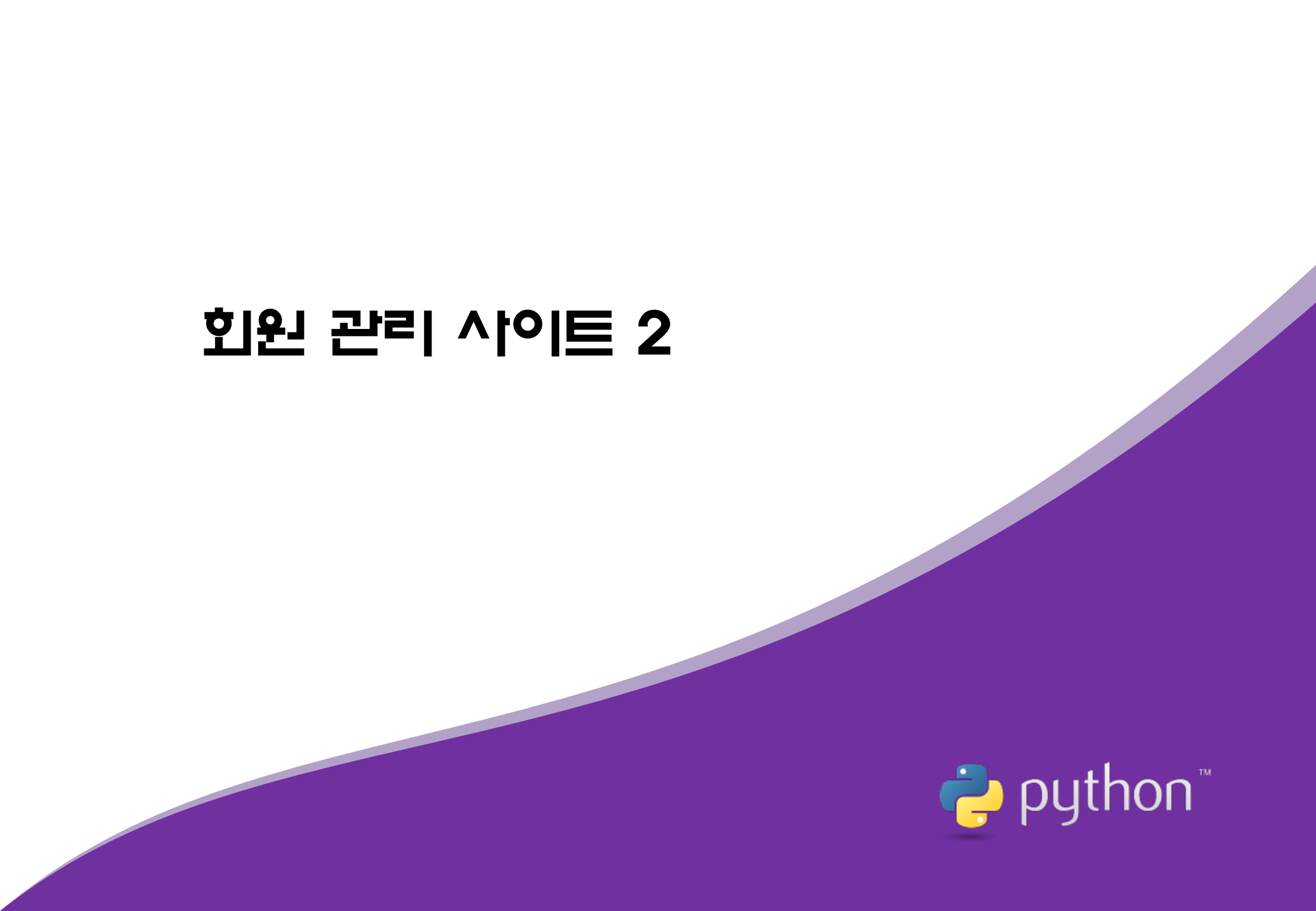
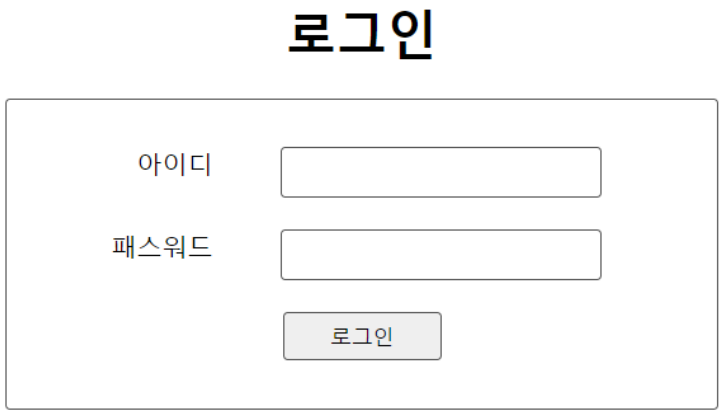
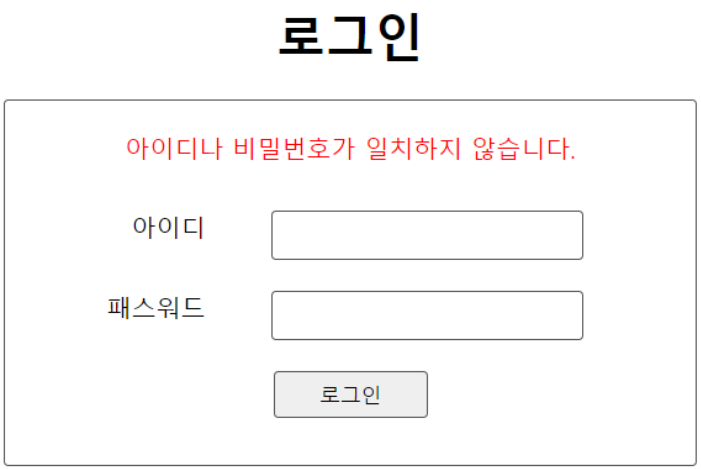
회원 관리 사이트 2



