8장 쿠키, 세션과 로그인

동의과학대학교
컴퓨터정보과
김진숙

이장의 목차

- HTTP(HyperText Transfer Protocol)
- 쿠키(Cookie)
- 세션(Session)

HTTP란?

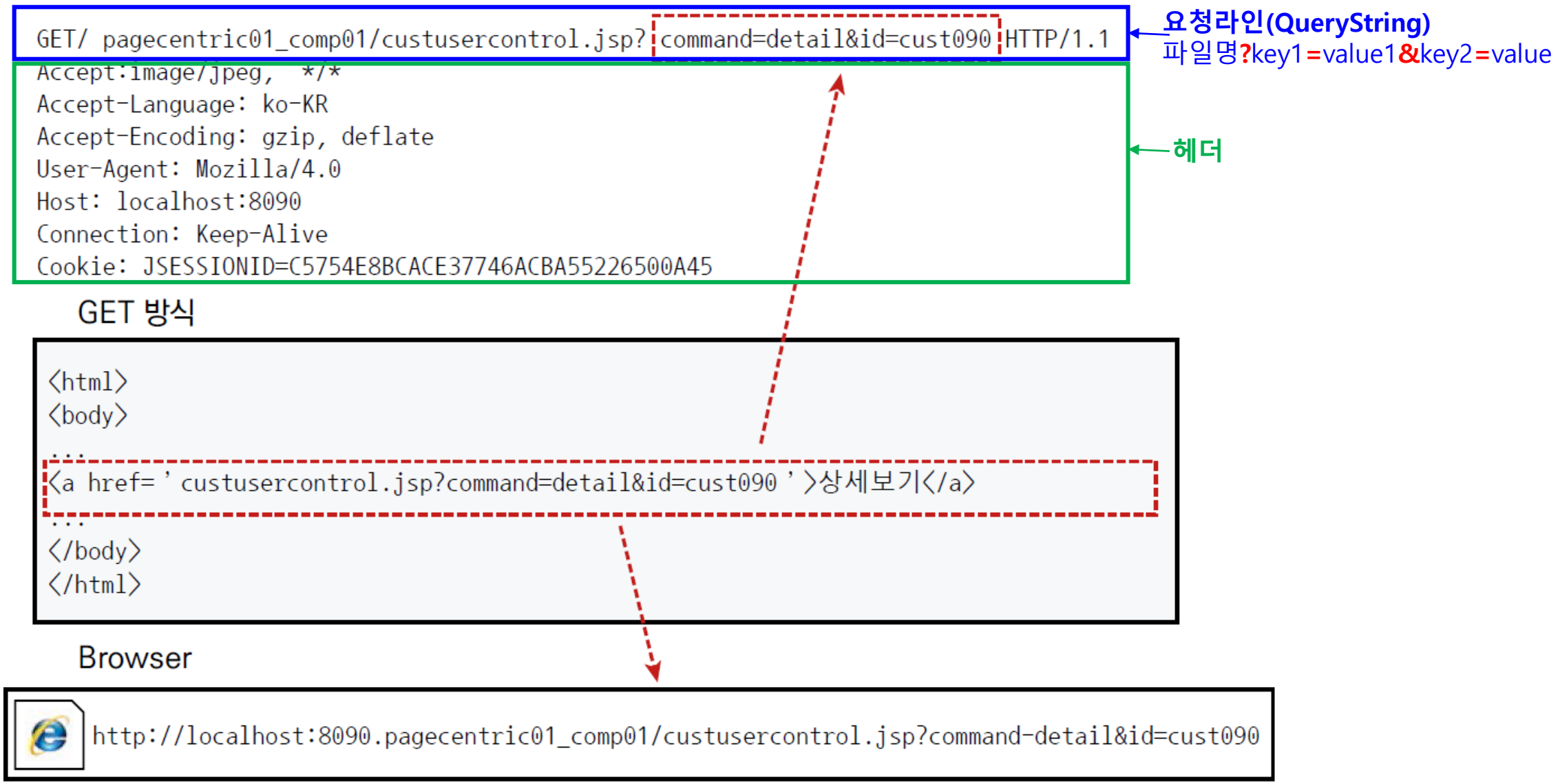
- HTTP는 웹 서버와 웹 클라이언트 간에 통신하기 위한 프로토콜
- 동작 방식
  - 웹 브라우저 : HTTP프로토콜에 맞게 요청(request)를 웹 서버에 전송
  - 웹 서버 : HTTP프로토콜에 맞게 응답(response)를 웹 브라우저에 전송
  - 요청과 응답은 세 부분으로 구조화되어 있음
    1. request / response line
    2. header
    3. entity body

요청(request)의 HTTP 포맷

1. 웹 클라이언트는 미리 설정된 포트(port)로 연결 시도
2. 연결되면 웹 클라이언트는 HTTP명령어, 문서주소, HTTP버전 정보를 웹 서버에 전달
3. 요청 라인 다음 줄에 헤더 정보, 구성정보, 받아들일 문서 포맷 등을 정보를 옵션으로 웹 서버에 전달
   - 모든 헤더정보는 한 줄에 하나씩 기술되며 헤더명과 값(key/value)으로 구성
   - 끝에 공백라인을 추가해 헤더의 끝 표시(**<CRLF>**)
4. 요청과 헤더정보를 보낸 다음 웹 클라이언트는 추가 정보를 보낼 수 있는데 보통 POST 방법으로 웹 서버에 전달

요청라인 → **<method><resource identifier><HTTP version><CRLF>**
헤더 → **[<header>:<value>] <CRLF>**
……
<CRLF>
엔티티 바디 → **[entity body]**

GET 요청 방식


GET/ pagecentric01_comp01/custusercontrol.jsp?command=detail&id=cust090 HTTP/1.1
Accept:image/jpeg, */*
Accept-Language: ko-KR
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Host: localhost:8090
Connection: Keep-Alive
Cookie: JSESSIONID=C5754E8BCACE37746ACBA55226500A45
요청라인(QueryString)
파일명?key1=value1&key2=value
헤더
GET 방식
<html>
<body>
...
상세보기
...
</body>
</html>
Browser
http://localhost:8090.pagecentric01_comp01/custusercontrol.jsp?command-detail&id=cust090

