

Student ID	2017314626
Name	이재훈

Report for Assignment #1

1. Environment

* 본 프로그램은, virtualBox와 같은 가상 머신에서의 테스트가 아닌, 같은 인터넷을 사용하는 두 개의 실제 기기에서, 우분투 기기에서 서버를 돌리고, 윈도우 기기에서 해당 우분투 기기의 ip로 접속하여 테스트를 완료하였습니다.

- Describe your development environment information in detail. (Versions of operating systems, what kinds of language or tools that you used, and etc).

-개발 환경

OS: Ubuntu 18.04/Windows 10 (두 버전 모두에서 번갈아가며 개발 및 테스트하였습니다.)

사용한 언어: html, python (version = 3.9)

개발 편집기: vs code (vs code 상으로 파이썬 서버 코드 실행하였습니다.)

-프로세서: i5-7200U

-메모리: 8GB

2. Design

- Explain how to design your program in the view of data structures and algorithms.

-소켓 프로그래밍

프로그램을 구현함에 있어서, 소켓 모듈을 사용하였습니다. 서버 소켓을 로컬 ip와 10090 포트를 가지는 주소에 bind를 시키고, 해당 소켓을 listen을 하면 클라이언트의 요청을 받을 수 있는 상태가 됩니다. 그 상태에서 webSocket (실제 웹소켓 모듈이 아닌, 제가 만든 소켓의 이름입니다.)으로 accept를 해주면, 이제 해당 webSocket을 통해 클라이언트-서버 간의 데이터 교환이 이루어질 수 있게 됩니다. 이러한 소켓을 통해 서버가 브라우저(클라이언트)와 http 메시지를 주고받는 방식으로 구현하였으며, 소켓의 recv 크기는 65535 바이트입니다. 이 크기는, recv가 한 번에 받을 수 있는 데이터의 크기를 뜻합니다. 즉, 전달된 데이터의 크기가 100000이라면 recv를 2번 해야 전체 데이터를 받을 수 있게 됩니다. 멀티쓰레딩을 이용하여

여러 브라우저에서 동시에 접속이 가능하도록 구현하였으며, http 1.1 이상에서 지원하는 persistent-http 기능을 활용하였습니다. 또한, 클라이언트에서의 쓸데없는 메시지인 favicon.ico가 들어오는 것을 방지하기 위해, 모든 html 파일에

```
<link rel="icon" href="data:,">
```

를 사용하였습니다. 또한 기본적으로 소켓을 통해 주고받는 데이터들은 encoding이 된 byte 상태로 전달이 됩니다. 따라서 메시지를 해석하기 위해서는 decoding의 과정이 필요합니다.

-http 메시지의 구조

소켓이 리시브하는 http의 메시지는 헤더와 바디로 이루어져 있으며, 둘은 'WrWnWrWn'으로 구분됩니다. 헤더에는 메시지의 종류, 크기 등 http 메시지의 전반적인 정보가 들어있으며, 바디에는 해당 메시지가 담고 있는 정보가 있습니다. 즉, id/pw를 post 방식으로 전달하거나, 파일 전송 등을 할 때 해당 파일의 데이터들이 http 바디에 포함이 됩니다. 이 때 파일이 들어올 경우 html에서 파일 전송을 위해 multipart 형식으로 form을 제공하기 때문에, 헤더에는 해당 메시지의 종류가 multipart라고 표시가 됩니다. 이를 통해 해당 메시지가 파일을 포함한 서버에 파일을 업로드하는 요청인지 다른 요청인지 구분하였습니다. 만약 해당 메시지(multipart)가 없을 시, 메시지는 로그인/index 사이트로 접근/쿠키 사이트로 접근 등 다른 작업을 위한 메시지라고 판단 가능합니다. 메시지에서 특정 메시지가 포함됐는지를 판단하기 위해서는 find/rfind 메소드가 사용되었습니다. 또한, 파일의 데이터는 utf-8로 decode가 불가능하여 오류가 날 수 있기 때문에, 파일이 들어온 경우 body 부분은 decode를 하지 않도록 구현하였습니다. (decode를 하지 않더라도 바이트로도 find 메소드가 사용 가능합니다.)

