5. Database

게시판 만들기

Why 게시판?

- 웹 처음 배우는 사람에게 처음 만들어보라고 하는 건 언제나 게시판 아니면 블로그
- 간단하게 만들 수 있으면서도, 웹 개발 전반에 대한 지식을 쌓을 수 있다

웹 서비스 모델링

- 게시판을 만들기 위해선, 어떤 게 필요할까?
- 회원 관련 기능
- 글 기능
- 댓글 기능
- ~~친구 기능~~
  - ~~좋아요(싫어요?)~~

필요한 기능 - 회원 관련 기능

- 회원 가입
- 정보 수정
  - 비밀번호 변경
- 로그인
- 로그아웃

필요한 기능 - 글

- 글 목록 보기
- 글 하나 보기
- 새 글 쓰기
- 기존 글 수정하기
- 글 삭제하기
- 추가 기능
  - 태그 추가/삭제
  - 좋아요/좋아요 취소

필요한 기능 - 댓글

- 댓글 쓰기
- 댓글 수정하기
- 댓글 삭제하기
- 추가 기능
  - 댓글 좋아요

CRUD

- 글, 댓글, 유저 관련 기능들이 보면 공통점이 있다
- 생성, 수정, 삭제 기능이 들어가 있다는 것
- 정보를 보는 것은 당연히 포함되어 있는 거고
- 이 4가지 기능(생성, 보기, 수정, 삭제)을 통틀어 CRUD(Create, Read, Update, Delete)라고 한다
- 문제: 그래서 이걸 어떻게 만들까?

Database

- 쉽게 말해, 엑셀 파일이라고 생각하면 편하다.
- 다만 훨씬 빠르고, 프로그래머의 입맛에 맞게 사용하기가 쉽다
- 하나의 데이터베이스는 여러 개의 테이블로 이루어져 있고, 테이블은 다시 레코드로 나뉘고, 레코드는 여러 개의 칼럼으로 이루어져 있다
  - 엑셀 파일 하나에 시트가 여러 개 있고, 시트는 여러 개의 행이 있고 여러 개의 열이 있다

Database 예시 - 유저

id	username	password	nickname	registered_at
1	appaz5	4858f2ee1aafa2058a6b295adf276d26b1962ebe	ApaaZ	2015-03-01 17:28:16
2	michael	0afe0d583a5c209e402d681f904a297abfa20056	Mr.Michael	2015-03-01 18:11:37
3	bicky	965f85c562b43bee7eaf980d28b394a408c42831	BickyBicky	2015-03-05 08:19:33
4	jd320	74e7ffb6ba82c98c093828d213d9348daec4ff41	JD.Kim	2015-11-16 21:04:22
5	cjkeng	9513ee5bd8c6241bfe28732480f9a1335ce56c01	CJK	2015-11-16 22:16:01

Database 예시 - 유저

id	username	password	nickname	registered_at
1	appaz5	4858f2ee1aafa2058a6b295adf276d26b1962ebe	ApaaZ	2015-03-01 17:28:16
2	michael	0afe0d583a5c209e402d681f904a297abfa20056	Mr.Michael	2015-03-01 18:11:37
3	bicky	965f85c562b43bee7eaf980d28b394a408c42831	BickyBicky	2015-03-05 08:19:33
4	jd320	74e7ffb6ba82c98c093828d213d9348daec4ff41	JD.Kim	2015-11-16 21:04:22
5	cjkeng	9513ee5bd8c6241bfe28732480f9a1335ce56c01	CJK	2015-11-16 22:16:01

