DATABASE APPLICTION 만들기

2017320114 최재원

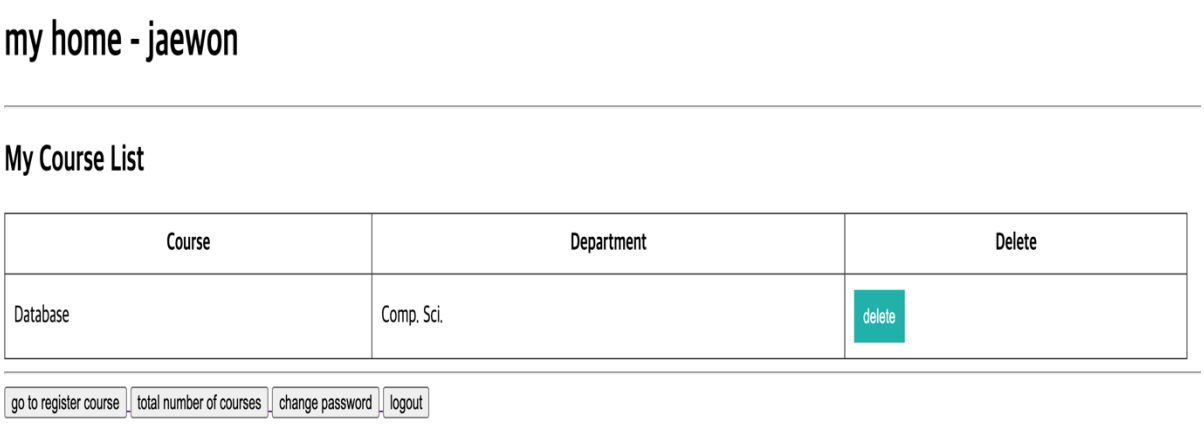
Database Application에 대한 간단한 설명

이 Application은 학생들을 위한 대학교 수강신청 시스템입니다.

1. 우선 학생은 회원가입(sign up)을 통하여 아이디와 비밀번호를 생성할 수 있습니다.

위의 사진처럼 회원가입이 가능하며, 회원가입을 하고 나서는 Login버튼을 통하여 간단히 사이트에 접속할 수 있습니다.

2. 아이디(id)와 비밀번호(password)를 입력 후 사이트에 들어가면 아래 사진과 같이 My Home이 보여지게 됩니다. My home에서는 현재 내가 수강 신청한 과목의 목록과 언제든지 삭제(delete)할 수 있는 삭제 칸이 보여지게 됩니다.



☑ **Go to register course**

Course Register Table of "jaewon"

Course	Department	Professor	College	register
Computer Network	Comp. Sci.	Kim	Informatics	register
Operating System	Comp. Sci.	Yoo	Informatics	register
Logic Design	Comp. Sci.	Baek	Informatics	register
Engineering Circuits	Comp. Sci.	Lee	Informatics	register
Deep Learning	Comp. Sci.	Baek	Informatics	register
Calculus	Math	Choi	Science	register
AI	Comp. Sci.	Kim	Informatics	register

이 버튼을 누르게 되면 수강신청 과목 목록 및 신청 페이지로 들어가게 됩니다. 수강신청 과목은 **과목명(course), 소속 학과(department), 교수님(professor), 그리고 해당 단과대학(College)** 이 정보로 보여지고 있으며 과목의 수강신청을 의미하는 register버튼을 누르게 되면 과목이 내 목록에 담긴 채로 my home으로 돌아가게 됩니다. 아래 사진에서는 Operating System을 신청해본 후의 결과입니다.

my home - jaewon

My Course List

Course	Department	Delete
Database	Comp. Sci.	delete
Operating System	Comp. Sci.	delete

go to register course | total number of courses | change password | logout

또한, 수강 신청을 완료한 과목은 나(jaewon)의 수강신청 강의 목록에서는 제거가 됩니다.

Course Register Table of "jaewon"

Course	Department	Professor	College	register
Computer Network	Comp. Sci.	Kim	Informatics	register
Logic Design	Comp. Sci.	Baek	Informatics	register
Engineering Circuits	Comp. Sci.	Lee	Informatics	register
Deep Learning	Comp. Sci.	Baek	Informatics	register
Calculus	Math	Choi	Science	register
AI	Comp. Sci.	Kim	Informatics	register

만약 신청을 한 과목들 중에서 삭제할 과목이 있으면 delete 버튼을 누르면 삭제가 됩니다.

☑ **Total number of courses**

- 수강 과목을 많이 신청한 학생들을 위해 현재 수강 신청한 과목의 개수를 보여주는 기능입니다. 버튼을 누르게 되면 아래 사진과 같이 나의 과목의 수가 보여집니다.

