

Servlet / JSP - 2

사용자 입력을 통한 GET 요청

```
index.html  Nana.java  web.xml  hello.html x
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
5 <title>Insert title here</title>
6 </head>
7 <body>
8   <div>
9     <form action = "hi">
10      <div>
11        <label>"안녕하세요"를 몇번듣고싶으세요?</label>
12      </div>
13      <div>
14        <input type="text" name = "cnt" />
15        <input type="submit" value = "출력" />
16      </div>
17    </form>
18  </div>
19 </body>
20 </html>
```

```
@WebServlet("/hi")
public class Nana extends HttpServlet {
    @Override
    protected void service(HttpServletRequest request, HttpServletResponse response) throws
        ServletException, IOException {
        response.setCharacterEncoding("UTF-8");
        response.setContentType("text/html; charset=UTF-8");

        PrintWriter out = response.getWriter();

        String cnt_ = request.getParameter("cnt");

        int cnt = 100;

        if(cnt_ != null && !cnt_.equals(""))
            cnt = Integer.parseInt(cnt_);

        for(int i=0; i<cnt; i++)
        {
            out.println((i+1)+" : 안녕 Servlet!! <br >");
        }
    }
}
```

"안녕하세요"를 몇번듣고싶으세요?

← → ↺ localhost:8000/hi?cnt=5

쿠팡! G마켓

1: 안녕 Servlet!!
2: 안녕 Servlet!!
3: 안녕 Servlet!!
4: 안녕 Servlet!!
5: 안녕 Servlet!!

- 입력할 내용이 많은 경우에는 POST 요청

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
5 <title>Insert title here</title>
6 </head>
7 <body>
8 <div>
9 <form action = "notice-reg" method="post">
10 <div>
11 <label>제목 :</label><input name="title" type="text">
12 </div>
13 <div>
14 <label>내용 :</label>
15 <textarea name="content"></textarea>
16 </div>
17 <div>
18 <input type="submit" value = "등록" />
19 </div>
20 </form>
21 </div>
22 </body>
23 </html>
```

```
@WebServlet("/notice-reg")
public class NoticeReg extends HttpServlet {
    @Override
    protected void service(HttpServletRequest request, HttpServletResponse response) {
        response.setCharacterEncoding("UTF-8");
        response.setContentType("text/html; charset=UTF-8");

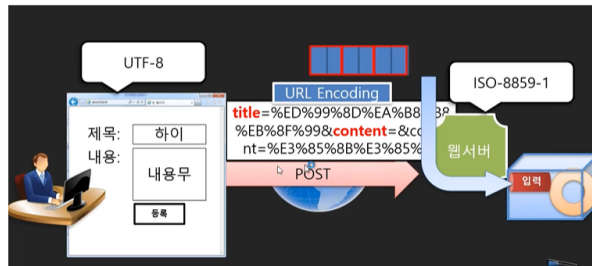
        PrintWriter out = response.getWriter();

        String title = request.getParameter("title");
        String content = request.getParameter("content");

