




## 7월 18일 - service, forward, rdirect, cookie, session

2018년 7월 18일 수요일 오전 9:09

W3schools 에서 참고할 만한 것들

Placeholder (해당 칸ㅇ 무엇을 써야하는지 힌트를 주는 것)

라디오 버튼



Run >

```
<!DOCTYPE html>
<html>
<body>



<h2>Radio Buttons</h2>

<form>
  <input type="radio" name="gender" value="남" checked> Male<br>
  <input type="radio" name="gender" value="female"> Female<br>
  <input type="radio" name="gender" value="other"> Other
</form>

</body>
</html>
```

### Radio Buttons

☒ Male  
☐ Female  
☐ Other



Run >

```
<!DOCTYPE html>
<html>
<body>

<h2>The target Attribute</h2>
<p>When submitting this form, the result will be opened in a new browser tab:</p>

<form action="/action_page.php" target="_blank">
  First name:<br>
  <input type="text" name="firstname" value="Mickey">
  <br>
  Last name:<br>
  <input type="text" name="lastname" value="Mouse">
  <br><br>
  <input type="submit" value="Submit">
</form>

</body>
</html>
```





### The target Attribute

When submitting this form, the result will be opened in a new browser tab:

First name:

Last name:

타겟을 지우지 않으면 새창으로 가지만



Run >

```
<!DOCTYPE html>
<html>
<body>

<h2>The target Attribute</h2>
<p>When submitting this form, the result will be opened in a new browser tab:</p>

<form action="/action_page.php" >
  First name:<br>
  <input type="text" name="firstname" value="Mickey">
  <br>
  Last name:<br>
  <input type="text" name="lastname" value="Mouse">
  <br><br>
  <input type="submit" value="Submit">
</form>

</body>
</html>
```

### Submitted Form Data

Your input was received as:

firstname=Mickey&lastname=Mouse

The server has processed your input and returned this answer.

**Note:** This tutorial will not teach you how servers are processing input. Processing input is explained in our [PHP tutorial](#).

타겟을 지웠을 때

키를 여러 개를 써서 각각 전송할 때  
Web query string 이라고 표현한다.

## When to Use GET?

The default method when submitting form data is GET.

However, when GET is used, the submitted form data will be **visible in the page address field**:

/action\_page.php?firstname=Mickey&lastname=Mouse

### Notes on GET:

- Appends form-data into the URL in name/value pairs
- The length of a URL is limited (about 3000 characters)
- Never use GET to send sensitive data! (will be visible in the URL)
- Useful for form submissions where a user want to bookmark the result
- GET is better for non-secure data, like query strings in Google

키를 여러 개를 써서 각각 전송할 때  
Web query string 이라고 표현한다.

데이터 양이 많거나 보안이 있으면 get으로 쓰면 안된다.

따라서 post를 써야 한다.

## When to Use POST?

Always use POST if the form data contains sensitive or personal information. The POST method does not display the submitted form data in the page address field.

### Notes on POST:

- POST has no size limitations, and can be used to send large amounts of data.
- Form submissions with POST cannot be bookmarked

```
<!DOCTYPE html>
<html>
<body>

<h2>The select Element</h2>
<p>The select element defines a drop-down list:</p>

<form action="/action_page.php">
  <select name="cars">
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
    <option value="fiat">Fiat</option>
    <option value="audi">Audi</option>
  </select>
  <br><br>
  <input type="submit">
</form>

</body>
</html>
```

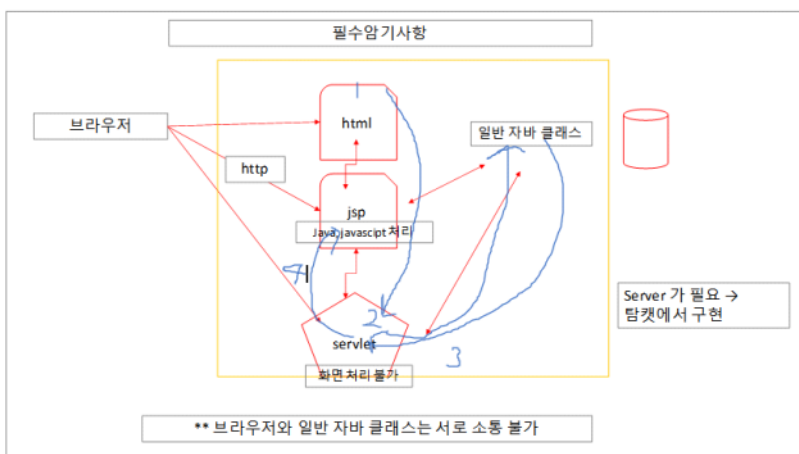
### The select Element

The select element defines a drop-down list:

Volvo ▼

제출

셀렉트를 통해 무엇을 먼저 오게 할지 바꿀 수 있다.



