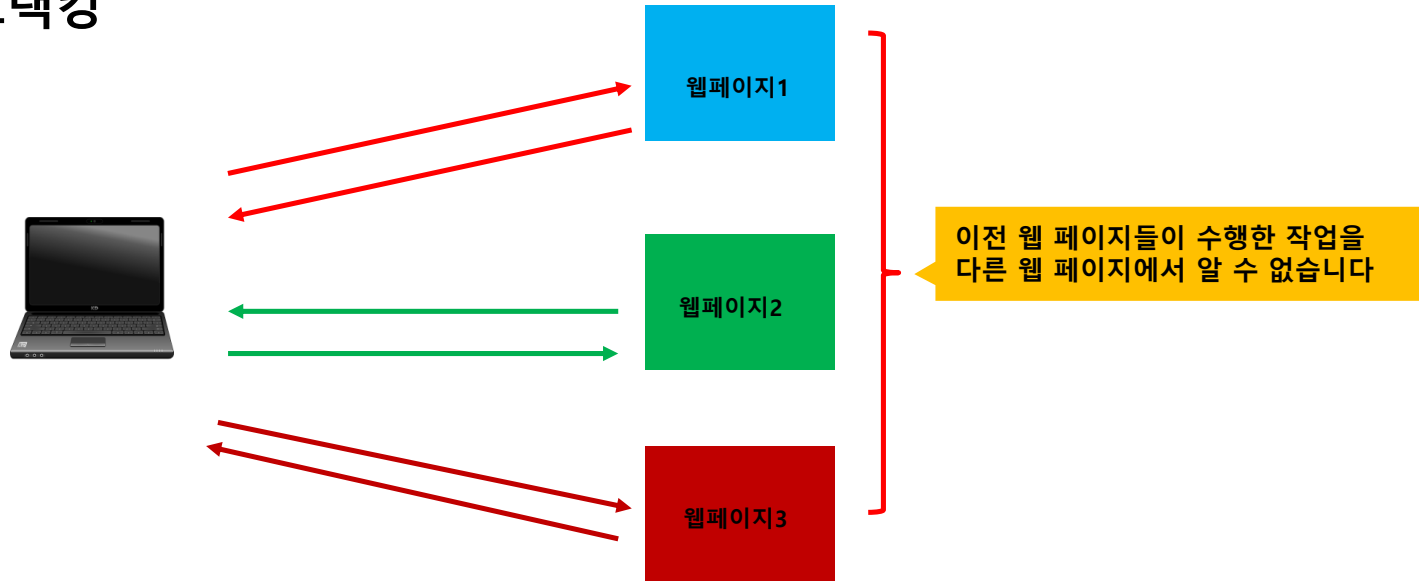


9장 쿠키와 세션 알아보기

1. 웹 페이지 연결 기능
2. <hidden> 태그와 URL Rewriting 이용해 웹 페이지 연동하기
3. 쿠키를 이용한 웹 페이지 연동기능
4. 세션을 이용한 웹 페이지 연동 기능
5. encodeURL() 사용법
6. 세션을 이용한 로그인 예제

1. 웹 페이지 연결 기능

1.1 세션 트래킹



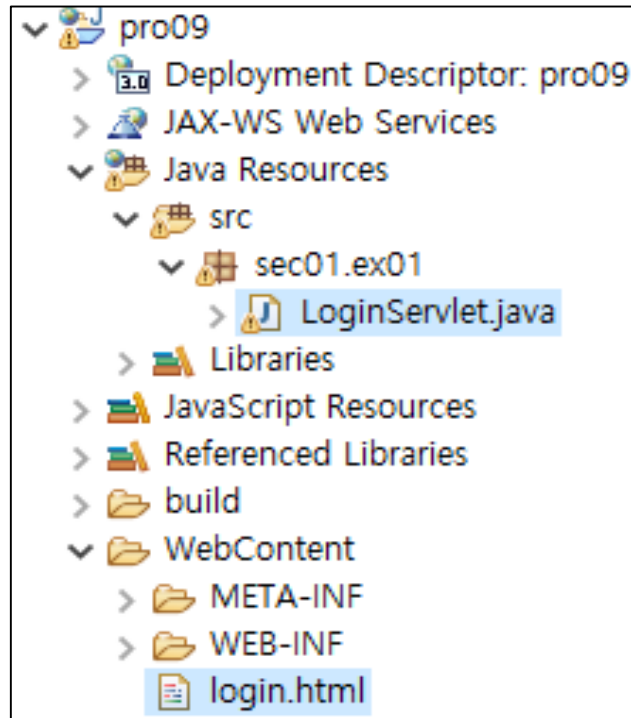
- HTTP 프로토콜은 서버-클라이언트 통신 시 stateless 방식으로 통신 **상태정보를 안 갖고 있다**
- 즉 브라우저에서 새 웹 페이지를 열면 기존의 웹 페이지나 서블릿에 관한 어떤 연결 정보도 알 수 없음
- 따라서 세션 트래킹을 이용해서 웹 페이지 간의 연결 기능을 구현함

- <hidden> 태그 와 URL ReWriting
- 쿠키와 세션

2. <hidden> 태그와 URL Rewriting 이용

2.1 <hidden> 태그를 이용한 세션 트래킹 실습

1. 새 프로젝트 pro09를 만들고 sec01.ex01 패키지를 생성한 후 다음과 같이 LoginServlet 클래스 파일과 login.html을 준비



2. <hidden> 태그와 URL Rewriting 이용

2. login.html을 다음과 같이 작성합니다. 로그인창에서 ID와 비밀번호를 입력하면 미리 <hidden> 태그에 저장된 주소, 이메일, 휴대폰 번호를 서블릿으로 전송합니다.

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="UTF-8">
  <title>로그인창</title>
</head>
<body>
  <form name="frmLogin" method="post" action="login" encType="UTF-8">
    아이디 :<input type="text" name="user_id"><br>
    비밀번호:<input type="password" name="user_pw"><br>
    <input type="submit" value="로그인">
    <input type="reset" value="다시 입력">
    <input type="hidden" name="user_address" value="서울시 성북구" />
    <input type="hidden" name="user_email" value="test@gmail.com" />
    <input type="hidden" name="user_hp" value="010-111-2222" />
  </form>
</body>
</html>
```

〈hidden〉 태그의 value 속성에 주소, 이메일, 전화번호를 저장한 후 서블릿으로 전송합니다.

2. <hidden> 태그와 URL Rewriting 이용

3. LoginServlet 클래스를 다음과 같이 작성합니다. getParameter() 메서드를 이용해 전송된 회원 정보를 가져온 후 브라우저로 다시 출력합니다.

```
package sec01.ex01;

...

@WebServlet("/login")
public class LoginServlet extends HttpServlet{
    public void init(){
        System.out.println("init 메서드 호출");
    }
```

```
protected public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    request.setCharacterEncoding("utf-8");
    response.setContentType("text/html;charset=utf-8");
    PrintWriter out = response.getWriter();
    String user_id = request.getParameter("user_id");
    String user_pw = request.getParameter("user_pw");
    String user_address=request.getParameter("user_address");
    String user_email=request.getParameter("user_email");
    String user_hp=request.getParameter("user_hp");
```

<hidden> 태그로 전송된 값을
getParameter() 메서드를 이용
해 가져옵니다.

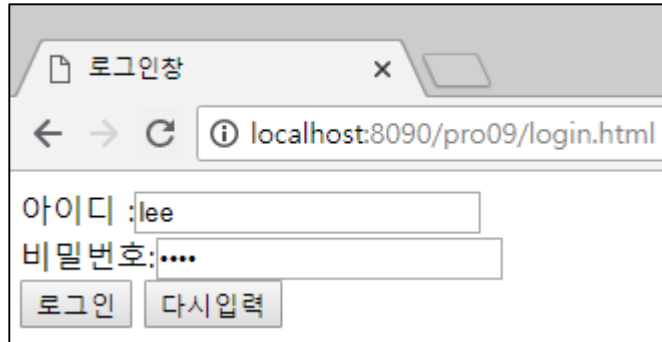
2. <hidden> 태그와 URL Rewriting 이용

```
data+="아이디 : "+user_id ;
data+="<br>";
data+="패스워드 : "+user_pw;
data+="<br>";
data+="주소 : "+user_address;
data+="<br>";
data+="email : "+user_email;
data+="<br>";
data+="휴대전화 : "+user_hp;
data+="</html></body>";
out.print(data);
}

public void destroy(){
    System.out.println("destroy 메서드 호출");
}
}
```

2. <hidden> 태그와 URL Rewriting 이용

4. `http://localhost:8090/pro09/login.html`로 요청하고 ID와 비밀번호를 입력한 후 서블릿으로 전송합니다.



로그인창

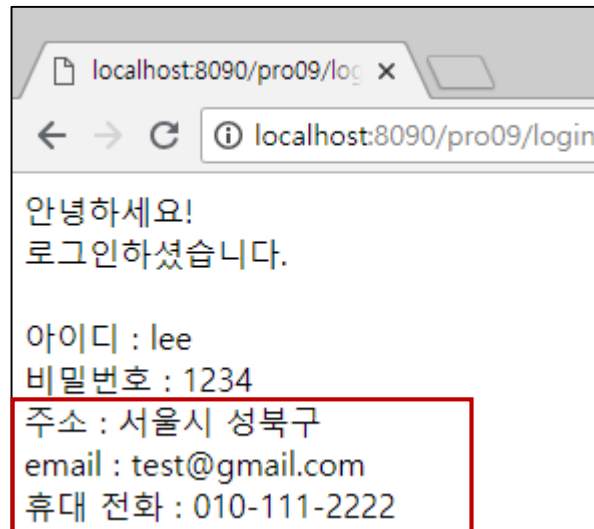
← → ↻ localhost:8090/pro09/login.html

아이디 : lee

비밀번호 :

로그인 다시입력

5. <hidden> 태그로 전송된 데이터도 출력합니다.



localhost:8090/pro09/log x

← → ↻ localhost:8090/pro09/login

안녕하세요!
로그인하셨습니다.

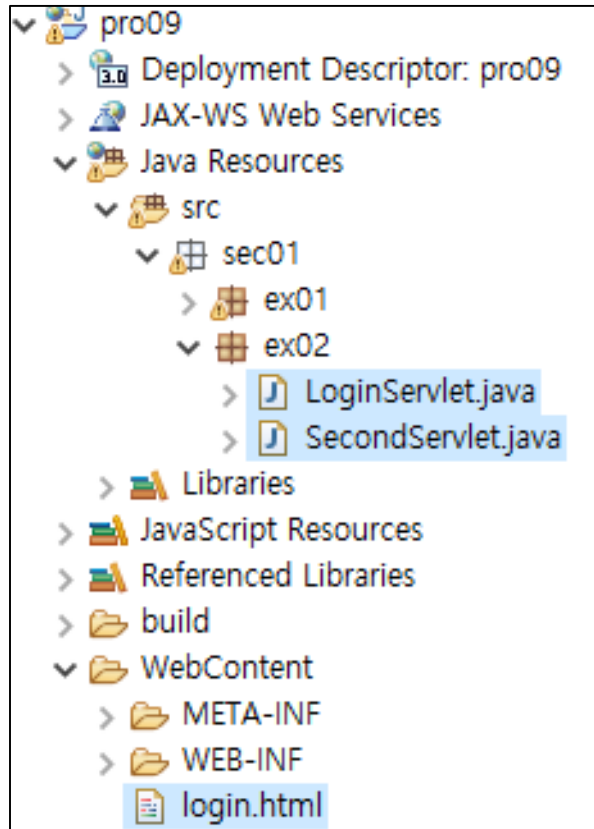
아이디 : lee
비밀번호 : 1234

주소 : 서울시 성북구
email : test@gmail.com
휴대 전화 : 010-111-2222

2. <hidden>태그와 URL Rewriting 이용

- 2.2 URL Rewriting을 이용한 세션 트래킹 실습

1. 새로운 패키지를 만들고 LoginServlet, SecondServlet 클래스 파일을 준비합니다.



2. <hidden>태그와 URL Rewriting 이용

- 2.2 URL Rewriting을 이용한 세션 트래킹 실습

2. LoginServlet 클래스를 다음과 같이 작성합니다.

```
package sec01.ex02;  
...  
@WebServlet("/login")
```

코드 9-2에서 주식 처리했다면 다시
주식 처리를 해제합니다.

```
public class LoginServlet extends HttpServlet{  
    public void init(){  
        System.out.println("init 메서드 호출");  
    }  
protected public void doPost(HttpServletRequest request, HttpServletResponse response)  
    throws ServletException, IOException {  
        request.setCharacterEncoding("utf-8");  
        response.setContentType("text/html;charset=utf-8");  
        PrintWriter out = response.getWriter();  
        String user_id = request.getParameter("user_id");  
        String user_pw = request.getParameter("user_pw");  
        String user_address=request.getParameter("user_address");
```

2. <hidden> 태그와 URL Rewriting 이용

```
String user_email=request.getParameter("user_email");
```

```
String user_hp=request.getParameter("user_hp");
```

```
String data="안녕하세요!<br>로그인하셨습니다.<br><br>";
```

```
data+="<html><body>";
```

```
data+="아이디 : "+user_id ;
```

```
data+="<br>";
```

```
data+="패스워드 : "+user_pw;
```

```
data+="<br>";
```

```
data+="주소 : "+user_address;
```

```
data+="<br>";
```

```
data+="email : "+user_email;
```

```
data+="<br>";
```

```
data+="휴대전화 : "+user_hp;
```

```
data+="<br>";
```

```
out.print(data);
```

```
user_address=URLEncoder.encode(user_address,"utf-8");
```

```
out.print("<a href='/pro09/second?user_id="+user_id+"&user_pw="+user_pw+  
"&user_address="+user_address+"'>두 번째 서블릿으로 보내기</a>");
```

```
data="</body></html>";
```

```
out.print(data);
```

```
}
```

GET 방식으로 한글을 전송하기
위해 인코딩합니다.

<a> 태그를 이용해 링크 클릭 시
서블릿 /second로 다시 로그인
정보를 전송합니다.

2. <hidden> 태그와 URL Rewriting 이용

3. SecondServlet 클래스를 다음과 같이 작성합니다. 첫 번째 서블릿에서 전송한 데이터 중 ID와 비밀번호를 가져왔으면 이미 첫 번째 서블릿에서 로그인한 것이므로 로그인 상태를 유지하도록 해줍니다.

```
package sec01.ex02;

...
@WebServlet("/second")
public class SecondServlet extends HttpServlet{
    public void init(){
        System.out.println("init 메서드 호출");
    }
}
```

2/ <hidden> 태그와 URL Rewriting 이용

```
public protected void doGet(HttpServletRequest request, HttpServletResponse response)
```

```
throws ServletException, IOException {  
    request.setCharacterEncoding("utf-8");  
    response.setContentType("text/html;charset=utf-8");  
    PrintWriter out = response.getWriter();
```

첫 번째 서블릿에서 전송한
로그인 정보를 가져옵니다.

```
String user_id = request.getParameter("user_id");  
String user_pw = request.getParameter("user_pw");  
String user_address = request.getParameter("user_address");
```

```
out.println("<html><body>");
```

```
if(user_id!=null && user_id.length()!=0){
```

첫 번째 서블릿의 ID 정보를 이용해
로그인 상태를 유지합니다.