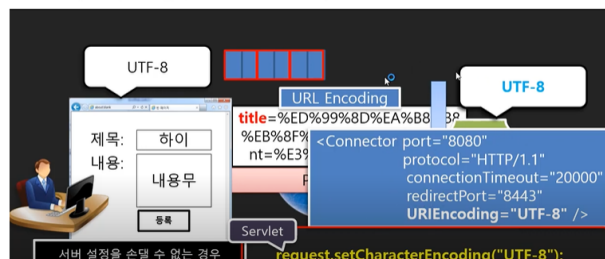
        out.println(title);
        out.println(content);
    }
}
```

멀티 바이트 문자 전송 문제: 사용자로부터 값 입력 받아서 전송하기

- 영어 제외 언어 한글같은 경우는 문자와 연결되어있는 코드값은 2바이트.



- UTF-8로 읽어야 한다.

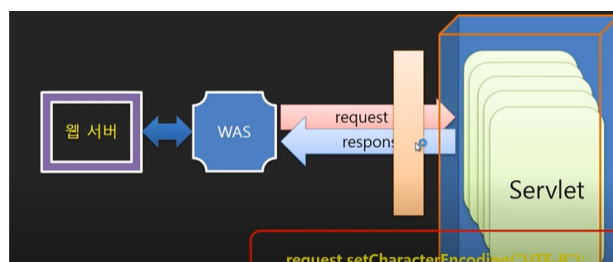


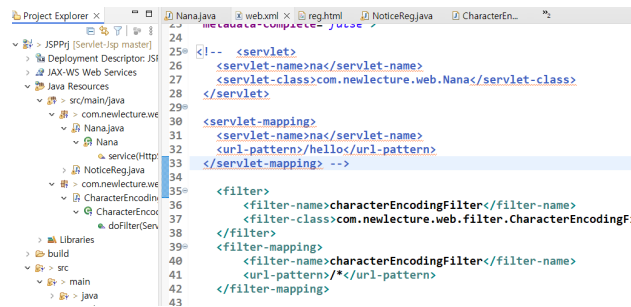
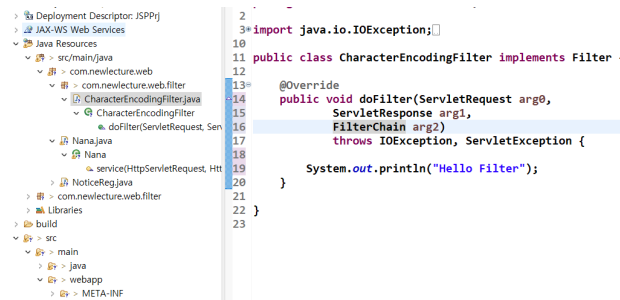
위와같이 톰캣서버의 설정을 바꿀 수 있다. 하지만 나의 설정을 위해 여러개의 서비스에 영향을 줄 수 있기 때문에 톰캣 서버의 설정을 바꾸는 경우는 드물다.

→ `request.setCharacterEncoding("UTF-8");` 사용

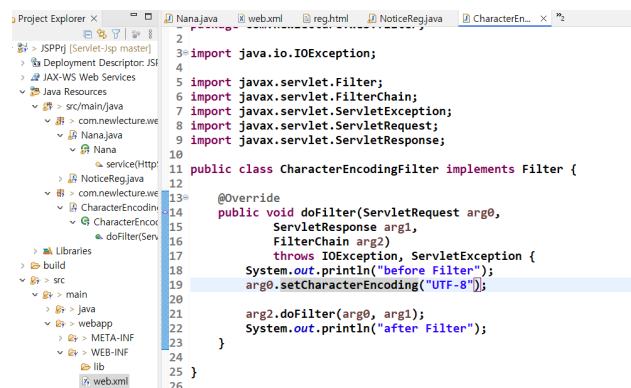
서블릿 필터

- 웹 어플리케이션 서버와 서블릿 컨테이너 사이에서 요청될때 한번, 응답할때 한번 실행, 모든 서블릿이 갖고있는 설정을 필터에서 설정할 수 있다.





- web.xml에 Filter 설정



- doFilter 를 활용해서 요청과 입력값에 대한 인코딩 설정
- web.xml 필터 설정보다는 어노테이션을 활용해서 설정하는 방법을 주로 사용한다

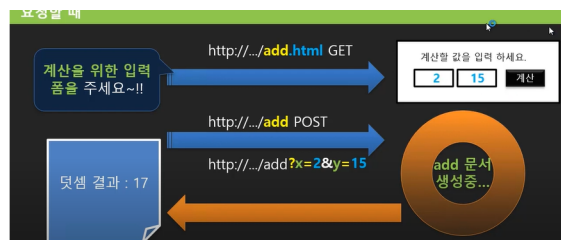
```