한글 데이터 입력 노노

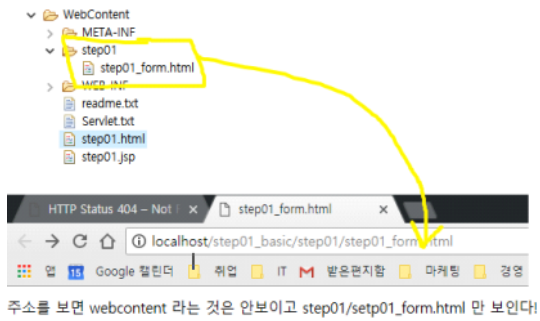
한글 받았다고 서버렛에서 처리 안함

단순히 서버렛에서 브라우저로 한글이 보일 수 있도록 코딩을 작업하는 것

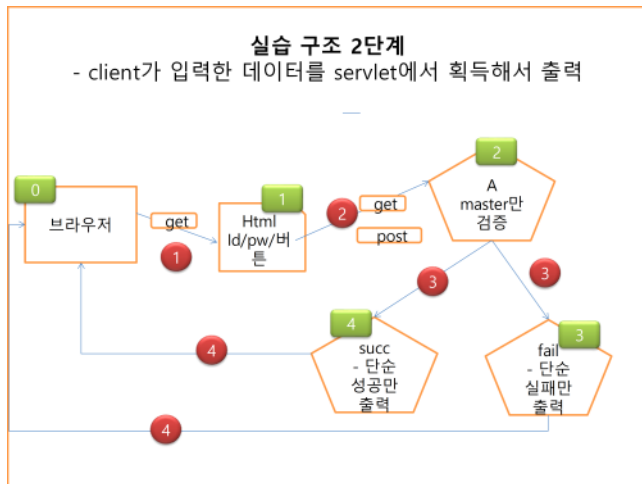
>> 한글 인코딩 작업 해야함 // 2byte 출력 가능한 객체 생성

아이디에다가 한글 데이터 쓰라고 하는 곳 없음

응답할때는 인코딩 설정만  
리퀘스트는 request.setCharacterEncoding("uft-8") 로 하기



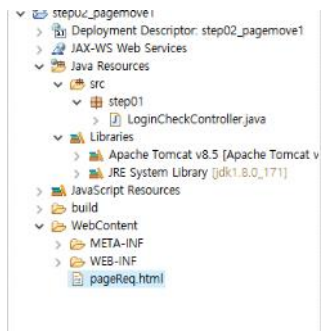
## Step2



서브렛에서master만 검증  
그런데 이것이 성공하면  
실패하면  
이라는 두가지 경우가 나뉘어짐

성공하면 >> 브라우저로 출력 (원하는 사이트 접속 가능)  
파일이라면 실패했다는 정보를 브라우저에 출력 (단순 실패)  
따라서 이 부분을 구현해야 한다.

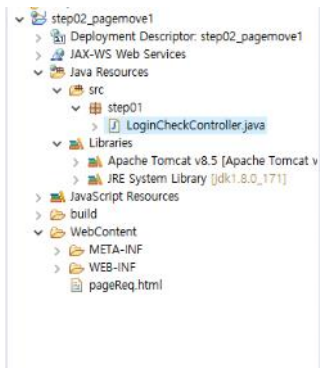
해보자!



```

1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="EUC-KR">
5 <title>Insert title here</title>
6 </head>
7 <body>
8 <h3> servlet을 이용한 다수의 웹 페이지 구성 학습</h3>
9 <form action="/cont" method="post">
10 id : <input type="text" name="id"> <br>
11 pw : <input type="password" name="pw"> <br>
12 <input type="submit" value="로그인">
13 </form>
14 </body>
15 </html>
16

```



```

1 package step01;
2
3 import java.io.IOException;
4
5 @WebServlet("/cont")
6
7 public class LoginCheckController extends HttpServlet {
8     private static final long serialVersionUID = 1L;
9
10    // service 는 get or post 모든 방식 처리하는 메소드, 재정의
11    protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
12    }
13
14 }
15
16
17
18
19
20

```

이제는 서비스로 포스트와 갯 모두를 처리해보자'

설정을 do get do post 빼고 service만 넣으면 된다.