```
    out.println("이미 로그인 상태입니다!<br><br>");  
    out.println("첫 번째 서블릿에서 넘겨준 아이디: " + user_id + "<br>");  
    out.println("첫 번째 서블릿에서 넘겨준 비밀번호: " + user_pw + "<br>");  
    out.println("첫 번째 서블릿에서 넘겨준 주소: " + user_address + "<br>");  
    out.println("</body></html>");
```

```
}else{
```

```
    out.println("로그인 하지 않았습니다.<br><br>");  
    out.println("다시 로그인하세요!!<br>");  
    out.println("<a href='/pro09/login.html'>로그인창으로 이동하기 </>");
```

```
}
```

```
}
```

로그인창을 거치지 않고 바로 요청한 경우에는
로그인창으로 다시 이동하도록 안내합니다.

2. <hidden> 태그와 URL Rewriting 이용

4. <hidden> 태그와 URL Rewriting 방식으로 데이터를 전송한 결과를 볼까요?

http://localhost:8090/pro09/login.html로 요청한 후 ID와 비밀번호를 입력하고 첫 번째 서블릿으로 전송합니다.

로그인창

← → ↻ ⓘ localhost:8090/pro09/login.html

아이디 : hong

비밀번호 :

로그인 다시입력

5. 첫 번째 서블릿에서 전달받은 로그인 정보를 출력한 후 두 번째 서블릿으로 보내기를 클릭합니다.

localhost:8090/pro09/log x

← → ↻ ⓘ localhost:8090/pro09/login

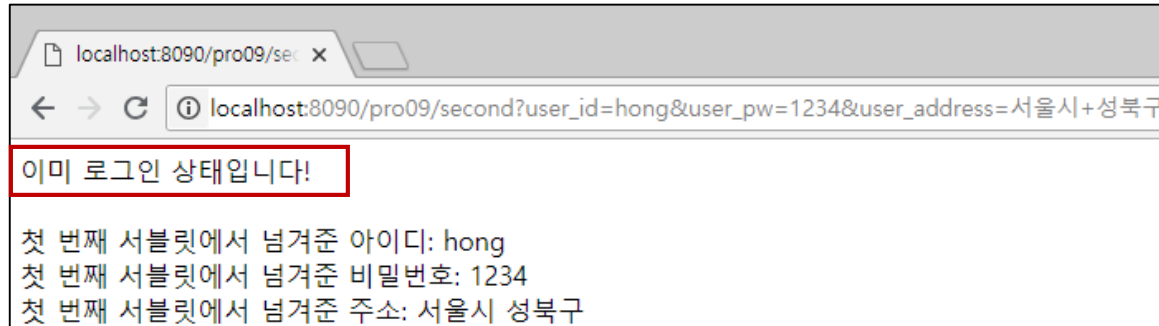
안녕하세요!
로그인하셨습니다.

아이디 : hong
비밀번호 : 1234
주소 : 서울시 성북구
email : test@gmail.com
휴대 전화 : 010-111-2222

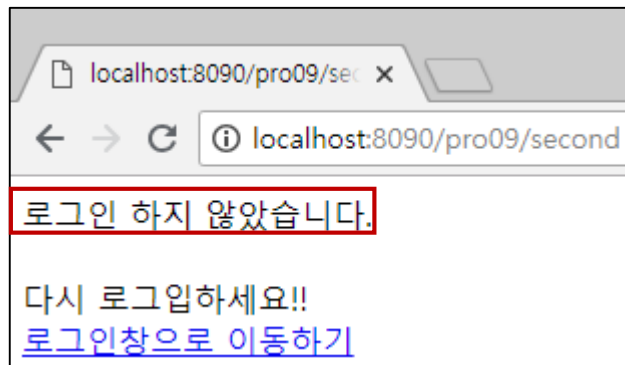
두 번째 서블릿으로 보내기

2. <hidden> 태그와 URL Rewriting 이용

6. 두 번째 서블릿에서 현재 로그인 상태와 회원 정보를 출력합니다.



7. 만약 브라우저에서 로그인창을 거치지 않고 바로 서블릿 /second를 요청하면 "로그인 하지않았습니다."라는 상태 안내 문구와 함께 '로그인창으로 이동하기'가 표시됩니다.



2. <hidden> 태그와 URL Rewriting 이용

<hidden> 태그와 URL Rewriting의 문제점

- 웹 페이지가 많아지면 일일이 로그인 정보를 전송해야 함
- GET 방식으로 전송하므로 보안에 취약함

3. 쿠키를 이용한 웹 페이지 연동 기능

쿠키(Cookie)

➤ 웹 페이지들 사이의 공유 정보를 클라이언트 PC에 저장해 놓고 사용하는 방법

쿠키(Cookie)의 특징

- 정보가 클라이언트 PC에 저장됨
- 저장 정보 용량에 제한이 있음(파일 용량은 4kb)
- 보안이 취약함
- 클라이언트 브라우저에서 사용 유무를 설정할 수 있음
- 도메인당 쿠키가 만들어짐(웹 사이트당 하나의 쿠키)

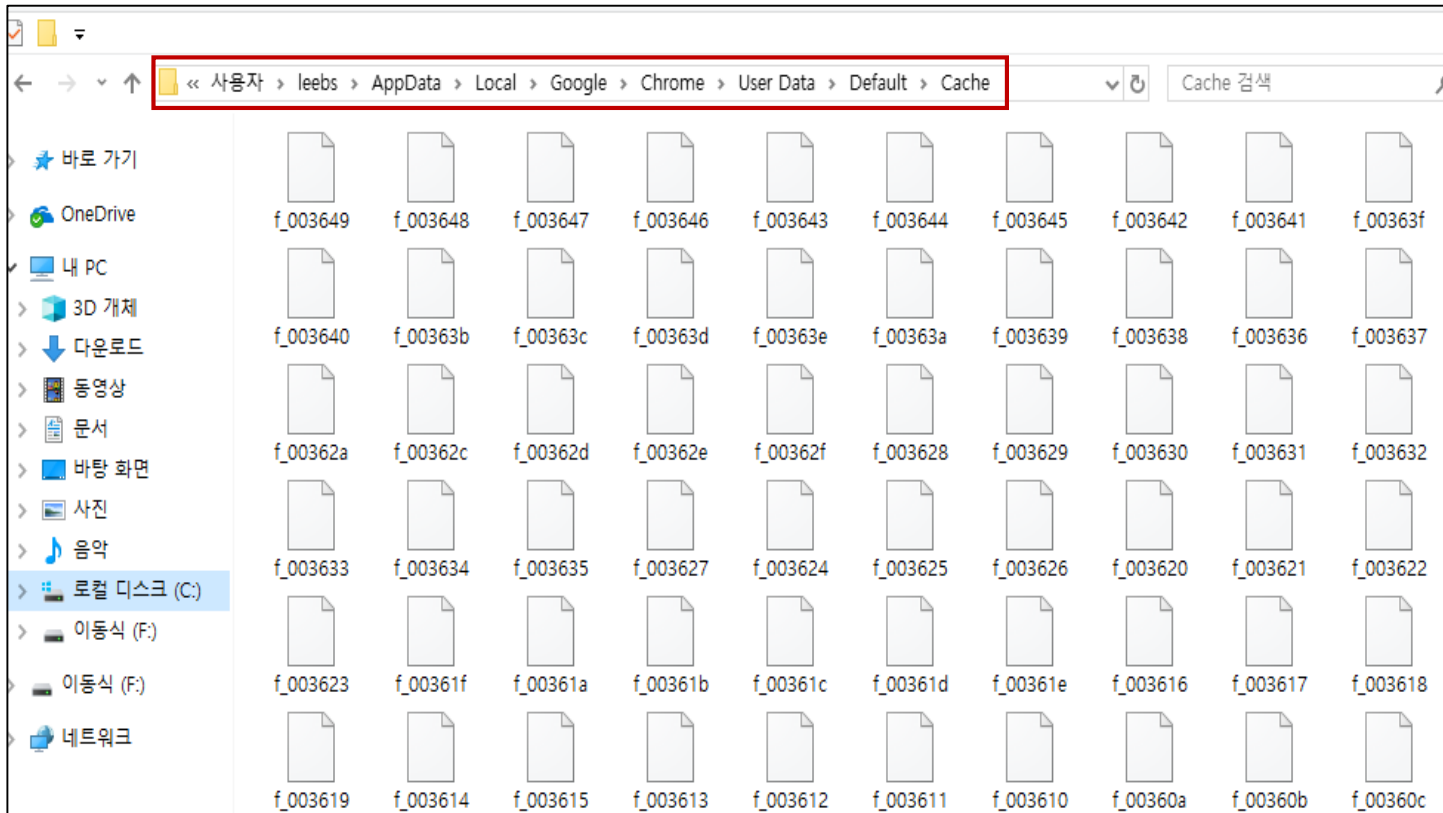
쿠키(Cookie)의 종류

속성	Persistence 쿠키	Session 쿠키
생성 위치	파일로 생성	브라우저 메모리에 생성
종료 시기	쿠키를 삭제하거나 쿠키 설정 값이 종료된 경우	브라우저를 종료한 경우
최초 접속 시 전송여부	최초 접속 시 서버로 전송	최초 접속 시 서버로 전송되지 않음
용도	로그인 유무 또는 팝업창을 제한할 때	사이트 접속 시 Session 인증 정보를 유지할 때

3. 쿠키를 이용한 웹 페이지 연동 기능


브라우저의 쿠키 생성 위치

1. 윈도우 탐색기를 열고 C:\Users\사용자\AppData\Local\Google\Chrome\User Data\Default\Cache로 이동하면 크롬에서 사용하는 쿠키 파일이 보일 것입니다.



일반	보안	개인 정보	내용	연결	프로그램	고급
----	----	-------	----	----	------	----

홈 페이지



홈페이지 탭을 만들려면 한 줄에 하나씩 주소를 입력하십시오(R).

현재 페이지(C)

기본값 사용(F)

새 탭 사용(U)

시작 옵션

☐ 마지막 세션의 탭으로 시작(B)
☒ 홈 페이지로 시작(H)

탭

탭 사용에 관련된 옵션을 변경합니다.

탭(T)

검색 기록

임시 파일, 열어본 페이지 목록, 쿠키, 저장된 암호 및 웹 양식 정보를 삭제합니다.

☐ 종료할 때 검색 기록 삭제(W)

삭제(D)...

설정(S)

모양

색(O)

언어(L)

글꼴(N)

접근성(E)

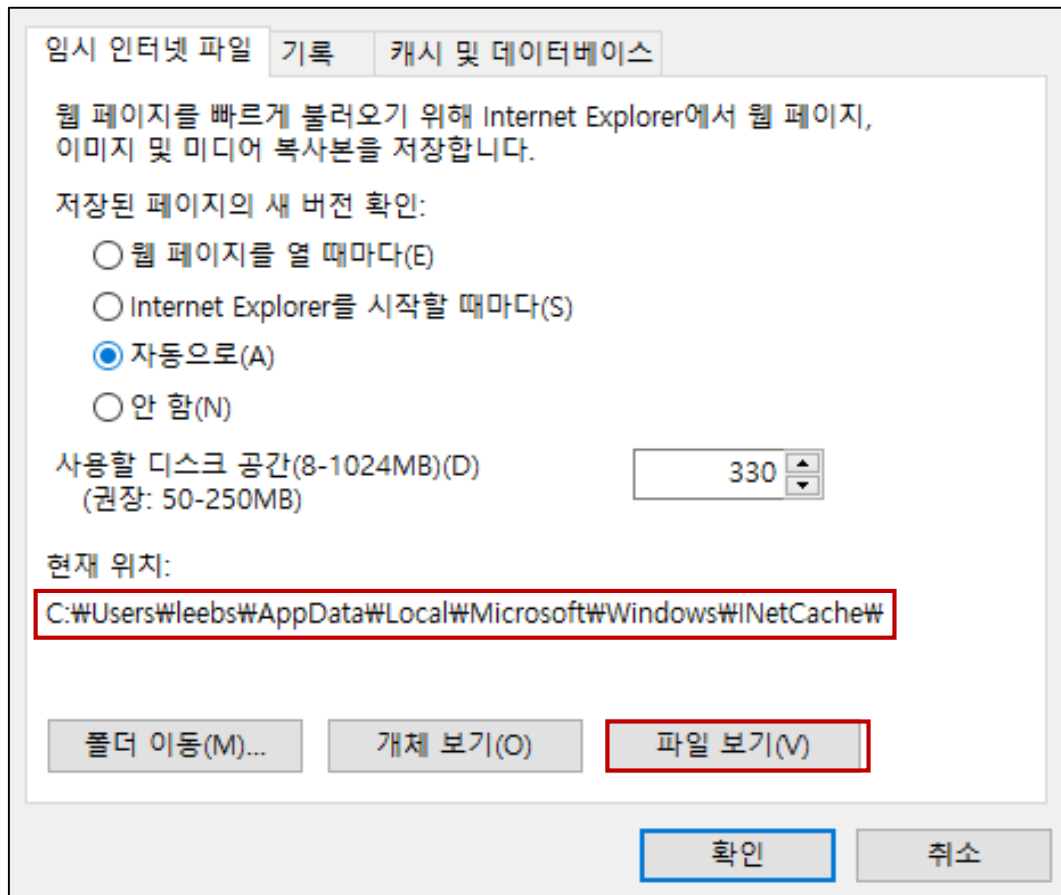
확인

취소

적용(A)

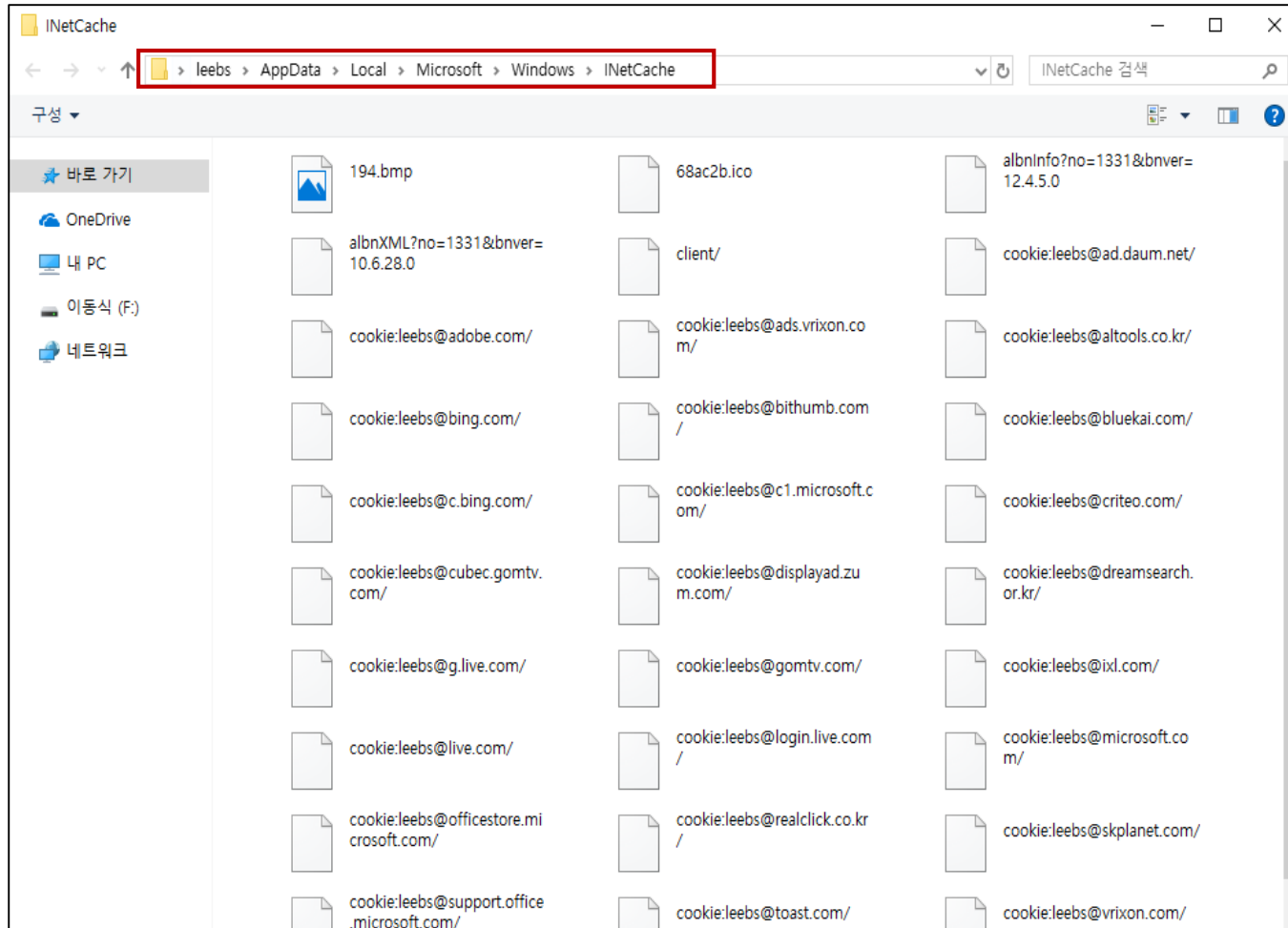
3. 쿠키를 이용한 웹 페이지 연동 기능

3. '임시 인터넷 파일' 탭에서 파일 보기를 클릭합니다.



3. 쿠키를 이용한 웹 페이지 연동 기능

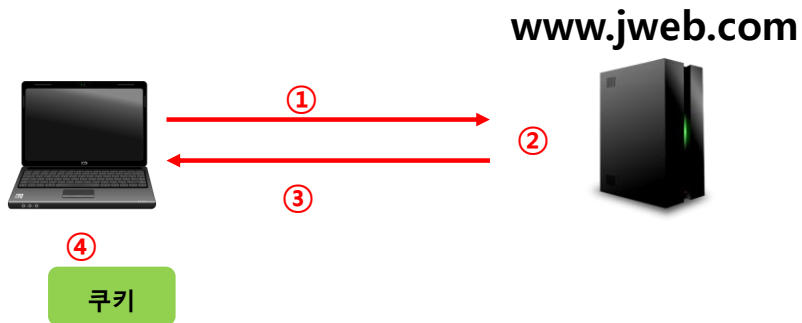
4. 인터넷 익스플로러에서 사용하는 쿠키 파일이 저장되어 있는 폴더가 나타납니다.



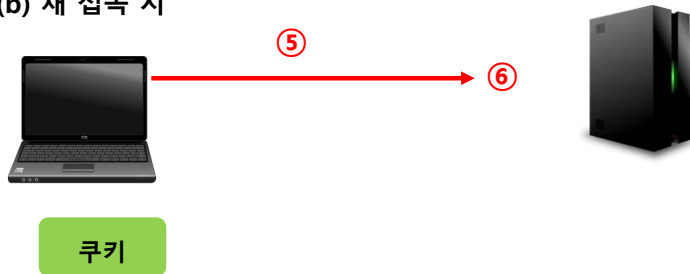
3. 쿠키를 이용한 웹 페이지 연동 기능

3.1 쿠키 기능 실행 과정

(a) 최초 사이트 접속 시



(b) 재 접속 시



- ① 브라우저로 사이트에 접속합니다.
- ② 서버는 정보를 저장한 쿠키를 생성합니다.
- ③ 생성된 쿠키를 브라우저로 전송합니다.
- ④ 브라우저는 서버로부터 받은 쿠키 정보를 쿠키 파일에 저장합니다.
- ⑤ 브라우저가 다시 접속해 서버가 브라우저에게 쿠키 전송을 요청하면 브라우저는 쿠키 정보를 서버에 넘겨줍니다.
- ⑥ 서버는 쿠키 정보를 이용해 작업을 합니다.

3. 쿠키를 이용한 웹 페이지 연동 기능

3.2 쿠키 API

쿠키 API의 특징

- javax.servlet.http.Cookie를 이용합니다.
- HttpServletResponse의 addCookie() 메서드를 이용해 클라이언트 브라우저에 쿠키를 전송한 후 저장합니다.
- HttpServletRequest의 getCookie() 메서드를 이용해 쿠키를 서버로 가져옵니다.

Cookie 클래스의 여러가지 메서드

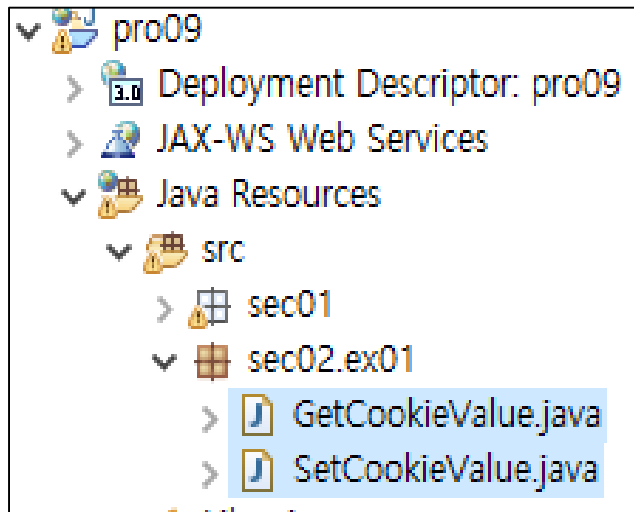
메서드	설명
getComment()	쿠키에 대한 설명을 가져옵니다.
getDomain()	쿠키의 유효한 도메인 정보를 가져옵니다.
getMaxAge()	쿠키 유효 기간을 가져옵니다.
getName()	쿠키 이름을 가져옵니다.
getPath()	쿠키의 디렉터리 정보를 가져옵니다.
getValue()	쿠키의 설정 값을 가져옵니다.
setComment(String)	쿠키에 대한 설명을 설정합니다.
setDomain(String)	쿠키의 유효한 도메인을 설정합니다.
setMaxAge(int)	쿠키 유효 기간을 설정합니다.
setValue(String)	쿠키 값을 설정합니다.

설정된 서블릿에서만 생성

3. 쿠키를 이용한 웹 페이지 연동 기능

- 3.3 서블릿에서 쿠키 사용하기

1. GetCookieValue, SetCookieValue 클래스 파일을 준비합니다.



3. 쿠키를 이용한 웹 페이지 연동 기능

2. SetCookieValue 클래스를 다음과 같이 작성합니다.