CharacterEncodingFilter.java x reg.html
4
5 import javax.servlet.Filter;
6 import javax.servlet.FilterChain;
7 import javax.servlet.ServletException;
8 import javax.servlet.ServletRequest;
9 import javax.servlet.ServletResponse;
10 import javax.servlet.annotation.WebFilter;
11
12 @WebFilter("/*")
13 public class CharacterEncodingFilter implements Filter {
14
15     @Override
16     public void doFilter(ServletRequest arg0,
17                         ServletResponse arg1,
18                         FilterChain arg2)
19         throws IOException, ServletException {
20         System.out.println("before Filter");
21         arg0.setCharacterEncoding("UTF-8");
22
23         arg2.doFilter(arg0, arg1);
24         System.out.println("after Filter");
25     }
26

```

연습해보기

- 사용자 입력을 통한 계산 요청
 - 숫자 두개를 입력해서 계산해볼것



- 결과

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
5 <title>Insert title here</title>
6 </head>
7 <body>
8     <form action = "add" method="post">
9         <div>
10             <label>계산할 값을 입력하세요</label>
11             <input name="add1" type="text">
12             <input name="add2" type="text">
13         </div>
14         <div>
15             <input type="submit" value = "계산하기" />
16         </div>
17     </form>
18 </body>
19 </html>

```

```

@WebServlet("/add")
public class add extends HttpServlet {
    @Override
    protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException {
        response.setCharacterEncoding("UTF-8");
        response.setContentType("text/html; charset=UTF-8");

        String x_ = request.getParameter("add1");
        String y_ = request.getParameter("add2");

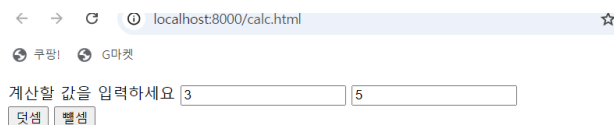
        int x = 0;
        int y = 0;

        if(!x_.equals("")) x = Integer.parseInt(x_);
        if(!y_.equals("")) y = Integer.parseInt(y_);

        int result = x+y;
        response.getWriter().printf("result is %d\n", result);
    }
}

```

- 여러 개의 submit 버튼이 필요할 경우도 처리할 수 있다.



```

1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
5 <title>Insert title here</title>
6 </head>
7 <body>
8 <form action="/calc" method="post">
9 <div>
10 <label>계산할 값을 입력하세요</label>
11 <input name="add1" type="text">
12 <input name="add2" type="text">
13 </div>
14 <div>
15 <input type="submit" name="operator" value = "덧셈" />
16 <input type="submit" name="operator" value = "빼셈" />
17 </div>
18 </form>
19 </body>
20 </html>

```

- submit 의 input 태그에 name을 지정해준다

```

public class Calc extends HttpServlet {
    @Override
    protected void service(HttpServletRequest request, HttpServletResponse response) throws
        ServletException, IOException {
        response.setCharacterEncoding("UTF-8");
        response.setContentType("text/html; charset=UTF-8");

        String x_ = request.getParameter("add1");
        String y_ = request.getParameter("add2");
        String op = request.getParameter("operator");

        int x = 0;
        int y = 0;

        if(!x_.equals("")) x = Integer.parseInt(x_);
        if(!y_.equals("")) y = Integer.parseInt(y_);

        int result=0;

        if(op.equals("덧셈"))
            result = x+y;
        else
            result = x-y;

        response.getWriter().printf("result is %d\n", result);
    }
}

```

- name(operator) 의 value값을 비교하여 덧셈이면 x+y, 그 외(뺄셈) 이면 x-y

입력 데이터 배열로 보내기

- 사용자로부터 여러 개의 입력값을 받아야 할때, form의 input 태그의 name을 x,y,z ... 등 계속 바꿔줘야할까?
- 같은 name을 주고 서버릿에서 배열로 받아올 수 있다.