log in 페이지

- login 페이지





log in 페이지

- **login.html**

```
<h1>로그인</h1>
<form action="{{ url_for('login') }}" method="post" class="login">
    <fieldset>
        {% if error  %}
        <p>{{error}}</p>
        {% endif %}
        <ul>
            <li>
                <label for="mid">아이디</label>
                <input type="text" id="mid" name="mid">
            </li>
            <li>
                <label for="passwd">패스워드</label>
                <input type="password" id="passwd" name="passwd">
            </li>
            <li>
                <input type="submit" value="로그인">
            </li>
        </ul>
    </fieldset>
```

log in 페이지

- **url 경로 설정 – /login/ 및 login() 함수 정의**

```python
@app.route('/login/', methods = ['GET', 'POST'])
def login():
    if request.method == "POST":
        id = request.form['mid']     # 자료 수집
        pwd = request.form['passwd']

        conn = getconn()    # db 연결
        cur = conn.cursor()
        sql = "SELECT * FROM member WHERE mid = '%s' AND passwd = '%s' " % (id, pwd)
        cur.execute(sql)
        rs = cur.fetchone()
        conn.close()
        if rs:
            session['userID'] = rs[0]  # 세션 발급
            session['userName'] = rs[2]
            return redirect(url_for('index'))
        else:
            error = "아이디나 비밀번호가 일치하지 않습니다."
            return render_template('login.html', error=error)
    else:
        return render_template('login.html')
```

log in 페이지

- **비밀키(암호키) 없으면 로그인 시 에러 발생!!**

```
RuntimeError

RuntimeError: The session is unavailable because no secret key was set.  Set the secret_key
to something unique and secret.
```

```
app = Flask(__name__)

app.secret_key = "abcdef" #비밀키 발행
```

log in 페이지

◆ style.css

```
/* login 스타일 */
section .login{width: 450px; margin: 0 auto;}
section .login fieldset{border:1px solid #555; border-radius: 3px}
section .login p{padding-top: 20px; color: #f00;}
section .login ul{list-style: none; padding: 10px;}
section .login ul li{margin: 20px; }
section .login ul li label{width: 100px; float: left; text-align: right;}
section .login ul li input{width: 200px; height: 30px;
  border:1px solid #555; border-radius: 3px}
section .login ul li input[type='submit']{width: 100px; height: 30px}

/* member_edit 스타일 */
section .tbl_edit{width: 450px; }
section .tbl_edit input{width: 200px; height: 30px;}
section .tbl_edit input[type='submit']{width: 100px; height:30px;}
section .tbl_edit input[type='reset']{width: 100px; height:30px;}
```

logout 페이지

- logout 페이지

```python
@app.route('/logout/')
def logout():
    #session.pop("userID")  #세션 삭제
    session.clear()        # 모든 세션 삭제
    return redirect(url_for('index'))
```

로그인후

Home (10001님)로그아웃 회원가입 회원목록

안녕하세요~ 회원 커뮤니티 사이트입니다.