```
package sec02.ex01;
...
@WebServlet("/set")
public class SetCookieValue extends HttpServlet{
protected public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    response.setContentType("text/html;charset=utf-8");
    PrintWriter out=response.getWriter();
    Date d=new Date();    key          values
    Cookie c=new Cookie("cookieTest",URLEncoder.encode("JSP프로그래밍입니다."
                                                         , "utf-8"));
    c.setMaxAge(24*60*60); • 초단위          유효 기간을 설정합니다.
    response.addCookie(c); •                  생성된 쿠키를 브라우저로 전송합니다.
    out.println("현재시간 : "+d);
    out.println("현재시간을 Cookie로 저장합니다.");
}
}
```


3. 쿠키를 이용한 웹 페이지 연동 기능

3. GetCookieValue 클래스를 다음과 같이 작성합니다.

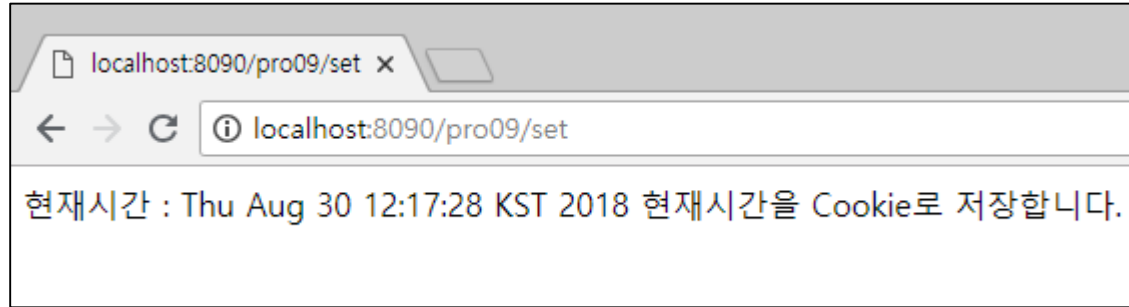
```
package sec02.ex01;
...
@WebServlet("/get")
public class GetCookieValue extends HttpServlet{
protected public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    response.setContentType("text/html;charset=utf-8");
    PrintWriter out=response.getWriter();
    Cookie[] allValues=request.getCookies();
    for(int i=0; i<allValues.length;i++){
        if(allValues[i].getName().equals("cookieTest")){
            out.println("<h2>Cookie 값 가져오기 :
                        "+URLDecoder.decode(allValues[i].getValue(),"utf-8"));
        }
    }
}
}
```

request의 getCookies() 메서드를 호출해 브라우저에게 쿠키 정보를 요청한 후 쿠키 정보를 배열로 가져옵니다.

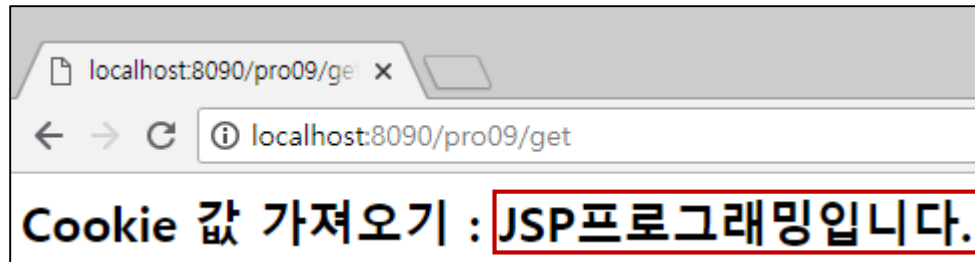
배열에서 ~~배열에~~ 저장할 때 사용한 쿠키 이름인 cookieTest로 검색해 쿠키 값을 가져옵니다.

3. 쿠키를 이용한 웹 페이지 연동 기능

4. 우선 set으로 첫 번째 서블릿을 요청합니다. 쿠키에 cookieTest 이름으로 문자열을 저장합니다.



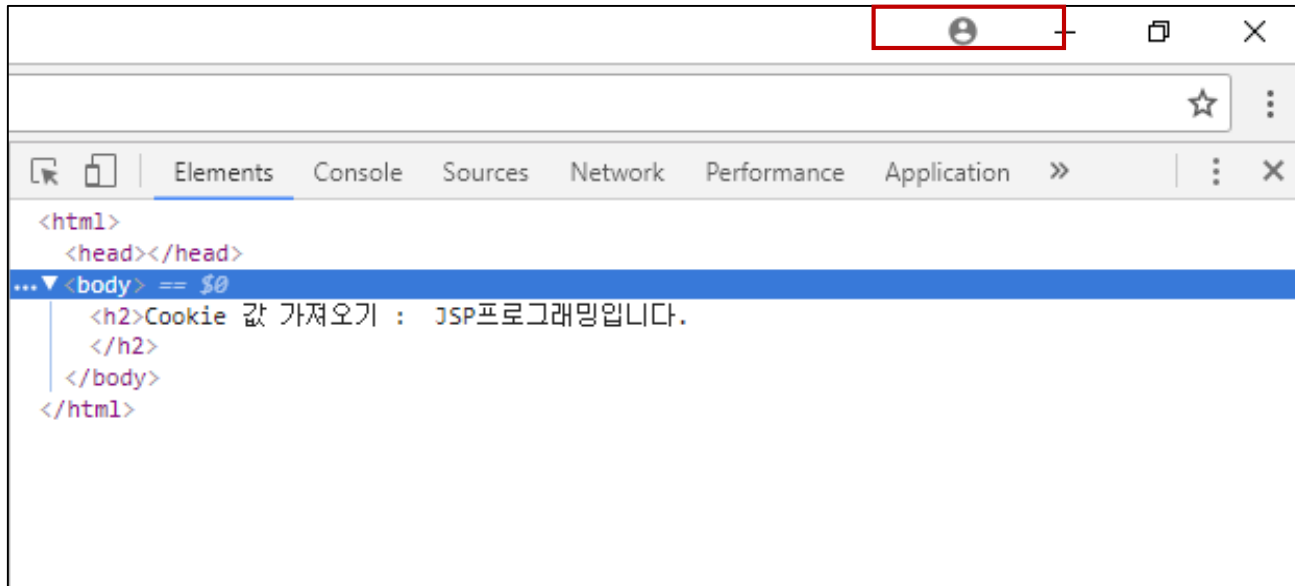
5. get으로 두 번째 서블릿을 요청하여 cookieTest로 쿠키 값을 가져와 브라우저에 출력합니다.



3. 쿠키를 이용한 웹 페이지 연동 기능

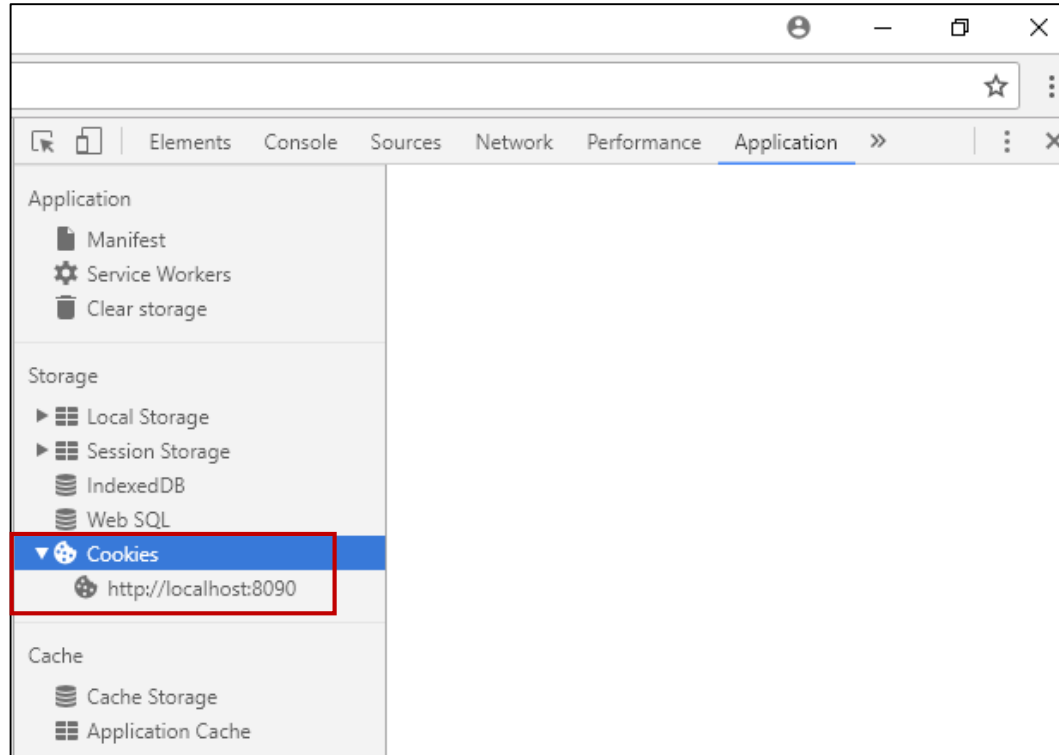
쿠키 생성 상태 확인하기

1. 크롬 브라우저를 실행하고 F12 를 눌러 디버그창을 나타냅니다. 그리고 상단 메뉴 바에서 Application을 클릭합니다.



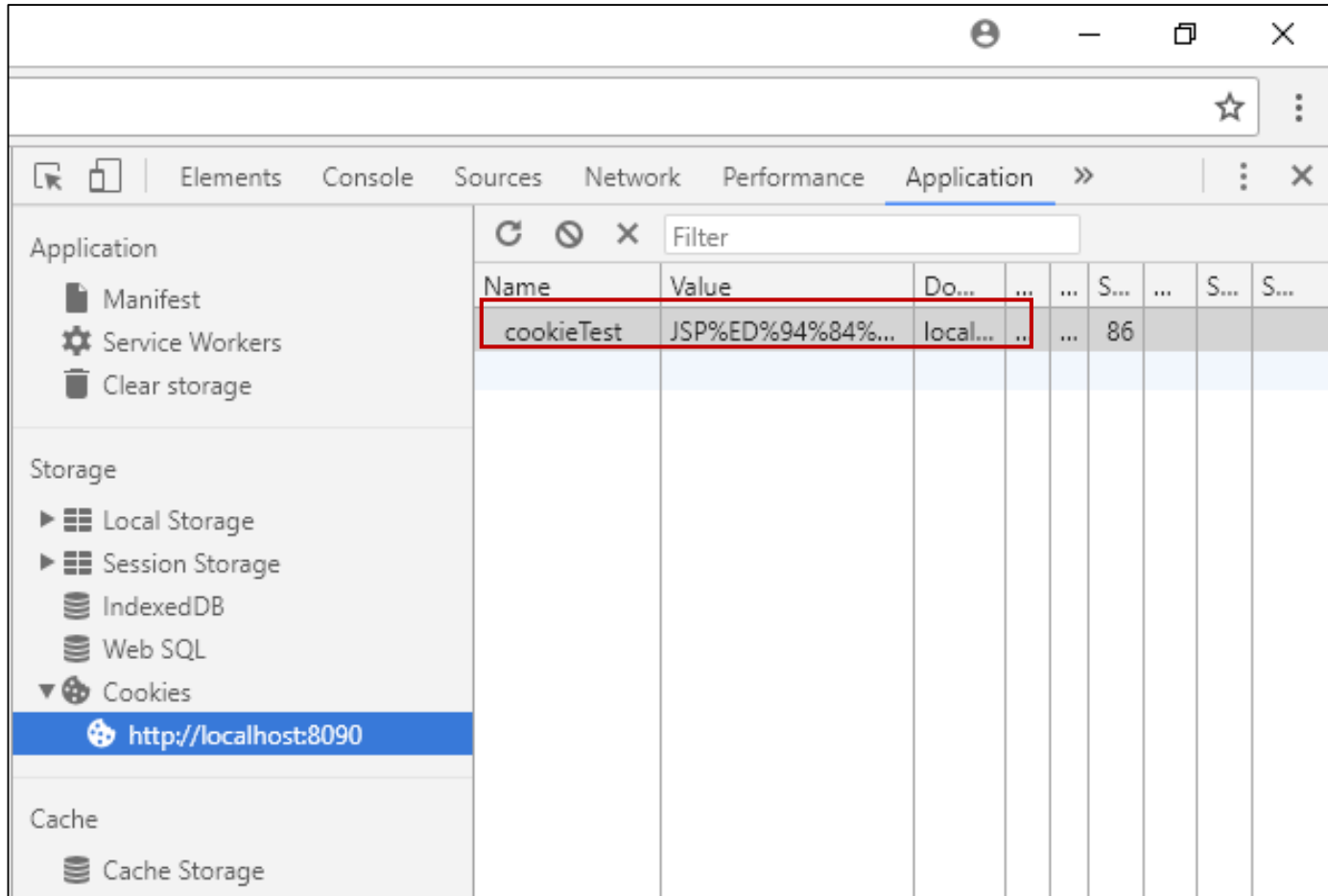
3. 쿠키를 이용한 웹 페이지 연동 기능

2. 왼쪽 메뉴에서 Cookies를 선택한 후 하위에 있는 <http://localhost:8090>을 클릭합니다.



3. 쿠키를 이용한 웹 페이지 연동 기능

3. 현재 애플리케이션에서 사용하고 있는 쿠키 정보가 표시됩니다.



3. 쿠키를 이용한 웹 페이지 연동 기능

• 3.4 세션 쿠키 사용하기

1. 다음과 같이 Cookie의 setMaxAge() 메서드를 이용해 유효 시간을 -1로 설정하면 세션 쿠키가 생성됩니다.

...