포워딩 방식

서브렛이 다르다고 하더라도 데이터 손실 안된다.

서블렛 api를 통해 페이지 이동을 해보자

실행은 서버렛에서 하나—개 아니다!

Data = admin인것과 아닌것

이제는 html 에서 servlet으로 데이터를 넘기는 것이 아니라

Servlet에서 servlet으로 데이터를 넘긴다.  
이때 메소드로 데이터를 담아서 준다.  
이 메소드 또한 정해져 있다.

## ■ 학습 내용 (웹 페이지 이동)

### 1.web page 이동 개발 기술

- html/jsp/servlet간의 이동
- 절대 일반 자바 처럼 단순 메소드 호출과 무관

### 2.이동 기술 종류

#### 1.html tag를 이용한 방식

- 1.링크 tag : <a>
- 2.form의 submit
- 3.일반 button의 onclick

#### 2.Servlet api를 이용한 방식

- 1.forward 방식
  - 데이터 손실 없음
  - 서로 다른 servlet이라 하더라도 하나의 servlet처럼 간주
- 2.redirect 방식
  - servlet간에 redirect가 발생시 무조건 새로운 요청으로 간주

### 3.주요 API

#### 1.javax.servlet.http.HttpServlet

- public void doGet()
- public void doPost()
- public void service()

#### 2.javax.servlet.http.HttpServletRequest

- 접속된 client만의 요청 정보 보유한 객체
- 1.String getParameter()
  - web query string 구조로 전송되는 데이터 값 획득
  - String으로만 반환
- 2.getRequestDispatcher()

#### 3.public void setAttribute(String key, Object value)

- 요청 객체에 데이터를 map 형태로 저장

#### 4.public Object getAttribute(String key)

- 요청 객체에 setAttribute()로 저장한 데이터만 반환

#### 3.javax.servlet.http.HttpServletResponse

- 1.setContentType()
- 2.getWriter()
- 3.sendRedirect()

### 4.uri의 변천사

#### 실행 process

#### 1.유효 : forward 방식

pageReq.html -> LoginCheckController -> Success  
: [http://localhost/step02\\_pageMove/cont](http://localhost/step02_pageMove/cont)  
주소 갈아타도, 내용 달라져도 주소줄은 계속 같다.  
ex) 네이버 카페

#### 2.무효 : redirect 방식

pageReq.html -> LoginCheckController -> Fail  
: [http://localhost/step02\\_pageMove/fail](http://localhost/step02_pageMove/fail)  
최종 uri 값이 그대로 오픈된다  
ex) 네이버 블로그

#### 참고)

get post 데이터를 어떻게 전송하느냐의 방식 // input 값을 어떻게 처리하느냐

링크랑 홈버튼, 리다이렉트, forward : 웹페이지 이동 방식

언제만 forward가 있고, 언제만 redirect 만 쓰게 아니다

메일의 경우, 복사는 redirect 로 복사되더라도 어차피 로그인을 해야 넘어가는 보안을 걸

어났기 때문에 굳이 forward를 쓰지 않아도 가능하다.

```

9 @WebServlet("/cont")
10 // container = server (웹킷과 같은) 즉, 웹킷이 호출을 한다!
11
12 public class LoginCheckController extends HttpServlet {
13
14     // service 는 get or post 모든 방식 처리하는 메소드, 자정의
15     protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
16         // client가 입력한 데이터 획득
17
18         String id=request.getParameter("id");
19         System.out.println(request.getRemoteAddr()); // 내 서버에 지수의 브라우저들 통해서 주소들 치고 들어오면 지수가 클라이언트가 된다. 지수의 ip 주소들 띄워주는 것 (콘솔창에)
20         System.out.println(request.getRemoteHost()); // addr. 같을 것!
21     }
22     //검증
23     /*
24     유효한 경우
25     - succ url 과 매핑된 servlet으로 이동
26     - forward로 이동
27     무효한 경우
28     - fail url 과 매핑된 servlet 으로 이동
29     - redirect로 이동
30     */
31     if(id.equals("admin")) {
32         request.setAttribute("newData", "요청에 새로운 데이터 저장");
33         // cont와 succ는 다른 servlet이긴 하나, 하나의 servlet으로 간주한다.
34         // 요청과 응답 객체가 공유가 되는 상황
35         request.getRequestDispatcher("succ").forward(request, response); // success 화면 이동
36     } else {
37         request.setAttribute("newData", "요청에 새로운 데이터 저장");
38         // 응답완료, 응답시 새로 요청할 fail url 정보를 client 브라우저에게 전송
39         // fail로 새로운 요청을 브라우저가 자동실행
40         // 단, fail 예전 새로운 요청이기 때문에 새로운 request, response를 새로 생성
41         response.sendRedirect("fail"); // fail 이라는 응답 >> 새로운 요청을 만든다 redirect
42     }
}