My total number courses

2

☑ **Change Password**

- 회원(학생)의 비밀번호 변경을 위한 버튼입니다. 버튼을 누르게 되면

사진과 같이 id는 현재 접속되어 있는 id로 고정되어서 보여지고 새로 바꾸려고 하는 비밀번호를 입력하고 change 버튼을 누르면 비밀번호가 update 된 채로 my home 페이지로 돌아가게 됩니다.

your id

jaewon

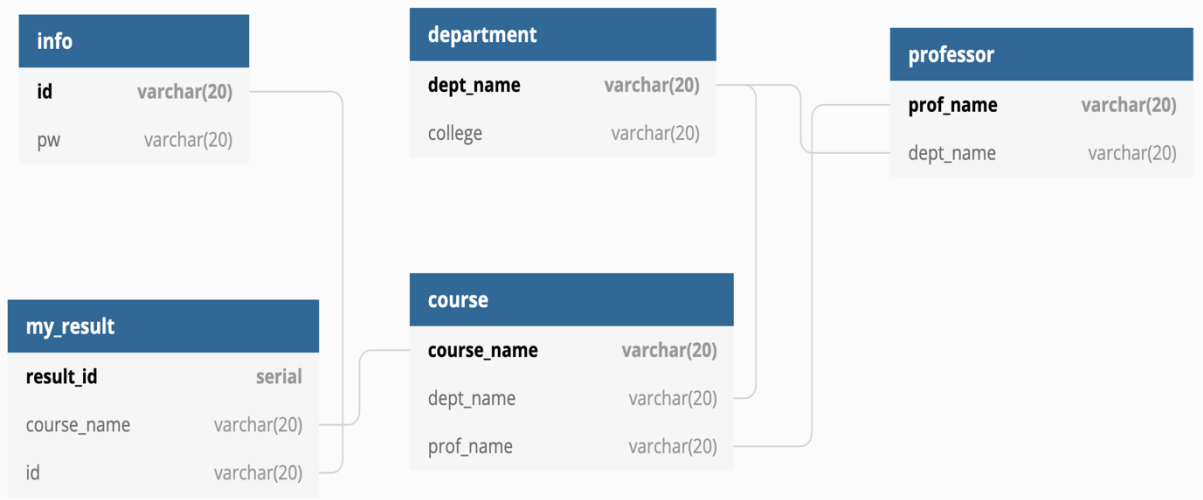
your new password

change

☑ **Logout**

- 회원 로그아웃을 위한 버튼입니다. 버튼을 누를 시 로그인 페이지로 넘어가게 됩니다.

Database Application에 사용한 Schema Diagram



구현한 함수들에 대한 설명

1. CREATE DATABASE **dbproject**를 통해 이번 프로젝트를 진행할 데이터베이스를 생성한 후 **\c dbproject** 그리고 **\i [filepath] /setting.sql**로 데이터베이스에 기본 세팅을 해줍니다. 파일명은 **setting.sql**입니다.

```
dbproject=# \i /Users/choijaewon/pythonProject/setting.sql;
DROP TABLE
DROP TABLE
DROP TABLE
DROP TABLE
DROP TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
INSERT 0 1
INSERT 0 1
```

사진과 같이 실행되는 것을 볼 수 있습니다. 사진은 윗 부분만 캡쳐를 진행했습니다.

2. 로그인 및 회원가입 함수 || SELECT, INSERT

```python
@app.route('/register', methods = ['POST'])
def register():
    id = request.form["id"]
    password = request.form["password"]
    button = request.form["send"]

    if button == "login":
        cur.execute("select * from info where id='{}' and pw = '{}';".format(id, password))
        result = cur.fetchall()
        if len(result) == 0:
            return "login error"
        else:
            return redirect(url_for("main_home", id=id))

    elif button == "sign up":
        cur.execute("insert into info values('{}','{}');".format(id, password))
        connect.commit()
        return f"<script>alert('Sign up success!');history.back();</script>)"

    return id + password + button
```

main.html에서 form = 'POST'를 통해 보내주는 값들을 위 함수에서 처리를 해줍니다. 만약 버튼이 login이면 로그인을 진행하고, sign up일 경우 아이디와 비밀번호를 입력을 하면 sign up success라는 문구를 띄워주고 다시 로그인 화면으로 돌아와 로그인을 할 수 있게 해줍니다. 또한 비밀번호 입력 오류도 해결해 주었습니다.