```
@WebServlet("/set")
public class SetCookieValue extends HttpServlet{
protected public void doGet(HttpServletRequest request,HttpServletResponse response)
throws ServletException, IOException {
    response.setContentType("text/html;charset=utf-8");
    PrintWriter out=response.getWriter();
    Date d=new Date();
    Cookie c=new Cookie("cookieTest",URLEncoder.encode("JSP프로그래밍입니다. ","utf-8"));
    //c.setMaxAge(24*60*60);
    c.setMaxAge(-1);
    response.addCookie(c);
    out.println("현재 시간 : "+d);
    out.println("현재 시간을 Cookie로 저장합니다.");
}
}
```

주석 처리합니다.

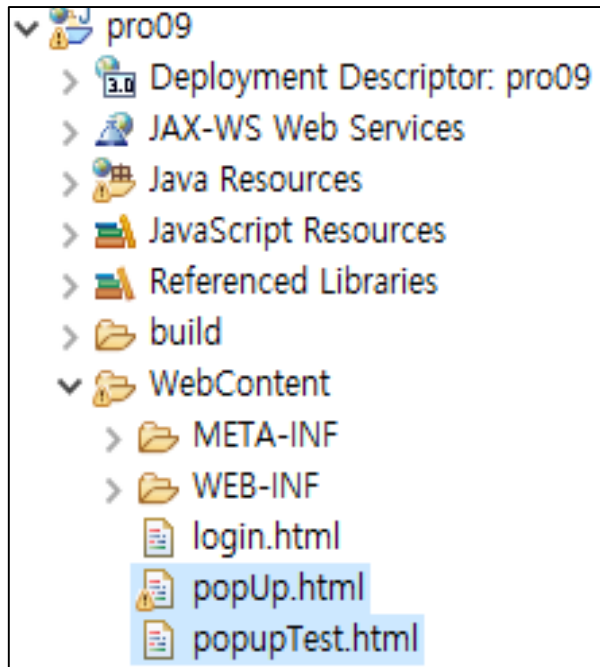
유효 시간을 음수로 지정하여 Session 쿠키를 만듭니다.

2. 톱캣을 재실행합니다. 출력 결과는 앞의 쿠키 예제(9.3.3절)와 같습니다.

3. 쿠키를 이용한 웹 페이지 연동 기능

- 3.5 쿠키 이용해 팝업창 제한하기

1. popUp.html, popupTest.html 이렇게 두 개의 html 파일을 준비합니다.



3. 쿠키를 이용한 웹 페이지 연동 기능

2. 먼저 popupTest.html을 다음과 같이 작성합니다.

```
<html>
<head>
  <meta charset="UTF-8">
  <title> 자바스크립트에서 쿠키 사용 </title>
  <script type = "text/javascript">
    window.onload = pageLoad;
    function pageLoad(){
      notShowPop =getCookieValue();
      if(notShowPop!="true"){
        window.open("popUp.html","pop","width=400,height=500,history=no,
          resizable=no,status=no,scrollbars=yes,menubar=no");
      }
    }
    function getCookieValue(){
      var result="false";
      if(document.cookie != ""){
        cookie = document.cookie.split(";");
        for(var i=0; i<cookie.length;i++){
          element=cookie[i].split("=");
          value=element[0];
          value=value.replace(/^\s*/, '');
          if(value=="notShowPop"){
            result= element[1];
          }
        }
      }
    }
  </script>
</head>
</html>
```

브라우저에 웹 페이지가 로드될 때 pageLoad() 함수를 호출하여 실행합니다.

notShowPop의 쿠키 값을 getCookieValue()를 호출하여 얻습니다.

notShowPop의 값이 true가 아니면 팝업창을 나타냅니다.

document의 cookie 속성으로 쿠키 정보를 문자열로 가져온 후 세미콜론(;)으로 분리해 각각의 쿠키를 얻습니다.

정규식을 이용해 쿠키 이름 문자열의 공백(ws)을 제거합니다.

쿠키 이름이 notShowPop이면 해당하는 쿠키 값을 가져와 반환합니다.

3. 쿠키를 이용한 웹 페이지 연동 기능

```
    }  
  }  
  return result;  
}  
  
function deleteCookie(){  
  document.cookie = "notShowPop=" + "false" + ";path=/; expires=-1" ;  
}  
</script>  
</head>  
<body>  
  <form>  
    <input type=button value="쿠키삭제" onClick="deleteCookie()" >  
  </form>  
</body>  
</html>
```

'쿠키삭제' 클릭 시 호출됩니다.
notShowPop 쿠키 값을 false로 설정합니다.

쿠키를 삭제합니다.

3. 쿠키를 이용한 웹 페이지 연동 기능

3. popUp.html에서는 오늘 더 이상 팝업창 띄우지 않기에 체크하면 자바스크립트 함수인 setPopUpStart() 함수를 호출해 notShowPop의 값을 true로 설정하여 재접속 시 팝업창을 나타내지 않도록 설정합니다.

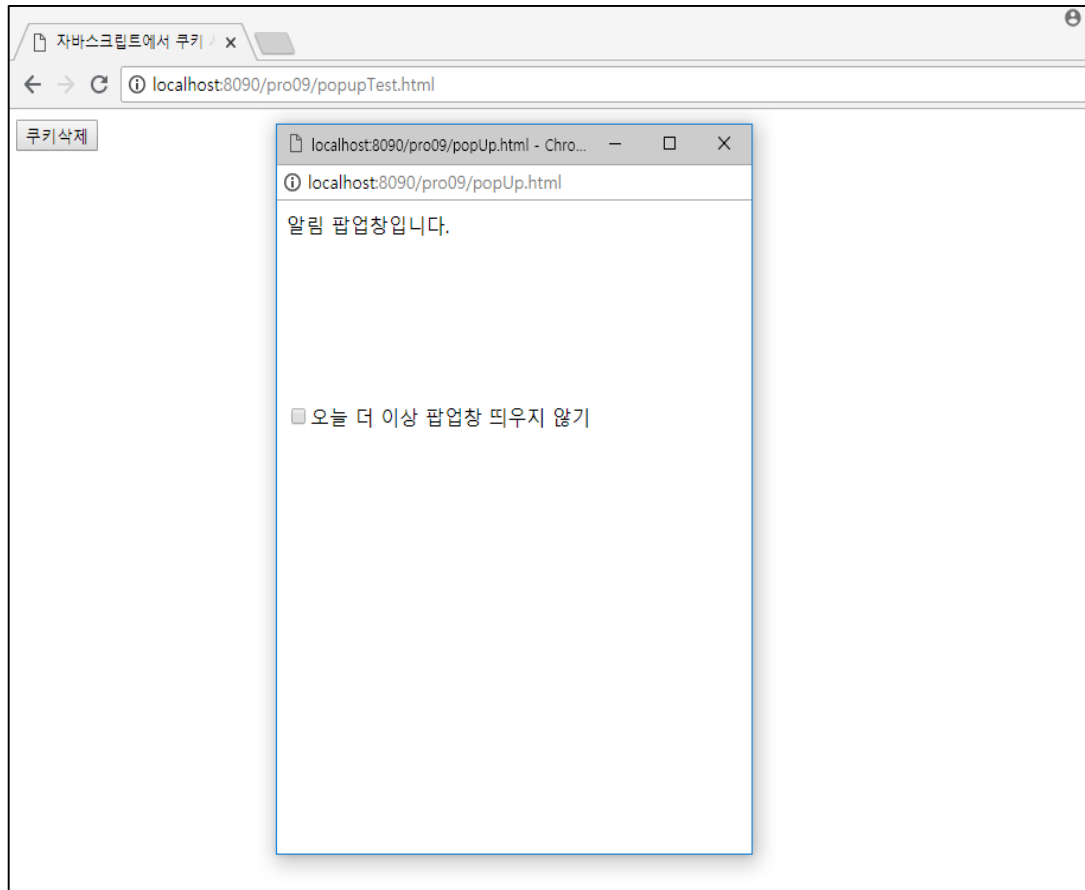
```
<html>
<head>
<meta charset="UTF-8">
<script type="text/javascript">
function setPopUpStart(obj){
    if(obj.checked==true){
        var expireDate = new Date();
        expireDate.setMonth(expireDate.getMonth() + 1);
        document.cookie = "notShowPop=" + "true" + ";path=/; expires=" +
            expireDate.toGMTString();
        window.close();
    }
}
</script>
</head>
<body>
    알림 팝업창입니다.
    <br><br><br><br><br><br><br>
    <form>
        <input type="checkbox" onClick="setPopUpStart(this)">오늘 더 이상 팝업창 띄우지 않기
    </form>
</body>
</html/>
```

쿠키 유효 시간을 한 달로 설정합니다.

오늘 더 이상 팝업창 띄우지 않기에 체크하면 notShowPop 쿠키 값을 true로 설정하여 재접속 시 팝업창을 나타내지 않습니다.

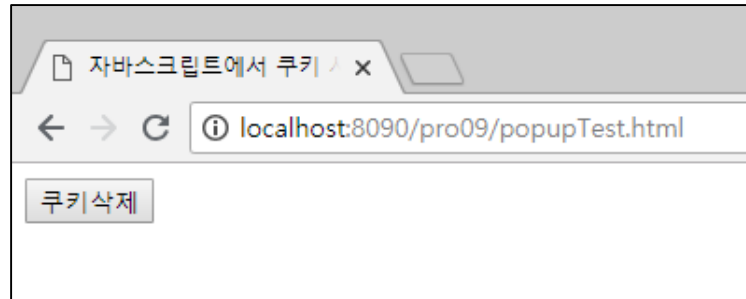
3. 쿠키를 이용한 웹 페이지 연동 기능

4. 브라우저에 최초 접속 시 팝업창을 나타냅니다.

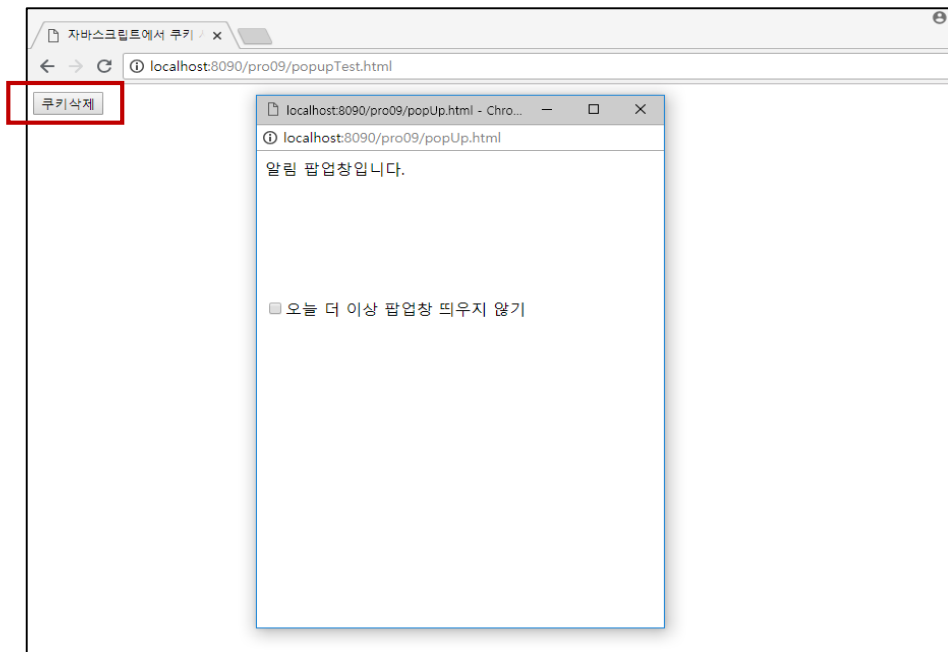


3. 쿠키를 이용한 웹 페이지 연동 기능

5. 오늘 더 이상 팝업창 띄우지 않기에 체크하고 재요청하면 더 이상 팝업창이 나타나지 않습니다.



6. 쿠키삭제를 클릭한 후 재요청하면 다시 팝업창이 나타납니다.



4. 세션을 이용한 웹 페이지 연동 기능

세션(Session)

➤ 웹 페이지들 사이의 공유 정보를 서버의 메모리에 저장해 놓고 사용하는 방법

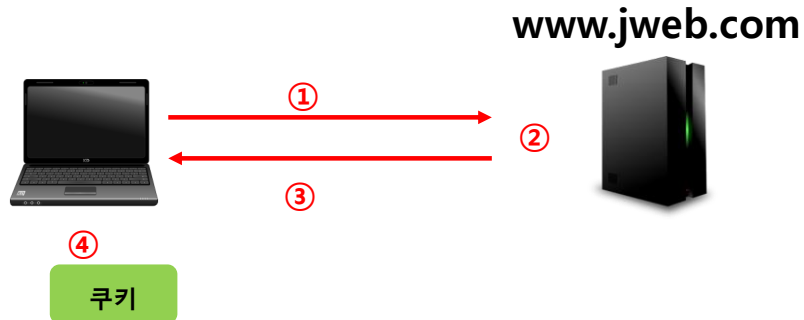
세션(Session)의 특징

- 정보가 서버의 메모리에 저장
- 브라우저의 세션 연동은 세션 쿠키를 이용함
- 쿠키보다 보안에 유리
- 서버에 부하를 줄 수 있음
- 브라우저(사용자)당 한 개의 세션(세션 id)이 생성됨
- 세션은 유효 시간을 가짐(기본 유효 시간은 30분)
- 로그인 상태 유지 기능이나 쇼핑물의 장바구니 담기 기능 등에 주로 사용됨

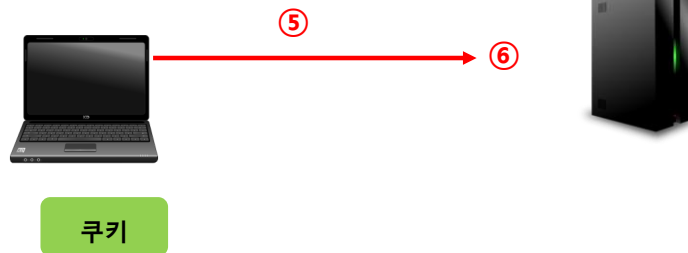
4. 세션을 이용한 웹 페이지 연동 기능

• 4.1 세션 기능 실행 과정

(a) 최초 사이트 접속 시



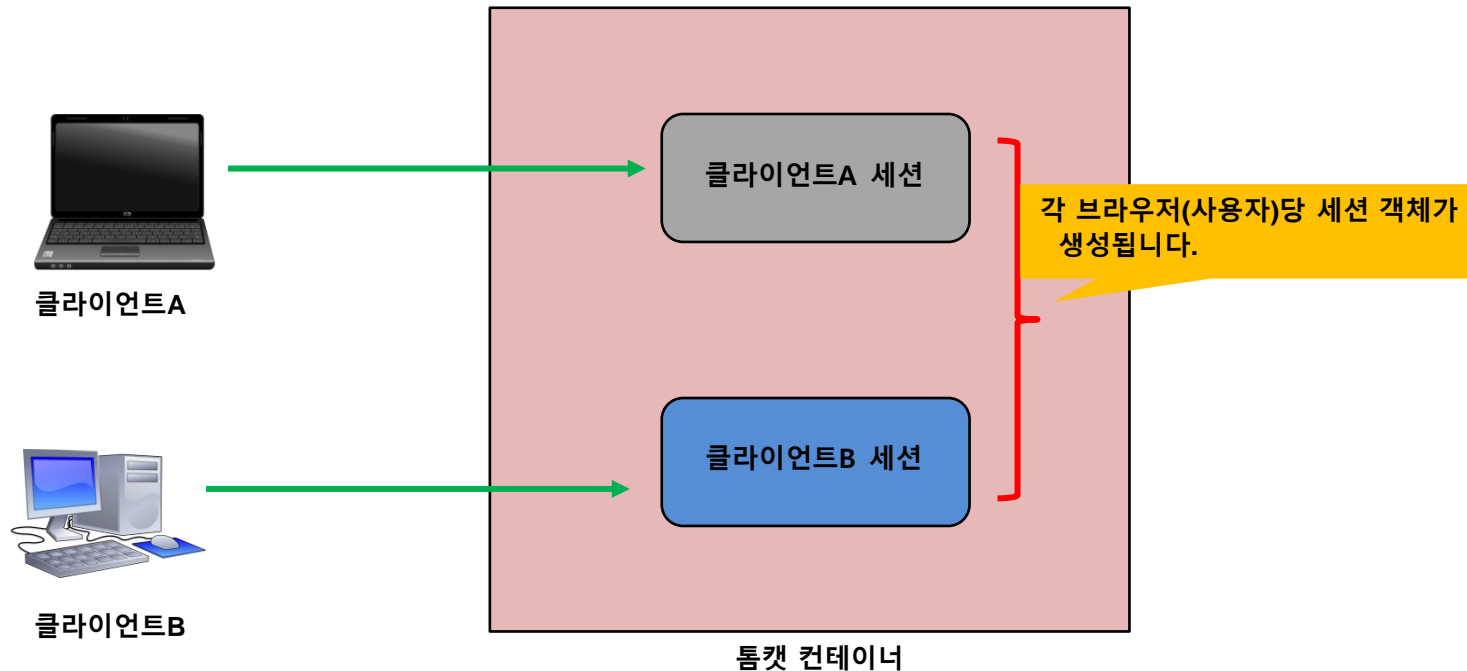
(b) 재 접속 시



- ① 브라우저로 사이트에 접속합니다.
- ② 서버는 접속한 브라우저에 대한 세션 객체를 생성합니다.
- ③ 서버는 생성된 세션 id를 클라이언트 브라우저에 응답합니다.
- ④ 브라우저는 서버로부터 받은 세션 id를 브라우저가 사용하는 메모리의 세션 쿠키에 저장합니다(쿠키 이름은 sessionId).
- ⑤ 브라우저가 재 접속하면 브라우저는 세션 쿠키에 저장된 세션 id를 서버에 전달합니다.
- ⑥ 서버는 전송된 세션 id를 이용해 해당 세션에 접근하여 작업을 수행합니다.

4. 세션을 이용한 웹 페이지 연동 기능

각 브라우저에 대한 세션 생성 상태



❖ 각 브라우저(사용자) 당 하나의 세션이 생성됨

4. 세션을 이용한 웹 페이지 연동 기능

- 4.2 세션 API의 특징과 기능

세션 얻는 방법

- 서블릿에서 HttpSession 클래스 객체를 생성해서 사용함
- HttpSession 객체는 HttpServletRequest의 getSession() 메서드를 호출해서 얻음

getSession() 종류

- getSession() : 기존의 세션 객체가 존재하면 반환하고, 없으면 새로 생성
- getSession(true) : 기존의 세션 객체가 존재하면 반환하고, 없으면 새로 생성
- getSession(false): 기존의 세션 객체가 존재하면 반환하고, 없으면 null을 반환

4. 세션을 이용한 웹 페이지 연동 기능

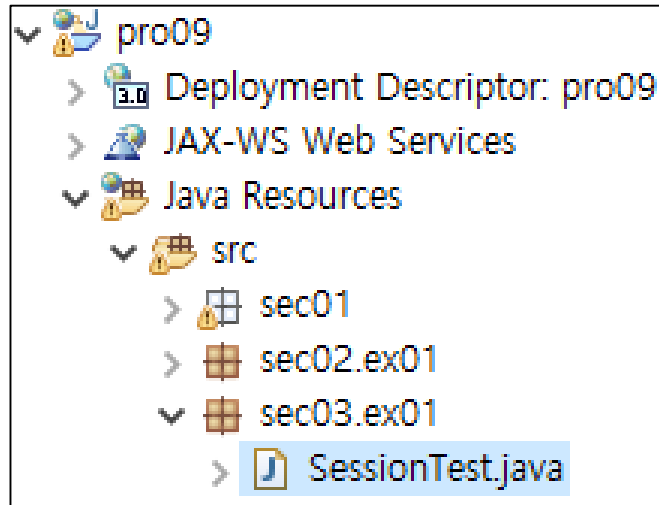
HttpSession 클래스의 여러가지 메서드

반환타입	메서드	설명
Object	getAttribute(String name)	속성 이름이 name인 속성 값을 Object 타입으로 반환합니다. 해당되는 속성 이름이 없을 경우 null 값을 반환합니다.
Enumeration	getAttributeNames()	세션 속성 이름들을 Enumeration 객체 타입으로 반환합니다.
long	getCreationTime()	1970년 1월 1일 0시 0초를 기준으로 현재 세션이 생성된 시간 까지 경과한 시간을 계산하여 1/1000초 값으로 반환합니다.
String	getId()	세션에 할당된 고유 식별자를 String 타입으로 반환합니다.
int	getMaxInactiveInterval()	현재 생성된 세션을 유지하기 위해 설정된 세션 유지 시간을 int 타입으로 반환합니다.
void	invalidate()	현재 생성된 세션을 소멸합니다.
boolean	isNew()	최초로 생성된 세션인지 기존에 생성되어 있었던 세션인지 판별 합니다.
void	removeAttribute (String name)	세션 속성 이름이 name인 속성을 제거합니다.
void	setAttribute (String name, Object value)	세션 속성 이름이 name인 속성에 속성 값으로 value를 할당합니 다.
void	setMaxInactiveInterval (int interval)	세션을 유지하기 위한 세션 유지 시간을 초 단위로 설정합니다.

4. 세션을 이용한 웹 페이지 연동 기능

- 4.3 서블릿에서 세션 API 이용하기

1. 다음과 같이 세션 테스트를 위한 실습 파일인 SessionTest 클래스를 준비합니다.



4 세션을 이용한 웹 페이지 연동 기능

2. SessionTest 클래스를 다음과 같이 작성합니다.

