第01课：创建 Django 项目和应用

本篇将会涉及以下内容：

- Virtualenv 的安装的使用
- Django 的安装
- Django 项目的创建
- Django 项目的运行
- Django 应用的创建

网站规划与设想

现在开始正式进入 Django Web 开发极简实战的达人课，我们将通过创建一个网站来入门 Django Web 开发，了解 Django 框架的基本结构和 Django Web 开发的基本流程。

首先在此规划一下，在接下来的课程中将会完成创建在线视频教程网站的一些基本页面和功能。

我们在本次达人课中创建的是一个提供 Python 视频教程在线播放的网站，网站主要由首页、视频列表页和视频详情页三个功能页面和其他辅助页面组成，出于教学演示的考虑，视频源存储在本地，同时对视频设置播放密码，来实现部分视频需要用户进行登录才能观看，网站大致有如下页面和功能。

页面

每一个内容丰富的网站都是由许许多多的页面构成的，对于我们需要创建的网站而言，以下几个页面是网站信息的主要载体，是必须必要存在的：

- 一个首页，用于展示各个分类的视频信息；
- 一个分类列表页面，用于显示特定分类的视频信息；
- 一个视频详情页面，用于显示视频的详情和控制视频的播放，也用于视频的评论和点赞；
- 一个观看历史页面，用于记录用户的观看历史，方便用户查看自己的学习记录；
- 一个注册页面，用于新用户的注册，通过输入 Email 地址、密码和验证码进行注册；
- 一个登录页面，用于用户的登录，输入用户名和密码即可进行登录。

功能

除了基本的展示外，我们的网站还需要有各种交互功能，而以下功能，是我们为网站所进行的设定。

- 用户认证功能：实现新用户的注册和老用户的登录以及登录用户的注销；
- 视频列表分页功能：在分类列表页面，能够通过“上一页”、“下一页”对视频列表进行翻页；
- 视频评论功能，能够对视频教程发表评论，此功能通过第三方评论工具来实现；
- 视频点赞功能，登录用户能够对视频进行点赞；
- 视频页面分享功能，能够将视频详情页面进行社会化分享，此功能通过百度分享接口实现。

上述的规划将作为接下来进行 Django 项目实战的指导，下面开始 Django Web 开发的实战应用吧！

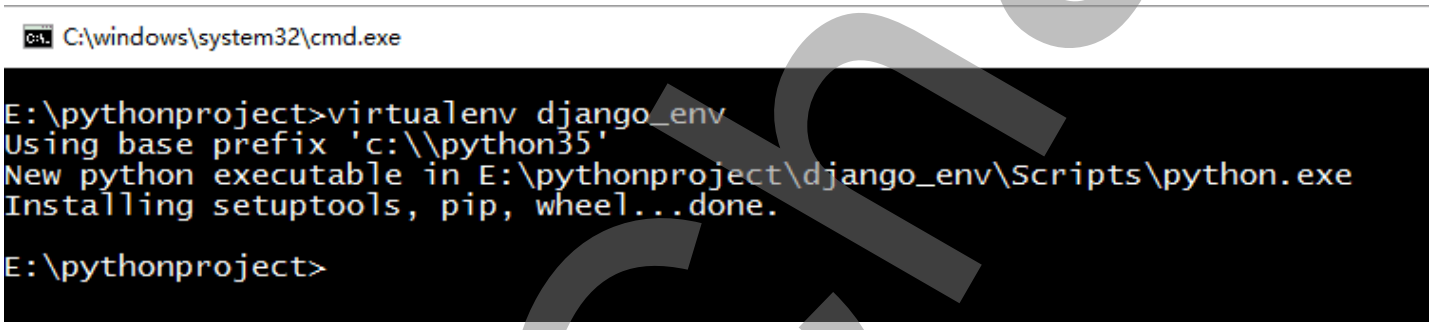
创建一个 Virtualenv 虚拟环境

为了在一个干净的 Python 环境中进行 Django 项目开发，我们使用 Virtualenv 工具创建一个隔离的虚拟 Python 环境作为 Django 项目的运行环境。

Virtualenv 是一个用于创建独立的 Python 环境的工具，用于解决 Python 开发中版本依赖和兼容的问题。

使用 pip install virtualenv 命令安装好 Virtualenv 模块之后，创建一个名为 `django_env` 的虚拟环境，在命令行终端使用 Virtualenv 命令进行创建，代码如下所示：

```
virtual django_env
```