POST 요청 방식

```
post/ pagecentric01_comp01/custusercontrol.jsp HTTP/1.1
```

← 요청라인

```
Accept:image/jpeg,  */*
Accept-Language: ko-KR
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Host: localhost:8090
Content-Length: 27
Connection: Keep-Alive
Cookie: JSESSIONID=C5754E8BCACE37746ACBA55226500A45
```

← 헤더

Body → `command=bfupdate&id=cust002`

빈 한 줄로 HEAD와 BODY구분
파라미터가 바디에 위치한다.

key1=value1&key2=value

POST 방식

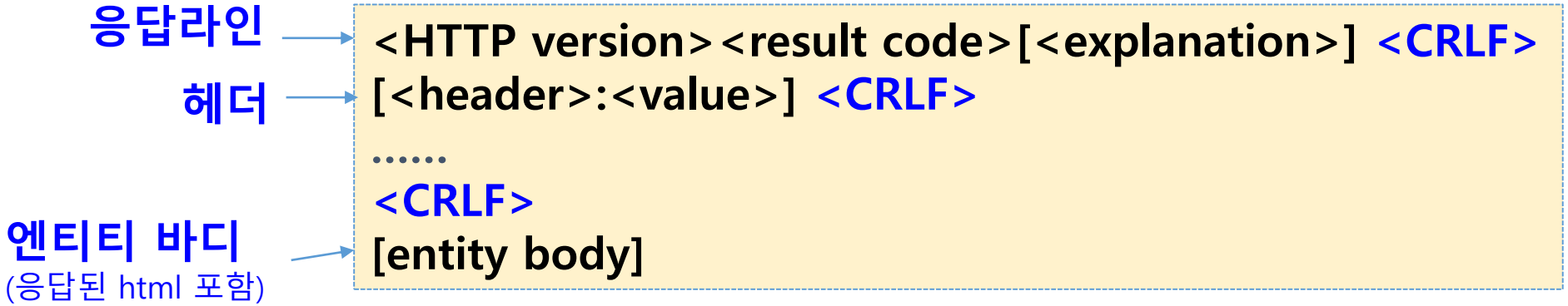
```
<form action="custusercontrol.jsp" method='post'>
        <input type='hidden' name='command' value='bfupdate'/>
        <input type='hidden' name='id' value='cust002'/>
        <input type='hidden' value='고객정보변경'/>
</form>
```

Browser

http://localhost:8090.pagecentric01_comp01/custusercontrol.jsp

응답(Response) 의 HTTP 포맷

1. HTTP버전, 상태코드, 설명으로 구성된 상태라인으로 응답
2. 상태코드는 클라이언트의 요청에 따른 서버 결과를 표현
3. 웹 서버는 자신과 요청된 문서에 대한 정보를 헤더를 통해 클라이언트에 전달
   - 헤더 끝은 공백라인 **<CRLF>** 로 표시
4. 클라이언트의 요청이 성공적으로 수행되었다면 요청된 자료 전송
   - 전송되는 자료는 파일 내용이거나 서블릿/JSP 프로그램의 응답이 될 수 있다.



상태코드와 응답 예

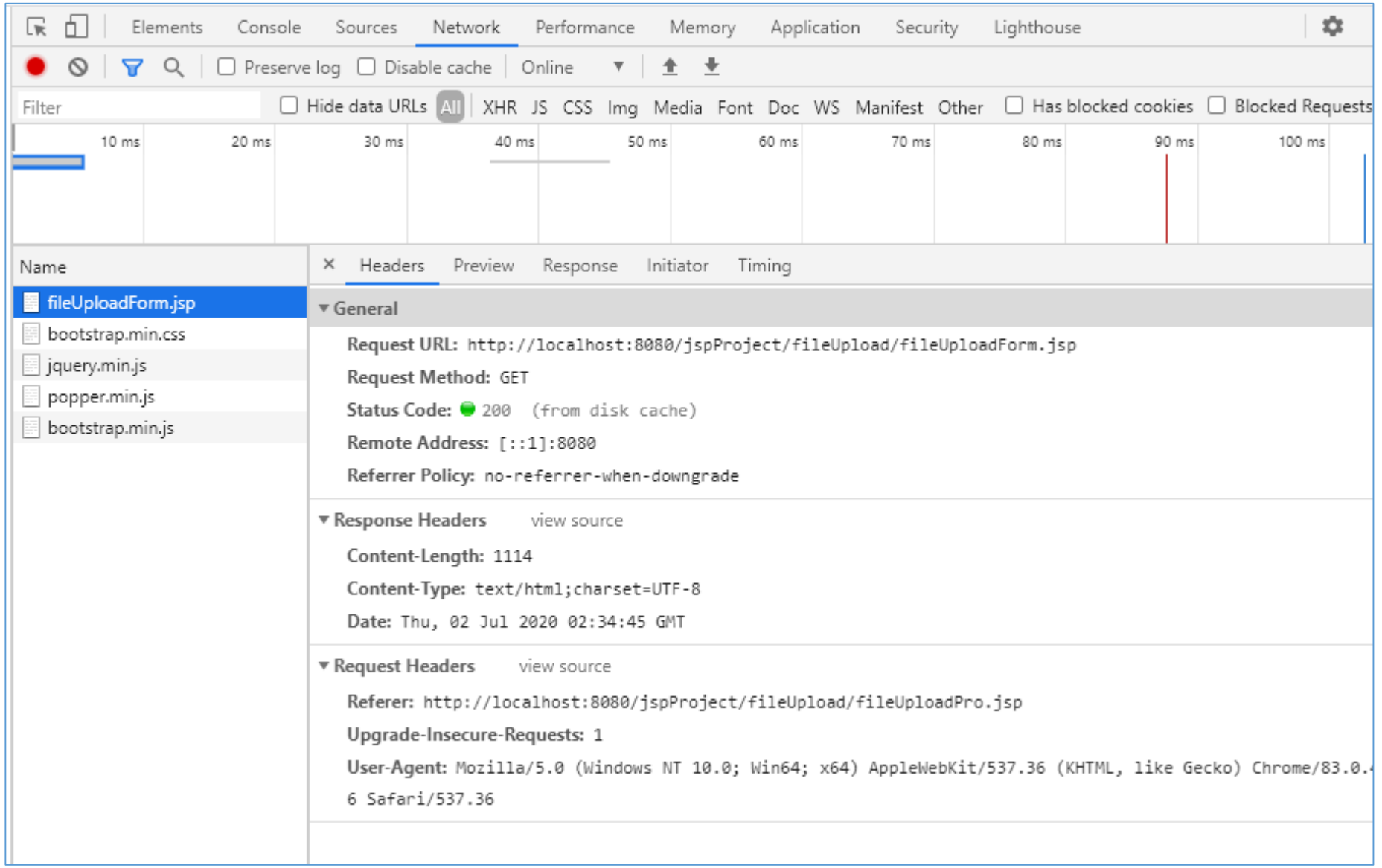
HTTP상태코드	코드	설명
1xx	정보	클라이언트로부터 일부분만 받았으니 나머지 요청정보를 요청
2xx	성공	에러 없이 전송함
3xx	경로변경	요청을 완전히 처리하기 위해 추가적 액션이 수행되어야 함을 의미
4xx	클라이언트 에러	요청 실패- 문법상 오류로 서버가 요청사항을 이해하지 못함
5xx	서버에러	정당한 요청을 서버가 처리하지 못함을 의미