메뉴바 변경

- **로그인 전과 후 변경**

Home 로그인 회원가입 게시판

안녕하세요~ 회원 커뮤니티 사이트입니다.

Home (김하나님)로그아웃 회원목록 게시판

회원 목록

번호	아이디	패스워드	이름	나이	가입일
1	sky12	m123456@	강하늘	24	2021-12-12 18:08:37
2	today	m123456@	김하나	21	2021-12-12 17:41:44

메뉴 변경

◆ navbar.html 변경

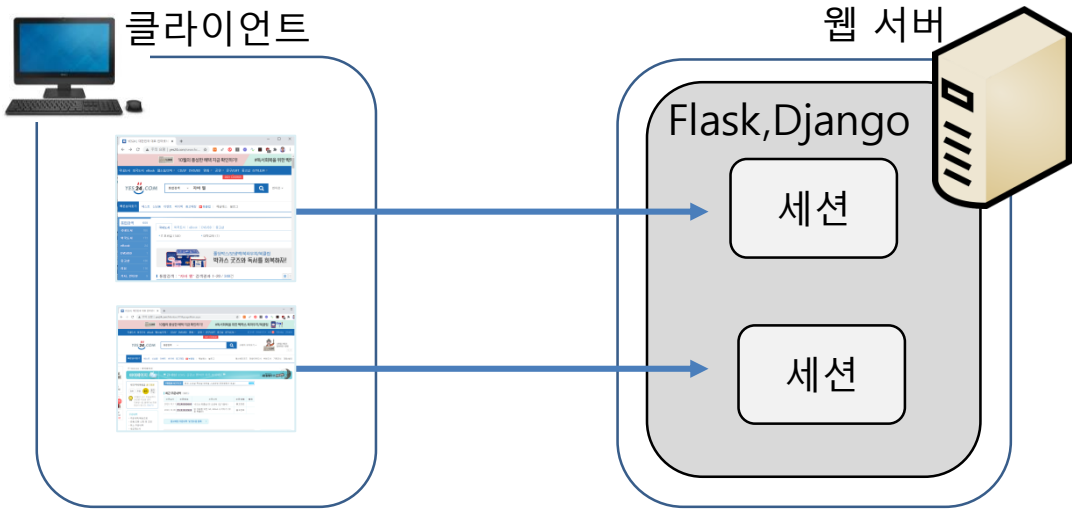
```
<header>
    <nav>
        <ul>
        {% if session %}
            <li><a href="{{ url_for('index') }}">Home</a></li>
            <li><a href="{{ url_for('logout') }}">
                ({{ session['userName'] }}님)로그아웃</a></li>
            <li><a href="{{ url_for('memberlist') }}">회원목록</a></li>
            <li><a href="{{ url_for('boardlist') }}">게시판</a></li>
        {% else %}
            <li><a href="{{ url_for('index') }}">Home</a></li>
            <li><a href="{{ url_for('login') }}">로그인</a></li>
            <li><a href="{{ url_for('register') }}">회원가입</a></li>
            <li><a href="{{ url_for('boardlist') }}">게시판</a></li>
        {% endif %}
        </ul>
    </nav>
</header>
```

세션(session)

세션(session)

- 클라이언트와 웹 서버 간의 상태를 지속적으로 유지하는 방법을 말한다.
- 사용자 인증을 통해 특정 페이지를 사용할 수 있도록 권한 상태 유지.

  예) 웹 쇼핑몰 – 장바구니나 주문 처리와 같은 회원 전용 페이지 로그인 후 다른 웹 페이지에 갔다가 돌아와도 로그인 상태 유지됨
- 세션은 오직 웹 서버에 존재하는 객체로 웹 브라우저마다 하나씩 존재하므로 브라우저를 닫기 전까지 웹 페이지를 이동하더라도 사용자 정보가 유지된다.