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
5 <title>Insert title here</title>
6 </head>
7 <body>
8     <form action = "add2">
9         <div>
10             <label>계산할 값을 입력하세요</label>
11             <input name="num" type="text">
12             <input name="num" type="text">
13             <input name="num" type="text">
14             <input name="num" type="text">
15         </div>
16         <div>
17             <input type="submit" value = "계산하기">
18         </div>
19     </form>
20 </body>
21 </html>

```

```

@WebServlet("/add2")
public class add2 extends HttpServlet {
    @Override
    protected void service(HttpServletRequest request, HttpServletResponse response) {
        response.setCharacterEncoding("UTF-8");
        response.setContentType("text/html; charset=UTF-8");

        String[] num_ = request.getParameterValues("num");

        int result = 0;

        for(int i=0; i<num_.length; i++) {
            int num = Integer.parseInt(num_[i]);
            result+=num;
        }

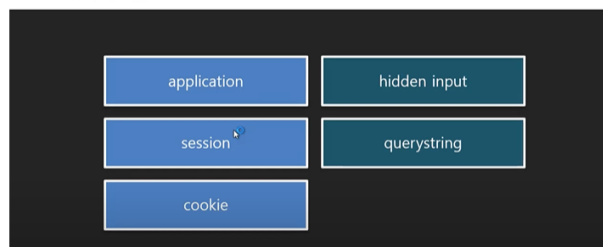
        response.getWriter().printf("result is %d\n", result);
    }
}

```

getParameterValues를 사용해서 num값을 배열로 가져와서 for문을 사용해 result에 값을 누적시킨다.

상태 유지의 필요성

상태 유지를 위한 5가지 방법



Application 객체

- 서버쪽에서 사용자가 전달한 값을 저장할 수 있고, 다음 요청에서 그 값을 꺼내볼 수 있는가?

```

@WebServlet("/calc2")
public class Calc2 extends HttpServlet {
    @Override
    protected void service(HttpServletRequest request, HttpServletResponse response) {
        response.setCharacterEncoding("UTF-8");
        response.setContentType("text/html; charset=UTF-8");
    }
}

```



```

String v_ = request.getParameter("v");
String op = request.getParameter("operator");

int v = 0;

if(!v_.equals("")) v = Integer.parseInt(v_);

//계산
if(op.equals("=")) {

    int x = (Integer)application.getAttribute("value")
    int y = v;
    String operator = (String)application.getAttribute("operator");

    int result=0;

    if(operator.equals("+"))
        result = x+y;
    else
        result = x-y;

    response.getWriter().printf("result is %d\n", result);
}
//저장
else {
    application.setAttribute("value", v);
    application.setAttribute("op", op);
}