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html;charser=euc-kr
Content-Length: 219
Date: Wed, 30 Dec 2009 09:10:30 GMT

<html>
<head><title>안녕하세요</title></head>
<body>...
```

엔티티 바디

구글 개발자 도구 (F12 또는 Ctrl+Shift+i) 살펴보기



HTTP 특성

- 무상태성(stateless)
  - HTTP를 통한 클라이언트와 서버 사이의 대화는 방금 전 대화를 기억하지 못하는**'무상태'**
  - 동일한 클라이언트가 다시 서버에 연결해도 서버는 이전 연결에 대한 클라이언트의 어떤 상태(state) 정보도 가지고 있지 않다.
- 비연결성(connectionless)
  - 서버는 웹 브라우저를 통해 받은 요청에 응답한 후 연결 종료

DB 서버는 클라이언트와 나눈 대화를 기억(**유상태**, statefull)한다. 대화가 끝난 후 다시 연결하지 않아도 요청을 할 수 있다. DB 서버가 대화를 끝내려면 명시적(close 호출)으로 끝내야 한다. DB 서버는 사용자가 제한적이므로 연결을 유지하는 편이 좋다.

HTTP의 비연결성의 장단점

- 장점
  - HTTP Session을 무상태로 한 이유는 많은 클라이언트들의 요청에 의한 **웹 서버의 과부하를 방지**하기 위한 것
  - 서버에 접속한 클라이언트의 수가 많더라도 **서버 부담이 적고 서버 자원을 효율적으로 사용**할 수 있음
- 단점
  - 인터넷 쇼핑몰에서 상품 구매 경우, 다른 페이지로 이동하면 현재 선택된 상품 목록과 **관련정보를 지속적으로 유지 관리 하기 어려움**

HTTP의 비연결성의 단점을 보완하고 페이지 간 지속성 서비스를 제공하기 위한 기법

- ▶ **쿠키(Cookie) : 클라이언트 컴퓨터에 정보 저장 관리**
- ▶ **세션(Session): 서버에 브라우저마다 다른 사용자 정보를 저장**
- ▶ URL Rewriting: 페이지 실행 시 그 페이지의 URL 파라미터를 붙여 실행이 되도록 하는 방법(정보노출)
- ▶ hidden form : 폼의 input 태그 속성 type을 'hidden'으로 하여 값을 서버로 넘길 수 있음(정보 노출이 안됨)

[참조] hidden form

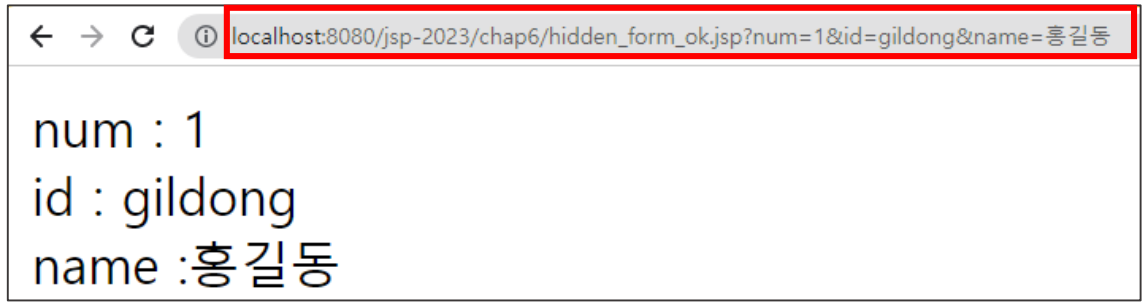
- hidden form 방식
  - form tag 내에 <input> 태그의 type을 hidden으로 하여 암묵적으로 데이터 전달 가능

```
<body>
  <form name="myform" action="hidden_form_ok.jsp" method="get">
    <input type="hidden" name="num" value="1">
    <input type="text" name="id"><br>
    <input type="text" name="name"><br>
    <input type="submit" value="확인"><br>
  </form>
</body>
```

```
<%
  request.setCharacterEncoding("utf-8");

  String num = request.getParameter("num");
  String id = request.getParameter("id");
  String name = request.getParameter("name");
%>

