

# 多人博客项目

## 概述

Django采用MVC架构设计的开源的WEB快速开发框架。

优点：

能够快速开发，自带ORM, Template, Form, Auth核心组件

MVC设计模式

实用的管理后台Admin

简洁的url设计

周边插件丰富

缺点：

框架重，因为东西大而全

同步阻塞

所以Django的设计目标就是一款大而全，便于企业快速开发项目的框架，因此企业应用较广。

## Django版本

| Django Version | Python Version          |
|----------------|-------------------------|
| 1.8            | 2.7, 3.2, 3.3, 3.4, 3.5 |
| 1.9, 1.10      | 2.7, 3.4, 3.5           |
| 1.11           | 2.7, 3.4, 3.5, 3.6, 3.7 |
| 2.0            | 3.4, 3.5, 3.6, 3.7      |
| 2.1, 2.2       | 3.5, 3.6, 3.7           |

## 安装Django

Python 使用3.6.x

Django的下载地址 <https://www.djangoproject.com/download/>

Python版本依赖，参看<https://docs.djangoproject.com/en/1.11/faq/install/#faq-python-version-support>

| Django version | Python版本  |
|----------------|---|
| 1.8            | 2.7, 3.2 (until the end of 2016), 3.3, 3.4, 3.5 |
| 1.9, 1.10      | 2.7, 3.4, 3.5                                   |
| 1.11(LTS)      | 2.7, 3.4, 3.5, 3.6, 3.7 (added in 1.11.17)      |
| 2.0            | 3.4, 3.5, 3.6, 3.7                              |
| 2.1            | 3.5, 3.6, 3.7                                   |

```
$ pip install django==1.11.20
```

本次使用Django 1.11版本，它也是长期支持版本LTS，请在虚拟环境中安装。

在虚拟环境路径中，Lib/site-packages/django/bin下有一个django-admin.py，一起从它开始。

```
$ django-admin --version
$ django-admin
```

Type '`django-admin help <subcommand>`' for help on a specific subcommand.

Available subcommands:

```
[django]
  check
  compilemessages
  createcachetable
  dbshell
  diffsettings
  dumpdata
  flush
  inspectdb
  loaddata
  makemessages
  makemigrations
  migrate
  runserver
  sendtestemail
  shell
  showmigrations
  sqlflush
  sqlmigrate
  sqlsequencereset
  squashmigrations
  startapp
  startproject
  test
  testserver
```

```
$ django-admin startproject --help
```

注意：本文如若未特殊声明，所有的命令操作都在项目根目录下

## 创建django项目

```
$ django-admin startproject blog .
```

上句命令就在当前项目根目录中构建了Django项目的初始文件。点 代表项目根目录。

```
F:\CLASSES\TPROJECTS\BLOG10
├─ manage.py
└─ blog
    ├─ settings.py
    ├─ urls.py
    ├─ wsgi.py
    └─ __init__.py
```

### 重要文件说明

- manage.py: 本项目管理的命令行工具。应用创建、数据库迁移等都使用它完成
- blog/settings.py: 本项目的配置文件。数据库参数等
- blog/urls.py: URL路径映射配置。默认情况下，只配置了/admin的路由。
- blog/wsgi: 定义WSGI接口信息。一般无须改动。

## 数据库配置

使用数据库，需要修改默认的数据库配置。

在主项目的settings.py下的DATABASES。默认使用的sqlite，修改为mysql。

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'blog',
        'USER': 'wayne',
        'PASSWORD': 'wayne',
        'HOST': '192.168.142.140',
        'PORT': '3306',
    }
}
```

HOST 数据库主机。缺省是空字符串，代表localhost。如果是'/'开头表示使用Unix Socket连接。

POST 端口

USER 用户名

PASSWORD 密码

NAME 库名

OPTIONS 选项，字典，参考MySQL文档

数据库引擎ENGINE

内建的引擎有

- 'django.db.backends.postgresql'
- 'django.db.backends.mysql'
- 'django.db.backends.sqlite3'
- 'django.db.backends.oracle'

## MySQL数据库驱动

<https://docs.djangoproject.com/en/2.0/ref/databases/>

Django支持MySQL 5.5+

Django官方推荐使用本地驱动mysqlclient 1.3.7 +