```

```

1 package step01;
2
3 import java.io.IOException;
4
11
12 @WebServlet("/succ")
13 public class SuccessView extends HttpServlet {
14     protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
15         // 한글로 무효합니다. 출력
16         // 한글 인코딩 처리, 응답 포맷 설정하기
17         response.setContentType("text/html;charset=utf-8");
18         // 출력
19         PrintWriter out = response.getWriter();
20         out.println("유료~~~ <br>");
21         out.println(request.getAttribute("newData"));
22     }
23

```

```

1 package step01;
2
3 import java.io.IOException;
4
11
12 @WebServlet("/fail")
13 public class FailView extends HttpServlet {
14     protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
15         // 한글로 무효합니다. 출력
16         // 한글 인코딩 처리, 응답 포맷 설정하기
17         response.setContentType("text/html;charset=utf-8");
18         // 출력
19         PrintWriter out = response.getWriter();
20         out.println("무효<br>");
21         out.println(request.getAttribute("newData"));
22     }
23

```

우리 입장에서 네이버 > 뉴스 > 해당 뉴스 기사 클릭

그런데 네이버에서는 이런 행동파악 다 몰라도 됨  
그냥 이런 url 을 들어왔다고 파악

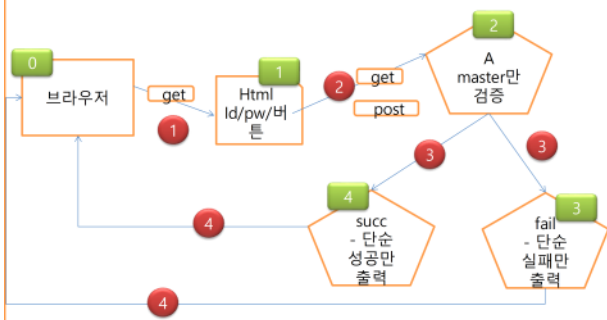
그런데 로그인 하고 나서는?  
로그인 절차 인증 절차 필수  
로그인 성공 → ... 로그아웃  
그동안의 모든 행동들 파악 가능

이는 쿠키와 server 개념과 연결된다.  
이를 좀더 자세히 배워보자

## Step 03

## 실습 구조 3단계 - 쿠키, 세션

- client가 입력한 데이터를 servlet에서 획득해서 출력



### 학습내용

#### 1. http기본 특징

- client의 정보를 기본적으로 저장 및 관리 하지 않음
- 무상태 연결 유지 (비연결 지향형)

#### 2. 로그인 ~ 로그아웃 할 때 까지의 상태 유지 기술

- 개발자가 직접 코드로 개발을 해야 한다.

#### 3. 메카니즘

##### 1. client pc에 상태 유지값 저장

- 문자열만 저장 가능
- 쿠키 개발 기술
  - : 오라클 jdk 설치할때 너 쿠키 허락해줘 강제 했던 것 처럼
  - : javax.servlet.http.Cookie 이름으로 저장하기
- 개발 단계
  - 1단계 : Cookie 객체 저장할 문자열 데이터 수만큼 생성
  - 2단계 : client 시스템에 잔존시킬 시간 설정
  - 3단계 : client 시스템에 전송해서 쿠키 저장
  - 4단계 : 쿠키가 저장되어 있으니, servlet으로 이동
  - 5단계 : 클라이언트 시스템으로 부터 쿠키 정보에 저장된 데이터 획득
  - 6단계 : 삭제