默认情况下，Virtualenv 创建一个只包含了 pip、wheel 和 setuptools 这三个模块安装工具的全新 Python 环境。

在当前目录下，我们可以发现已经多出了一个新的文件夹——`django_env`，点击进去可以查看我们创建的 `django_env` 虚拟环境的结构，如图所示：

在 Windows 平台下，我们可以运行 Scripts 子目录下的 activate 脚本来激活当前的虚拟环境，代码如下所示：

```
Scripts\activate
```

```
E:\pythonproject>cd ./django_env/

E:\pythonproject\django_env>Scripts\activate

(django_env) E:\pythonproject\django_env>
```

当看到命令行中目录前出现了一个括号包含着虚拟环境的名称，说明已经激活了虚拟环境。

如果需要退出当前的虚拟环境，可以使用 Script 子目录下的 deactivate 脚本来实现虚拟环境的退出，代码如下所示：

```
Scripts\deactivate
```

安装 Django

在激活当前的虚拟环境之后，使用 pip 命令在虚拟环境内安装 Django 模块，代码如下所示：

```
pip install django
```

```
(django_env) E:\pythonproject\django_env>pip install django
Collecting django
  Downloading Django-1.11.7-py2.py3-none-any.whl (6.9MB)
    100% |                                        | 7.0MB 19kB/s
Collecting pytz (from django)
  Downloading pytz-2017.3-py2.py3-none-any.whl (511kB)
    100% |                                        | 512kB 53kB/s
Installing collected packages: pytz, django
Successfully installed django-1.11.7 pytz-2017.3

(django_env) E:\pythonproject\django_env>_
```

安装完成之后，进行 Python 的 shell 测试一下是否安装成功：

```
(django_env) E:\pythonproject\django_env>python
Python 3.5.3 (v3.5.3:1880cb95a742, Jan 16 2017, 16:02:32) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import django
>>>
```