num : <%=num%><br>
id : <%=id%><br>
name :<%=name%><br>
```



쿠키와 세션 비교

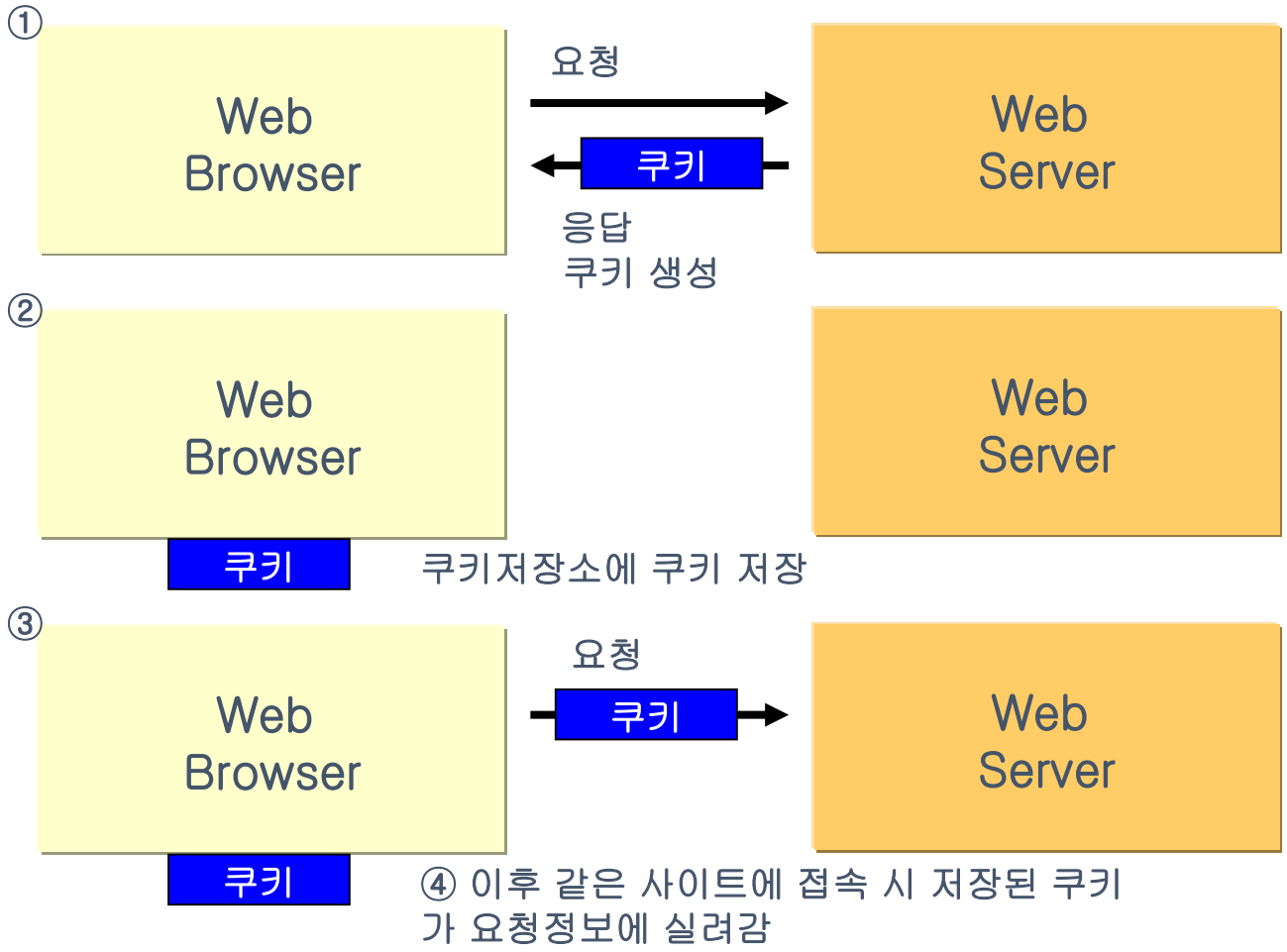
구분	쿠키	세션
사용 클래스/ 인터페이스	Class javax.servlet.http.**Cookie**	Interface javax.servlet.http.**HttpSession**
관련 내장객체	response, request	session
저장 값 유형	문자열 형태만 가능	자바의 모든 객체
저장 장소	**클라이언트**에 저장	**서버**에 저장
정보 크기	총 1.2MB로 제한 있음	제한 없음
보안	어려움	강력함

1. 쿠키(Cookie)

- 쿠키의 개요
  - 서버에서 만들어진 **작은 정보의 단위**로 서버에서 클라이언트의 브라우저로 전송되어 **사용자의 컴퓨터에 저장**됨
  - 쿠키는 다시 해당 웹사이트에 접속할 때 브라우저에서 서버로 전송될 수 있음
  - 쿠키는 특정 웹 사이트에 접속할 때 생성되는 정보를 담은 임시 파일
    - 한 쿠키의 크기는 4KB 이하
    - 클라이언트 당 최대용량은 1.2MB
  - 저장 구조
    - **이름(name)과 값(value)**로 구성된 자료를 저장
    - 이외에도 주석, 경로, 유효기간, 버전, 도메인 같은 추가 정보 저장 가능
  - 사용 용도 : **사용자 id**, **방문 횟수** 기록 등
  - 클라이언트는 원하지 않는다면 웹 서버에서 전달된 쿠키를 삭제하거나 받지 않을 수 있음

1.쿠키(Cookie)

- 쿠키의 동작 방식



1. JSP에서 쿠키를 사용하기 위해서는 javax.servlet.http 패키지에 있는 Cookie 클래스의 객체를 생성
2. 생성된 쿠키에는 각각의 웹 브라우저를 판별할 수 있는 정보가 포함
3. 생성된 쿠키는 웹 서버가 웹 브라우저의 요청에 응답할 때, response객체에 실려서 **사용자의 웹 브라우저에 저장**
4. 웹 브라우저에 저장된 쿠키는 다시 사용자가 웹 서버에 요청을 할 때 request객체에 실려서 웹 서버에 전달

쿠키 관련 메소드

- Cookie 클래스의 주요 메소드

반환 유형	메소드	기능
int	**getMaxAge()**	쿠키의 최대 지속시간을 반환
String	**getName()**	쿠키의 이름을 String으로 반환