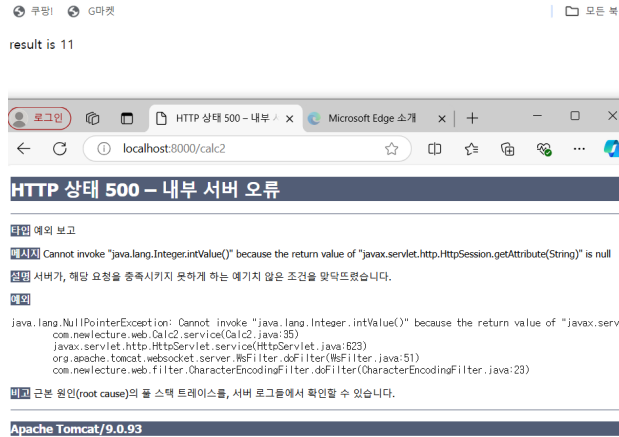
```

application의 getAttribute와 setAttribute를 활용해서 저장하고 가져올 수 있다.

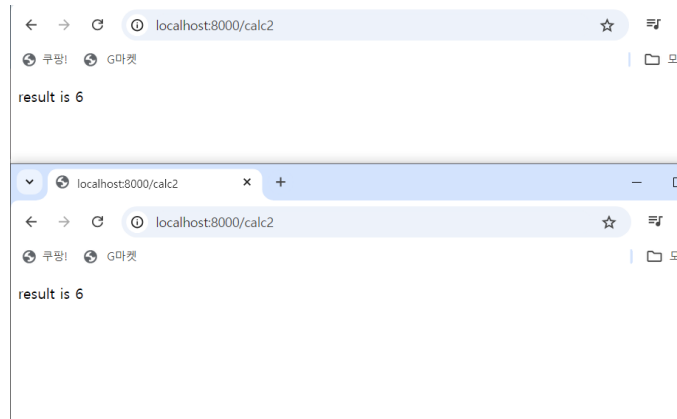
Session 객체로 상태값 저장하기(Application객체와 차이점)

- 현재 접속한 사용자를 뜻함
- 접속한 사용자마다 공간이 달라진다
- 테스트 - 서로 다른 브라우저(다른 사용자)를 이용해서 세션에 저장된 값을 테스트한다

○



- 크롬에서 session에 저장했던 값이 엣지 브라우저에서는 존재하지 않음을 알수 있다 → 사용자마다 공간이 다르다

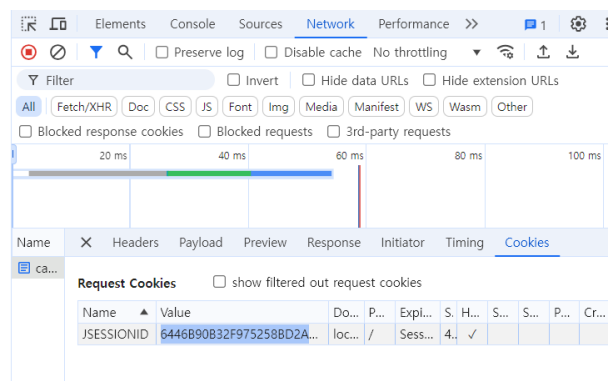
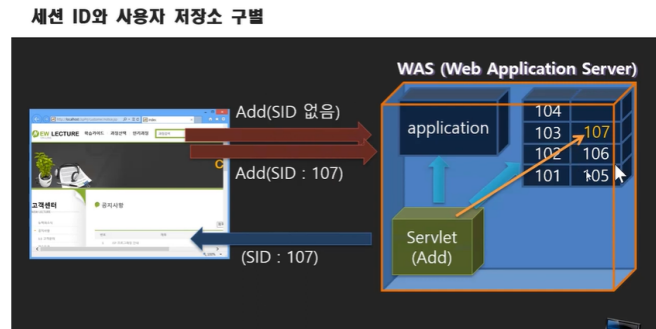


같은 브라우저면 같은 사용자로 인식한다.

Google Chrome(17)	13.1%	1,122.2MB	0.1MB/s	0Mbps
Google Chrome	6.3%	259.0MB	0MB/s	0Mbps
Google Chrome	0%	7.6MB	0MB/s	0Mbps
Google Chrome	1.9%	231.2MB	0MB/s	0Mbps
Google Chrome	0%	2.1MB	0MB/s	0Mbps
Google Chrome	3.5%	255.0MB	0MB/s	0Mbps
Google Chrome	0%	9.8MB	0MB/s	0Mbps
Google Chrome	0%	12.9MB	0MB/s	0Mbps
Google Chrome	0%	14.5MB	0MB/s	0Mbps
Google Chrome	0%	32.1MB	0MB/s	0Mbps
Google Chrome	0%	10.9MB	0MB/s	0Mbps
Google Chrome	0%	85.4MB	0MB/s	0Mbps
Google Chrome	0%	29.5MB	0MB/s	0Mbps
Google Chrome	0.2%	2.2MB	0MB/s	0Mbps
Google Chrome	0%	24.8MB	0MB/s	0Mbps
Google Chrome	0%	17.6MB	0.1MB/s	0Mbps
Google Chrome	0%	0.8MB	0MB/s	0Mbps

- 하나의 프로세스에 하위 흐름을 갖고있는 스레드의 개념으로 창을 띄우기 때문에 프로세스가 가지고 있는 자원을 스레드가 공유하기 때문에 같은 사용자로 인식한다.
- 즉, 창만 여러개 떠있을 뿐 같은 사용자

그렇다면 웹서버는 사용자를 어떤식으로 구분하는가?

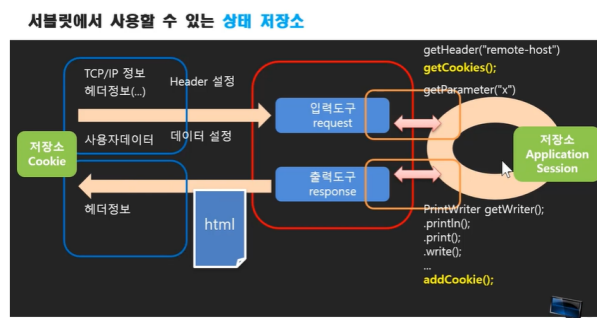


sessionid가 같으면 같은 사용자로 인식한다

- `void setAttribute(String name, Object value)`
 - 지정된 이름으로 객체를 설정
- `Object getAttribute(String name)`
 - 지정한 이름의 객체를 반환
- **`void invalidate()`**
 - 세션에서 사용되는 객체들을 바로 해제
- `void setMaxInactiveInterval(int interval)`

- 세션 타임아웃을 정수(초)로 설정
- 기본 타임아웃은 30분
 - 30분이 지나면 새로운 사용자로 인식된다
- boolean isNew()
 - 세션이 새로 생성되었는지를 확인
- Long getCreationTime()
 - 세션이 시작된 시간을 반환, 1970년 1월 1일을 시작으로 하는 밀리초
- long getLastAccessedTime()
 - 마지막 요청 시간, 1970년 1월 1일을 시작으로 하는 밀리초

Cookie를 이용해 상태값 유지하기



- 클라이언트가 서버에 요청할 때 값을 가지고다닐 수 있다

쿠키 사용하기

쿠키 저장하기