-서버에서 클라이언트로 html 전송

브라우저의 주소창에 주소를 입력하면, 클라이언트에서는 서버로 get 방식으로 메시지를 전달합니다. 이 때, get뒤에 어떤 주소를 요청하였는지가 헤더 메시지의 두번째 인덱스(message.split()[1])에 나타납니다. (eg. Index.html, cookie.html, storage.html) 이를 활용하여, 헤더를 포함한 메시지의 두번째 인덱스에서 클라이언트가 요청하는 html 파일을 알아낸 후, 해당 페이지를 다시 클라이언트로 전송시켜 주는 것이 기본적인 서버-클라이언트 간 통신 방식입니다. 이 때, 해당 페이지는 http의 헤더 부분 (http/1.1 200 OK ... (생략) WrWnWrWn) 이후 바디 부분으로 포함되어 보내집니다.

-index 페이지에서 로그인 할 때

Index 페이지에서 로그인을 하면, index page의 input data인 id와 pw가 서버에 post 방식으로 전달이 됩니다. Post 방식의 데이터는 http 메시지의 바디 부분에 나타납니다. 따라서 서버에서는 해당 메시지의 바디 부분에서 id가 무엇인지 알아낸 후, storage.html 파일을 열고

user1 부분 (기본으로 설정된 유저의 id값)을 바디에서 얻어온 id값으로 바꾸게 됩니다. 또한 이 때 받은 id를 이용하여 os 모듈을 통해 해당 id의 디렉토리가 없다면 만들어줍니다. (해당 디렉토리에 파일 저장을 하기 위해서) 로그인하여 넘어간 storage 페이지에는 user의 id와 해당 id가 가진 파일 목록이 보여야 하기 때문에, os 모듈을 이용하여 디렉토리에 파일 목록을 http 메시지의 바디(html)에 추가하여 줍니다. 파일 목록을 보는 것 또한 os 모듈을 이용합니다. 그 후 http 메시지를 클라이언트로 전송하여 user의 id와 파일 목록이 포함된 페이지가 클라이언트에 보이게 됩니다.

-storage 페이지의 업로드 기능

업로드 기능은, 파일을 클라이언트에서 서버로 저장하는 기능입니다. 파일을 전송하기 위해서는 위에서 말했듯 multipart/data 형식을 사용해야 합니다. (html에서 enctype을 설정해주지 않고 기본 설정을 사용할 시, 파일의 실제 데이터는 전달이 되지 않습니다.) 파일은 다른 메시지들이 그렇듯 인코딩되어 전달이 되는데, 바디에서 파일의 데이터라고 표시해주는 문구가 webkit입니다. Webkit 문구가 헤더에서 나오게 되고, 바디에서는 해당 webkit 문구로 데이터들이 구분이 됩니다. 그러나 해당 데이터를 저장하기 위해서는 메시지에서 webkit 문구 및 헤더 부분이 삭제된, 진짜 데이터 부분만 파일로 저장을 해야 합니다. 또한 파일의 경우 데이터가 크기 때문에, 위의 (-소켓 프로그래밍) 부분에서 말했듯 recv를 여러 번 해야 할 수 있습니다. 따라서 while문을 통해 파일의 모든 데이터를 받을 때까지 recv를 반복하였고, 반복하며 얻는 데이터는 기존에 받은 데이터에 계속하여 append 형식으로 저장하였습니다. Webkit의 마지막 부분은 '--wrwn'으로 끝나기 때문에, 추가적으로 받은 새 메시지의 마지막 부분이 '--wrwn'이 된다면 while문을 탈출하도록 하였습니다. 이런 과정을 통해 전체 파일의 데이터를 받으면, 전체 데이터를 저장할 차례입니다. 파일의 이름 또한 메시지에서 'filename' 부분을 find하여 찾고, os 모듈을 사용하여 현재 디렉토리에 '/user_id(로그인 또는 쿠키를 통해 받음)/filename' 형식의 파일을 open 하고, 헤더 및 webkit 문구를 뺀 전체 데이터를 write하면 클라이언트->서버로의 파일 전송이 완료됩니다. 파일 전송을 완료하면, 유저의 파일 리스트가 갱신이 되기 때문에 새로운 html 코드가 포함된 http 메시지를 다시 클라이언트로 전송해줍니다.