String	**getValue()**	쿠키의 값을 String으로 반환
void	**setMaxAge(int expiry)**	쿠키의 만료시간을 초단위로 설정
void	**setValue(String value)**	쿠키에 새로운 값을 설정할 때 사용

- Cookie관련 객체/메소드

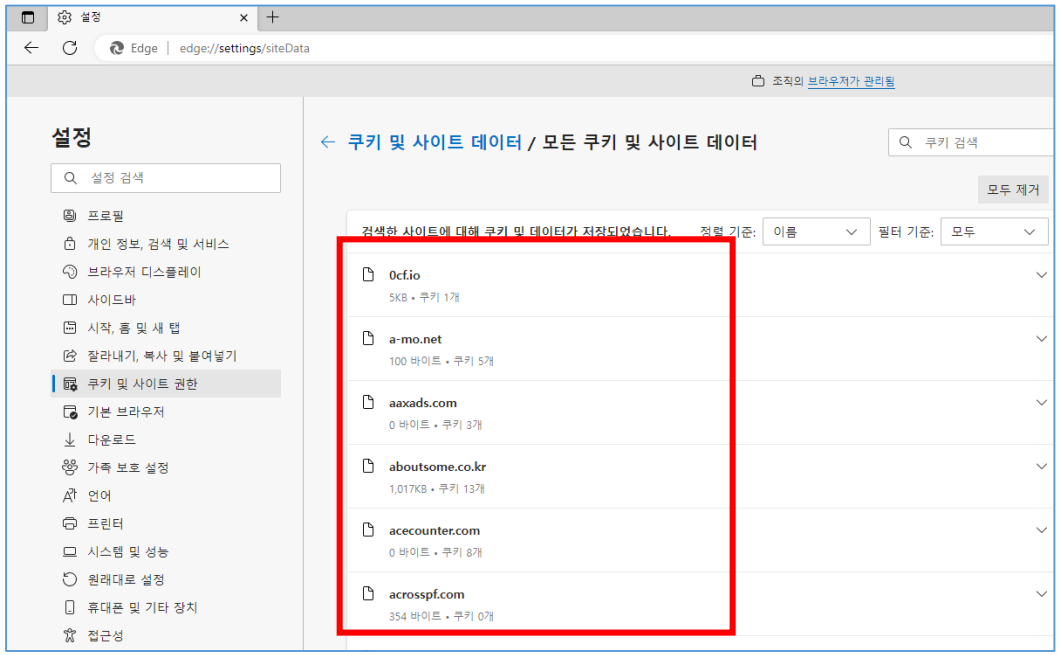
객체	메소드	기능
response	**addCookie(Cookie c)**	쿠키만들기 : 클라이언트 컴퓨터에 파일 형태로 정보 c를 저장
request	**getCookies()**	쿠키사용 : 클라이언트 컴퓨터에 저장된 쿠키를 조회 반환값은 저장된 모든 쿠키의 배열로 쿠키가 없으면 null 반환

1.쿠키(Cookie)

- 쿠키의 문제점
  - 웹 브라우저가 거쳐 간 웹 사이트 및 개인 정보가 기록됨
    - **보안 문제 유발** 가능성
  - 보안문제를 유발하기 때문에 웹 브라우저 자체에 **쿠키 거부 기능**이 들어 있음
    - 쿠키 거부 기능이 웹 브라우저에 설정되어 있으면 쿠키 본래의 목적인 웹 브라우저와의 연결을 지속시키는 기능을 수행할 수 없음
    - 이것이 쿠키의 가장 치명적인 단점
  - 쿠키는 최대 300개 저장할 수 있으며 그 이상이 되면 가장 오래된 것부터 삭제
  - 쿠키 저장 장소

쿠키 저장 장소

- **Edge**



- **chrome**
  - 설정 – 개인정보 및 보안 – 쿠키 및 기타 사이트 데이터 – 모든 쿠키 및 사이트 데이터 보기

1.쿠키(Cookie)

- 쿠키의 생성
  1. 쿠키 객체 생성

     Cookie cookie = new Cookie(String name, String value);

     key **value**

  2. 쿠키 속성 설정(예제) - 선택

     cookie.setMaxAge(2*60); //쿠키의 유효기간을 2분으로 설정
     cookie.setMaxAge(-1); //브라우저가 종료되면 쿠키도 삭제
     cookie.setMaxAge(0); //쿠키 삭제(무효화)

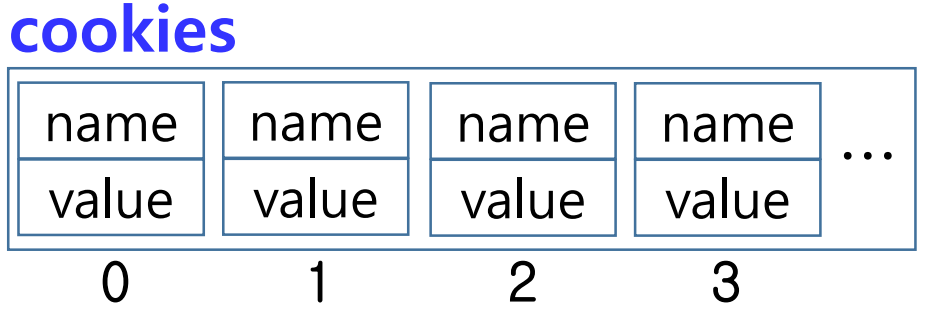
  3. 쿠키 전송(클라이언트에 저장)

     response.addCookie(cookie);

1.쿠키(Cookie)

- 쿠키의 사용
  1. 쿠키 얻기

