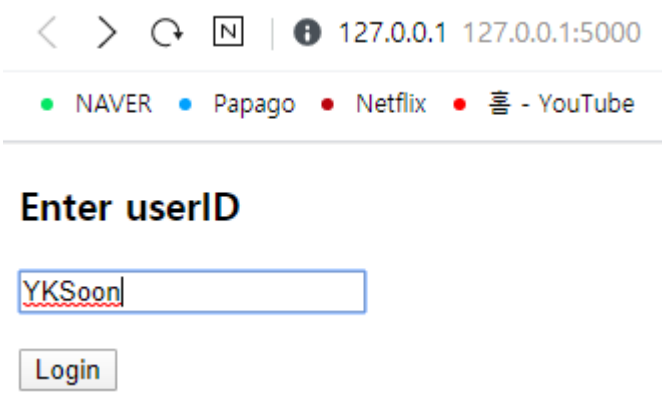
쿠키와 세션 다루기

쿠키(Cookie)

쿠키는 클라이언트PC에 텍스트 파일 형태로 임시저장 되는것이고 세션은 웹 서버쪽에 임시 저장 되는 것이다.
보통 세션과 더불어
자동 로그인, 팝업창에서 “오늘은 이 창을 더이상 보지 않기” 등의 기능을 클라이언트에 저장해놓기 위해 사용된다.

```
routes.py

from flask import Flask, render_template
app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')


if __name__ == '__main__':
    app.run(debug=True)
```

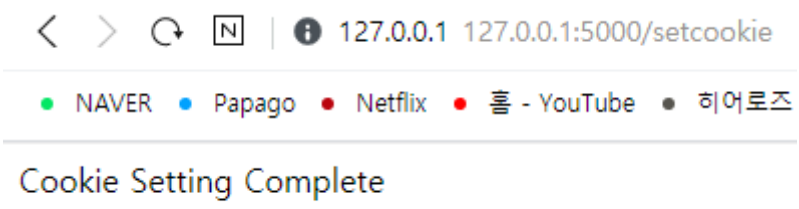
```
templates\index.html

<!DOCTYPE html>
<html>
<head>

</head>
<body>
    <form action="/setcookie" method="POST">
        <p><h3>Enter userID</h3></p>
        <p><input type="text" name="nm" /></p>
        <p><input type="submit" value="Login" /></p>
    </form>
</body>
</html>
```



> 하지만 setcookie페이지를 구현하지 않아 에러가 발생한다.

코드를 아래와 같이 수정한다.

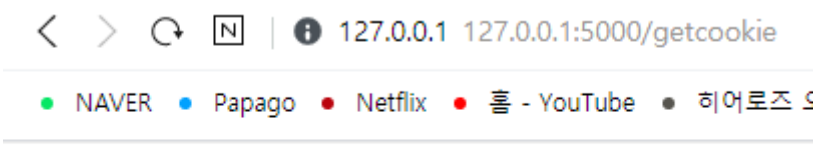
```
routes.py

from flask import Flask, render_template, request, make_response
app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/setcookie', methods = ['POST', 'GET'])
def setcookie():
    if request.method == 'POST':
        user = request.form['nm']

    resp = make_response("Cookie Setting Complete")
    resp.set_cookie('userID', user)

    return resp

if __name__ == '__main__':
    app.run(debug=True)
```



> 이제 오류 없이 잘 넘어간다.

아래는 쿠키를 꺼내는 방법이다.

```
routes.py

from flask import Flask, render_template, request, make_response
app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/setcookie', methods = ['POST', 'GET'])
def setcookie():
    if request.method == 'POST':
        user = request.form['nm']

    resp = make_response("Cookie Setting Complete")
    resp.set_cookie('userID', user)  # 쿠키ID, 쿠키값

    return resp

@app.route('/getcookie')
def getcookie():
    name = request.cookies.get('userID')
    return '<h1>welcome '+name+'</h1>'

if __name__ == '__main__':
    app.run(debug=True)
```



welcome YKSoon

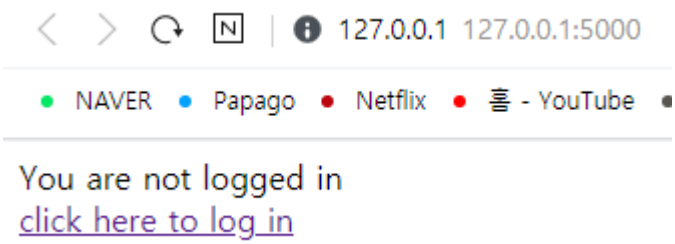
세션(Session)

쿠키와 다르게 세션과 관련된 데이터는 서버에 저장된다.
서버에서 관리할 수 있다는 점에서 안전성이 좋아서 보통 로그인 관련으로 사용되고 있다.

플라스크에서 세션은 딕셔너리의 형태로 저장되며
키를 통해 해당 값을 불러올 수 있다.