```
$ pip install mysqlclient
```

windows下安装错误 error: Microsoft Visual C++ 14.0 is required.解决方法

1、下载Visual C++ Redistributable Packages 2015、2017安装，但是即使安装后，确实看到了V14库，也不保证安装mysqlclient就成功

2、直接安装编译好的wheel文件

mysqlclient-1.3.13-cp35-cp35m-win\_amd64.whl , python 3.5使用

mysqlclient-1.3.13-cp36-cp36m-win\_amd64.whl , python 3.6使用

mysqlclient-1.4.2-cp37-cp37m-win\_amd64.whl, python 3.7使用

```
$ pip install mysqlclient-1.3.13-cp35-cp35m-win_amd64.whl
```

参考 <https://stackoverflow.com/questions/29846087/microsoft-visual-c-14-0-is-required-unable-to-find-vcvarsall-bat>

下载地址

<https://www.lfd.uci.edu/~gohlke/pythonlibs/>

## 创建应用

创建用户应用 `$ python manage.py startapp user`

该应用完成以下功能

- 用户注册
- 用户登录

创建应用后，项目根目录下产生一个user目录，有如下文件：

- admin.py：管理站点模型的声明文件
- models.py：模型层Model类定义
- views.py：定义URL响应函数
- migrations包：数据迁移文件生成目录
- apps.py：应用的信息定义文件

## 注册应用

在settings.py中，增加user应用。  
目的是为了**后台管理**admin使用，或**迁移**migrate使用

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'user',  
]
```

## 模型Model



| 字段类              | 说明  |
|------------------|---|
| AutoField        | 自增的整数字段。<br>如果不指定，django会为模型类自动增加主键字段   |
| BooleanField     | 布尔值字段，True和False<br>对应表单控件CheckboxInput   |
| NullBooleanField | 比BooleanField多一个null值   |
| CharField        | 字符串，max_length设定字符长度<br>对应表单控件TextInput   |
| TextField        | 大文本字段，一般超过4000个字符使用<br>对应表单控件Textarea   |
| IntegerField     | 整数字段  |
| BigIntegerField  | 更大整数字段，8字节  |
| DecimalField     | 使用Python的Decimal实例表示十进制浮点数。max_digits总位数，decimal_places小数点后的位数  |
| FloatField       | Python的Float实例表示的浮点数  |
| DateField        | 使用Python的datetime.date实例表示的日期<br>auto_now=False每次修改对象自动设置为当前时间。auto_now_add=False对象第一次创建时自动设置为当前时间。auto_now_add、auto_now、default互斥<br>对应控件为TextInput，关联了一个Js编写的日历控件 |
| TimeField        | 使用Python的datetime.time实例表示的时间，参数同上  |
| DateTimeField    | 使用Python的datetime.datetime实例表示的时间，参数同上  |
| FileField        | 一个上传文件的字段   |
| ImageField       | 继承了FileField的所有属性和方法，但是对上传的文件进行校验，确保是一个有效的图片  |

## 字段选项

| 值           | 说明                                     |
|-------------|--|
| db_column   | 表中字段的名称。如果未指定，则使用属性名                   |
| primary_key | 是否主键                                   |
| unique      | 是否是唯一键                                 |
| default     | 缺省值。这个缺省值不是数据库字段的缺省值，而是新对象产生的时候被填入的缺省值 |
| null        | 表的字段是否可为null，默认为False                  |
| blank       | Django表单验证中，是否可以不填写，默认为False           |
| db_index    | 字段是否有索引                                |

## 关系类型字段类

| 类               | 说明   |
|-----------------|--|
| ForeignKey      | 外键，表示一对多<br>ForeignKey('production.Manufacturer')<br>自关联ForeignKey('self') |
| ManyToManyField | 表示多对多。   |
| OneToOneField   | 表示一对一。   |

一对多时，自动创建会增加\_id后缀。

- 从一访问多，使用 `对象.小写模型类_set`
- 从一访问一，使用 `对象.小写模型类`

访问id `对象.属性_id`

## 创建User的Model类

- 基类 `models.Model`
- 表名不指定默认使用 `<appname>_<model_name>`。使用Meta类修改表名

```
from django.db import models

# Create your models here.

class User(models.Model):
    class Meta:
        db_table = 'user'
    id = models.AutoField(primary_key=True)
    name = models.CharField(max_length=48, null=False)
    email = models.CharField(max_length=64, unique=True, null=False)
    password = models.CharField(max_length=128, null=False)

    def __repr__(self):
```

```
        return '<user {} {}>'.format(self.id, self.name)

    __str__ = __repr__
```