如果没有报错，那就说明安装成功了。

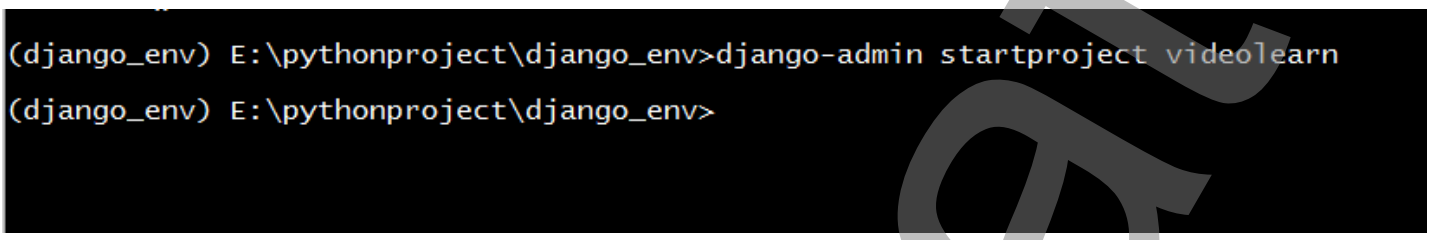
创建一个 Django 项目

在 Django 中，每一个 Web 网站都以项目 project 的形式来呈现，Django Project 通俗的理解，就是一个包含各个子系统的网站容器。

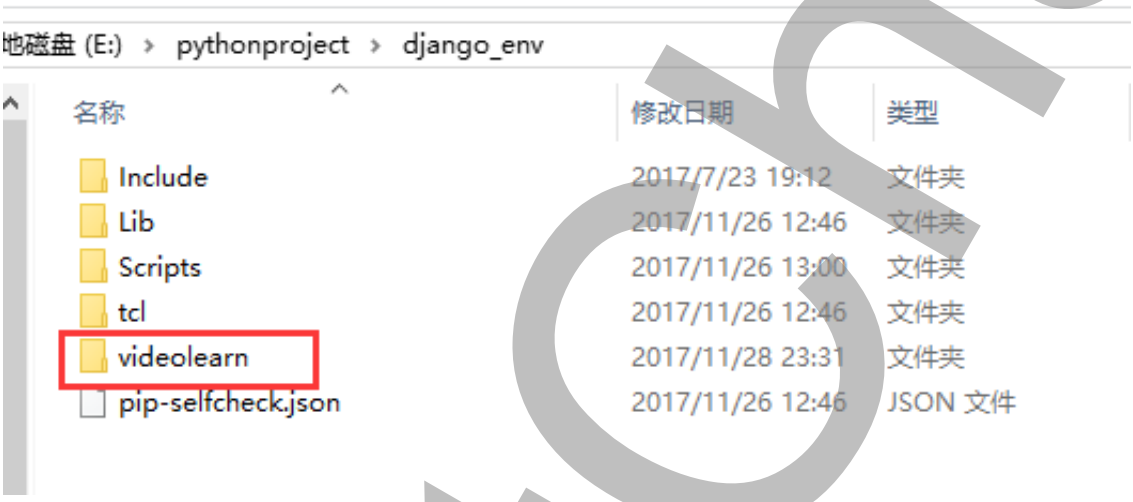
在 Django 安装完成之后，其会生成很多工具脚本，可以使用其中的 Django 管理工具——django-admin 来创建一个项目，使用参数 startproject，后接具体的项目名称，代码如下所示：

```
django-admin startproject videolearn
```

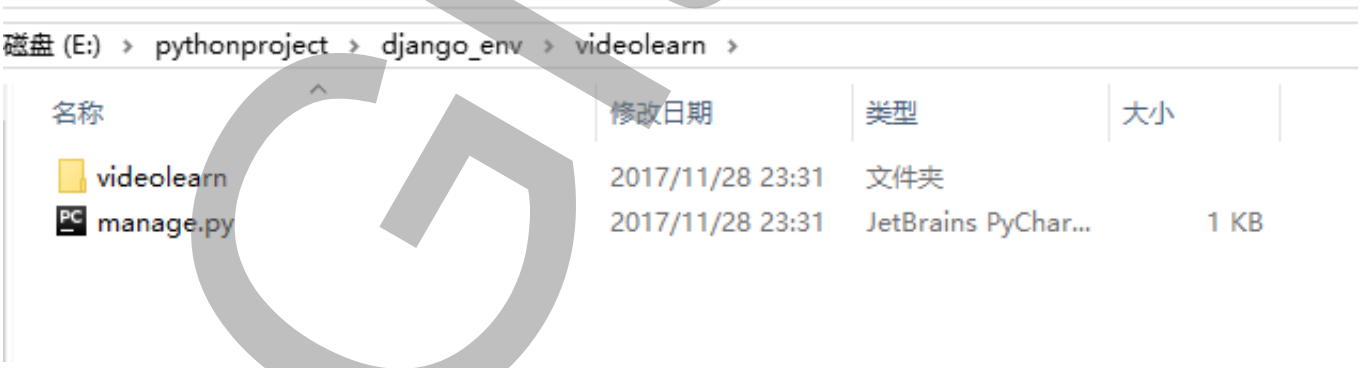
在命令行终端运行上述命令，就在当前目录下创建了一个名为 vidolearn 的 Django 项目，结果如下图所示：



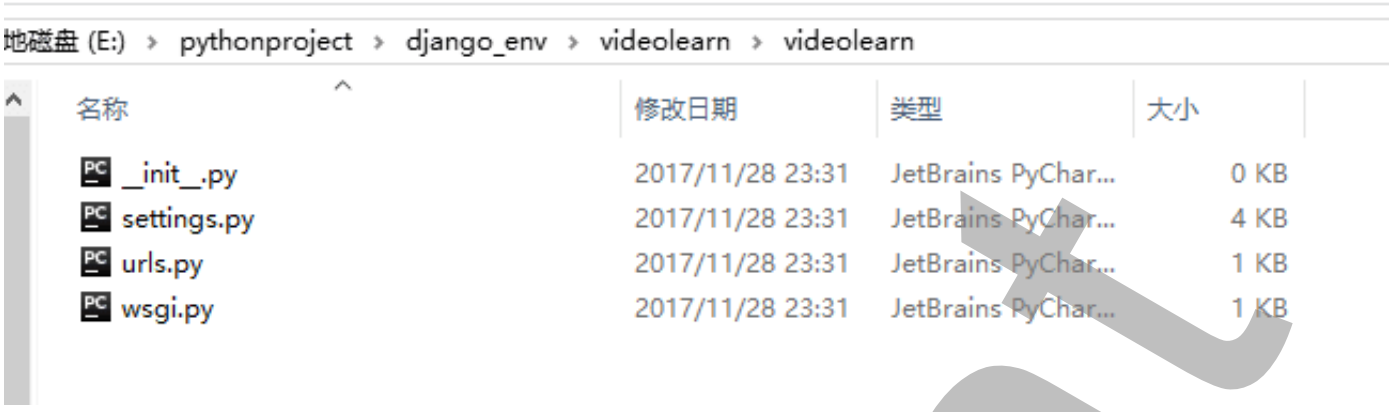
完成上述操作后，可以打开虚拟环境所在的文件夹，看看刚刚创建的 Django 项目，结果如下图所示：