세션을 사용하기 위해서는 해당 값을 암호화 하기 위한 key값을 코드에서 지정해주어야 한다.

```
routes.py

from flask import Flask, render_template, request, make_response
from flask import session, redirect, url_for

app = Flask(__name__)
app.secret_key = 'any random string'

@app.route('/')
def index():
    if 'username' in session:
        username = session['username']
        return 'Logged in as' + username + '<br>' + \
            "<b><a href= '/logout'>click here to log out</a></b>"

    return "You are not logged in <br><a href = '/login'></b>" + \
        "click here to log in</b></a>"

@app.route('/setcookie', methods = ['POST', 'GET'])
def setcookie():
    if request.method == 'POST':
        user = request.form['nm']

    resp = make_response("Cookie Setting Complete")
    resp.set_cookie('userID', user)  # 쿠키ID, 쿠키값

    return resp

@app.route('/getcookie')
def getcookie():
    name = request.cookies.get('userID')
    return '<h1>welcome '+name+'</h1>'

if __name__ == '__main__':
    app.run(debug=True)
```

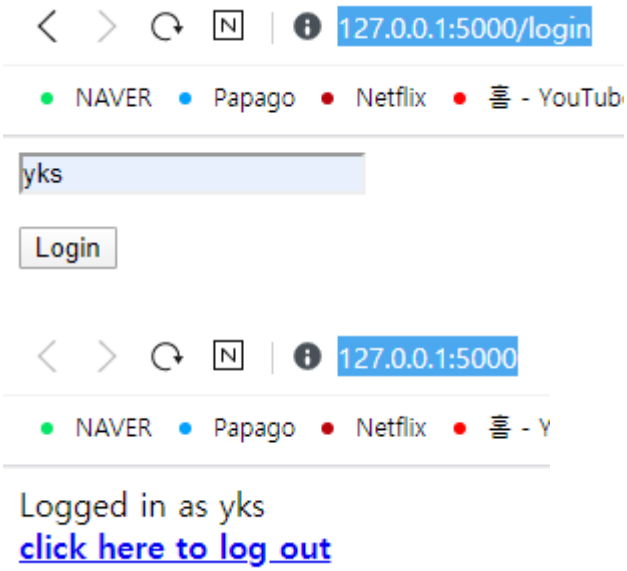


맨처음 접속했을때는 GET 메소드로 요청이 오게 되므로,
로그인을 하기 위한 폼을 전송한다.

폼을 통해 POST 요청이 오면 username 이라는 세션을 생성하여
입력받은 폼의 데이터를 세션에 저장한 후, 맨 처음 페이지로 리다이렉션한다.

routes.py

```
from flask import Flask, render_template, request, make_response
from flask import session, redirect, url_for

app = Flask(__name__)
app.secret_key = 'any random string'

@app.route('/')
def index():
    if 'username' in session:
        username = session['username']
        return 'Logged in as ' + username + '<br>' + \
            "<b><a href= '/logout'>click here to log out</a></b>"

    return "You are not logged in <br><a href = '/login'></b>" + \
        "click here to log in</b></a>"

@app.route('/setcookie', methods = ['POST', 'GET'])
def setcookie():
    if request.method == 'POST':
        user = request.form['nm']

    resp = make_response("Cookie Setting Complete")
    resp.set_cookie('userID', user)  # 쿠키ID, 쿠키값

    return resp

@app.route('/getcookie')
def getcookie():
    name = request.cookies.get('userID')
    return '<h1>welcome '+name+'</h1>'

@app.route('/login', methods = ['GET', 'POST'])
def login():
    if request.method == 'POST':
        session['username'] = request.form['username']
        return redirect(url_for('index'))
    return '''
        <form action = "" method = "post">
            <p><input type = text name = username /></p>
            <p><input type = submit value = Login /></p>
        </form>
'''

if __name__ == '__main__':
    app.run(debug=True)
```



로그아웃의 경우 해당 세션 정보를 제거하는 것으로 연결을 끊는 것이 가능하다. 세션의 제거는 pop 메소드를 사용해서 아래처럼 짜주면 된다.

```
routes.py

@app.route('/logout')
def logout():
    session.pop('username', None)
    return redirect(url_for('index'))
```

그러나 서버 측의 세션 정보는 사라지지 않기 때문에 이미 사라져버린 세션을 계속 잡고 있게 된다. 플라스크에서도 이런 경우를 대비하여 각 세션의 유효기간이 정해져 있으며 아무 설정도 하지 않을 경우, 31일로 설정되어 있다.

만약 직접 세션의 유효기간을 직접 설정하고 싶다면, 아래와 같은 함수를 추가해주면 된다.

```
routes.py

from datetime import timedelta
from flask import session, app

@app.before_request
def make_session_permanent():
    session.permanent = True
    app.permanent_session_lifetime = timedelta(minutes=5)
```