*http 메시지에서 헤더 및 바디의 내용을 추출한 방식

위에서 http 메시지에서 헤더 및 바디를 구분하고, webkit 메시지를 제거한 진짜 파일 데이터만을 추출한다고 하였습니다. 해당 과정의 자세한 과정을 서술합니다. 먼저, 헤더와 바디는 가장 첫번째로 오는 'wrwnwrwn'에 의해 구분이 됩니다. 이를 이용하여 헤더를 먼저 얻습니다. 특이한 점은, 헤더와 바디가 함께 오는 경우가 있고, 첫번째 recv에는 헤더만 오고 두번째부터 바디가 오는 경우가 있습니다. 이 두 경우를 모두 처리하였습니다. 파일 전송 시 생기는 Webkit 메시지의 경우에도, '-----webkit: ... (생략)' 등의 문구를 parsing하여 모두 삭제하였습니다.

-storage 페이지의 다운로드 기능

해당 기능은, 클라이언트가 서버에 저장해 놓은 파일을 다시 다운로드 받는 기능입니다. 요청하는 파일의 이름이 메시지를 통해 서버로 전달이 되면, 서버는 해당 이름으로 파일을 찾아서 rb(바이트모드)로 열고, 읽은 후 http 바디에 포함시켜 http 헤더와 함께 클라이언트로 전송해줍니다. 이 때 http 헤더에는 Content-Disposition: attachment를 포함시켜, 클라이언트에서 파일이 다운로드 될 수 있도록 합니다.

-storage 페이지의 삭제 기능

해당 기능은, 서버에 저장된 파일을 클라이언트의 요청에 의해 삭제하는 기능입니다. 클라이언트에서 삭제할 파일을 메시지를 통해 전달받고, os 모듈을 통해 서버에서 삭제합니다. 그 후 새로 갱신된 파일 목록을 가진 페이지를 클라이언트에 전송해줍니다.

-쿠키 구현

http 헤더에는 쿠키도 포함시킬 수 있습니다. 로그인을 한다면 새로운 쿠키를 http 헤더에 포함시켜 보내줍니다. (Set-cookie 이용) 또한, 쿠키를 주는 시간을 datetime 모듈을 이용하여 저장합니다. 브라우저가 4개가 가능하기 때문에 유저도 많을 수 있어서 dictionary를 활용하여 유저별로 쿠키 등록 시간을 저장하였습니다. 또한 datetime을 통해 현재 시간과 쿠키 등록 시간의 차를 구하여 쿠키 페이지에 표시하였으며, 쿠키가 없다면 쿠키 사이트든 storage 사이트든 모두 이용 불가능하게 403 error 처리를 하였습니다. 또한 쿠키가 만료된다면 파일 업로드/다운로드/삭제 시도 시에도 403 error 처리를 하였습니다.

-유저 별 저장소 및 접근 금지

쿠키가 있을 때만 해당 쿠키에 등록된 유저의 storage 및 쿠키 사이트에 접근이 가능하며, post 방식으로 id/pw를 넘겨주기 때문에 다른 유저의 저장소로 url을 통해 접근할 수도 없습니다.

-persistent http

http 1.1 이상에서는 메시지를 클라이언트로 줄 때 content-length를 헤더부분에 포함하여 전송하면 연결이 끊기지 않고 persistent를 유지합니다. 따라서 헤더부분에 content-length를 항상 포함하여 주었으며, 실제 소켓이 close하지 않고 계속하여 사용되는 것을 확인하였습니다.

-multithreading

while문을 통해 service 메소드(소켓으로 http 통신하는 부분)를 멀티쓰레딩으로 실행하도록 구현하였습니다. 다중 유저의 이용이 가능합니다.

3. Implementation

- Describe which functionalities were successfully implemented (with evidence / screenshots).

◆ Scenario #1. Displaying list of files, Upload/Download/Delete files

■ View file list (10 pts)

jaehun's Storage Server

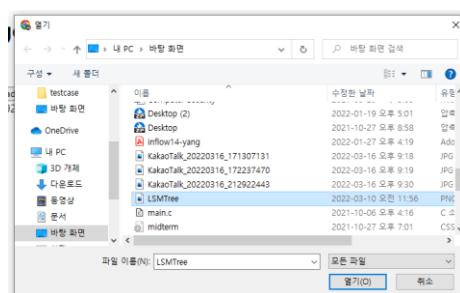


```
else:
    #파일이 들어왔거나,
    f = open('./storage.html', 'rt', encoding='utf-8')
    createFolder(user_id)
    file_list = os.listdir(user_id)
    file_list.reverse()
```

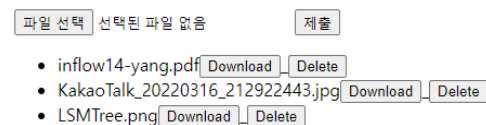
```
file_list_idx = outputdata.find('</ul>')
for i in file_list:
    outputdata = outputdata[:file_list_idx] + '<li>' + i + '<a href="/>
```

로그인 시 모습. 파일 리스트가 보입니다.