因为我们激活了虚拟环境，所以 Django 项目创建在虚拟环境的目录下，继续打开 Videolearn 文件夹，里面的内容如下图所示：



可以发现，在这个文件夹下有一个项目的主文件夹和一个名为 manage.py 的 Python 文件。



manage.py 文件是以后在 Django 项目开发的过程中经常会使用到的一个管理文件，其提供了 Django 项目的一系列操作，诸如创建应用、创建数据模型、运行服务器等。

在命令行终端运行如下命令：

```
python manage.py -h
```

可以看到 manage.py 工具所支持的各种可用功能：

其中一些常用的参数如下：

- changepassword：用于更改用户密码；
- createsuperuser：用于创建超级用户，后台管理员；
- dbshell：用于进入 Django 模型 shell 中；
- loaddata：用于从文件中加载数据；
- migrate：用于检测数据模型的改变；
- makemigtations：用于创建数据模型改变的迁移；
- startapp：用于创建一个 Django 应用；
- startproject：用于创建一个 Django 项目；
- collectstatic：用于收集汇总静态文件；
- runserver：运行自带调试服务器，默认运行在本地 8000 端口之上。

在 Django 开发过程中，会频繁使用到一些 manage 命令，大家不需要死记硬背，使用得多了自然就熟练了。

继续来看创建的项目文件夹，内层的 viedolearn 目录是创建的 django 项目的包，里面是我们这个 django 项目创建后的初始文件，默认生成了 `__init__.py` 、settings.py、urls.py、wsgi.py这四个文件。

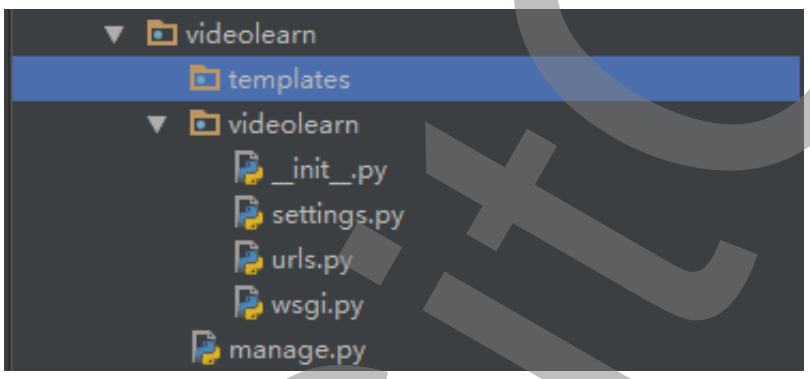
- `\_\_init__.py` 文件用于将这个文件夹变成一个可以被其他 Python 文件引用的模块；
- setting.py 文件是 Django 项目的全局配置文件，数据库的信息、模板的配置、静态文件和资源的配置、站点配置基础配置等都在这里面进行设置；
- urls.py 为 Django 项目的路由映射文件，里面为对 url 路由的处理和转发规则；
- wsgi.py 文件则是 Django 的默认 WSGI 配置，WSGI（Web Server Gateway Interface，Web 服务网关接口），是 Python Web 服务器和应用的标准，我们可以根据项目的需要对 wsgi.py 文件做出调整。

对项目进行基本设置

在了解了创建 Django 项目后生成的文件后来对 Django 项目进行一些基础的设置，以方便我们接下来的项目开发。

设置模板目录

我们先创建一个与 Videolearn 同层级的目录 templates，用来放置以后需要使用到的 HTML 模板文件，这些模板文件用于进行数据的渲染。



然后在 settings.py 文件中，将新创建的 templates 目录设为模板引擎的路径，settings.py 中的源代码如下：

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

将其修改为：

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [
            os.path.join(BASE_DIR,'templates')
        ],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