```
Cookie cookie = new Cookie("c", String.valueOf(result));
response.addCookie(cookie);
```

쿠키 읽기

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

if (cookies != null)
    for (Cookie cookie : cookies)
        if ("c".equals(cookie.getName()))
            _c = cookie.getValue();
```

NEW LECTURE ... <http://www.newlecture.com>

```
@WebServlet("/calc2")
public class Calc2 extends HttpServlet {
```

```

@Override
protected void service(HttpServletRequest request, HttpServletResponse application = request.getServletContext()
    HttpSession session = request.getSession();
    Cookie[] cookies = request.getCookies();

    response.setCharacterEncoding("UTF-8");
    response.setContentType("text/html; charset=UTF-8");

    String v_ = request.getParameter("v");
    String op = request.getParameter("operator");

    int v = 0;

    if(!v_.equals("")) v = Integer.parseInt(v_);

    //계산
    if(op.equals("=")) {

//        int x = (Integer)application.getAttribute("value");
//        int x = (Integer)session.getAttribute("value");
        int x = 0;

        for(Cookie c : cookies)
            if(c.getName().equals("value")) {
                x = Integer.parseInt(c.getValue());
                break;
            }

        int y = v;

//        String operator = (String)application.getAttribute("operator");
//        String operator = (String)session.getAttribute("operator");

        String operator = "";
        for(Cookie c : cookies) {
            if(c.getName().equals("op"))
                operator = c.getValue();
        }
    }
}

```

```

        break;
    }

    int result=0;

    if(operator.equals("+"))
        result = x+y;
    else
        result = x-y;

    response.getWriter().printf("result is %d\n", res
}
//저장
else {
//    application.setAttribute("value", v);
//    application.setAttribute("op", op);
//    session.setAttribute("value", v);
//    session.setAttribute("op", op);
    Cookie valueCookie = new Cookie("value", String.v
    Cookie opCookie = new Cookie("op", op);
    response.addCookie(valueCookie); //사용자에게 쿠키를
    response.addCookie(opCookie);
}

}

}

