>
13        <td width=200><input type=text name=userID size=20></td>
14      </tr>
15      <tr>
16        <td width=100>패스워드</td>
17        <td width=200><input type=password name=userPW size=20></td>
18      </tr>
19    </table>
20    <table width=300>
21      <tr>
22        <td align=center><input type=submit value="LOGIN"></td>
23      </tr>
24    </table>
25  </form>
26 </body>
27 </html>
```

```
1 package chapter07;
2
3 import java.io.IOException;
4 import java.io.PrintWriter;
5
6 import javax.servlet.ServletException;
7 import javax.servlet.annotation.WebServlet;
8 import javax.servlet.http.HttpServlet;
9 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11 import javax.servlet.http.HttpSession;
12
13 @WebServlet("/SessionLogin")
14 public class SessionLogin extends HttpServlet {
15     private static final long serialVersionUID = 1L;
16
17     protected void doGet(HttpServletRequest request, HttpServletResponse response)
18         throws ServletException, IOException {
19
20         response.setContentType("text/html; charset=utf-8");
21         PrintWriter out = response.getWriter();
22
23         HttpSession session = request.getSession();
24
25         // 실제 패스워드를 가정함
26         String realID="kim";
27         String realPW="kimpass";
28         String userID = request.getParameter("userID");
29         String userPW = request.getParameter("userPW");
```

```
30
31     if( userID.equals(realID) && userPW.equals(realPW)) {
32
33         session.setAttribute("user_ID", userID);
34         String user_ID = (String)session.getAttribute("user_ID");
35         out.println("안녕하세요 " + user_ID + "님<br><br>");
36         out.println("<a href='SessionLogout'>로그아웃</a>");
37
38     } else {
39
40         out.println("아이디나 비밀번호가 다릅니다.<br>");
41         out.println("<a href='ch06/Session-form.jsp'>로그인</a>");
42
43     }
44
45 }
46
47 protected void doPost(HttpServletRequest request, HttpServletResponse response)
48     throws ServletException, IOException {
49     doGet(request, response);
50 }
51
52 }
```

```
1 package chapter07;
2
3 import java.io.IOException;
4 import javax.servlet.ServletException;
5 import javax.servlet.annotation.WebServlet;
6 import javax.servlet.http.HttpServlet;
7 import javax.servlet.http.HttpServletRequest;
8 import javax.servlet.http.HttpServletResponse;
9 import javax.servlet.http.HttpSession;
10
11 @WebServlet("/SessionLogout")
12 public class SessionLogout extends HttpServlet {
13     private static final long serialVersionUID = 1L;
14
15     protected void doGet(HttpServletRequest request, HttpServletResponse response)
16         throws ServletException, IOException {
17
18         HttpSession session = request.getSession();
19         session.invalidate();
20         response.sendRedirect("SessionLoginForm.jsp");
21     }
22
23     protected void doPost(HttpServletRequest request, HttpServletResponse response)
24         throws ServletException, IOException {
25         doGet(request, response);
26     }
27 }
28 }
```


세션 실습



수고하셨습니다