Meta类的使用，参考 [https://docs.djangoproject.com/en/1.11/ref/models/options/#django.db.models.Options.db\\_table](https://docs.djangoproject.com/en/1.11/ref/models/options/#django.db.models.Options.db_table)

## 迁移Migration

迁移：从模型定义生成数据库的表

### 1、生成迁移文件``

```
$ python manage.py makemigrations
Migrations for 'user':
  user/migrations/0001_initial.py
    - Create model User
```

生成如下文件

```
user
├─ migrations
│   └─ 0001_initial.py
│   └─ __init__.py
```

修改过Model类，还需要调用makemigrations，然后migrate，迁移文件的序号会增加。

注意：

迁移的应用必须在settings.py的INSTALLED\_APPS中注册。

不要随便删除这些迁移文件，因为后面的改动都是要依据这些迁移文件的。

```
# 0001_initial.py 文件内容如下
class Migration(migrations.Migration):

    initial = True

    dependencies = [
    ]

    operations = [
        migrations.CreateModel(
            name='User',
            fields=[
                ('id', models.AutoField(primary_key=True, serialize=False)),
                ('name', models.CharField(max_length=48)),
                ('email', models.CharField(max_length=64, unique=True)),
                ('password', models.CharField(max_length=128)),
            ],
            options={
                'db_table': 'user',
            },
        ),
    ]
```



## 2、执行迁移生成数据库的表

```
$ python manage.py migrate
```

执行了迁移，还同时生成了admin管理用的表。

查看数据库，user表创建好了，字段设置完全正确。

# Django后台管理

## 1、创建管理员

管理员用户名admin

密码adminadmin

```
$ manage.py createsuperuser
Username (leave blank to use 'wayne'):admin
Email address:
Password:
Password (again):
Superuser created successfully.
```

## 2、本地化

settings.py中设置语言、时区

语言名称可以查看 django\contrib\admin\locale 目录

```
LANGUAGE_CODE = 'zh-Hans' #'en-us'
USE_TZ = True
TIME_ZONE = 'Asia/Shanghai' #'UTC'
```

## 3、启动WEB Server

```
$ python manage.py runserver
```

默认启动8000端口



#### 4、登录后台管理

后台登录地址 `http://127.0.0.1:8000/admin`

Django administration

Username:

Password:

Log in

#### 5、注册应用模块

在user应用的admin.py添加

```
from django.contrib import admin
from .models import User
# Register your models here.

admin.site.register(User) # 注册
```



user就可以在后台进行增删改了。

## 路由\*\*

路由功能就是实现URL模式匹配和处理函数之间的映射。

路由配置要在项目的urls.py中配置，也可以多级配置，在每一个应用中，建立一个urls.py文件配置路由映射。

url函数

url(regex, view, kwargs=None, name=None), 进行模式匹配

regex: 正则表达式，与之匹配的 URL 会执行对应的第二个参数 view

view: 用于执行与正则表达式匹配的 URL 请求

kwargs: 视图使用的字典类型的参数

name: 用来反向获取 URL

urls.py内容如下

```

from django.conf.urls import url
from django.contrib import admin

from django.http import HttpRequest, HttpResponse
def index(request:HttpRequest):
    """视图函数：请求进来返回响应"""
    return HttpResponse(b'welcome to magedu.com')

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^$', index),
    url(r'^index$', index),
]

```

url(r'^index/\$', index)

<http://127.0.0.1:8000/index/> 可以访问

<http://127.0.0.1:8000/index> 可以访问，但会补一个/

url(r'^index\$', index)

<http://127.0.0.1:8000/index> 可以访问

<http://127.0.0.1:8000/index/> 不可以访问

请求信息测试和JSON响应

```

from django.http import HttpRequest, HttpResponse, JsonResponse
def index(request:HttpRequest):
    """视图函数：请求进来返回响应"""
    d = {}
    d['method'] = request.method
    d['path'] = request.path
    d['path_info'] = request.path_info
    d['GETparams'] = request.GET

    return JsonResponse(d)

```

在项目中首页多数使用HTML显示，为了加载速度快，一般多使用静态页面。如果首页内容多，还有部分数据需要变化，将变化部分使用AJAX技术从后台获取数据。

本次，为了学习模板技术，只将首页采用Django的模板技术实现。