Django 的数据模型

> **Organization**: 千锋教育 Python 教学部 **Date**: 2019-03-12 **Author**: 张旭

Web 开发中数据的存储

一个网站从上线开始，会逐渐积累大量的用户，而这些用户又会产生出大量的数据。

如果运营得力，这些数据最终都会产生价值。所以说，这些数据是网站最宝贵的资产。

面对大量的数据应该如何存储呢？答案就是：数据库

常见的数据库有很多种，用得最多的一般是关系型数据库。比如: MySQL、PostgreSQL、Oracle、SQLite3 等

数据库的使用有自己的规范，也有自己的语法，一般叫做 SQL (结构化查询语句)

但 SQL 语句的使用相对来说比较繁琐，通常 SQL 的使用相对繁琐，写在程序里面也显得比较突兀。

为了更方便的操作数据库，人们开发出了 ORM 系统

数据库配置

打开 project/settings.py 文件，可以看到默认的数据库配置:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

通常，这个配置文件使用 SQLite 作为默认数据库。如果只是想尝试下 Django，这是最简单的选择。

Python 内置 SQLite，所以无需安装额外东西来使用它。

当开始一个真正的项目时，你可能更倾向使用一个功能更加丰富的数据库，例如 MySQL。

如果你想使用其他数据库，你需要安装合适的 database bindings ，然后改变设置文件中 DATABASES 'default' 项目中的一些键值：

1. ENGINE 可选值有:
    - `'django.db.backends.sqlite3'`
    - `'django.db.backends.postgresql'`
    - `'django.db.backends.mysql'`
    - `'django.db.backends.oracle'`
2. NAME - 数据库的名称

- 如果使用的是 SQLite，数据库将是你电脑上的一个文件，在这种情况下， NAME 应该是此文件的绝对路径。 默认值 `os.path.join(BASE_DIR, 'db.sqlite3')`
- 如果不使用 SQLite，则必须添加一些额外设置，比如 USER、PASSWORD、HOST 等等。

app 配置及 数据库初始化

1. 我们通过 `startapp` 命令创建的每一个 app 都需要将其在 settings.py 文件中声明一下

```
INSTALLED_APPS = [
    # Django 默认配置
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    # 个人定义的配置
    'joke',
]
```

2. app 配置完以后, 可以通过以下命令初始化相关的数据库表格

```
python manage.py migrate
```

数据模型的定义

数据模型是程序运行的基础，设计优良的数据模型能为后续开发带来很大的助力。

数据模型定义在 models.py 文件中，详情如下：

```
from django.db import models

class Article(models.Model):
    CATEGORY = [
        ('event', '热门事件'),
        ('anime', '动漫'),
        ('military', '军事'),
    ]
    title = models.CharField(max_length=32, unique=True)
    category = models.CharField(max_length=16, choices=CATEGORY)
    date = models.DateTimeField(auto_now_add=True)
    content = models.TextField()
    author = models.ForeignKey(User, on_delete=models.CASCADE)
```

Field 详解

1. 常用的 Field 的类型
    - `IntegerField`: 整型类型
    - `FloatField`: 浮点类型
    - `BooleanField`: 布尔类型
    - `CharField`: 字符串类型
    - `TextField`: 长文本类型
    - `DateTimeField`: 日期时间类型，包含完整的年、月、日、时、分、秒
    - `DateField`: 日期类型，只包含年、月、日
    - `TimeField`: 时间类型，只包含时、分、秒
    - `ForeignKey`: 外键，关联到另一个 Model，用来表示数据模型之间的关系
2. Field 中的一些选项
    - `null`: 针对数据库, 允许数据库该字段为 Null
    - `blank`: 针对 Model 本身, 允许传入字段为空. blank 为 True 时, 对数据库来说, 该字段依然为必填项
    - `default`: 尽量使用 default, 少用 null 和 blank
    - `choices`: 针对 CharField 类型的可选项，限定该字段的取值范围
    - `primary_key`: 非必要时不要设置, 用默认 id, 保持条目自增、有序、唯一
    - `unique`: 表明该字段是唯一字段，每个值只能出现一次，如果出现多次会报错
    - `db_index`: (True | False)
    - `max_length`: CharField 的最大长度
    - `auto_now`: 每次 save 时，更新为当前时间
    - `auto_now_add`: 只记录创建时的时间, 保存时不更新

Model.objects 及 QuerySet 的一些操作语句

1. Django ORM 的大多数操作都封装在一个名为 objects 的类属性上

```
# 取出所有文章
articles = Article.objects.all()

# 检查 id = 3 的文章是否存在
if Article.objects.filter(id=3).exists():
    print('存在')
else:
    print('不存在')
```

2. objects 常用方法有:
    - `all()`: 取出所有数据
    - `filter()`: 根据条件过滤出某些数据
    - `exclude()`: 根据条件排除某些数据
    - `create()`: 创建一个对象
    - `order_by()`: 根据某个字段进行增序或者降序排列
    - `count()`: 统计取出的数据有多少个

- `exists()`: 检查某条数据是否存在
- `latest()`: 取出最新的一条数据
- `earliest()`: 取出最早的一条数据
- `first()`: 取出第一条数据
- `last()`: 取出最后一条数据

3. filter 中常见的过滤条件:
    - 过滤出包含在某个列表中的数据: `filter(id__in=[123, 555, 231])`
    - 过滤出在某个范围内的数据: `filter(id__range=[123, 456])`
    - 过滤出包含某文本的数据: `filter(name__contains='123')`
    - 过滤出数值小于等于某值的数据: `filter(id__lte=123)`
    - 其他的比较过滤条件: `gt / gte / lt / lte`