##### 2. server (container, tomcat) 의 상태 유지값 저장

- 객체 타입으로 저장가능
  - 세션 개발 기술
    - : javax.servlet.http.HttpSession
  - 개발 단계
    - 1단계 : HttpSession 객체를 하나만 생성
    - 2단계 : server 메모리에 저장할 데이터 수 만큼 세션에 데이터 저장 함 (쿠키는 문자였는데, 세션은 데이터 수 만큼 생성 // 중요중요)
      - setAttribute () 키 밸류 로 저장
      - 데이터 수 만큼 setAttribute 호출 >> 10개면 10개를 만들면 되는데, key 값이 달라진다.
    - 3단계 : servlet 이동
    - 4단계 : HttpSession 으로 부터 저장된 데이터 획득 - getAttribute()
    - 5단계 : 삭제
      - HttpSession 세션 기능 아예 무효화
        - invalidate()
        - null 값으로 정리
      - 생략시 발생하는 문제
        - 쿠키는 삭제 안해도 클라이언트 서버이니까 굳이 내 거에 영향을 미치지 않는데, server의 경우, 우리 서버 터질 수 있다.
- client 시스템에 잔존시킬 시간 설정

이제 쿠키를 만들어보자

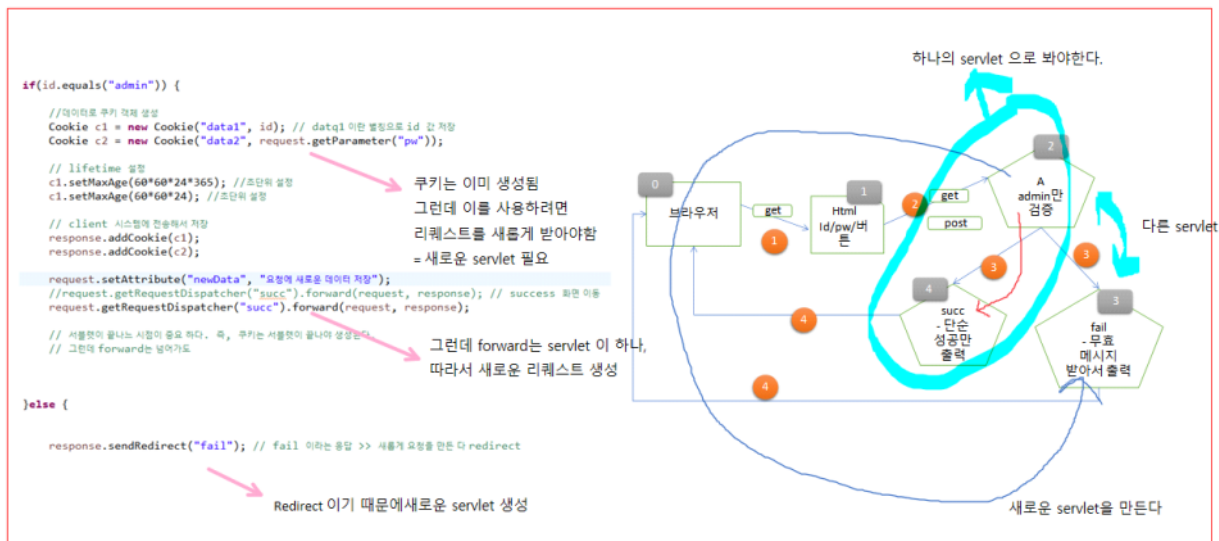
```

1 package step01.cookie;
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.Cookie;
9 import javax.servlet.http.HttpServlet;
10 import javax.servlet.http.HttpServletRequest;
11 import javax.servlet.http.HttpServletResponse;
12
13 @WebServlet("/succ")
14 public class SuccessView extends HttpServlet {
15     protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
16         // 한글로 무표합니다. 출력
17         // 한글 인코딩 처리, 응답 포맷 설정하기
18         response.setContentType("text/html;charset=utf-8");
19         // 출력
20         PrintWriter out = response.getWriter();
21
22         /*
23          client 시스템에 저장된 쿠키 삭제 획득 및 출력
24          쿠키 삭제 획득 메소드
25          1. 요청 객체가 제공
26             Cookie[] getCookies()
27          2. Cookie 객체가 제공
28             String getName() - key 반환
29             String getValue() - value 반환
30          */
31
32         Cookie[] all = request.getCookies();
33         for(int i = 0; i < all.length; i++) {
34             // Cookie c1 = new Cookie("data1", id); 이를 위해
35             if(all[i].getName().equals("data1")) {
36                 out.print(all[i].getValue());
37             }
38         }
39     }
40
41     out.println("유료~~~ <br>");
42     out.println(request.getAttribute("newData"));
43 }
44 }
45 }
46