■ Upload file (10 pts)



jaehun's Storage Server



파일 업로드 모습. 파일 선택 후 제출 시 파일 업로드가 됩니다.

```

while True:
    if message_org[len(message_org)-4:len(message_org)] != b'--\r\n': #한번에 다들어온게 아닌 케이스에서는, 남은 데이터를 받기.
        message = websocket.recv(65535)
        message_no_header = message_no_header + message #combine all messages
    if i==1:
        i+=1
        if message.find(b'filename') != -1:
            filename_idx_start = message.find(b'filename=') + 10
            filename_idx_end = message.find(b'\r\nContent-Type')
            fn_from_client = message[filename_idx_start:filename_idx_end].decode() #filename 추출
            write_file = open(os.getcwd()+"/"+user_id+"/"+fn_from_client, "wb")

        if i==2:
            webkit_idx_end = message_no_header.find(b'\r\n\r\n') + 4
            webkit_msg = message_no_header[webkit_idx_end:] #find webkit msg
            webkit_submsg = webkit_msg[0:webkit_msg.find(b'\r\nContent:')]
            i+=1

        if message[len(message)-4:len(message)] == b'--\r\n':
            break

#remove webkit messages (3 cases)
message_no_header = message_no_header[:webkit_idx_end:]

webkit_endmsg_idx = message_no_header.rfind(b'\r\n-----WebKit')
message_no_header = message_no_header[:webkit_endmsg_idx]
webkit_endmsg_idx = message_no_header.rfind(b'\r\n-----WebKit')
message_no_header = message_no_header[:webkit_endmsg_idx]
while message_no_header.find(webkit_submsg) != -1:
    message_no_header = message_no_header[:message_no_header.find(webkit_submsg)] + message_no_header[message_no_header.find(webkit_submsg)+len(webkit_submsg):]
write_file.write(message_no_header)

```

파일 데이터를 모두 받을 때까지 반복적으로 recv 합니다.

■ Download file (10 pts)

jaehun's Storage Server

파일 선택

선택된 파일 없음

제출

- inflow14-yang.pdf

Download

Delete
- KakaoTalk_20220316_212922443.jpg

Download

Delete
- LSMTree.png

Download

Delete



```

elif filename.decode().find('down') != -1:
    down_file_idx = filename.decode().rfind('/')
    down_file = filename.decode()[down_file_idx+1:]
    #file 전송 여기서 하기.
    sendfile = open(user_id+"/"+down_file,"rb")
    senddata = sendfile.read()
    sendfile.close()
    senddata = b'HTTP/1.1 200 OK\r\nContent-Disposition: attachment\r\nContent-Length: '+bytes(str(len(senddata)),encoding='utf8') + b'\r\n\r\n' + senddata
    websocket.send(senddata)

```

파일 다운로드 버튼 클릭 시 다운로드가 됩니다.

■ Delete file (10 pts)

jaehun's Storage Server

파일 선택

선택된 파일 없음

제출

- inflow14-yang.pdf

Download

Delete
- LSMTree.png

Download

Delete

```
if filename.decode().find('delete') != -1:  
    del_file_idx = filename.decode().rfind('/')  
    del_file = filename.decode()[del_file_idx+1:]  
    (del_file)  
    os.remove(os.getcwd()+"/"+user_id + "/" + del_file)
```

Delete 버튼 클릭 시 파일 삭제가 됩니다. (kakao~.jpg 삭제한 모습)

◆ Scenario #2. Account Management

■ Login Page (5 pts)

Storage Server Login

Enter your ID and Password

User ID :

Password:

```
if (filename.decode() == '/' or filename.decode() == '/index.html'):  
    f = open('./index.html', 'rt', encoding='utf-8')
```

로그인 페이지 모습입니다. index.html을 열어서 클라이언트로 주면 됩니다.

