Python05

前后端分离项目 和 前后端不分离项目

- 前后端分离：目前主流
  - 面临的问题：跨域
  - 优点：加载速度快，降低服务器压力
  - 缺点：不利于SEO优化--搜索优化
- 前后端不分离：之前主流
  - java ，php 的常见做法.
  - 优点：SEO优化更好
  - 确定：加载速度慢，服务器压力大

ORM

ORM 对象关系映射：是一种使用 面向对象的类 来操作数据库的方式；取代了SQL；

出现原因：

数据库的品牌有很多种-- mysql, sql server, sqlite, sql anywhere, oracle, mongodb...

这些数据库都是用 SQL 进行操作，但是细节上有所不同.

面临的挑战：

如果项目更换数据库，则SQL需要微调. 微调所有sql 工作量极大!

解决：

ORM，把数据库语句使用 面向对象进行封装. 然后利用面向对象进行数据库操作，底层会根据不同的数据库转化成不同的 sql 语句.

模块安装

python需要安装 mysqlclient 模块才能拥有 ORM 操作mysql的功能

> 有极大概率无法自动安装成功， 则可以使用手动方式

```
pip install mysqlclient
```

查看： `pip list`

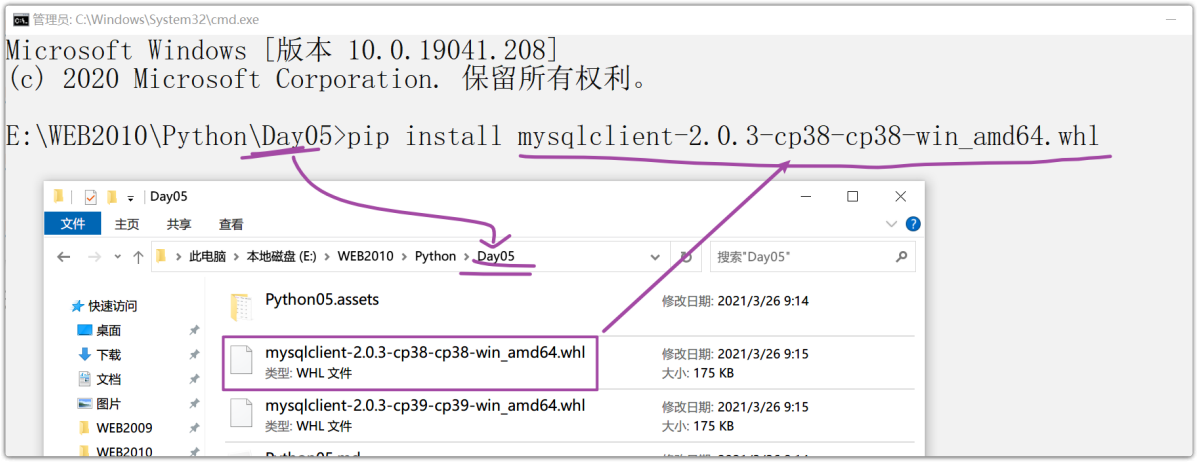
手动下载

https://pypi.org/project/mysqlclient/#files

使用 `py -V` 查看python版本

Filename, size 对应不同版本的python	File type	Python version	Upload date	Hashes
mysqlclient-2.0.3-cp36-cp36m-win_amd64.whl (179.0 kB)	Wheel	cp36	Jan 1, 2021	View
mysqlclient-2.0.3-cp37-cp37m-win_amd64.whl (179.0 kB)	Wheel	cp37	Jan 1, 2021	View
mysqlclient-2.0.3-cp38-cp38-win_amd64.whl (179.4 kB)	Wheel	cp38	Jan 1, 2021	View
mysqlclient-2.0.3-cp39-cp39-win_amd64.whl (179.4 kB)	Wheel	cp39	Jan 1, 2021	View
mysqlclient-2.0.3.tar.gz (88.9 kB)	Source	None	Jan 1, 2021	View

安装方式： 文件下载之后所在的目录下，打开cmd，执行 `pip install xxxx.whl`



新建项目包： `django-admin startproject djorm`

配置项

ORM不需要sql语句就可以操作数据库，其中搭建数据库链接的操作 由系统自动完成；

我们需要把数据库的连接相关配置进行书写.

settings.py

djorm > settings.py > ...

```
73  # Database
74  # https://docs.djangoproject.com/en/3.1/ref/settings/#databases
75
76  DATABASES = {
77      'default': {
78          'ENGINE': 'django.db.backends.mysql',
79          'NAME': 'web2010',
80          'USER': 'root',
81          'PASSWORD': '',
82          'HOST': '127.0.0.1',
83          'PORT': '3306',
84      }
85  }
```

ORM的引入，要求项目更加规范：

通常一个项目要分很多个模块：广告 用户 商品 购物车 ...