```
Cookie[] cookies = request.getCookies();
```



  2. 쿠키 배열 추출
     - 쿠키는 이름, 값의 쌍으로 된 배열형태로 리턴 반환
     - 반환된 쿠키의 배열에서 쿠키 이름을 가져옴
     - 쿠키이름을 통해서 해당 쿠키에 설정된 값을 추출

```
if(cookies != null){
      for(int i = 0; i < cookies.length; i++){
        out.println(cookies[i].getName() + ":" +cookies[i].getValue() + "<br>");
      }
}else{
      out.println("쿠키가 없습니다.");
}
```

실습1 – 쿠키에 속성을 쓰고 읽기

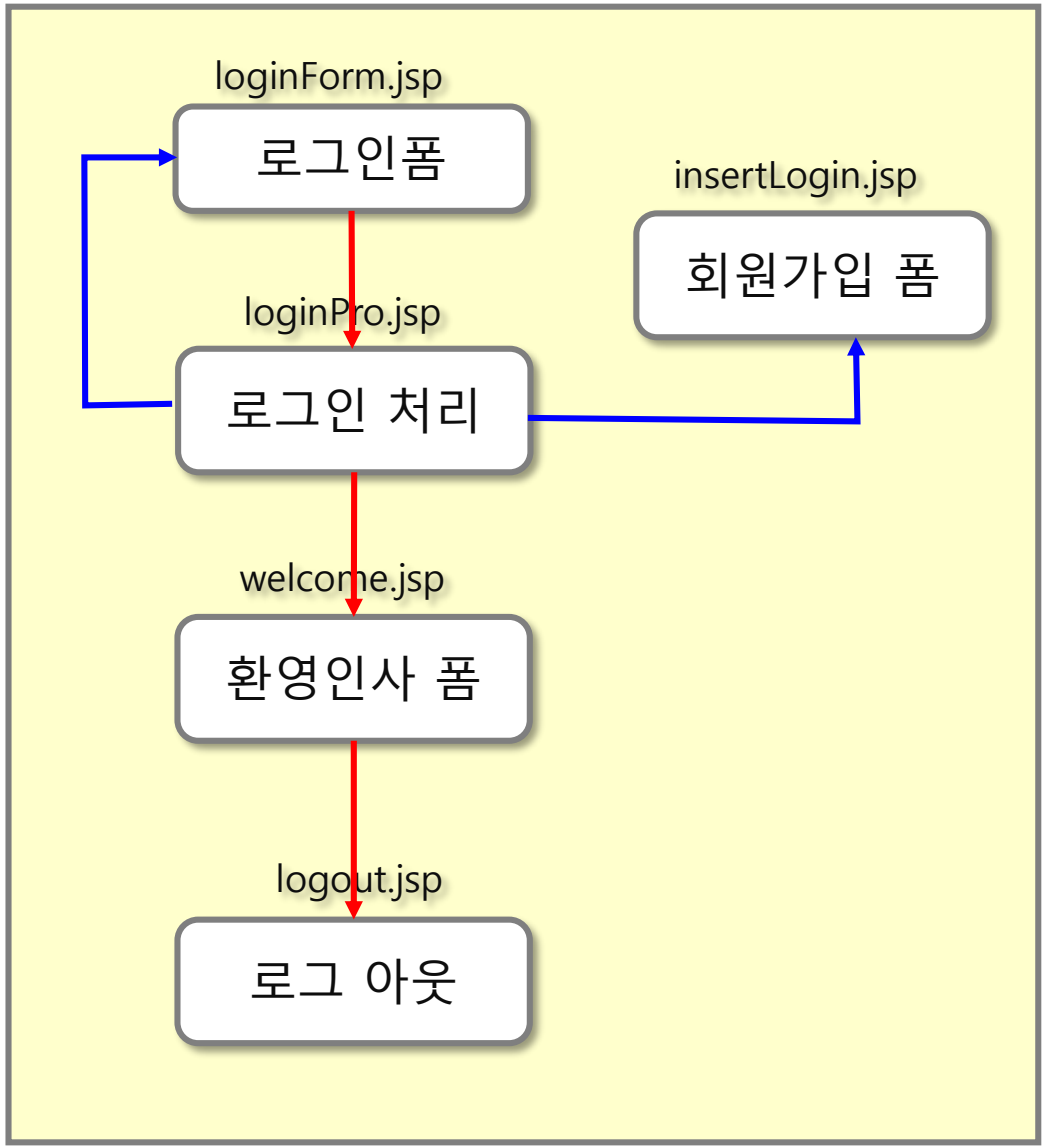
```
 7  <title>쿠키 생성 : makeCookie.jsp</title>
 8  </head>
 9  <body>
10  <%
11      String cookieName = "id";
12      //1. 쿠키 객체 생성 : 속성명은 id , 속성값은 자신의 이름
13      Cookie cookie = new Cookie(cookieName, "jinsook");
14
15      //2. 쿠키 속성 설정 : 2분간 쿠키가 유효하도록 설정
16      cookie.setMaxAge(60*2);
17
18      //3. 쿠키를 클라이언트에 전송
19      response.addCookie(cookie);
20  %>
21      <%=cookieName %>가 생성되었습니다.<br>
22
23      <form action="useCookie.jsp" method="post">
24          <input type="submit" value="생성된 쿠키 확인">
25      </form>
26  </body>
```

실습1 – 쿠키에 속성을 쓰고 읽기

```
7  <title>웹브라우저에 저장된 쿠키 가져오기: useCookie.jsp</title>
8  </head>
9  <body>
10    <h3>쿠키 보기</h3>
11    <%
12       //1. 사용자 컴퓨터에 저장된 쿠키배열 가져오기
13       Cookie[] cookies = request.getCookies();
14
15       //2. 쿠키배열이 null이 아니면 쿠키에서 "id" 속성이 있는지
16       //   확인하여 화면에 쿠기 속성명과 속성값을 출력
17       if(cookies != null){
18          for(int i=0; i<cookies.length; i++){
19             if(cookies[i].getName().equals("id")){
20    %>
21                쿠키의 속성명: <%=cookies[i].getName() %> <
22                쿠키의 값: " <%=cookies[i].getValue() %>
23    <%
24             }
25          }
26       }
27    %>
28 </body>
```

```
10  <%
11  Cookie[] cookies = request.getCookies();
12
13  for(Cookie c:cookies){
14    if(c.getName().equals("id")){
15      out.println(c.getValue());
16    }
17  }
18
19  %>
```

실습2 – 쿠키를 사용한 DB 회원 인증

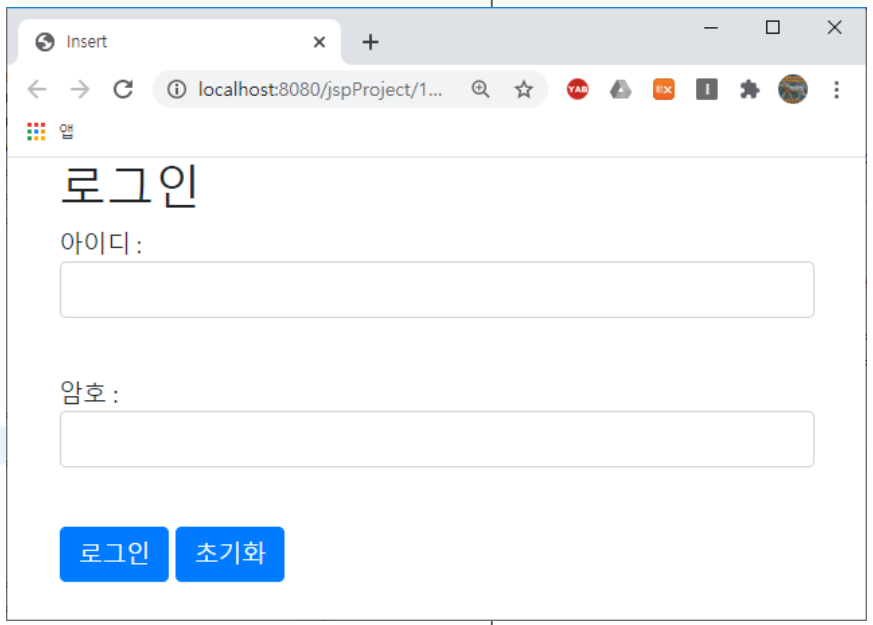


1. 다음의 흐름으로 DB 인증을 진행한다.
2. DB에 있는 유효한 사용자인 경우 사용자 id를 쿠키에 저장하고 welcome.jsp페이지를 출력
3. DB에 없는 사용자인 경우에는 회원가입폼이 출력되도록 한다.
4. welcome 페이지에는 logout 버튼으로 쿠키에서 지워지도록 한다.

로그인 폼

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Insert</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"></script>
</head>
<body>
<div class="container">
  <h2>로그인</h2>

  <form action="loginPro.jsp" method="post">
    <div class="form-group">
      아이디 : <input class="form-control" type="text" name="id"><br>
    </div>
    <div class="form-group">
      암호 : <input class="form-control" type="password" name="pwd"><br>
    </div>
    <input class="btn btn-primary" type="submit" value="로그인">
    <input class="btn btn-primary" type="reset" name="재작성">
  </form>
</div>
</body>
</html>
```