■ Access Control (15 pts)

← → ↻ 127.0.0.1:10090/storage.html

Gmail YouTube https://icampus.skk... Papago

James's Storage Server

파일 선택 선택된 파일 없음 제출

← → ↻ 127.0.0.1:10090/storage.html

Gmail YouTube https://icampus.skk... Papago

403Forbidden

```
<form action="storage.html" method="post">
  User ID : <input type="text" name="id" placeholder="Your ID"><br>
  Password:<input type="password" name="pw" placeholder="Your Password"><br><br>
  <input type="submit" value="Login">
```

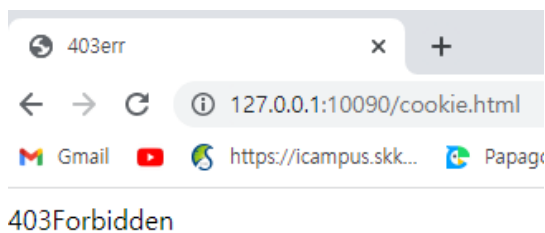
```
else: #쿠키 정보 없을때
    while True:
        if message[len(message)-4:len(message)] == b'--\r\n':
            break
        message = websocket.recv(66535)

    f = open('./403err.html', 'rt', encoding='utf-8')
    outputdata = f.read()
    sendingData = 'HTTP/1.1 200 OK'+'\r\nContent-Length: '+str(len(outputdata))+'\r\n\r\n' + outputdata
    websocket.send(sendingData.encode('utf-8'))
    #websocket.close()
    continue
```

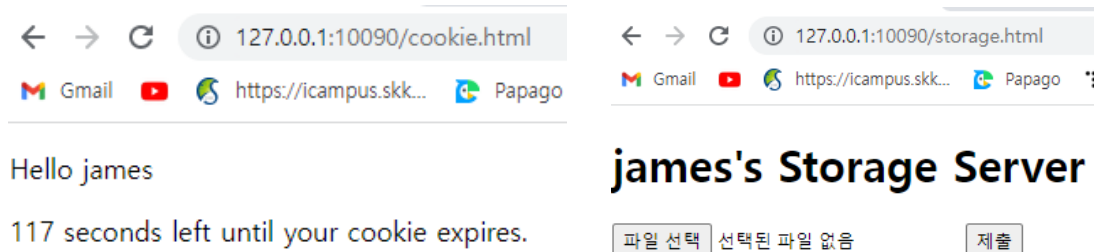
Post 방식으로 id 및 pw를 전달하기 때문에 다른 유저에게 접근이 불가능합니다.

쿠키가 없다면 로그인을 하지 않고 storage로 접근이 불가능합니다.

■ Detailed Cookie Management (10 pts)



쿠키 만료 시 cookie.html 및 storage.html 접근 불가합니다.



```
if (id_idx != -1): #login 단계
    user_id = message.decode()[id_idx+3:pw_idx]
    cookie_msg = '\r\nSet-Cookie: ckd='+user_id+'; max-age=120;' #set new cookie only when it is login step
    cookie_time[user_id] = datetime.now()
```

```
else: #로그인
    user_idx = message.decode().find('ckd=')
    user_idx2 = message.decode().find('\r\n',user_idx)
    if user_idx != -1: #there is cookie info
        user_id = message.decode()[user_idx+4:user_idx2]
```

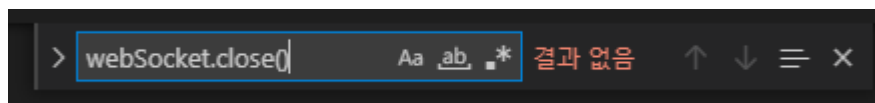

쿠키 남은 시간이 보입니다. 쿠키가 있다면 storage로 바로 접근 가능합니다.

◆ Scenario #3. Persistent HTTP

■ Persistent HTTP (20 pts)

```
sendingData = 'HTTP/1.1 200 OK'+'\r\nContent-Length: '+str(len(outputdata))+'\r\n\r\n' + outputdata
senddata = b'HTTP/1.1 200 OK\r\nContent-Disposition: attachment\r\nContent-Length: '+bytes(str(len(senddata)),encoding='utf8') + b'\r\n\r\n' + senddata
```

persistent http를 위하여 Content-length를 추가한 모습



소켓을 닫지 않습니다. (socket close 없음) 기존 소켓을 계속 씁니다.