```

오류가 나는 이유

Request 가 새로 요청되었을 때 당시의 쿠키 정보가 있나 없나가 중요하다  
Request가 쿠키를 쓸 수 있나 없나



따라서 이를 고치면 하기와 같다.



```

1 package step01.cookie;
2
3 import java.io.IOException;
4 @WebServlet("/cont")
5
6 public class LoginCheckController extends HttpServlet {
7
8     protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
9
10         String id=request.getParameter("id");
11
12         /*
13         학습내용
14         1. 쿠키 데이터 사용 시점에 대한 test
15         2. request 객체가 생성되는 시점에 브라우저에 Cookie 정보 존재 여부 확인 필수
16         3. 논리적으로 쿠키 생성 코드는 응답 완료된 화면이동방식 사용
17             (link, button click, redirect ) >> 예내들끼리는 응답완료되고 새로운 객체 생성이라 계속해서 request 생성이라고 보면 된다.
18         */
19
20         if(id.equals("admin")) {
21
22             request.setAttribute("newData", "요청에 새로운 데이터 저장");
23             //request.getRequestDispatcher("succ").forward(request, response); // success 화면 이동
24             request.getRequestDispatcher("succ").forward(request, response);
25
26             // 서블렛이 끝나는 시점이 중요하다. 즉, 쿠키는 서블렛이 끝나야 생성된다.
27             // 그런데 forward는 넘어가도
28
29         }else {
30             //데이터로 쿠키 객체 생성
31             Cookie c1 = new Cookie("data1", id); // data1이란 별칭으로 id 값 저장
32             Cookie c2 = new Cookie("data2", request.getParameter("pw"));
33
34             // lifetime 설정
35             c1.setMaxAge(60*60*24*365); //조단위 설정
36             c2.setMaxAge(60*60*24); //조단위 설정
37
38             // client 시스템에 전송해서 저장
39             response.addCookie(c1);
40             response.addCookie(c2);
41
42             response.sendRedirect("fail"); // fail 이라는 응답 >> 새롭게 요청을 만든 다 redirect
43         }
44     }
45 }

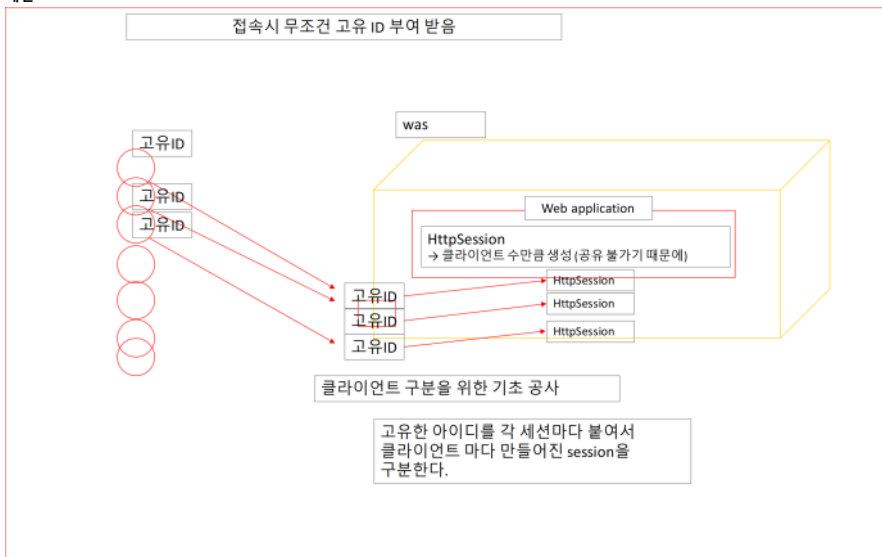
```