```

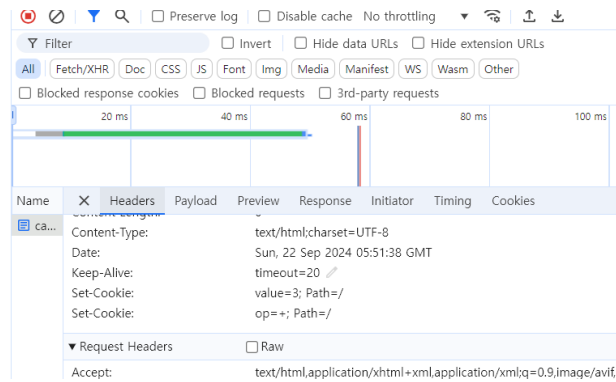
Cookie의 path 옵션

- Cookie를 사용할 때 url과 관련된 서블릿에게만 값이 전달될 수 있도록 활용할 수 있다.
-

```

Cookie valueCookie = new Cookie("value", String.valueOf(v)); //value값이 반드시 문자열 형태(url에 사용할 수 있는 형태)로 보내야한다
Cookie opCookie = new Cookie("op", op);
valueCookie.setPath("/"); //이 쿠키가 어느경우의 사용자로부터 전달되어야 하는지의 경로
opCookie.setPath("/");
response.addCookie(valueCookie); //사용자에게 쿠키를 보냄
response.addCookie(opCookie);
}

```



- path 값이 설정된 것을 알 수 있다

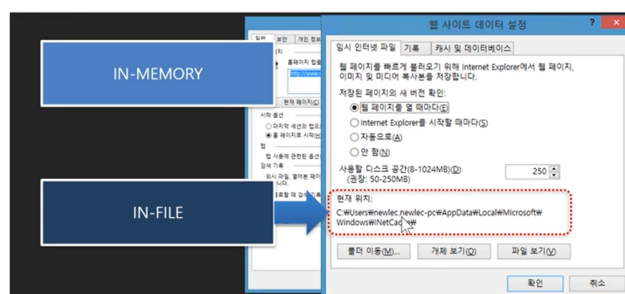
→ setPath를 사용하면 쿠키(response header)에 지정한 path경로가 같이 전송된다.

해당 경로로 접근하지 않으면 쿠키가 따라오지 않는다.

해당 경로로 접근한 경우 저장한 쿠키값이 달려 오는 것을 확인할 수 있다.

이는 불필요한 데이터 전송 방지, 변수이름 중복 방지의 효과가 있다.

Cookie의 maxAge 옵션



- 만료날짜에 따라서 쿠키에 대한 기간을 설정할 수 있다.

```

Cookie valueCookie = new Cookie("value", String.valueOf(v)); //value를 반드시 문자열 형태(url에 사용할 수 있는 형태)로 보내야한다
Cookie opCookie = new Cookie("op", op);
valueCookie.setPath("/calc"); //이 쿠키가 어느경우의 사용자로부터 전달되어야 하는지의 경로
valueCookie.setMaxAge(24*60*60); //이 쿠키를 보내면 그 시간으로부터 이후동안 쿠키값을 유지한다.
opCookie.setPath("/calc");
response.addCookie(valueCookie); //사용자에게 쿠키를 보냄
response.addCookie(opCookie);
}

```

Application / Session / Cookie의 차이점 정리

• Application

- 사용범위 : 전역 범위에서 사용하는 저장 공간
- 생명주기 : WAS가 시작해서 종료할 때 까지
- 저장위치 : WAS 서버의 메모리

• Session

- 사용 범위 : 세션 범위에서 사용하는 저장 공간
- 생명 주기 : 세션이 시작해서 종료할 때 까지
- 저장 위치 : WAS 서버의 메모리

• Cookie

- 사용 범위 : Web Browser별 지정한 path 범주 공간
- 생명 주기 : Browser에 전달한 시간부터 만료시간 까지
- 저장 위치 : Web Browser의 메모리 또는 파일
- 저장할 기간이 길다 → Cookie를 사용해야한다
- 특정 URL (범위) 에서만 데이터를 쓰는 것 또한 쿠키를 쓰는것이 바람직하다. (서버자원에 대한 부담을 줄일 수 있다.)

•