改动的地方就是将变量 DIRS 的值从空列表，设为了一个包含 templates 目录路径的列表，代码如下所示：

```
'DIRS': [
            os.path.join(BASE_DIR,'templates')
        ],
```

修改中文语言和时区

Django 的默认显示语言为英语，并且时区为标准格林威治时间，为了方便查看，我们将其显示语言修改为中文，显示时间的时区修改为上海时区。

settings.py 文件中的源代码如下所示：

```
LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'
```

将其修改为如下所示：

```
LANGUAGE_CODE = 'zh-hans'

TIME_ZONE = 'Asia/Shanghai'
```

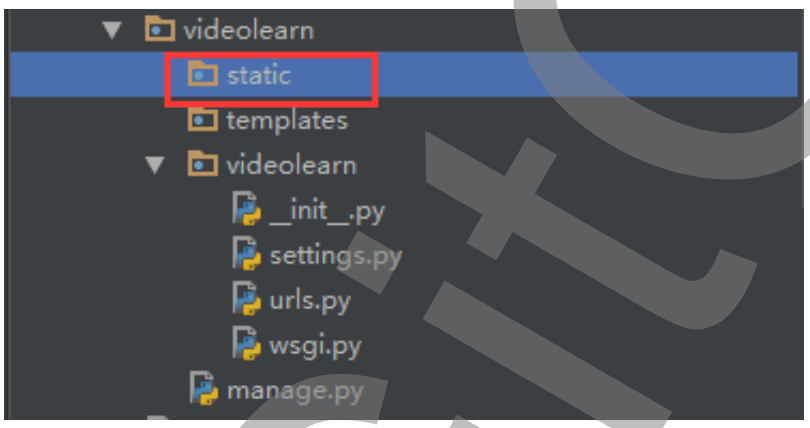
设置静态文件路径

在 Django 中，使用静态文件（JS、CSS、图片等）需要首先声明静态文件的路径，以便于 Django 从声明的路径中寻找到静态文件。

settings.py 文件中已经默认生成了一个静态文件的 URL，代码如下所示：

```
STATIC_URL = '/static/'
```

继续在项目的内层 Videolearn 文件夹同层级创建一个 static 文件夹，用于存放静态资源文件：



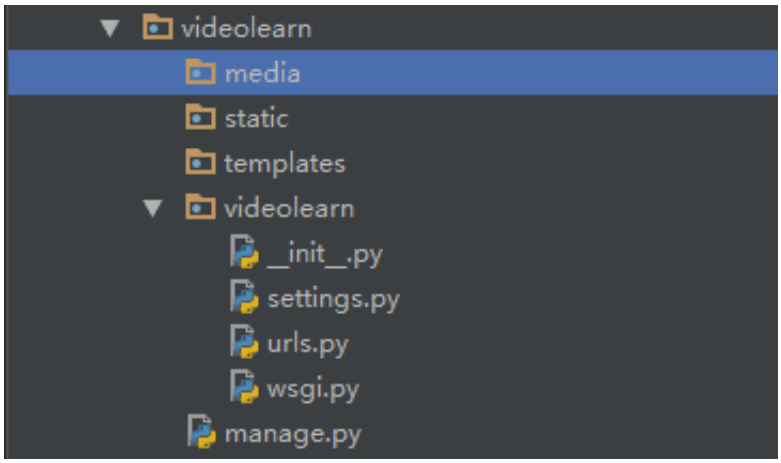
然后在 settings.py 文件中指定这个静态资源文件的路径，代码如下所示：

```
STATIC_URL = '/static/'
STATIC_ROOT = os.path.join(BASE_DIR,'static')
```

设置媒体文件路径

在我们的网站中可能会需要上传一些图片或是文件，这就需要在 settings.py 文件中声明媒体文件的路径。

我们同样在项目的内层 Videolearn 文件夹同层级创建一个 media 文件夹，用于存放媒体资源文件：



然后在 settings.py 文件中设置变量 `MEDIA_URL` 和 `MEDIA_ROOT` ，分别指定媒体文件的 URL 路由地址和媒体文件的本地路径，代码如下所示：

```
MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR,'media')
```