不同的模块分开书写

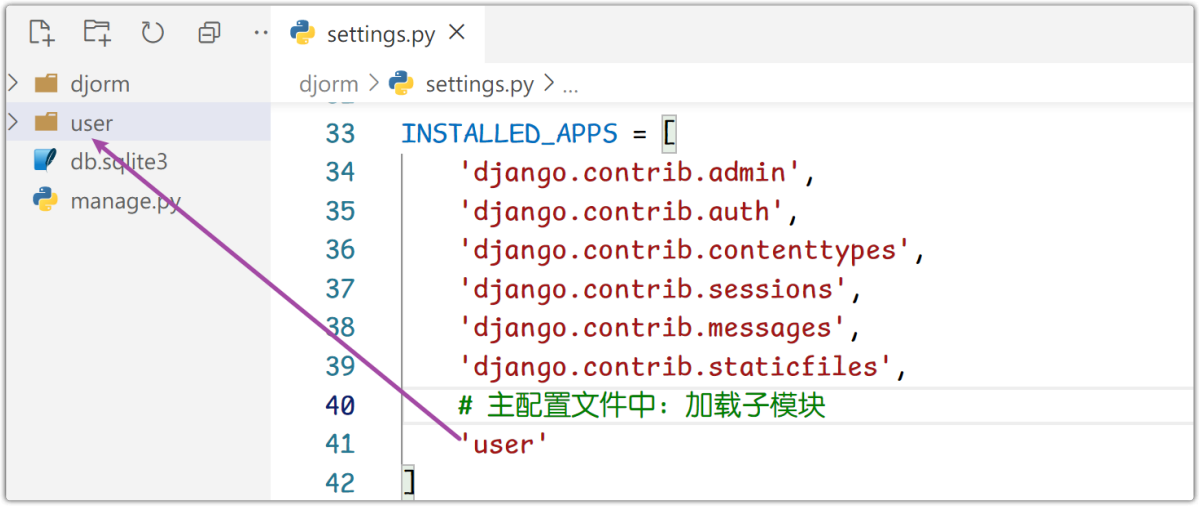
创建模块的命令行：必须在项目下执行

```
django-admin startapp 模块名
```

例如： `django-admin startapp user`

```
E:\WEB2010\Python\Day05\djorm>django-admin startapp user
```

配置文件中，加载子模块



models.py

```python
from django.db import models

# Create your models here.

# ORM 对象关系映射
'''
程序员发现 面向对象写法 和 表结构非常相似

表名            类名
表字段          类属性
表数据          实例化对象

例如：用户表 User
CREATE TABLE IF NOT EXISTS user(
    id INT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(255) NOT NULL,
    age TINYINT UNSIGNED NOT NULL,
    phone CHAR(11) NOT NULL UNIQUE,
    email VARCHAR(255) NOT NULL UNIQUE,
    gender TINYINT UNSIGNED NOT NULL,
) CHARSET=utf8
'''

# 实现表->类的转化
```

```python
class User(models.Model):
    # 必须继承父类，才是合格的ORM类
    # id：由于主键是必备的，所以父类中已经默认具备此属性．子类无需书写
    name = models.CharField(max_length=255, null=False)
    # 数字类型 都自带无符号特性
    age = models.SmallIntegerField(null=False)
    phone = models.CharField(max_length=11, null=False)
    email = models.CharField(max_length=255, null=False)
    gender = models.SmallIntegerField(null=False)


'''
需要通过 django 提供的命令，让 ORM 参考 User 类 生成对应的表

$ 是linux的风格，代表接下来的是命令行

检测是否有 ORM 类变更
$ py manage.py makemigrations

把检测的变更应用到数据库
$ py manage.py migrate

'''
```

ORM生成表命令，必须在项目下执行，即 `manage.py` 文件所在目录

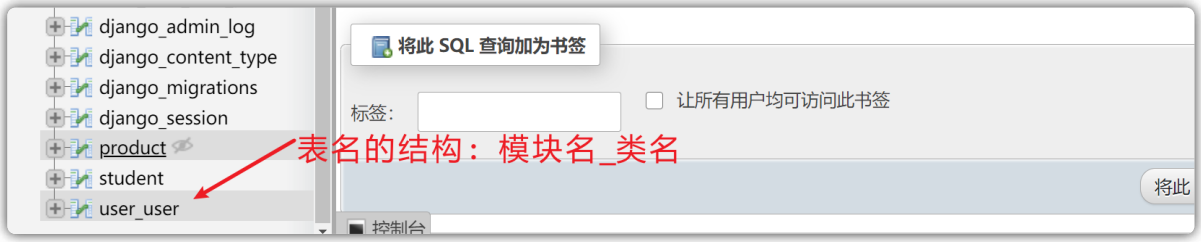
```
需要通过 django 提供的命令，让 ORM 参考 User 类 生成对应的表
$ 是linux的风格，代表接下来的是命令行

检测是否有 ORM 类变更
$ py manage.py makemigrations

把检测的变更应用到数据库
$ py manage.py migrate
```

```
E:\WEB2010\Python\Day05\djorm>py manage.py makemigrations
Migrations for 'user':
  user\migrations\0001_initial.py
    - Create model User

E:\WEB2010\Python\Day05\djorm>
```

```
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
  Applying user.0001_initial... OK

E:\WEB2010\Python\Day05\djorm>
```

用户表

py manage.py migrate
首次执行会生成很多默认文件



表名的结构：模块名_类名