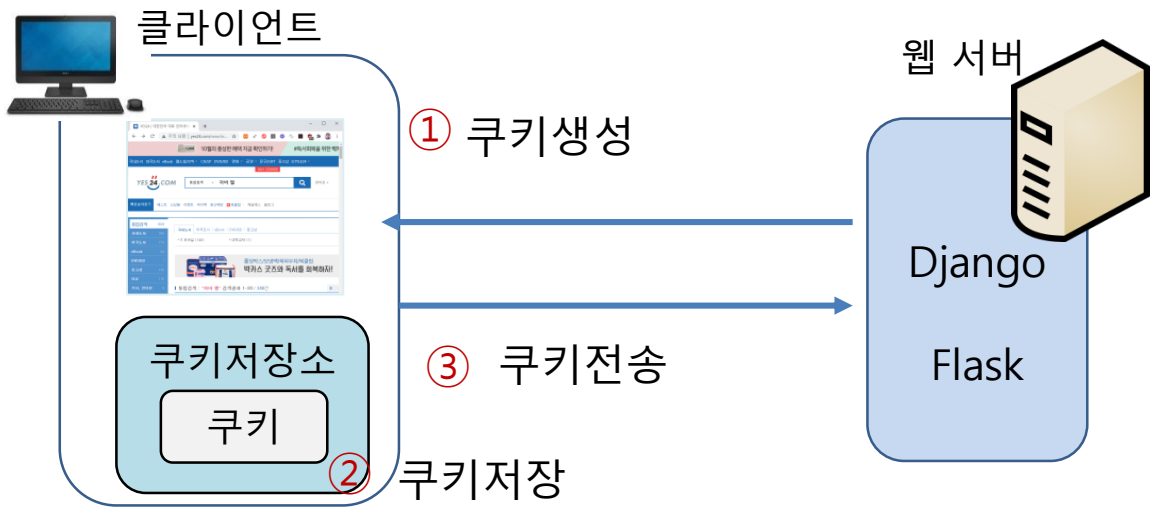


쿠키(cookie)

쿠키(cookie)

- 클라이언트와 웹 서버간의 상태를 지속적으로 유지하는 방법으로 쿠키와 세션이 있다.
- 쿠키는 세션과 달리 상태 정보를 웹 서버가 아닌 클라이언트에 저장한다.