```
package sec03.ex01;

...
@WebServlet("/sess")
public class SessionTest extends HttpServlet{
    protected public void doGet(HttpServletRequest request , HttpServletResponse response )
        throws ServletException, IOException {
        response.setContentType("text/html;charset=utf-8");
        PrintWriter out = response.getWriter();
        HttpSession session = request.getSession();
        out.println("세션 아이디: " + session.getId() + "<br>");
        out.println("최초 세션 생성 시각: " + new Date(session.getCreationTime()) + "<br>");
        out.println("최근 세션 접근 시각 : " +
            new Date(session.getLastAccessedTime()) + "<br>");
        out.println("세션 유효 시간 : " + session.getMaxInactiveInterval() + "<br>");
        if(session.isNew()){
            out.print("새 세션이 만들어졌습니다.");
        }
    }
}
```

getSession()을 호출하여 최초 요청 시 세션 객체를 새로 생성하거나 기존 세션을 반환합니다.

생성된 세션 객체의 id를 가져옵니다.

최초 세션 객체 생성 시간을 가져옵니다.

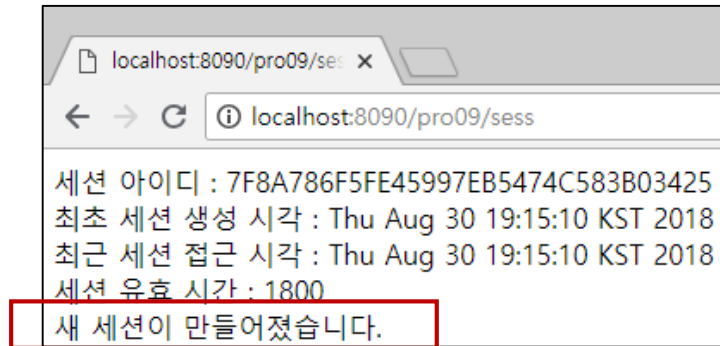
세션 객체에 가장 최근에 접근한 시간을 가져옵니다.

세션 객체의 유효 시간을 가져옵니다.

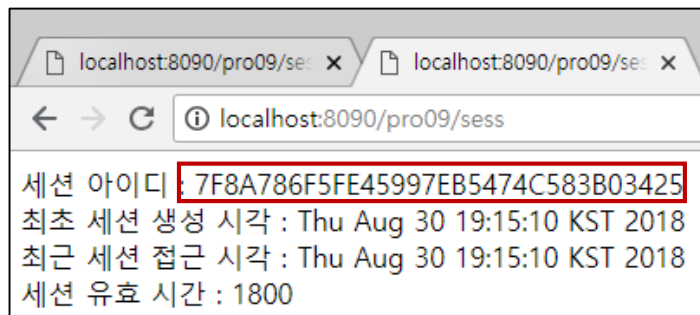
최초 생성된 세션인지 판별합니다.

4. 세션을 이용한 웹 페이지 연동 기능

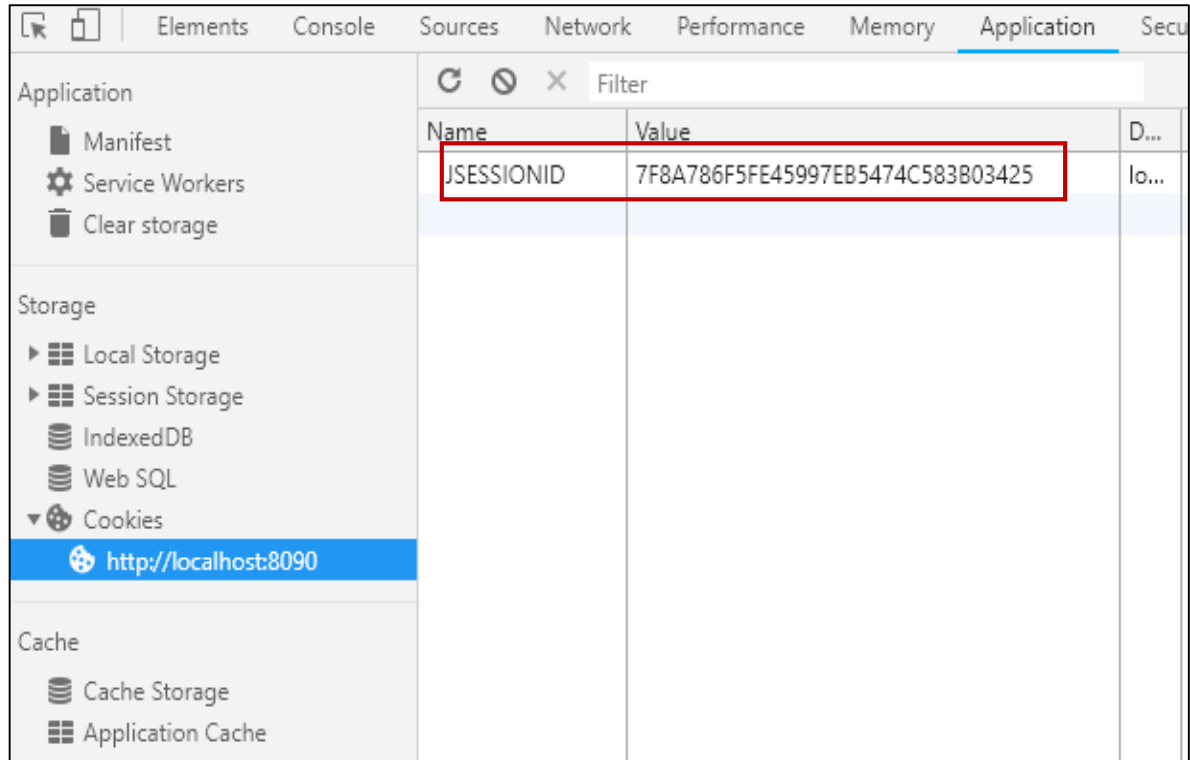
3. 브라우저에서 최초 요청 시 생성된 세션 객체에 할당된 세션 id와 여러 가지 정보를 출력합니다.
최초 생성된 세션이므로 "새 세션이 만들어졌습니다."라는 메시지가 출력됩니다.



4. 같은 브라우저에서 다른 탭을 열고 요청하면 같은 세션 id를 출력하므로 최초 생성된 세션을 재사용합니다.
따라서 "새 세션이 만들어졌습니다."라는 메시지는 출력되지 않습니다.



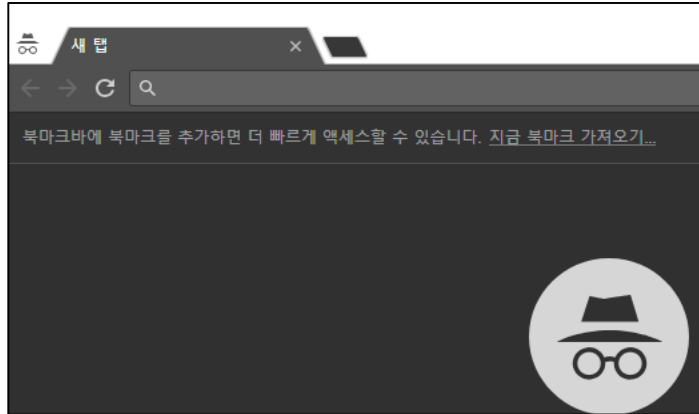
4. 세션을 이용한 웹 페이지 연동 기능



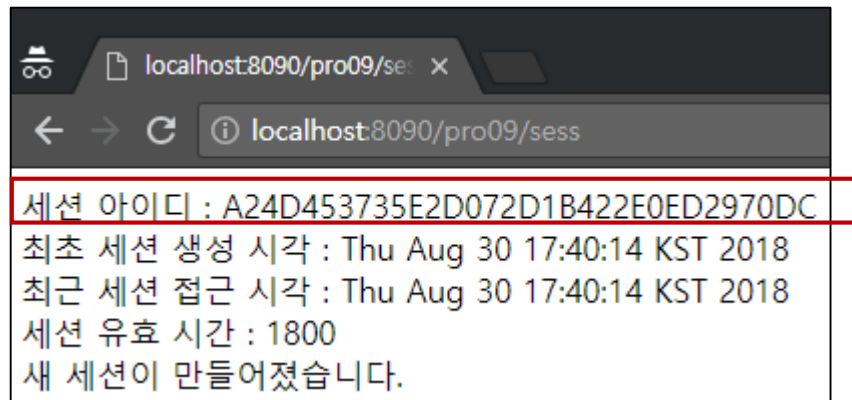
4. 세션을 이용한 웹 페이지 연동 기능

4.4 다른 브라우저에서 새 세션 만들기

브라우저에서 Ctrl+ Shift + N 키를 눌러 시크릿 모드의 크롬을 실행합니다

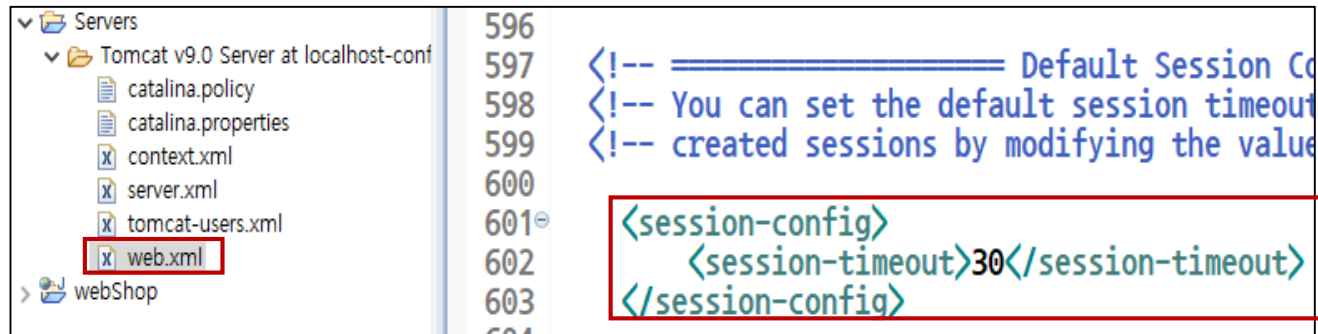


주소창에서 /sess로 요청하면 새로운 세션을 생성한 후 다른 세션 id를 출력합니다.



4. 세션을 이용한 웹 페이지 연동 기능

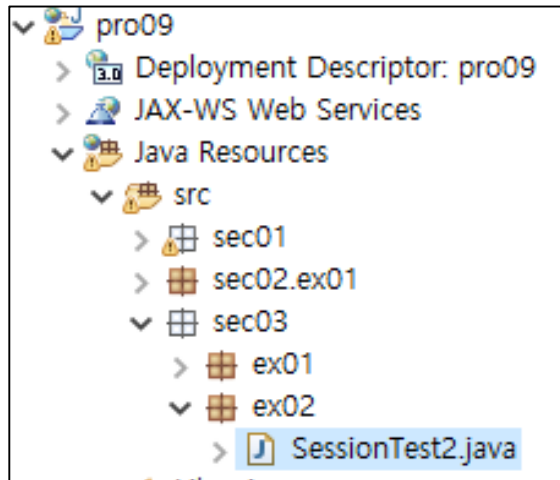
다음과 같이 톰캣 컨테이너의 web.xml에 세션 기본 유효 시간이 설정된 것을 확인할 수 있습니다.



4. 세션을 이용한 웹 페이지 연동 기능

세션 유효 시간 재설정하기

1. 다음과 같이 SessionTest2 클래스를 준비합니다.



4. 세션을 이용한 웹 페이지 연동 기능

2. SessionTest2 클래스를 다음과 같이 작성합니다. setMaxInactiveInterval() 메서드를 이용해 세션 유효 시간을 5초로 재설정합니다.