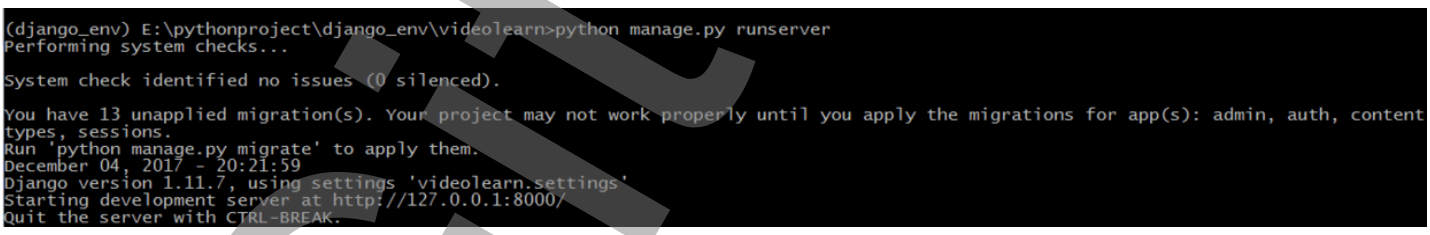
测试运行 Django 项目

在完成上述基本设置后，我们使用 Django 内置的测试服务器来运行一下 Django 项目，看看是否正常。

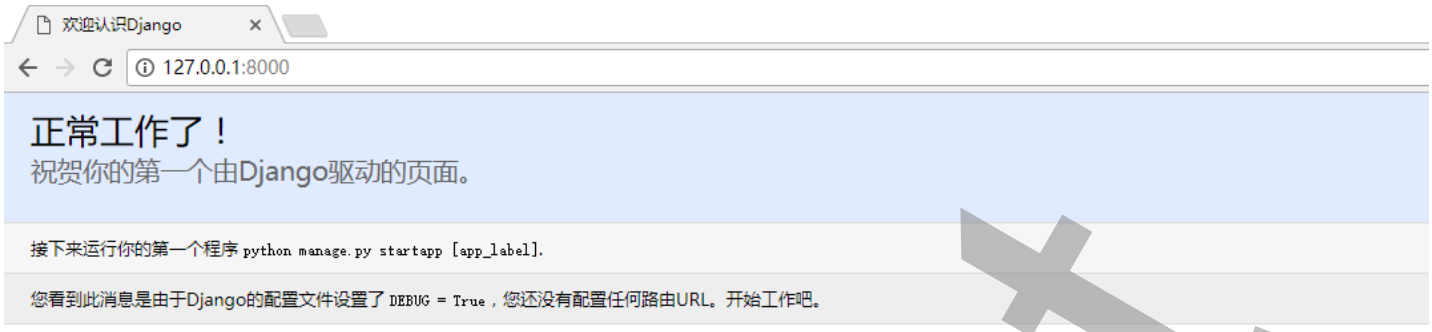
在命令行终端中输入命令，命令如下所示：

```
python manage.py runserver
```

Django 项目默认会运行在本地的 8000 端口上，当出现下图的提示，说明测试服务器启动成功：



在浏览器中访问命令行终端中显示的地址：http://127.0.0.1:8000/：



出现了如上图所示的页面，说明 Django 项目创建并测试运行成功了。

为 Django 项目创建一个 App

刚刚创建了一个 Django 项目并进行了基本的设置，同时成功地将创建的项目使用 manage 工具的 runserver 命令运行在本地的 8000 端口上。

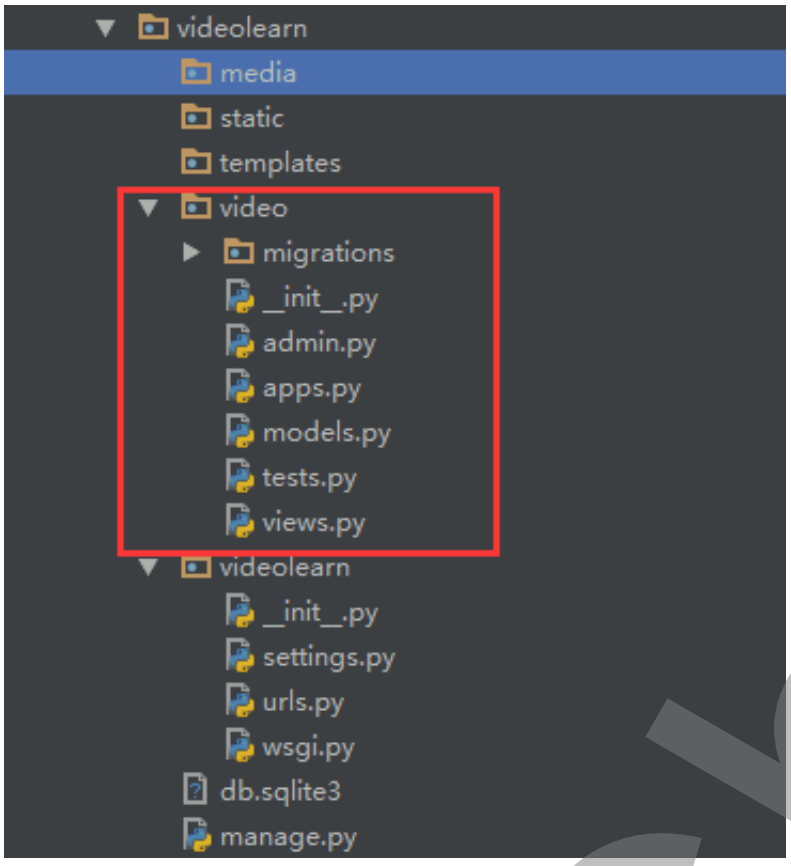
下面接着介绍 Django 的应用（App），在 Django 中，项目（project）是 Django 网站的大框架和容器，那么 App（应用）又是什么呢？