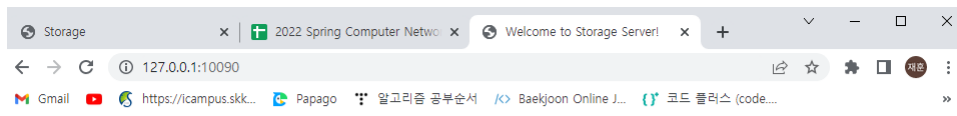
4. Execution Screenshot

- **Important:** Insert a screenshot that explicitly shows your program is working correctly with your execution command. TAs will run your program based on this information.

(Capturing the initialization part is enough, not the entire operation of scenarios)

```
PS C:\Users\jaehun\Downloads\pal_jaehun\pal_jaehun> & 'C:\Python310\python.exe' 'c:\Users\jaehun\.vscode\extensions\ms-python.python-2022.4.0\pythonFiles\lib\python\debugpy\launcher' '3376' '--' 'c:\Users\jaehun\Downloads\pal_jaehun\pal_jaehun\2017314626.py'
before accept
```

Vscode에서 파이썬 (2017314626.py) 실행



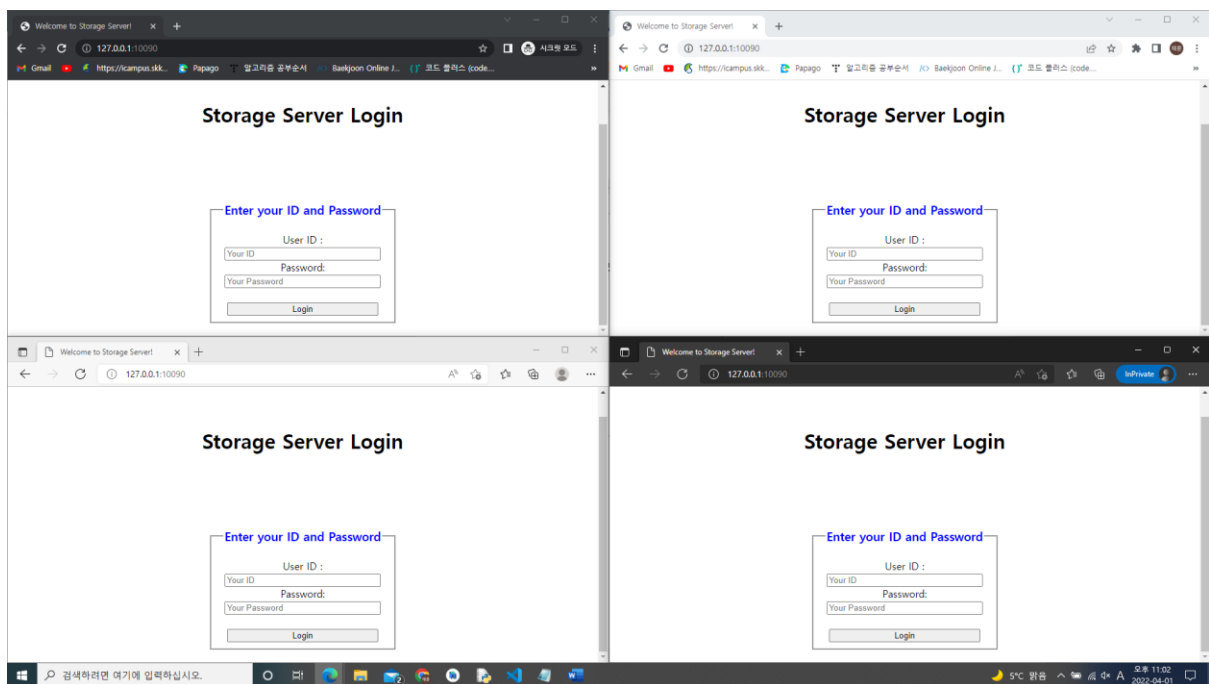
Storage Server Login

Enter your ID and Password

User ID :

Password:

크롬(클라이언트)에서 접속



브라우저 4개 (엣지, 크롬 및 2개의 시크릿 모드)에서 접속이 되는 모습