테이블

Database 예시 - 유저

id	username	password	nickname	registered_at
1	appaz5	4858f2ee1aafa2058a6b295adf276d26b1962ebe	ApaaZ	2015-03-01 17:28:16
2	michael	0afe0d583a5c209e402d681f904a297abfa20056	Mr.Michael	2015-03-01 18:11:37
3	bicky	965f85c562b43bee7eaf980d28b394a408c42831	BickyBicky	2015-03-05 08:19:33
4	jd320	74e7ffb6ba82c98c093828d213d9348daec4ff41	JD.Kim	2015-11-16 21:04:22
5	cjkeng	9513ee5bd8c6241bfe28732480f9a1335ce56c01	CJK	2015-11-16 22:16:01

레코드

Database 예시 - 유저

id	username	password	nickname	registered_at
1	appaz5	4858f2ee1aafa2058a6b295adf276d26b1962ebe	ApaaZ	2015-03-01 17:28:16
2	michael	0afe0d583a5c209e402d681f904a297abfa20056	Mr.Michael	2015-03-01 18:11:37
3	bicky	965f85c562b43bee7eaf980d28b394a408c42831	BickyBicky	2015-03-05 08:19:33
4	jd320	74e7ffb6ba82c98c093828d213d9348daec4ff41	JD.Kim	2015-11-16 21:04:22
5	cjkeng	9513ee5bd8c6241bfe28732480f9a1335ce56c01	CJK	2015-11-16 22:16:01

칼럼

CRUD 다시 정의하기

- Create: 테이블에 새로운 레코드 추가하기
- Read: 테이블에서 특정한 조건의 레코드 가져오기
- Update: 어떤 레코드의 칼럼 값 수정하기
- Delete: 테이블의 특정 레코드 삭제하기

본격적으로 들어가기 전 - OOP

- OOP - Object Oriented Programming
- 모델들을 만들고, 모델에 속성과 할 수 있는 일을 정의해서 모델간의 상호작용을 통해 프로그램을 작성하는 것
- 사람이라는 모델이 있고, 사람은 키, 몸무게 등의 속성을 가지고 있고, 먹거나, 자거나 할 수 있다
- 우리가 만들 게시판에도 여러 모델이 존재한다
  - 게시글
  - 댓글
  - 사용자
- 게시글은 어떤 속성을 가져야 할까?

게시글이 가져야 할 속성

- 제목
- 내용
- 작성자
- 글 쓴 시각

게시글 기능 구현

- cmd를 열고, pip install flask-sqlalchemy 를 실행, Flask-SQLAlchemy를 설치하자
  - SQLAlchemy는 Python 라이브러리 중 하나로, 데이터베이스를 Python 내에서 '잘'~~쉽다고는 안 했다~~ 다룰 수 있게 하는 라이브러리
  - Flask-SQLAlchemy 는 Flask를 만든 사람이, 그래도 쉽게 SQLAlchemy를 이용할 수 있게 만들어 놓은 라이브러리

```
# app.py
import datetime

from flask import Flask, render_template
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///db.sqlite'
db = SQLAlchemy(app)

class Article(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.Unicode(255))
    content = db.Column(db.UnicodeText)
    author = db.Column(db.Unicode(255))
    published_at = db.Column(db.DateTime)

@app.route('/')
def index():
    articles = Article.query.all()
    return render_template('index.html', articles=articles)
```

```
# init_db.py
from app import db

if __name__ == '__main__':
    db.drop_all()
    db.create_all()
```

```
app.config['SQLALCHEMY_DATABASE_URI'] =
'sqlite:///db.sqlite'

db = SQLAlchemy(app)

class Article(db.Model):

    id = db.Column(db.Integer,
primary_key=True)

    title =
db.Column(db.Unicode(255))

    content =
db.Column(db.UnicodeText)

    author =
db.Column(db.Unicode(255))

    published_at =
db.Column(db.DateTime)
```

app.config 는 웹서비스에 대한 설정들이 담겨 있는 사전이다. 여기에는 우리가 맘대로 키를 설정하고 값을 넣을 수도 있다.