```
package sec03.ex02;

...

@WebServlet("/sess2")
public class SessionTest2 extends HttpServlet{
    protected public void doGet(HttpServletRequest request , HttpServletResponse response )
        throws ServletException, IOException {
        response.setContentType("text/html;charset=utf-8");
        PrintWriter out = response.getWriter();
        HttpSession session = request.getSession();
        out.println("세션 아이디: "+session.getId()+"<br>");
        out.println("최초 세션 생성 시각: " + new Date(session.getCreationTime())+"<br>");
        out.println("최근 세션 접근 시각 : "+
            new Date(session.getLastAccessedTime())+"<br>");
        out.println("기본 세션 유효 시간 : " + session.getMaxInactiveInterval()+"<br>");
        session.setMaxInactiveInterval(5);
        out.println("세션 유효 시간 : " + session.getMaxInactiveInterval()+"<br>");
        if(session.isNew()){
            out.print("새 세션이 만들어졌습니다.");
        }
    }
}
```

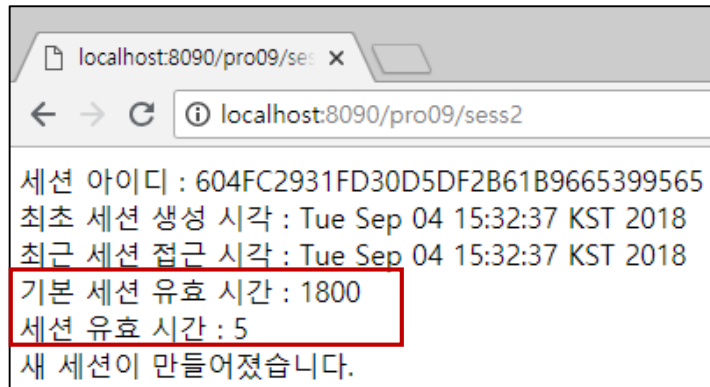
톰캣의 기본 세션 유효 시간을 출력합니다.

세션의 유효 시간을 5초로 설정합니다.

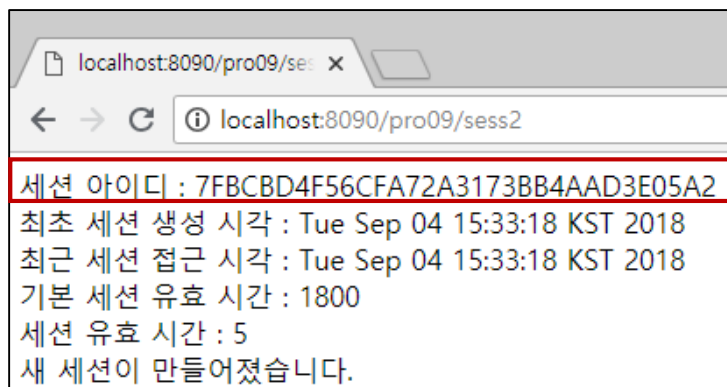
유효 시간을 재설정한 후 세션 유효 시간을 출력합니다.

4. 세션을 이용한 웹 페이지 연동 기능

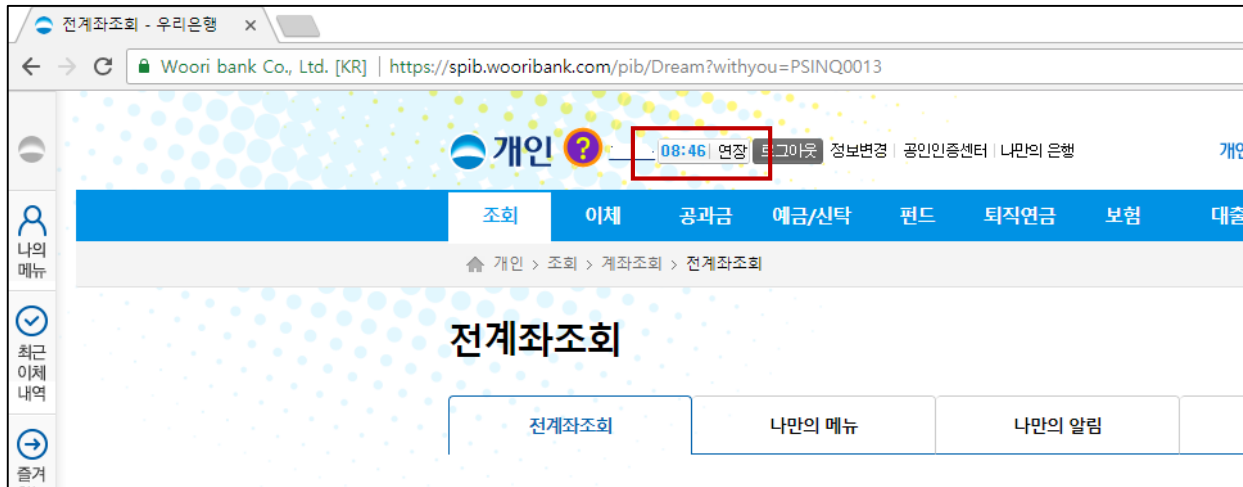
3. 최초에 /sess2로 요청하여 설정 전 유효 시간과 설정 후 유효 시간을 출력합니다.



4. 5초가 지난 후 같은 브라우저에서 재요청하면 다시 새 세션이 생성됩니다.



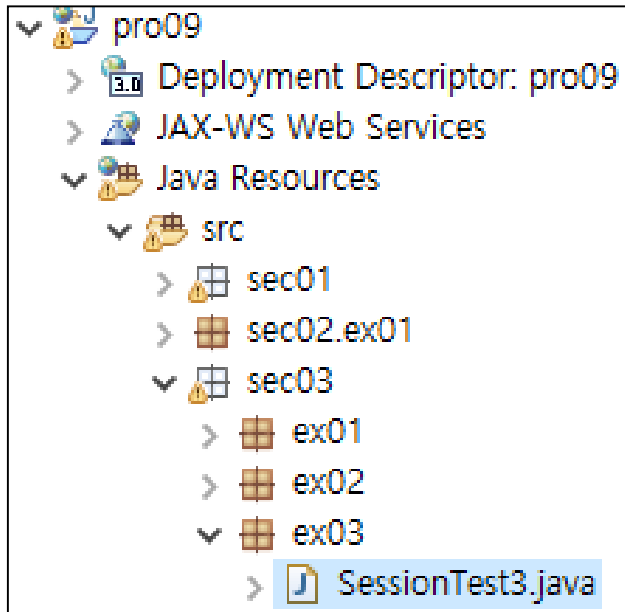
4. 세션을 이용한 웹 페이지 연동 기능



4. 세션을 이용한 웹 페이지 연동 기능

강제로 세션 삭제하기

1. 다음과 같이 실습 파일을 준비합니다.



4. 세션을 이용한 웹 페이지 연동 기능

2. SessionTest3 클래스를 다음과 같이 작성합니다. invalidate() 메서드를 이용해 강제로 세션을 삭제합니다.

```
package sec03.ex03;

...

@WebServlet("/sess3")
public class SessionTest3 extends HttpServlet{

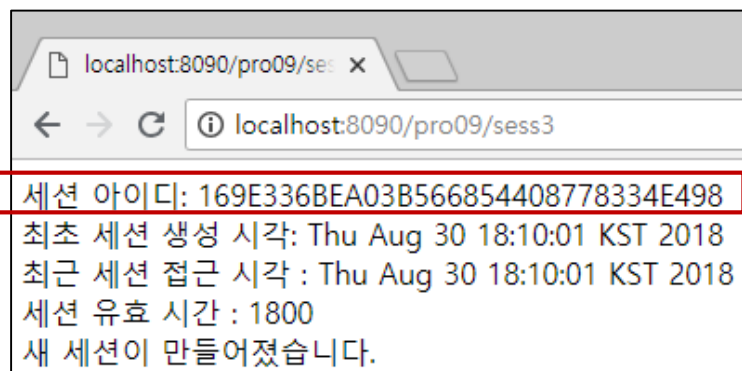
protected public void doGet(HttpServletRequest request , HttpServletResponse response )
    throws ServletException, IOException {
    response.setContentType("text/html;charset=utf-8");
    PrintWriter out = response.getWriter();
    HttpSession session = request.getSession();
    out.println("세션 아이디: "+session.getId()+"<br>");
    out.println("최초 세션 생성 시각: "+ new Date(session.getCreationTime())+"<br>");
    out.println("최근 세션 접근 시각 : "+
                new Date(session.getLastAccessedTime())+"<br>");
    out.println("세션 유효 시간 : "+ session.getMaxInactiveInterval()+"<br>");
    if(session.isNew()){
        out.print("새 세션이 만들어졌습니다.");
    }
    session.invalidate();
}

}
```

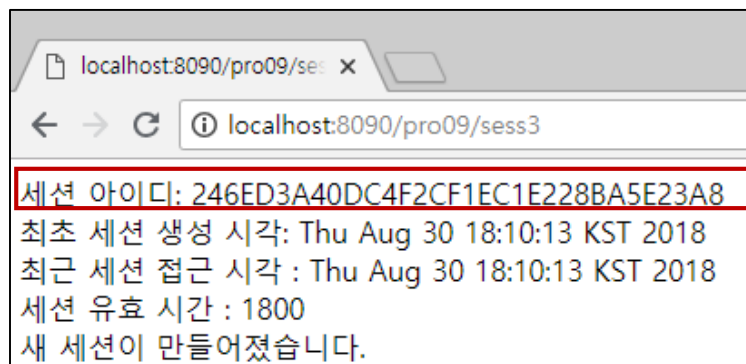
invalidate()를 호출하여 생성된 세션 객체를 강제로 삭제합니다.

4. 세션을 이용한 웹 페이지 연동 기능

3. 최초 요청 시 새 세션이 생성된 후 invalidate() 메서드가 호출되므로 바로 소멸됩니다.



4. 재 요청 시 다른 세션이 생성됩니다.

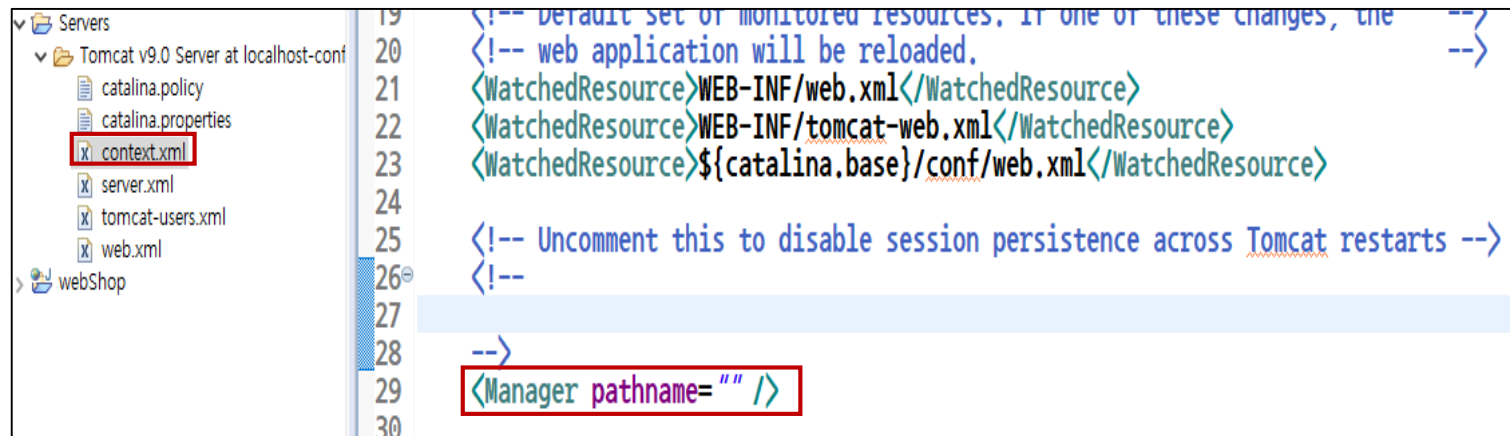


4. 세션을 이용한 웹 페이지 연동 기능

• 4.5 세션을 이용한 로그인 정보 바인딩 실습

➤ 로그인 시 로그인 상태를 세션에 저장해서 사용하면 편리함

1. 실습하기 전에 톰캣이 종료된 후에도 세션이 메모리에서 삭제되지않는 경우가 있으므로 톰캣 설정 파일인 context.xml을 열어 <Manager pathname="" /> 태그의 주석을 해제해야 합니다.

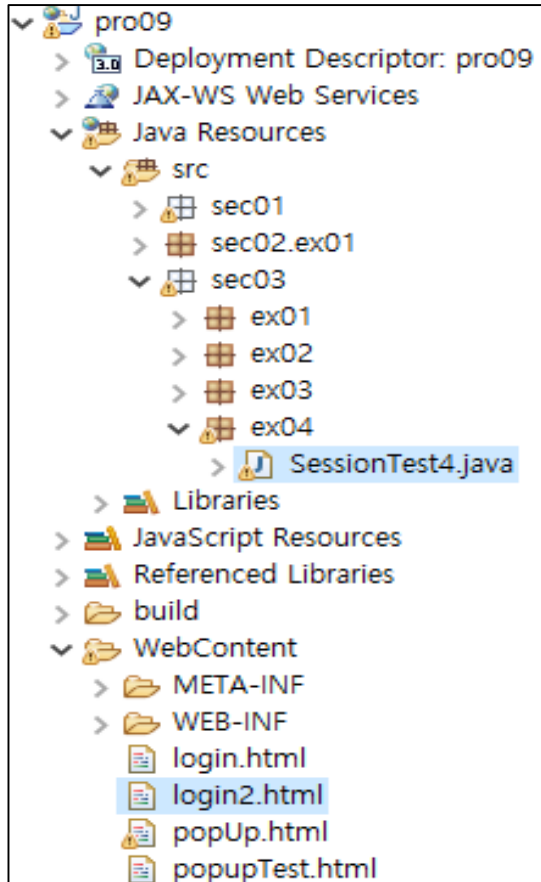


The screenshot shows the Tomcat configuration files in the 'Servers' panel on the left. The 'context.xml' file is selected and highlighted with a red box. The main panel displays the content of 'context.xml' with line numbers 19 through 30. The code includes comments about monitored resources and a commented-out <Manager> tag. The <Manager> tag is highlighted with a red box.

