

장고 차근차근 시작하기 2/E

당신의 파이썬/장고 페이스메이커가 되겠습니다. ☺

EP-09. 장고 모델 (ORM)

애플리케이션의 다양한 데이터 저장방법


- 데이터베이스 : **RDBMS**, NoSQL 등
- 파일 : 로컬, 외부 정적 스토리지
- 캐시서버 : memcached, redis 등

데이터베이스와 SQL

- 데이터베이스의 종류
 - **RDBMS** (관계형 데이터베이스 관리 시스템)
 - PostgreSQL, MySQL, SQLite, MS-SQL, Oracle 등
 - NoSQL : MongoDB, Cassandra, CouchDB, Google BigTable 등
- 데이터베이스에 쿼리하기 위한 언어 → SQL
 - 같은 작업을 하더라도, 보다 적은 수의 SQL, 보다 높은 성능의 SQL
 - 직접 SQL을 만들어내기도 하지만, ORM(Object-relational mapping)을 통해 SQL을 생성/실행하기도 합니다. → Not Magic.
 - **중요**) ORM을 쓰더라도, 내가 작성된 ORM코드를 통해 어떤 SQL이 실행되고 있는 지, 파악을 하고 이를 최적화할 수 있어야 합니다.

장고 ORM인 모델은 RDB만을 지원

- 장고 2.1.1 기준으로 기본 제공되는 backends

Tag: 2.1.1 ▾	django / django / db / backends /	Create new file	Upload files	Find file	History
 jdufresne and timgraham [2.1.x] Refs #29015 -- Added database name to PostgreSQL database nam...					Latest commit cae8490 on Aug 18
..					
base	[2.1.x] Fixed #29496 -- Fixed crash on Oracle when converting a non-u...	3 months ago			
dummy	Refs #27656 -- Updated django.db docstring verbs according to PEP 257.	2 years ago			
mysql	[2.1.x] Fixed #29544 -- Fixed regex lookup on MariaDB.	3 months ago			
oracle	[2.1.x] Fixed #29496 -- Fixed crash on Oracle when converting a non-u...	3 months ago			
postgresql	[2.1.x] Refs #29015 -- Added database name to PostgreSQL database nam...	a month ago			
postgresql_psycopg2	Removed postgresql_psycopg2.version	a year ago			
sqlite3	Refs #29350 -- Fixed 'invalid escape sequence' warning in SQLite intr...	4 months ago			
__init__.py	Fixed #22603 -- Reorganized classes in django.db.backends.	4 years ago			
ddl_references.py	Fixed #28465 -- Unified index SQL creation in DatabaseSchemaEditor	a year ago			
signals.py	Fixed #13798 -- Added connection argument to the connection_created s...	8 years ago			
utils.py	Used Decimal.scaleb() in backends.utils.format_number() and DecimalFi...	9 months ago			

다양한 파이썬 ORM ([awesome-python#orm](https://t.me/awesome-python-orm))

- Relational Databases
 - Django Models
 - SQLAlchemy
 - Orator
 - Peewee
 - PonyORM
- NoSQL Databases
 - django-mongodb-engine
 - hot-redis
 - MongoEngine
 - PynamoDB

장고의 강점은 Model과 Form

- 물론, 장고에서도 다양한 ORM, 라이브러리 사용 가능.
- 강력한 Model/Form
- 물론, 적절하게 섞어쓰실 수도 있습니다.
- SQL을 직접 실행하실 수도 있습니다.
 - 직접 SQL문자열을 조합하지마시고, 인자로 처리하세요. → SQL Injection 방지

```
from django.db import connection
```

```
with connection.cursor() as cursor:  
    cursor.execute("UPDATE bar SET foo = 1 WHERE baz = %s", [self.baz])  
    cursor.execute("SELECT foo FROM bar WHERE baz = %s", [self.baz])  
    row = cursor.fetchone()  
    print(row)
```

Django Model

장고 내장 ORM

- <데이터베이스 테이블>과 <파이썬 클래스>를 **1:1로 매핑**
 - 모델 클래스명은 **단수형**으로 지정 - 예: Posts (X), Post (O)
 - 클래스이기에 필히 "첫 글자가 대문자인 CamelCase 네이밍"
 - 매핑되는 모델 클래스는 DB 테이블 필드 내역이 일치해야합니다.
 - 모델을 만들기 전에, 서비스에 맞게 **데이터베이스 설계가 필수.**
 - 이는 **데이터베이스**의 영역. ➔ 데이터베이스 학습도 필요합니다.

```
from django.db import models
```

```
class Post(models.Model):  
    title = models.CharField(max_length=100)  
    content = models.TextField()  
    created_at = models.DateTimeField(auto_now_add=True)  
    updated_at = models.DateTimeField(auto_now=True)
```

모델 활용 순서

- 장고 모델을 통해, 데이터베이스 형상을 관리할 경우
 1. 모델 클래스 작성
 2. 모델 클래스로부터 마이그레이션 파일 생성 → makemigrations 명령
 3. 마이그레이션 파일을 데이터베이스에 적용 → migrate 명령
 4. 모델 활용
- 장고 외부에서, 데이터베이스 형상을 관리할 경우
 - 데이터베이스로부터 모델 클래스 소스 생성 → inspectdb 명령
 - 모델 활용

모델명과 DB 테이블명

- DB 테이블명 : 디폴트 "앱이름_모델명"
- 예
 - blog 앱
 - Post 모델 → "blog_post"
 - Comment 모델 → "blog_comment"
 - shop 앱
 - Item 모델 → "shop_item"
 - Review 모델 → "shop_review"
- 커스텀 지정
 - 모델 Meta 클래스의 db table 속성

새로운 모델을 만들어봅시다.

shop/models.py

```
from django.db import models
```

```
class Item(models.Model):  
    name = models.CharField(max_length=100)  
    desc = models.TextField(blank=True)  
    price = models.PositiveIntegerField()  
    created_at = models.DateTimeField(auto_now_add=True)  
    updated_at = models.DateTimeField(auto_now=True)
```

적용순서

- Item 모델 정의
- 마이그레이션 파일 생성
- 마이그레이션 파일 적용
- 데이터베이스 확인
 - DB 종류에 따라 다양한 방법
 - <https://sqlitebrowser.org/>

```
Windows PowerShell
PS C:\dev\askcompany> python .\manage.py migrate shop
Operations to perform:
  Apply all migrations: shop
Running migrations:
  Applying shop.0001_initial... OK
PS C:\dev\askcompany> python .\manage.py dbshell
SQLite version 3.23.1 2018-04-10 17:39:29
Enter ".help" for usage hints.
sqlite> .tables
auth_group                                django_admin_log
auth_group_permissions                    django_content_type
auth_permission                           django_migrations
auth_user                                 django_session
auth_user_groups                          shop_item
auth_user_user_permissions
sqlite> .schema shop_item
CREATE TABLE IF NOT EXISTS "shop_item" ("id" integer NOT NULL PR
IMARY KEY AUTOINCREMENT, "name" varchar(100) NOT NULL, "desc" te
xt NOT NULL, "price" integer unsigned NOT NULL, "created_at" dat
etime NOT NULL, "updated_at" datetime NOT NULL);
sqlite> .quit
PS C:\dev\askcompany>
```

인생은 짧습니다.
파이썬/장고를 쓰세요.