SQLALCHEMY_DATABASE_URI 는 Flask-SQLAlchemy 가 쓰는 키. 이 키에 들어있는 값인 'sqlite:///db.sqlite'는, SQLite 라는 데이터베이스를 쓰겠으며, 그 파일 이름은 db.sqlite 라는 뜻이다.

```
app.config['SQLALCHEMY_DATABASE_URI'] =
'sqlite:///db.sqlite'

db = SQLAlchemy(app)

class Article(db.Model):

    id = db.Column(db.Integer,
primary_key=True)

    title =
db.Column(db.Unicode(255))

    content =
db.Column(db.UnicodeText)

    author =
db.Column(db.Unicode(255))

    published_at =
db.Column(db.DateTime)
```

SQLAlchemy 는 from flask_sqlalchemy import SQLAlchemy 로 불러온 것이다.

인자로 우리가 만든 app이 들어가 있음을 주목하라. SQLAlchemy 함수 내부에서는 app.config 에서 SQLALCHEMY_DATABASE_URI 값을 가져와서 이걸 기반으로 데이터베이스를 이용할 준비를 하게 된다.

```
app.config['SQLALCHEMY_DATABASE_URI'] =
'sqlite:///db.sqlite'

db = SQLAlchemy(app)

class Article(db.Model):

    id = db.Column(db.Integer,
primary_key=True)

    title =
db.Column(db.Unicode(255))

    content =
db.Column(db.UnicodeText)

    author =
db.Column(db.Unicode(255))

    published_at =
db.Column(db.DateTime)
```

Article이라는, 새로운 객체를 만들겠다...라는 뜻.

db.Model 이라고 되어 있는 부분은, Article 객체가 우리가 데이터베이스를 다루는데 쓸 모델이라고 선언하는 것.

Article 모델은 데이터베이스에선 article 테이블과 연결(mapping) 된다.

```
app.config['SQLALCHEMY_DATABASE_URI'] =
'sqlite:///db.sqlite'

db = SQLAlchemy(app)

class Article(db.Model):

    id = db.Column(db.Integer,
primary_key=True)

    title =
db.Column(db.Unicode(255))

    content =
db.Column(db.UnicodeText)

    author =
db.Column(db.Unicode(255))

    published_at =
db.Column(db.DateTime)
```

id라는 이름의 칼럼이 Article 객체 안에 있다... 라는 것.

db.Integer는 이 칼럼이 integer 타입만 받을 수 있다는 뜻.

primary_key=True 속성은, 레코드의 id는 테이블 전체에서 유일해야 한다는 제약조건(Constraint)

id가 1인 레코드가 두 개 이상 존재할 수 없다.

```
app.config['SQLALCHEMY_DATABASE_URI'] =
'sqlite:///db.sqlite'

db = SQLAlchemy(app)

class Article(db.Model):

    id = db.Column(db.Integer,
primary_key=True)

    title =
db.Column(db.Unicode(255))

    content =
db.Column(db.UnicodeText)

    author =
db.Column(db.Unicode(255))

    published_at =
db.Column(db.DateTime)
```

title, content, author, published_at 이라는 칼럼을 만들었다.

title은 Unicode 타입에, 최대 길이가 255자까지. author도 마찬가지.

content는 UnicodeText 타입 (길이 제한이 없는 문자열)

published_at은 DateTime 타입으로, 날짜를 받는다.

```
app.config['SQLALCHEMY_DATABASE_URI'] =
'sqlite:///db.sqlite'

db = SQLAlchemy(app)

class Article(db.Model):

    id = db.Column(db.Integer,
primary_key=True)

    title =
db.Column(db.Unicode(255))

    content =
db.Column(db.UnicodeText)

    author =
db.Column(db.Unicode(255))

    published_at =
db.Column(db.DateTime)
```

title, content, author, published_at 이라는 칼럼을 만들었다.

title은 Unicode 타입에, 최대 길이가 255자까지. author도 마찬가지.