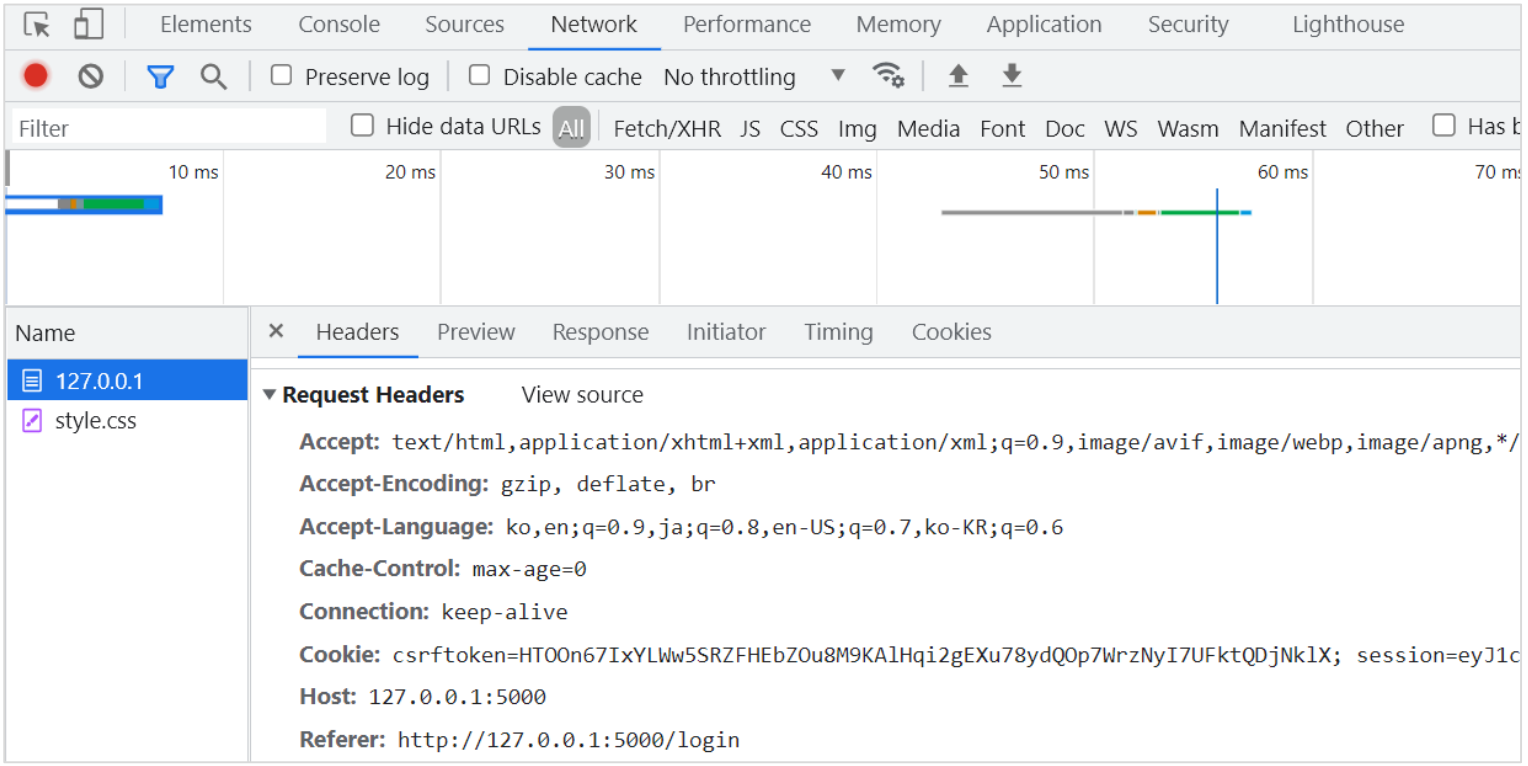
  예) 어떤 웹 사이트를 처음 방문한 사용자가 로그인 인증을 하고 나면 아이디와 비밀번호를 기록한 쿠키가 만들어지고, 그 다음부터 사용자가 그 사이트에 접속하면 별도의 절차를 거치지 않고 쉽게 접속할 수 있다.
- 쿠키는 클라이언트의 일정 폴더에 정보를 저장하므로 웹 서버의 부하를 줄일 수 있으나 개인 정보 기록이 남기 때문에 보안에 문제가 있다.