쿠키 저장

```jsp
<%@page import="csdit.LoginDAO"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
  request.setCharacterEncoding("utf-8");

  String id = request.getParameter("id");
  String pwd = request.getParameter("pwd");

  //checkUser메소드 사용을 위해 LoginDAO 생성
  LoginDAO dbPro = new LoginDAO();
  int check = dbPro.checkUser(id, pwd);//checkUser메소드 호출 및 반환값 얻어옴

  if(check==1){ //아이디와 비밀번호가 맞는 유효한 사용자
    Cookie cookie = new Cookie("id", id);  //쿠키 생성
    cookie.setMaxAge(20*60); //20분
    response.addCookie(cookie);             //쿠키 전송
    response.sendRedirect("welcome.jsp?id="+id);//welcome.jsp에 id 값 보냄
  }else if(check==0){
%>
<script>
  alert("비번을 잘못입력하셨습니다. 다시 로그인하시기 바랍니다.");
  location.href="loginForm.jsp";
</script>
<%}else{
%>
<script>
  alert("없는 사용자입니다. 회원가입화면으로 이동합니다.");
  location.href="../12주차/insertDB.jsp";
</script>
<%}%>
```

쿠키 읽어오기

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
     String id = "";
     Cookie[] cookie = request.getCookies();

     if(cookie!=null || !cookie.equals("")){
       for(int i=0; i<cookie.length; i++){
         if(cookie[i].getName().equals("id"))
           id = cookie[i].getValue();
       }
     }else
       response.sendRedirect("loginForm.jsp");
%>
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title></title>
</head>
<body>
  반갑습니다!! <%=id%>님이 로그인하셨습니다! 반갑습니다.<br>
  <br><br>
  <input type="button" onclick="location.href='logout.jsp'" value="로그아웃 ">
</body>
</html>
```

쿠키 무효화

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
  String id = "";
  Cookie[] cookie = request.getCookies();

  if(cookie!=null || !cookie.equals("")){  //id로 저장된 쿠키 찾기
   for(int i=0; i<cookie.length; i++){
     if(cookie[i].getName().equals("id"))
       cookie[i].setMaxAge(0);     //유효기간을 0로 만들어 쿠키를 무효화시킴
       response.addCookie(cookie[i]);
   }
  }
%>
<script>
  alert("로그아웃되었습니다!");
  location.href="loginForm.jsp";
</script>
```

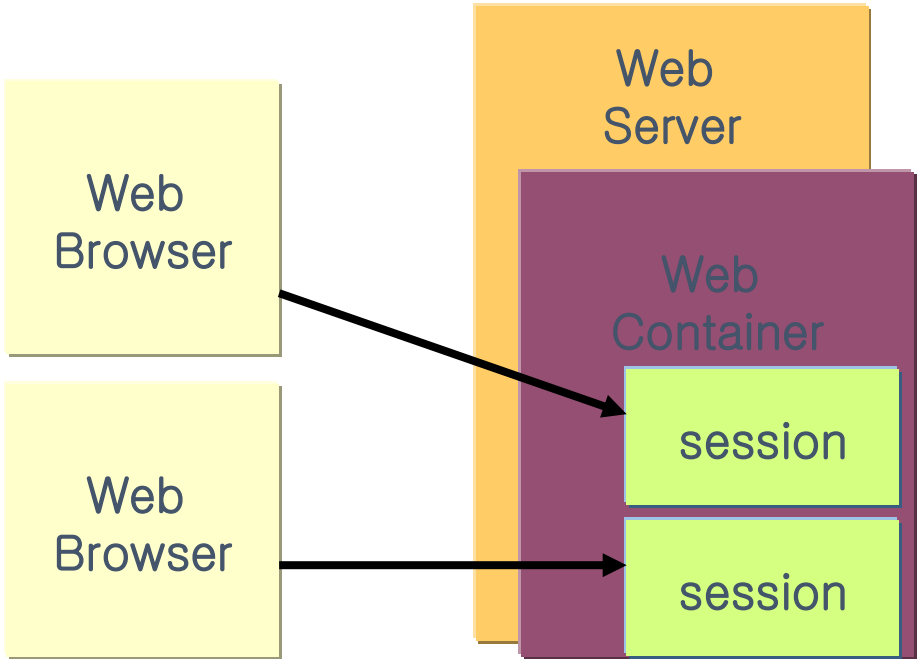
2.세션(Session)

