Python Web Development

powered by Flask

Tweeter 소개

우리가 이번에 만들어볼 웹사이트의 이름은 Tweeter이다.
Tweeter는 Twitter와 유사하게 누구든지 홈 화면에 글을 작성할 수 있고, 다른 사람들이 작성한 글을 볼 수 있는 서비스이다.

최종적으로 우리가 보게 될 웹사이트는 여기서 미리 볼 수 있다.

데이터베이스

지난 주에는 Flask를 이용하여 간단한 Fake 로그인 어플리케이션을 만들어 보았다. 이번에는 실제로 데이터베이스를 연동하여 회원가입과 로그인을 구현해보도록 한다. 이를 위해 몇 가지 준비가 필요하다.

데이터베이스 선택

데이터베이스는 웹서비스의 요구 사항에 가장 잘 맞는 것을 선택하면 된다. 여기서는 유명한 NoSQL 데이터베이스 중 하나인 MongoDB를 사용하여 설명을 진행한다.

(MongoDB를 설치하는 방법은 별도의 문서를 참고한다)

SQL 기반 데이터베이스와 MongoDB

SQL 기반 데이터베이스에 익숙한 개발자들을 위해 유명한 SQL 기반 데이터베이스인 MySQL과 MongoDB를 비교해보면 다음과 같다:

- MySQL의 데이터베이스(Database)와 MongoDB의 데이터베이스(Database)는 동일한 개념이다.
- MySQL에 테이블(Table)이 존재했다면 MongoDB에는 컬렉션(Collection)이 존재한다.
- 또한 MySQL에서 데이터 하나를 나타내는 행(Row) 대신 MongoDB에는 다큐먼트(Document)가 존재한다.

MongoDB의 구성요소

MongoDB의 구성요소인 데이터베이스, 컬렉션, 다큐먼트에 대해 간단하게 알아보자.

데이터베이스

데이터베이스는 컬렉션의 집합이다. 데이터베이스를 생성하는 코드를 직접 작성하지 않아도 데이터베이스는 하위 컬렉션에 다큐먼트가 생성될 때 자동으로 생성된다.

컬렉션

컬렉션은 다큐먼트의 집합이다. 컬렉션도 데이터베이스와 마찬가지로 하위에 다큐먼트가 생성될 때 자동으로 생성된다.

다큐먼트

다큐먼트는 JSON 포맷으로 표현된다. 예를 들어, 유저 정보를 담고 있는 다큐먼트는 다음과 같다:

```
{
    "_id": ObjectId("911bb8e73724fe9b06a7e1e3e3176c7e"),
    "id": "hallazzang",
    "name": "Kim Hanjun",
    "password_hash": "72d50e7fd049fe3b3aacbe22faec1622",
}
```

눈여겨 볼 것은 `_id` 키다. `id`는 우리가 생성한, 유저의 로그인 아이디를 저장하는 키이지만 `_id`는 MongoDB의 모든 다큐먼트에 필수적으로 존재하는 다큐먼트 자체의 아이디이다. `_id`는 MongoDB에 의해 자동으로 생성된다.

MongoDB 사용해보기

MongoDB를 설치했다면 MongoDB 쉘(Shell)을 이용하여 MongoDB에 적응하는 시간을 가져보도록 한다.

MongoDB 쉘은 다음과 같이 실행한다:

```
$ mongo [dbname]
```

`[dbname]`에는 접속할 데이터베이스의 이름을 입력한다. `[dbname]`이 명시되지 않으면 기본값으로 `test` 데이터베이스가 선택된다. 그러므로 우리는 다음을 입력하여 `test` 데이터베이스에 접속하도록 하자:

```
$ mongo
```

다큐먼트 생성하기

MongoDB 쉘에서 `test` 데이터베이스에 성공적으로 접속했다면 다음을 입력하여 `users` 컬렉션에 새로운 다큐먼트들을 생성할 수 있다:

```
> db.users.insert({id: 'hallazzang', name: 'Kim Hanjun', school: 'PKNU'})
WriteResult({ "nInserted" : 1 })
> db.users.insert({id: 'gildong', name: 'Hong Gildong', school: 'PKNU'})
WriteResult({ "nInserted" : 1 })
> db.users.insert({id: 'gnazzallah', name: 'Mik nujnah', school: 'SNU'})
WriteResult({ "nInserted" : 1 })
```

(맨 앞의 `>`는 MongoDB 쉘이라는 것을 나타내는 표시이므로 입력하지 않도록 주의한다. 맨 앞에 `>`가 없는 행들은 출력 결과를 나타낸다)