```
19 <!-- Default set of monitored resources. If one of these changes, the
20 <!-- web application will be reloaded. -->
21 <WatchedResource>WEB-INF/web.xml</WatchedResource>
22 <WatchedResource>WEB-INF/tomcat-web.xml</WatchedResource>
23 <WatchedResource>${catalina.base}/conf/web.xml</WatchedResource>
24
25 <!-- Uncomment this to disable session persistence across Tomcat restarts -->
26 <!--
27
28 -->
29 <Manager pathname="" />
30
```

4 .세션을 이용한 웹 페이지 연동 기능

2. 다음과 같이 실습 파일을 준비합니다.



4. 세션을 이용한 웹 페이지 연동 기능

3. 로그인창에서 ID와 비밀번호를 입력한 후 서블릿으로 전송할 수 있도록 login2.html 파일을 작성합니다.

```
<!DOCTYPE html>
<html>
<head>...</head>
<body>
  <form name="frmLogin" method="post " action="login" encType="UTF-8">
    아이디 :<input type="text" name="user_id"><br>
    비밀번호:<input type="password" name="user_pw"><br>
    <input type="submit" value="로그인">
    <input type="reset" value="다시 입력">
  </form>
</body>
</html>
```

4. 세션을 이용한 웹 페이지 연동 기능

4. SessionTest4 클래스를 다음과 같이 작성합니다.

```
...
private public void doHandle(HttpServletRequest request , HttpServletResponse response )
    throws ServletException, IOException {
    request.setCharacterEncoding("utf-8");
    response.setContentType("text/html;charset=utf-8");
    PrintWriter out = response.getWriter();
    HttpSession session = request.getSession();
    String user_id = request.getParameter("user_id");
    String user_pw = request.getParameter("user_pw");
    if (session.isNew()){
        if(user_id != null){
            session.setAttribute("user_id", user_id);
            out.println("<a href='login'>로그인 상태 확인</a>");
        }else {
            out.print("<a href='login2.html'>다시 로그인 하세요!!</a>");
            session.invalidate();
        }
    }else {
        user_id = (String) session.getAttribute("user_id");
        if (user_id != null && user_id.length() != 0) {
            out.print("안녕하세요 " + user_id + "님!!!");
        } else {
            out.print("<a href='login2.html'>다시 로그인 하세요!!</a>");
            session.invalidate();
        }
    }
}
```

로그인창에서 전송된 ID와 비밀번호를 가져옵니다.

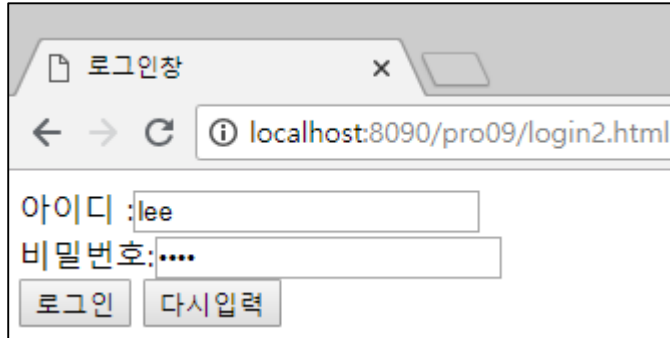
최초 요청 시 수행합니다.

로그인창에서 서버로 요청했다면 ID가 null이 아니므로 세션에 ID를 바인딩합니다.

재요청 시 세션에서 ID를 가져와 이전에 로그인했는지 여부를 확인합니다.

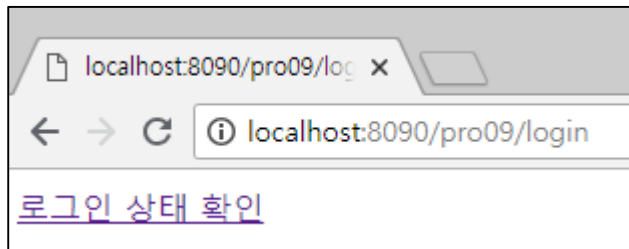
4. 세션을 이용한 웹 페이지 연동 기능

5. 로그인창 요청 후 ID와 비밀번호를 입력하고 전송합니다.



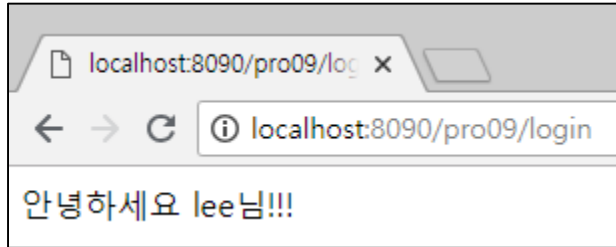
A screenshot of a web browser window. The title bar shows a tab labeled '로그인창' (Login Window). The address bar displays 'localhost:8090/pro09/login2.html'. The page content includes a login form with two input fields: '아이디 :lee' (ID) and '비밀번호:....' (Password). Below the fields are two buttons: '로그인' (Login) and '다시입력' (Re-enter).

6. 최초 로그인 시 세션에 ID를 바인딩합니다.

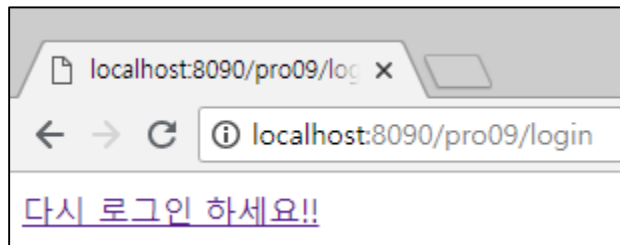


4. 세션을 이용한 웹 페이지 연동 기능

7. 다시 로그인 상태 확인을 클릭해 /login으로 재요청하면 현재 로그인 상태를 출력합니다.



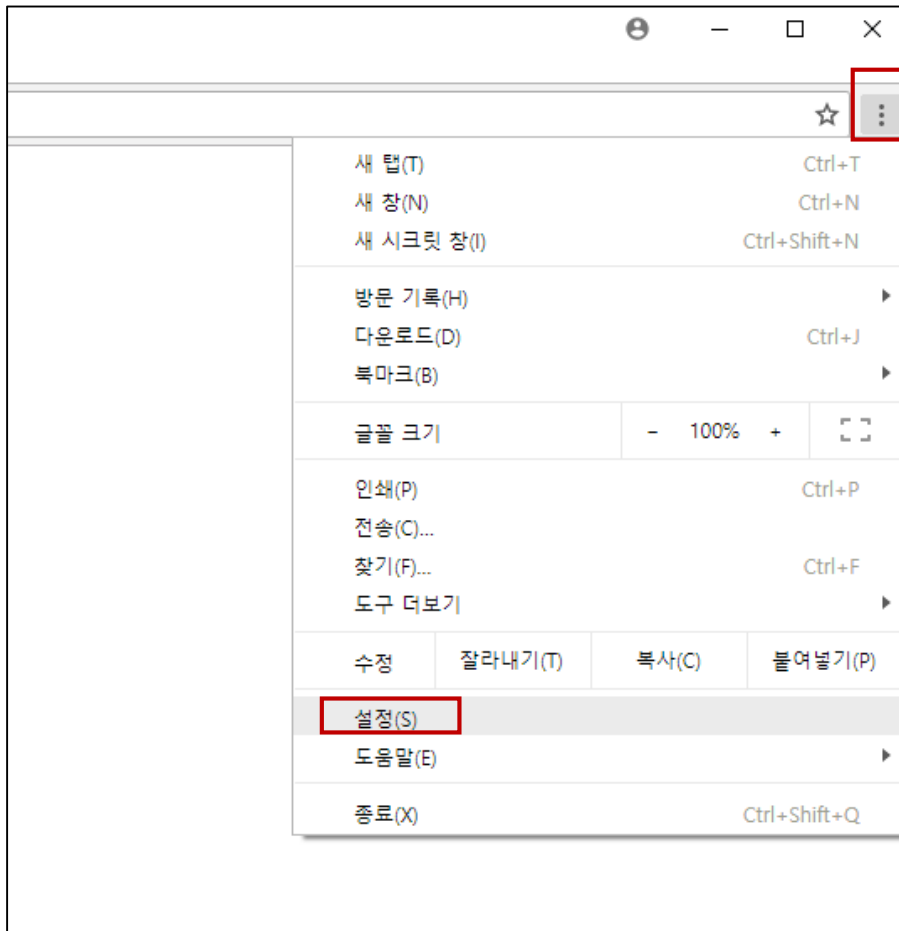
8. 톰캣 재실행 후 로그인 창을 거치지 않고 바로 /login으로 요청하면 세션에 ID가 없으므로 "다시 로그인 하세요!!"라는 메시지가 출력됩니다.



5. encodeURIComponent() 사용법

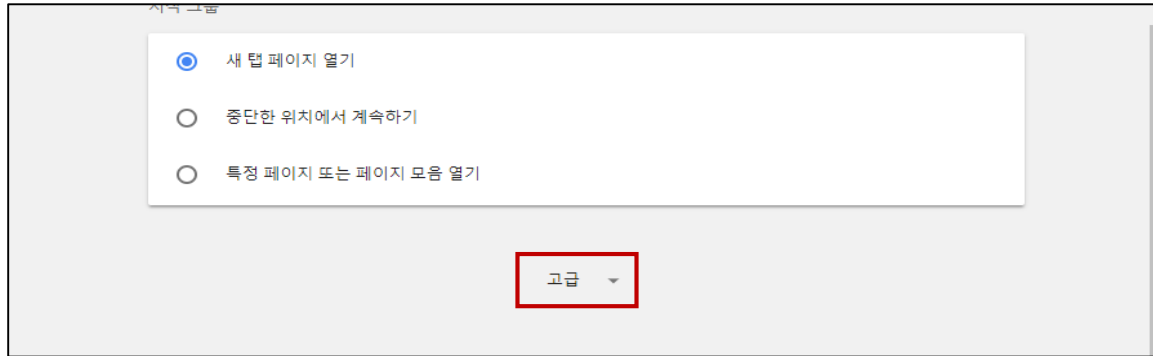
- 5.1 브라우저에서 쿠키 사용 금지하기

1. 크롬 브라우저를 실행하고 오른쪽 상단에서 더 보기 아이콘 클릭 후 설정을 클릭합니다.



5. encodeURIComponent() 사용법

2. 하단에서 고급을 클릭합니다.

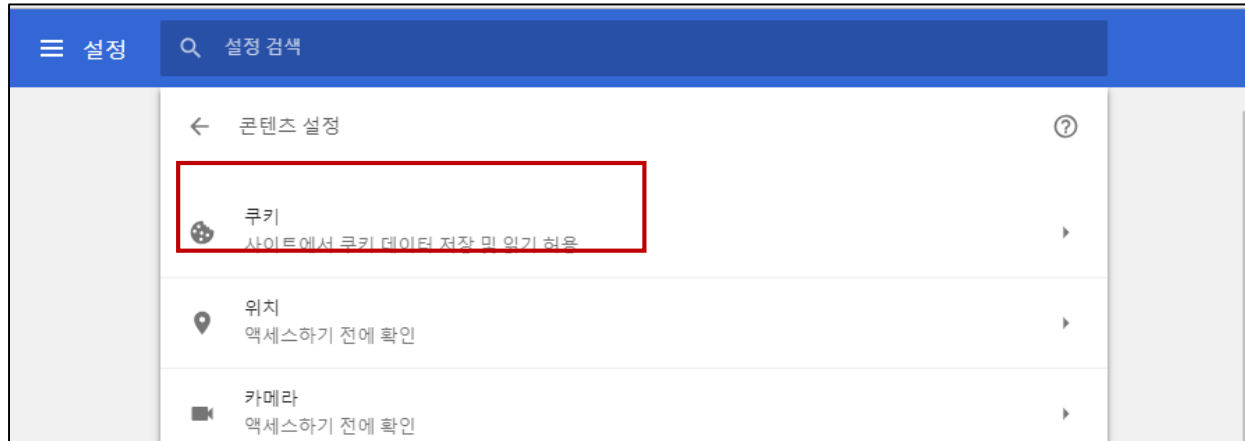


3. '개인정보 보호 및 보안'에서 콘텐츠 설정을 클릭합니다.



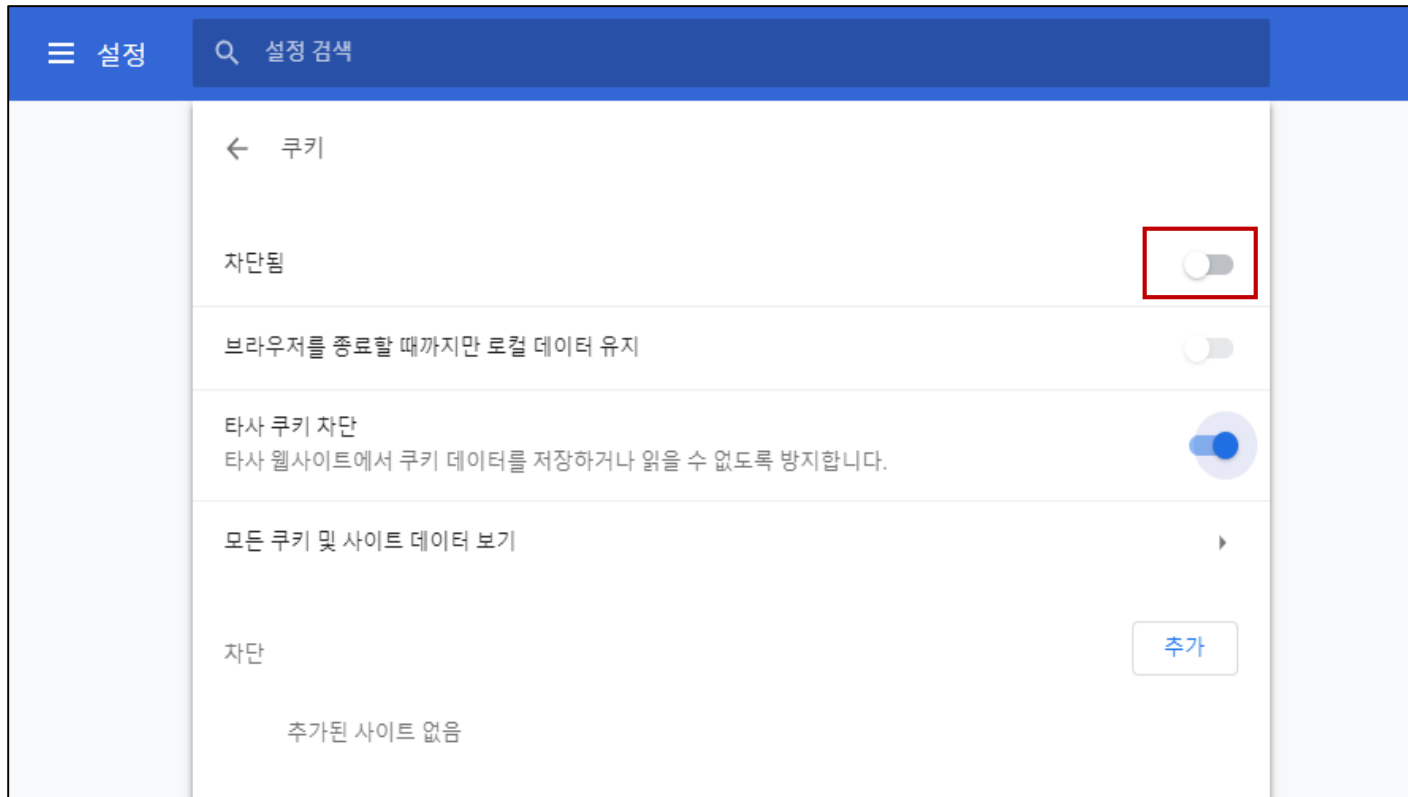
5. encodeURIComponent() 사용법

4. 쿠키를 클릭해 '사이트에서 쿠키 데이터 저장 및 읽기 허용'을 차단하도록 합니다.



5 encodeURIComponent() 사용법

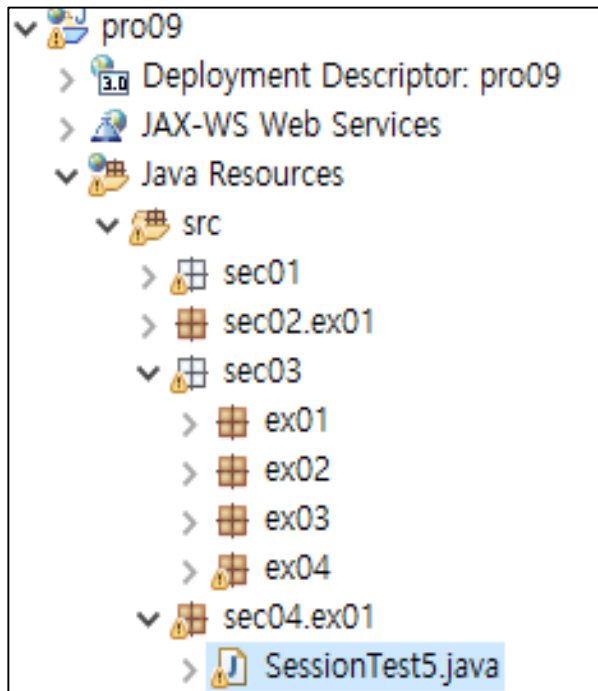
5. '차단됨' 옆의 옵션을 클릭하여 차단으로 설정합니다.



5. encodeURL() 사용법

- 5.2 encodeURL() 메서드를 이용한 세션 사용 실습

1. 다음과 같이 실습 파일을 준비합니다.



5. encodeURL() 사용법

2. SessionTest5 클래스를 다음과 같이 작성합니다.