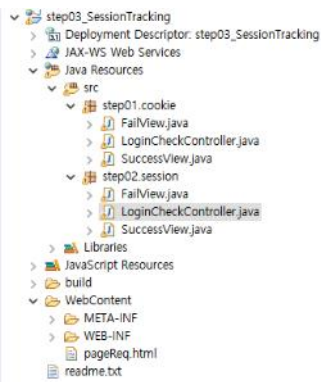
쿠키 배웠으니, 이제는 세션에 대해 배워보자

참고 ) 메타 정보 : 클라이언트에게 보여주고자 하는 정보는 아니지만 컨테이너 만의 정보

## 세션

세션





```
1 package step02.session;
2
3 import java.io.IOException;
11 @WebServlet("/cont2")
12
13 public class LoginCheckController extends HttpServlet {
14
15     protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
16
17         String id=request.getParameter("id");
18
19
20
21         if(id.equals("admin")) { // 코난
22             //새로 세션 객체 생성 - 이 user 에게 id 값으로 비교해서 id가 동일한 세션 객체 없었다 라고 하면 생성
23             HttpSession session = request.getSession();
24             // 세션에 이름 저장
25             session.setAttribute("name", "코난"); // session 완전 중요
26
27
28             request.setAttribute("newData", "요청에 새로운 데이터 저장");
29             request.getRequestDispatcher("succ2").forward(request, response);
30
31         }else {
32             response.sendRedirect("fail2"); // fail 이라는 응답 >> 새롭게 요청을 만든 다 redirect
33         }
34     }
35 }
36 }
37 }
38
39
```

```
1 package step02.session;
2
3 import java.io.IOException;
13
14 @WebServlet("/succ2")
15 public class SuccessView extends HttpServlet {
16     protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
17         // 한글로 무표합니다. 출력
18         // 한글 인코딩 처리, 응답 포맷 설정하기
19         response.setContentType("text/html;charset=utf-8");
20         // 출력
21         PrintWriter out = response.getWriter();
22
23         // 이미 존재하는 HttpSession 객체를 받아옴
24         // 컨테이너는 server (tomcat)로 부터 id를 비교 당해서 존재하면 반환
25         // 없으면 새로 생성
26         HttpSession session = request.getSession();
27
28         // 세션에 저장된 이름을 화면에 출력
29         out.print(session.getAttribute("name") + "<br>");
30         session.invalidate();
31
32         out.print(2 + " " +session.getAttribute("name")); // 진짜 지워졌는지 확인하기
33         session = null;
34
35         out.println("유료~~~ <br>");
36         out.println(request.getAttribute("newData"));
37
38
39     }
40 }
41 }
42
```

\*\* 주의! 실행은 pageReq.html 에서 실행해야된다.

복습

오늘 배운거 복습해보자!

4. 웹 페이지 이동 데이터 저장 방식 (웹 페이지 이동 방식에 따른 데이터 저장 및 활용 방식)

1. 페이지 이동
  1. html
    - 링크 / 버튼 (일반버튼 (onclick), or form의 submit)
  2. servlet api
    - 포워드 / 리다이렉트
2. 포워드의 특징에 대해서 보자.
  - 다중 서블릿이라고 하더라도, 응답 요청 객체가 한번만 생성되고, 이를 공유하는 관점
3. 리다이렉트
  - 이동되는 페이지는 새로운 요청으로 인식이 되어 요청 객체와 응답 객체가 새롭게 생성이 됨.
  - 따라서 클라이언트의 정보를 리퀘스트로부터 뺏아올 수 있다. (획득 가능)
4. 상태 유지를 위한 데이터 저장 방식
  1. cookie
    - client 시스템에 문자열로만 저장
    - 생성 >> 잔존시간 설정 >> client 에 전송 >> 이동된 servlet에서 Cookie 객체들 한번에 배열로 뺏음 >> getName() / getValue()로 쿠키 정보 활용 가능
  2. HttpSession을 위한 것
    - server 시스템 메모리에 저장
    - page 이동 방식과 무관하게 서버 메모리의 데이터를 저장 및 활용 가능

5. 최종 정리

1. request.setAttribute(key, value)
  - 이 요청 객체에 저장한 데이터를 활용할 수 있는 page 이동 방식?
  - 여기서 저장한 데이터란 쿠키를 말하는 것이 아니다.
  - 답 : forward
2. session.setAttribute()
  - 이 세션 객체에 저장한 데이터를 활용할 수 있는 page 이동 방식?
  - forward / redirect 둘 다 이용
  - 서버에 데이터가 다 있기 때문에