content는 UnicodeText 타입 (길이 제한이 없는 문자열)

published_at은 DateTime 타입으로, 날짜를 받는다.

```
@app.route('/')
def index():
    articles = Article.query.all()
    return render_template('index.html', articles=articles)
```

Article 객체에 query(질의)를 날리는데, 무슨 질의냐하면 all(), 즉 모델 안에 있는 모든 값을 가져오라는 뜻이다.

```
@app.route('/')
def index():
    articles = Article.query.all()
    return render_template('index.html', articles=articles)
```

우리의 index.html 파일은 이제 게시판의 글 목록을 보여주는 기능을 하기로 했다. 그러려면 index.html 파일에서 Article 목록에 접근할 수 있어야 한다.

render_template 함수에 인자가 늘었다. 우리는 html에서 쓸 변수들을 render_template 함수를 통해 자유롭게 넘길 수 있다.

```
<!doctype html>
...
<body>
  <h1>글 목록</h1>
  <table>
    <tr><th>ID</th><th>제목</th><th>글쓴이</th></tr>
    {% for article in articles %}
    <tr>
      <td>{{ article.id }}</td>
      <td>{{ article.title }}</td>
      <td>{{ article.author }}</td>
    </tr>
    {% endfor %}
  </table>
</body>
</html>
```

테이블에 아무 것도 없어요!

- 넣은 글이 없으니 당연.
- https://goo.gl/6HpAm7
  - 글을 약간 넣었습니다.

글 읽기

```
@app.route('/<int:article_id>')
def read(article_id):
    article =
Article.query.filter_by(id=article_id).first()
    if article is None:
        # 위에서 from flask import redirect
        # 만약 찾는 글이 없으면, 첫 페이지로 이동한다.
        return redirect('/')
    return render_template('read.html', article=article)
```

글 읽기

```
...
<h1>{{ article.title }}</h1>
<p>{{ article.author }}</p>
<div>
{{ article.content }}
</div>
<a href="{{ url_for('index') }}">글 목록</a>
...
```

글 쓰기

```
@app.route('/write', methods=['GET', 'POST'])
def write():
    if request.method == 'GET':
        return render_template('write.html')
    title = request.form.get('title')
    content = request.form.get('content')
    author = request.form.get('author')
    article = Article(title=title, content=content,
author=author)
    db.session.add(article)
    db.commit()
    return redirect(url_for('read', article_id=article.id))
```

글 쓰기

```
...
<h1>글쓰기</h1>
<form action="{{ url_for('write') }}" method="POST">
  <input type="text" name="author" placeholder="글쓴이"><br>
  <input type="text" name="title" placeholder="제목"><br>
  <textarea name="content" placeholder="내용"></textarea> <br>
  <button type="submit">글쓰기</button>
</form>
...
```

글 지우기

```
@app.route('/delete/<int:article_id>')
def delete(article_id):
    article = Article.query.filter_by(id=article_id).first()
    if article is None:
        return redirect('/')
    db.session.delete(article)
    db.session.commit()
    return redirect('/')
```

글 수정하기

```
@app.route('/modify/<int:article_id>', method=['GET', 'POST'])
def modify(article_id):
    article = Article.query.filter_by(id=article_id).first()
    if article is None:
        return redirect('/')
    if request.method == 'GET':
        return render_template('modify.html', article=article)
    article.title = request.form.get('title')
    article.content = request.form.get('content')
    article.author = request.form.get('author')
    db.session.commit()
    return redirect(url_for('read', article_id=article_id))
```

글 수정하기

```
...
<h1>수정하기</h1>
<form action="{{ url_for('modify', article_id=article.id) }}" method="POST">
  <input type="text" name="author" placeholder="글쓴이" value="{{ article.author }}"><br>
  <input type="text" name="title" placeholder="제목" value="{{ article.title }}"><br>
  <textarea name="content" placeholder="내용">{{ article.content }}</textarea> <br>
  <button type="submit">수정하기</button>
</form>
...
```