- 세션의 개요
  - 세션은 웹 서버 쪽의 웹 컨테이너에 상태를 유지하기 위한 정보 저장
    - 웹 컨테이너는 사용자의 상태를 **서버 컴퓨터의 메모리에 저장**
    - 메모리에 저장된 내용은 **JSESSIONID** 값(식별자)으로 지정된 고유한 세션ID를 통해 접근 가능
    - 웹 브라우저에 전달되는 것이 세션ID임
  - javax.servlet.http 패키지의 HttpSession 인터페이스를 구현
  - 로그인과 같이 보안상 중요한 작업은 세션을 사용
  - 세션의 유지시간은 기본이 30분 – 기본값 변경 가능
    - [servers]-web.xml-<session-config>에 설정되어 있음

```
624   <!-- ==================== Default Session Configuration ================= -->
625   <!-- You can set the default session timeout (in minutes) for all newly   -->
626   <!-- created sessions by modifying the value below.                       -->
627
628     <session-config>
629         <session-timeout>30</session-timeout>
630     </session-config>
```

2.세션(Session)

- 세션은 웹 브라우저당 1개씩 생성되어 웹 컨테이너에 저장



2.세션(Session)

- Session 객체의 주요 메소드

반환 유형	메소드명 및 설명
java.lang.Object	**getAttribute**(java.lang.String name) :name이란 이름에 해당되는 속성값을 Object 타입으로 반환. 해당되는 이름이 없을 경우에는 null 값 반환.
java.util.Enumeration	**getAttributeNames():** 속성명을 Enumeration 타입으로 반환
long	**getCreationTime():** 1970년 1월 1일 자정을 기준으로 현재 세션이 생성된 시간까지 계산하여 1/1000초로 반환
java.lang.String	**getId**() : 세션에 할당된 고유ID를 String 타입으로 반환
int	**getMaxInactiveInterval**(): 현재 생성된 세션을 유지하기 위해 설정된 최대 시간을 정수형으로 반환
void	**invalidate**(): 현재 생성된 세션을 무효화 시킴
void	**removeAttribute**(java.lang.String name): name으로 지정한 속성 값 제거
void	**setAttribute**(java.lang.String name, java.lang.Object value): name으로 지정한 이름에 value 값 할당
void	**setMaxInactiveInterval(int interval):** 세션의 최대 유지시간을 초 단위로 설정

2.세션(Session)

- 세션 속성의 설정

  session.setAttribute(“id", “jinsook@dit.ac.kr");

- 세션의 속성을 사용

  String id= (String)session.getAttribute(“id");

반환유형이 Object 이므로 String으로 캐스팅(형변환) 해주어야 함

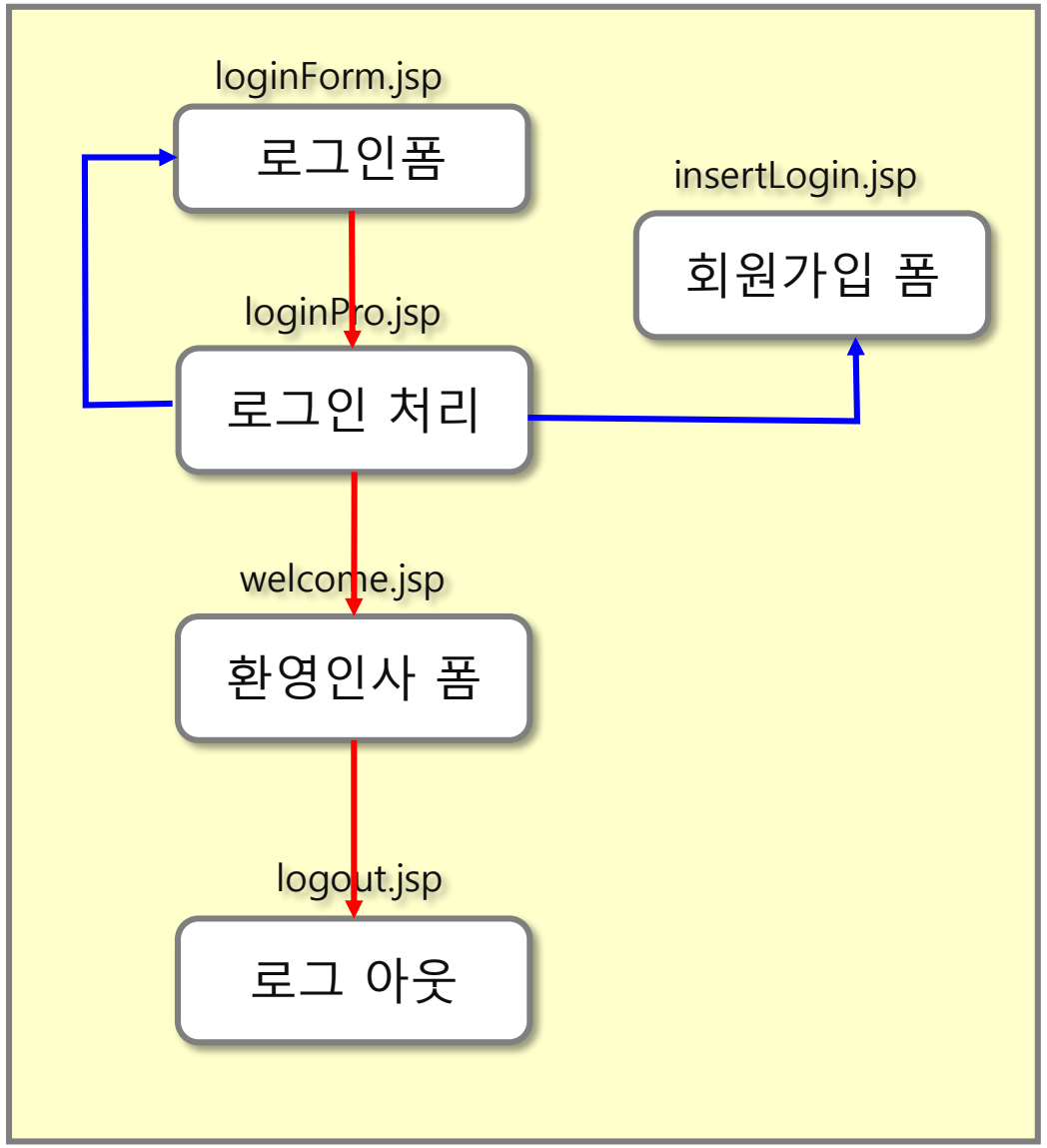
- 세션의 속성을 삭제

  session.removeAttribute(“id");

- 세션의 모든 속성 삭제

  session.invalidate();

실습 3– 세션을 사용한 DB 회원 인증



1. 쿠키로 작성되었던 것을 세션으로 작성할 것