Session & Cookie

◆ 세션과 쿠키 확인

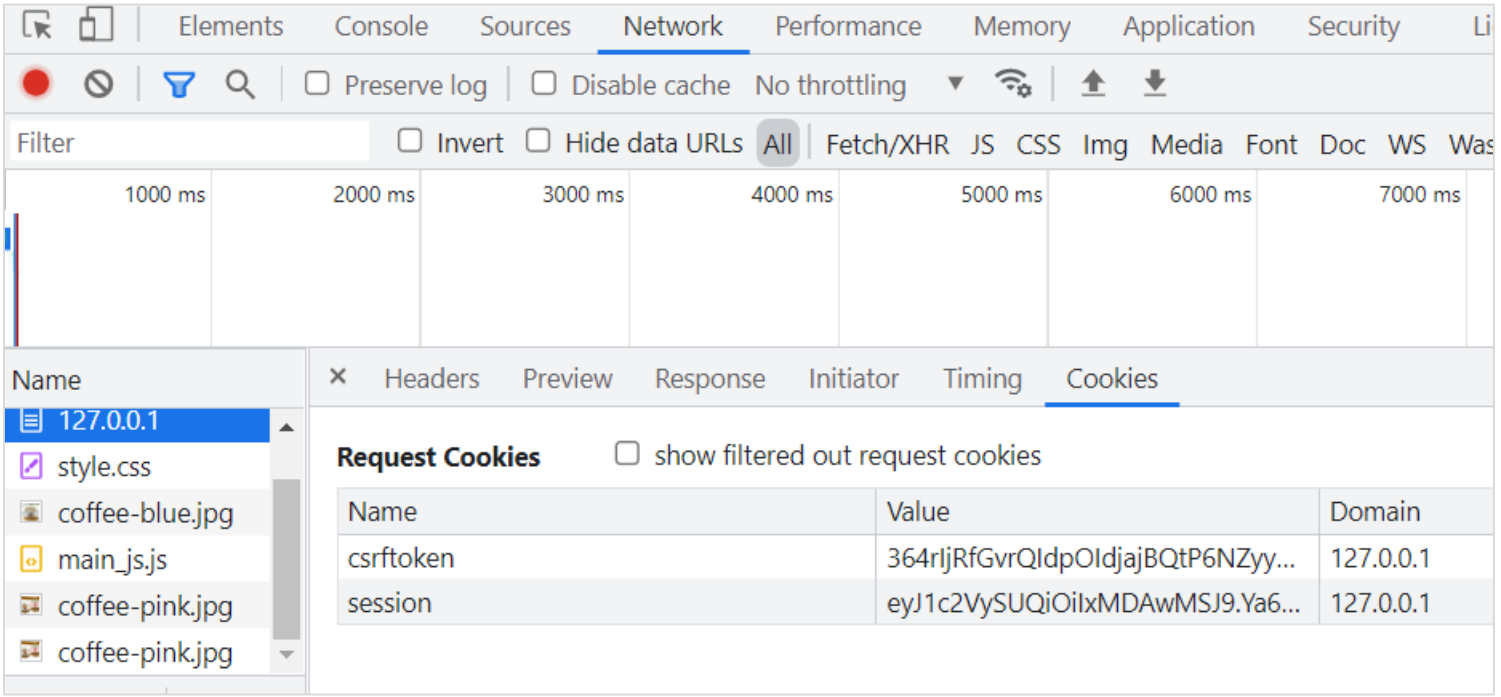
개발자도구 > Network > 127.0.0.1 > Headers



Session & Cookie

◆ 세션과 쿠키 확인

개발자도구 > Network > 127.0.0.1 > Cookies



Session 권한

◆ 수정과 탈퇴 메뉴 보이기

Session 권한

◆ member_view.html 변경

```
<td colspan="2" style="background-color: #fff">
    <a href="{{ url_for('memberlist') }}" >목록</a>
    {% if session['userID'] == rs[0] %}
    <a href="{{ url_for('member_edit', id=rs[0]) }}" >수정</a>
    <a onclick="return confirm('정말로 삭제하시겠습니까?')"
       href="{{ url_for('member_del', id=rs[0]) }}">탈퇴</a>
    {% endif %}
</td>
```

회원 탈퇴

◆ 회원 탈퇴 – confirm 창

회원 탈퇴

◆ url 경로 설정 – /member_del/<string:id>/ 및 member_del()

```
@app.route('/member_del/<string:id>/')
def member_del(id):
    conn = getconn()
    cur = conn.cursor()
    sql = "DELETE FROM member WHERE mid = '%s' " % (id)
    cur.execute(sql)
    conn.commit()
    conn.close()
    return redirect(url_for('memberlist'))
```

회원 정보 수정

◆ 회원 정보 수정

회원 정보 수정

- member_edit.html

```
<section>
<h1>회원 정보</h1>
    <form action="{{ url_for('member_edit', id=rs[0]) }}" method="post">
        <table class="tbl_edit">
            <colgroup>
                <col style="background-color: #eee">
                <col>
            </colgroup>
            {% if rs %}
            <tr>
                <td>아이디</td>
                <td><input type="text" name="mid" value="{{ rs[0] }}"></td>
            </tr>
            <tr>
                <td>비밀번호</td>
                <td><input type="password" name="passwd" value="{{ rs[1] }}"></td>
            </tr>
            <tr>
                <td>이름</td>
                <td><input type="text" name="name" value="{{ rs[2] }}"></td>
            </tr>
```

회원 정보 수정

- member_edit.html

```
            <tr>
                <td>나이</td>
                <td><input type="text" name="age" value="{{ rs[3] }}"></td>
            </tr>
            <tr>
                <td >가입일</td>
                <td><input type="text" name="regDate" value="{{ rs[4] }}"></td>
            </tr>
            <tr>
                <td colspan="2" style="background-color: #fff">
                    <input type="submit" value="저장">
                    <input type="reset" value="취소">
                </td>
            </tr>
            {% endif %}
        </table>
    </form>
</section>
```