다큐먼트 조회하기

우리가 생성한 다큐먼트들을 조회하기 위해서는 다음과 같이 입력한다:

```
> db.users.find({id: 'hallazzang'})
{ "_id" : ObjectId("583efe124464b57a64f19608"), "id" : "hallazzang",
  "name" : "Kim Hanjun", "school" : "PKNU" }
> db.users.find({school: 'PKNU'})
{ "_id" : ObjectId("583efe124464b57a64f19608"), "id" : "hallazzang",
  "name" : "Kim Hanjun", "school" : "PKNU" }
{ "_id" : ObjectId("583efe1f4464b57a64f19609"), "id" : "gildong",
  "name" : "Hong Gildong", "school" : "PKNU" }
```

쿼리를 살펴보면 SQL보다 훨씬 직관적인 인터페이스를 제공하는 것을 알 수 있다.

SQL 쿼리:

```
SELECT * FROM users WHERE id = "hallazzang"
SELECT * FROM users WHERE school = "PKNU"
```

MongoDB 쿼리:

```
db.users.find({id: 'hallazzang'})
db.users.find({school: 'PKNU'})
```

MongoDB의 쿼리 문법은 여러가지가 있으나 여기서 모두 다루지는 않는다.

다큐먼트 삭제하기

이번에는 생성한 다큐먼트들을 삭제하기 위해 다음과 같이 입력한다:

```
> db.users.remove({id: 'gildong'})
WriteResult({ "nRemoved" : 1 })
> db.users.remove({})
WriteResult({ "nRemoved" : 2 })
```

`db.users.remove({})`는 컬렉션 내의 모든 다큐먼트를 삭제하는 쿼리이다.

이상으로 MongoDB의 기능들을 간략하게 살펴보았다. 더 많은 기능들은 공식 문서를 살펴보거나 MongoDB 쉘의 `help()`를 이용해 알아볼 수 있다.

파이썬과 MongoDB

지금까지는 MongoDB 쉘에서 작업하는 방법을 다뤘다. 우리 웹사이트는 파이썬으로 작성될 것이므로 파이썬을 이용해 MongoDB에 접속하고, 데이터베이스를 다루는 방법을 설명한다.

우선 파이썬용 MongoDB API를 제공하는 PyMongo 라이브러리를 설치한다:

```
$ pip install pymongo
```

파이썬에서 이를 사용하기 위해서는 다음의 코드를 입력한다:

```
from pymongo import MongoClient

db = MongoClient().test
```

MongoDB 쉘에서와 파이썬에서의 문법은 거의 동일하다. 가장 큰 차이점은 MongoDB에서는 키를 따옴표로 감싸주지 않았다면 파이썬은 감싸줘야 한다는 것이다.

즉, MongoDB에서는 다음과 같이 입력했던 것을:

```
> db.users.insert({id: 'hallazzang'})
```

파이썬에서는 다음과 같이 입력해야 한다:

```
db.users.insert({'id': 'hallazzang'})
```

또한 MongoDB 쉘은 `find()`의 결과를 알아서 출력해주는데, PyMongo는 그렇지 않다(그리고 당연히 그렇지 않아야 한다).

PyMongo에서 `find()`의 결과는 `Cursor` 타입인데, 이는 `for` 문을 이용해 반복이 가능한 객체이다. 즉, 다음과 같은 코드로 `find()`의 결과를 순회할 수 있다:

```
result = db.users.find({'school': 'PKNU'})
for item in result:
    print item['id'], item['name'], item['school']
```

나머지는 필요할 때 부연 설명을 하기로 한다.

데이터베이스 구조 구상하기

이제 실제 회원가입과 로그인을 위한 데이터베이스의 구조를 구상할 차례다. 단순히 회원가입과 로그인만 있는 사이트는 허전하므로 게시글 작성 등의 기능도 추가해보도록 한다. 우리가 최종적으로 만드는 웹사이트는 방명록과 비슷한 모습을 가지게 될 것이다.

회원 정보

회원 한 명은 아이디, 비밀번호 해쉬(비밀번호는 **절대** 평문으로 저장하면 안 된다), 이름의 정보를 가진다고 가정한다.

따라서 회원 한 명을 나타내는 다큐먼트는 다음과 같이 표현될 것이다:

```
{
    "id": "hallazzang",
    "name": "Kim Hanjun",
    "password_hash": "72d50e7fd049fe3b3aacbe22faec1622"
}
```

(`_id` 키는 생략했음을 유의한다)

게시글

게시글은 작성자, 제목, 내용, 그리고 등록 시간을 가져야 한다.

따라서 게시글 하나를 나타내는 다큐먼트는 다음과 같이 표현될 것이다:

```
{
    "title": "Hello, World!",
    "content": "Lorem ipsum dolor sit amet, consectetur adipiscing ...",
    "author": "hallazzang",
}
```

Live coding again