3. Main Home, 나의 수강신청 목록 함수 || SELECT, NATURAL JOIN

```python
@app.route("/main_home")
def main_home():
    # get my_result course
    id = request.args.get("id")
    cur.execute("select distinct course_name, dept_name from my_result natural join course where id = '{}';".format(id))
    my_results = cur.fetchall()
    return render_template("home.html", id=id, my_results=my_results)
```

로그인/회원가입 함수에서 로그인을 누르면 들어가지는 함수입니다. 로그인 된 id parameter를 form = "GET"을 통해서 받습니다. 이 함수를 통해 현재 나의(전달받은 id 값) 수강 과목의 과목명과 과목의 해당 학과명을 My_result 와 course 테이블의 natural join을 이용해서 표현해 보았습니다. Home.html을 통해서 My home에 있는 여러가지 버튼들을 구현해 내었습니다.

```html
<a href="/course?id={{id}}">
    <button>go to register course</button>
</a>
<a href="/total?id={{id}}">
    <button>total number of courses</button>
</a>
<a href="/pwchange?id={{id}}">
    <button>change password</button>
</a>
<a href="/">
    <button>logout</button>
```

Logout 버튼을 누르면 로그인 화면으로 연결되는 것도 볼 수 있습니다.

4. 수강신청 가능한 과목 목록 및 정보 제공 함수 || SELECT, NESTED SUBQUERY, CARTESIAN PRODUCT

```
@app.route("/course", methods = ['GET'])
def course():
    id = request.args.get("id")
    button = request.args.get("send")
    cur.execute(f"select course_name, department.dept_name, professor.prof_name, college from course, department, professor where course.dept_name = depa
    courses = cur.fetchall()

    return render_template("course.html", id=id, courses=courses)
```

Home.html에서 go to register course 버튼의 연결된 함수로 "GET"을 통해 id를 받습니다. 이후 SQL문을 실행해 과목명(Course), 학과(Department), 교수님(Professor), 단과대학(College) 등의 정보를 course.html을 열어서 제공을 해줍니다. SQL문은 각각의 정보들을 위해 course, department, 그리고 professor 테이블의 cartesian product를 실행했으며, nested subquery를 이용해 이미 신청을 한 과목은 목록에서 제거될 수 있도록 했습니다.

SQL문이 사진에서 잘린 관계로 직접 적겠습니다.

select course_name, department.dept_name, professor.prof_name, college from course, department, professor where course.dept_name = department.dept_name and professor.prof_name = course.prof_name and course_name not in (select course_name from my_result where id ='{id}'';

5. 나의 신청 목록에 과목 추가/ 나의 신청 목록에서 과목 삭제 || INSERT, DELETE

```
@app.route("/course_register/<id>/<course_name>", methods = ['GET'])
def course_register(id, course_name):
    cur.execute("insert into my_result values(default, '{}', '{}');".format(course_name, id))
    connect.commit()
    return redirect(url_for("main_home", id=id))

@app.route("/course_delete/<id>/<course_name>", methods = ['GET'])
def course_delete(id, course_name):
    cur.execute(f"delete from my_result where id='{id}' and course_name='{course_name}'")
    connect.commit()
    return redirect(url_for("main_home", id=id))
```

```
href="{{ url_for('course_register', id=id, course_name=course[0])}}">register</a></button>
```

```
href="{{ url_for('course_delete', id=id, course_name=my_result[0])}}">delete</a></button>
```

두 함수 모두 나의 수강신청목록(my_result)에서 신청 및 삭제를 해주고 있으며, course.html, home.html에서 확인할 연결되는 것을 확인할 수 있습니다. SQL문은 Insert 와 delete를 사용했습니다.

6. 총 신청한 과목의 개수 제공 함수 || SELECT, AGGREGATE FUNCTION

```
@app.route("/total", methods=['GET'])
def total_course():
    id = request.args.get("id")
    cur.execute("select count(course_name) from my_result where id='{}';".format(id))
    total = cur.fetchall()
    return render_template("total.html", id=id, total=total)
```

Home.html에서 total number of courses 버튼을 누르면 연결되는 함수로 GET method를 통해 로그인된 id를 parameter로 받고 있습니다. SQL문에서는 aggregation function인 count를 사용해서 course_name의 개수를 세어주고 있습니다. 총 개수는 total.html파일을 통해 보여지고 있습니다.

7. 비밀번호 변경 함수 || UPDATE

```
@app.route("/pwchange", methods=['POST','GET'])
def pwchange():
    if request.method == 'GET':
        id = request.args.get("id")
        return render_template("pwchange.html", id=id)
    elif request.method =='POST':
        id = request.form["id"]
        password = request.form["password"]
        cur.execute(f"update info set pw='{password}' where id='{id}';")
        connect.commit()
        return redirect(url_for("main_home", id=id))
```

GET 과 POST method를 둘 다 사용하며 get을 요청하는 경우에는 pwchange.html을 열어주고 post일 경우에는 SQL문 update를 사용하여 사용자 비밀번호를 변경해준 후 나의 Main home으로 연결해 주는 함수입니다.