회원 정보 수정

◆ url 경로 설정 – /member_edit/<string:id>/ 및 member_edit()

```python
@app.route('/member_edit/<string:id>/', methods = ['GET', 'POST'])
def member_edit(id):
    if request.method == "POST":
        # 데이터 수집
        id = request.form['mid']
        pwd = request.form['passwd']
        name = request.form['name']
        age = request.form['age']

        conn = getconn()  # db 연결
        cur = conn.cursor()
        sql = "UPDATE member SET passwd = '%s', name = '%s', age = '%s' " \
              "WHERE mid = '%s' " % (pwd, name, age, id)
        cur.execute(sql)
        conn.commit()
        conn.close()
        return redirect(url_for('member_view', id=id))
```

회원 정보 수정

◆ url 경로 설정 – /member_edit/<string:id>/ 및 member_edit()

```
else:
    conn = getconn()
    cur = conn.cursor()
    sql = "SELECT * FROM member WHERE mid = '%s' " % (id)
    cur.execute(sql)
    rs = cur.fetchone()
    conn.close()
    return render_template('member_edit.html', rs=rs)
```

회원 가입시 가입일 자동 생성

- 가입일 입력 상자 삭제 및 자동 로그인

회원 가입시 가입일 자동 생성

- **register() 함수 – 가입일 자동 생성**

```
@app.route('/register/', methods = ['GET', 'POST'])
def register():
    if request.method == "POST":
        # 데이터 수집
        id = request.form['mid']
        pwd = request.form['passwd']
        name = request.form['name']
        age = request.form['age']
        # date = request.form['regDate']

        conn = getconn()  #db 연결
        cur = conn.cursor()
        sql = "INSERT INTO member (mid, passwd, name, age) VALUES ('%s', '%s', '%s', '%s')" \
              % (id, pwd, name, age)
        cur.execute(sql)
        conn.commit()
```

회원 가입시 자동 로그인하기

- **register() 함수 수정 – 가입 후 자동 로그인 및 세션 발급**

```
    # 가입후 자동 로그인 및 세션 발급
    sql = "SELECT * FROM member WHERE mid = '%s' AND passwd = '%s' " % (id, pwd)
    cur.execute(sql)
    rs = cur.fetchone()
    conn.close()
    if rs:
        session['userID'] = rs[0]    # 세션 발급 - mid
        session['userName'] = rs[2]  # 세션 발급 - name
        return redirect(url_for('memberlist'))
else:
    return render_template('register.html')
```

회원 가입시 유효성 검사

- **회원가입 – 유효성 검사**

회원 가입시 유효성 검사

- register.html 변경

```
    <script src="{{ url_for('static', filename='js/checkMember.js') }}"></script>
</head>
<body>
  <div id="container">
      {% include 'navbar.html' %}
      <section>
      <h1>회원 가입</h1>
        <form action="{{ url_for('register') }}" method="post" class="join" name="regForm">
            <fieldset>
```

```
      <li>
          <input type="button" value="가입" onclick="checkMember()">
          <input type="reset" value="취소">
      </li>
  </ul>
```

회원 가입시 유효성 검사

- **checkmember.js**

```
function checkMember(){
    var form = document.regForm;  //폼 이름
    var id = form.mid.value;      //입력된 mid의 값
    var pwd1 = form.passwd.value;
    var pwd2 = form.passwd_confirm.value;
    var name = form.name.value

    //비밀번호 정규 표현식 설정
    var pwd_pat1 = /[0-9A-Za-z]/   //영문자, 숫자
    var pwd_pat2 = /[!@#$%^&*]/    //특수문자

    if(id.length != 5){
        alert("5자리까지 가능합니다.");
        form.mid.select();
        return false;
    }
```

회원 가입시 유효성 검사

```
else if(!pwd_pat1.test(pwd1) || !pwd_pat2.test(pwd1) || pwd1.length !=8 ){
    alert("비밀번호는 영문자, 숫자, 특수문자 포함 8자까지 가능합니다.");
    form.passwd.select();  //선택 영역 지정
    return false;
}
else if(pwd1 != pwd2){
    alert("비밀번호를 동일하게 입력해 주세요.");
    form.passwd_confirm.select();
    return false;
}
else if(name == ""){
    alert("이름은 필수 입력사항입니다.");
    form.name.focus();  //커서 이동
    return false;
}
else{
    form.submit(); //폼 전송
}
```