```
package sec04.ex01;

...
@WebServlet("/login")
public class SessionTest5 extends HttpServlet{

protected public void doGet(HttpServletRequest request , HttpServletResponse response )
    throws ServletException, IOException {
    doHandle(request, response);
}

protected public void doPost(HttpServletRequest request , HttpServletResponse response )
    throws ServletException, IOException {
    doHandle(request, response);
}
```

5. encodeURL() 사용법

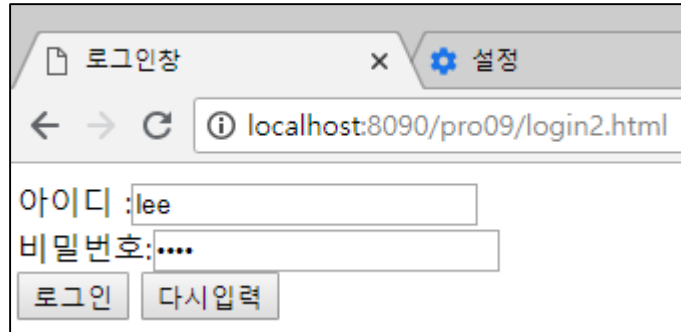
```
private public void doHandle(HttpServletRequest request , HttpServletResponse response )
throws ServletException, IOException {
    request.setCharacterEncoding("utf-8");
    response.setContentType("text/html;charset=utf-8");
    PrintWriter out = response.getWriter();
    HttpSession session = request.getSession();
    String user_id = request.getParameter("user_id");
    String user_pw = request.getParameter("user_pw");
    if (session.isNew()){
        if(user_id != null){
            session.setAttribute("user_id", user_id);
            String url=response.encodeURL("login");
            out.println("<a href="+url+">로그인 상태 확인</a>");
        }else {
            out.print("<a href='login2.html'>다시 로그인하세요!!</a>");
            session.invalidate();
        }else {
            user_id = (String) session.getAttribute("user_id");
            if (user_id != null && user_id.length() != 0) {
                out.print("안녕하세요 " + user_id + "님!!!");
            } else {
                out.print("<a href='login2.html'>다시 로그인하세요!!</a>");
                session.invalidate();
            }
        }
    }
}
```

변수 url에 encodeURL()을 이용해 응답 시
미리 sessionId를 저장합니다.

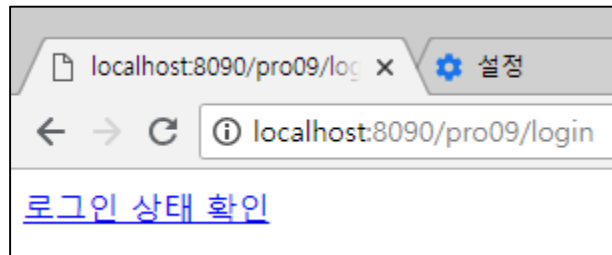
로그인 상태 확인 클릭 시
sessionId를 서블릿으로 다
시 전송합니다.

5. encodeURIComponent() 사용법

3. 로그인창에서 ID와 비밀번호를 입력하고 로그인합니다.

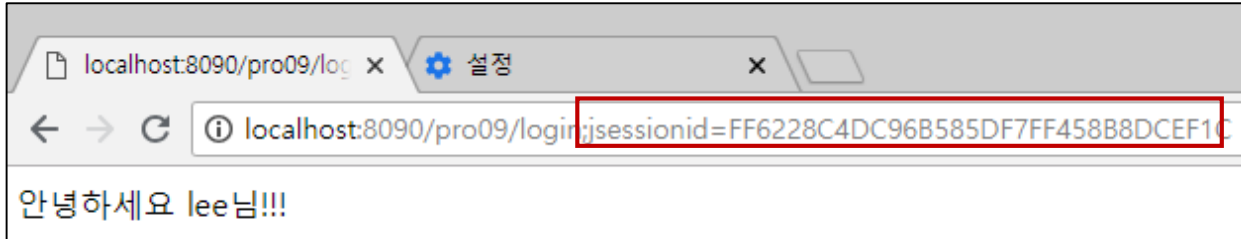


4. 로그인 상태 확인을 클릭합니다.



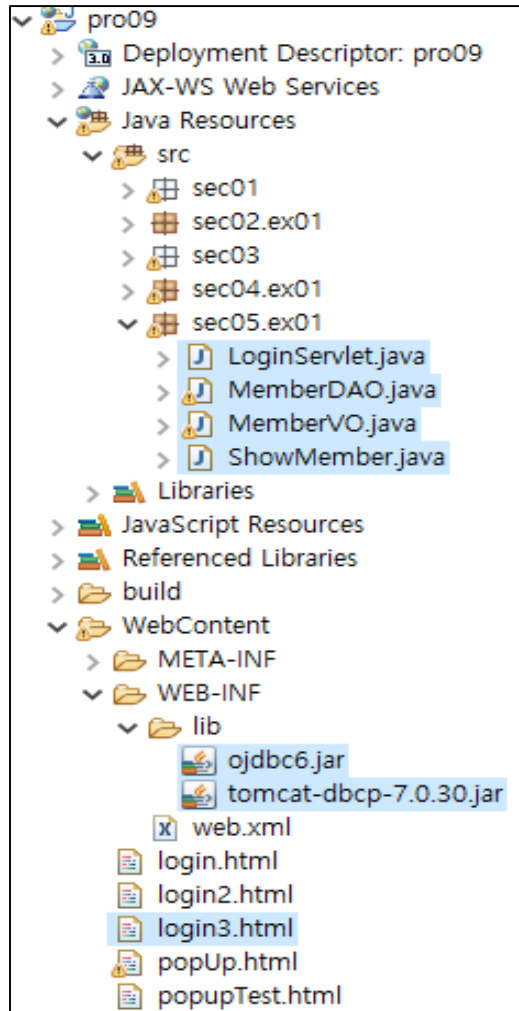
5 encodeURIComponent() 사용법

5. 서버릿에 jsessionId 쿠키 값을 전송해 로그인 상태를 유지합니다.



6. 세션을 이용한 로그인 예제

1. 먼저 데이터베이스 연동과 관련된 설정을 해줍니다. 앞장에서 실습한 회원 기능 실습 자바 클래스 파일인 MemberDAO.java와 MemberVO.java를 복사하여 붙여 넣습니다.



6. 세션을 이용한 로그인 예제

2. 사용자의 ID와 비밀번호를 입력한 후 /login 서블릿으로 전송하도록 login3.html을 작성합니다.

코드 9-16 pro09/WebContent/login3.html

```
<!DOCTYPE html>
<html>
<head>...</head>
<body>
  <form name="frmLogin" method="post" action="login" encType="UTF-8">
    아이디 :<input type="text" name="user_id"><br>
    비밀번호:<input type="password" name="user_pwd"><br>
    <input type="submit" value="로그인">
    <input type="reset" value="초기화">
  </form>
</body>
</html>
```

6. 세션을 이용한 로그인 예제

3. 로그인창의 요청을 처리하는 LoginServlet 클래스를 다음과 같이 작성합니다.

코드 9-17 pro09/src/sec05/ex01/LoginServlet.java

```
package sec06.ex01;
```

```
private public void doHandle(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    request.setCharacterEncoding("utf-8");
    response.setContentType("text/html;charset=utf-8");
    PrintWriter out = response.getWriter();
    String user_id = request.getParameter("user_id");
    String user_pwd = request.getParameter("user_pwd");
    MemberVO memberVO = new MemberVO();
    memberVO.setId(user_id);
    memberVO.setPwd(user_pwd);
    MemberDAO dao = new MemberDAO();
    boolean result = dao.isExisted(memberVO);
```

로그인창에서 전송된 ID와 비밀번호를 가져옵니다.

MemberVO 객체를 생성하고 속성에 ID와 비밀번호를 설정합니다.

MemberDAO의 isExisted() 메서드를 호출하면서 memberVO를 전달합니다.

6. 세션을 이용한 로그인 예제

```
if (result) {
```

```
    HttpSession session = request.getSession();
```

```
    session.setAttribute("isLogon", true);
```

조회한 결과가 true이면 isLogon 속성을 true로 세션에 저장합니다.

```
    session.setAttribute("login.id", user_id);
```

```
    session.setAttribute("login.pwd", user_pwd);
```

조회한 결과가 true이면 ID와 비밀번호를 세션에 저장합니다.

```
    out.print("<html><body>");
```

```
    out.print("안녕하세요 " + user_id + "님!!!<br>");
```

```
    out.print("<a href='show'>회원정보 보기</a>");
```

```
    out.print("</body></html>");
```

```
} else {
```

```
    out.print("<html><body><center>회원 아이디가 틀립니다.");
```

```
    out.print("<a href='login3.html'> 다시 로그인하기</a>");
```

```
    out.print("</body></html>");
```

```
}
```

```
}
```

```
}
```

6. 세션을 이용한 로그인 예제

4. MemberDAO 클래스를 다음과 같이 작성합니다.

코드 9-18 pro09/src/sec05/ex01/MemberDAO.java

```
package sec05.ex01;

...

public class MemberDAO {
    private DataSource dataFactory;
    public MemberDAO(){
        try{
            Context ctx=new InitialContext();
            Context envContext = (Context) ctx.lookup("java:/comp/env");
            dataFactory = (DataSource) envContext.lookup("jdbc/oracle");
        }catch(Exception e){
            e.printStackTrace();
        }
    }
    ...
    public boolean isExisted(MemberVO memberVO) {
        boolean result = false;
        String id = memberVO.getId();
        String pwd = memberVO.getPwd();
        try {
            con = dataFactory.getConnection();
```

6. 세션을 이용한 로그인 예제

```
String query = "select decode(count(*),1,'true','false') as result from t_member";
    query += " where id=? and pwd=?";

    pstmt = con.prepareStatement(query);
    pstmt.setString(1, id);
    pstmt.setString(2, pwd);
    ResultSet rs = pstmt.executeQuery();
    rs.next();
    result = Boolean.parseBoolean(rs.getString("result"));
    System.out.println("result=" + result);
} catch (Exception e) {
    e.printStackTrace();
}
return result;
}
```

오라클의 decode() 함수를 이용해 조회하여 ID와 비밀번호가 테이블에 존재하면 true를, 존재하지 않으면 false를 조회합니다.

메서드로 전달된 ID와 비밀번호를 이용해 SQL문을 작성한 후 데이터베이스에 조회합니다.

커서를 첫 번째 레코드로 위치시킵니다.

6. 세션을 이용한 로그인 예제

5. ShowMember 클래스를 다음과 같이 작성합니다.

```
package sec05.ex01;

...

@WebServlet("/show")
public class ShowMember extends HttpServlet{
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=utf-8");
        PrintWriter out = response.getWriter();
        String id = "", pwd = "" ;
        Boolean isLogon = false;
        HttpSession session = request.getSession(false);
        if( session != null){
            isLogon = (Boolean)session.getAttribute("isLogon");
            if(isLogon == true){
                id = (String)session.getAttribute("login.id");
                pwd = (String)session.getAttribute("login.pwd");
                out.print("<html><body>");
                out.print("아이디: " + id + "<br>");
                out.print("비밀번호: " + pwd + "<br>");
            }
        }
    }
}
```

이미 세션이 존재하면 세션을 반환하고, 없으면 null을 반환합니다.

먼저 세션이 생성되어 있는지 확인합니다.

isLogon 속성을 가져와 로그인 상태를 확인합니다.

isLogon이 true면 로그인 상태이므로 회원 정보를 브라우저에 표시합니다.

6. 세션을 이용한 로그인 예제

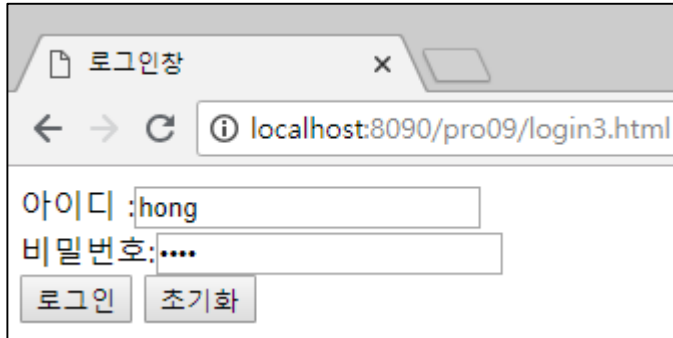
```
    out.print("</body></html>");
  }else{
    response.sendRedirect("login3.html");
  }
}else{
  response.sendRedirect("login3.html");
}
}
```

로그인 상태가 아니면 로그인창으로 이동합니다.

세션이 생성되지 않았으면 로그인창으로 이동합니다.

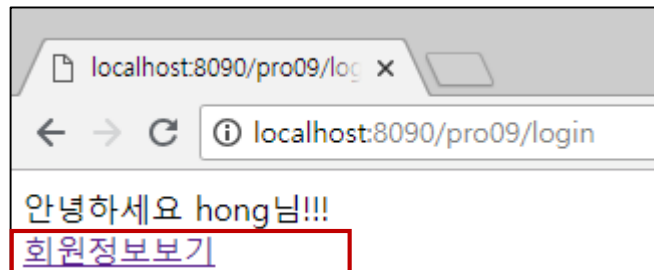
6. 세션을 이용한 로그인 예제

6. 로그인창에서 ID와 비밀번호를 입력한 후 전송합니다.



A screenshot of a web browser window. The title bar shows a tab labeled '로그인창' (Login Window). The address bar shows the URL 'localhost:8090/pro09/login3.html'. The page content includes a form with two input fields: '아이디 :hong' (ID) and '비밀번호:....' (Password). Below the fields are two buttons: '로그인' (Login) and '초기화' (Reset).

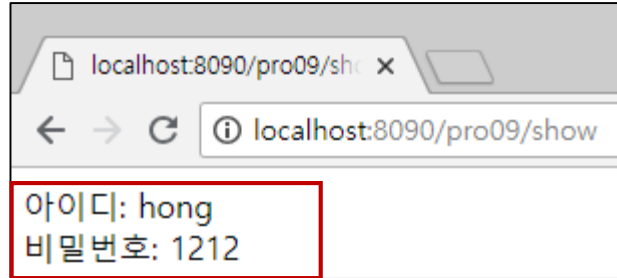
7. 회원 테이블에 입력한 ID와 비밀번호가 존재하면 로그인 성공 메시지가 출력됩니다.



A screenshot of a web browser window. The title bar shows a tab labeled 'localhost:8090/pro09/log'. The address bar shows the URL 'localhost:8090/pro09/login'. The page content displays a message '안녕하세요 hong님!!!' (Hello hong!!!) and a link '회원정보보기' (View Member Information) which is highlighted with a red rectangular box.

6. 세션을 이용한 로그인 예제

8. 회원정보보기를 클릭하면 이미 로그인 상태이므로 세션에 저장된 회원 정보가 표시됩니다.



9. 다음은 톱갯을 재실행한 후 로그인창을 거치지 않고 바로 /show로 요청한 경우입니다. 로그인을 하지 않았으므로 다시 로그인창으로 리다이렉트됩니다.