应用（App）是一个具体的 Web 应用程序，用来实现具体的功能和完成具体的事项，被 Django 项目（project）所包含。一个 project 里面可以有很多个 App，就像一个网站可以有文章系统、论坛系统、后台系统等，每一个 App 也可以剥离出来作为独立的模块组件。

还记得上一篇介绍过的 manage.py 文件和它的一些常用功能命令吗？利用 manage.py 这个命令行小工具，通过 startapp 命令，可以很方便快速地创建一个 django app，代码如下所示：

```
python manage.py startapp video
```

在命令行终端运行上述命令，就创建了一个名为 video 的 Django App，我们看看文件夹中的新变化：



Django 项目 viedolearn 目录中多出了一个 video 文件夹，里面有6个文件，其中：

- admin.py：用于设置 Django 自带的强大管理后台；
- apps.py：声明了这个的 App 信息；
- models.py：用于定义数据模型（数据库表）；
- tests.py：用于单元测试；
- views.py：用于定义 App 的视图，也就是业务函数。

这包含了一个 Web 应用的后台管理、数据库定义、逻辑视图，在结合项目的路由映射，就是一个 Django App 最基本的结构。

migrations 文件夹目前是一个空的文件夹，其中会记录应用的数据模型迁移的情况。

创建完 video 应用之后，我们需要使用 manage.py 文件的 migrate 和 makemigrations 命令，创建一些默认的数据库表，命令如下所示：

```
python manage.py migrate
```

运行命令，会显示将会对数据库进行很多个表的实例化和迁移，细看一下，基本上是属于认证和后台管理的数据库名。

```
(django_env) E:\pythonproject\django_env\videolearn>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  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 sessions.0001_initial... OK
```

接着，可以使用 makemigrations 命令来检测项目中数据模型的数据迁移变化，命令如下所示：

```
python manage.py makemigrations
```

直接输入这个命令，终端可能会提示没有任何改变完成（No changes detected），结果如下图所示：

```
(django_env) E:\pythonproject\django_env\videolearn>python manage.py makemigrations
No changes detected
```

面对这种情况，可以在命令后显式地指定具体的应用名，命令如下所示：

```
python manage.py makemigrations video
```

但是可能依然不会成功，命令行中提示“video”App 并没有被发现，因为刚刚创建的 App——video 并没有添加到 videolearn 项目的 App 列表中：

```
(django_env) E:\pythonproject\django_env\videolearn>python manage.py makemigrations video
App 'video' could not be found. Is it in INSTALLED_APPS?
```

怎么添加呢？添加方法还是在 settings.py 文件中，找到 `INSTALLED_APPS` 变量，源代码如下所示：

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

我们将 video 以一个字符串的形式添加到列表最后，代码如下所示：

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

添加完成后，再尝试使用 makemigrations 和 migrate 命令，如下所示：

```
python manage.py makemigrations
python manage.py migrate
```

结果就会顺利的完成一些数据模型的创建和更改。

创建超级用户

接下来为我们的项目创建一个超级管理员，通过这个超级管理员账户，可以快速地使用 Django 强大的后台功能，对数据模型进行管理，同样使用 manage.py 这个工具，利用它的 createsuperuser 命令来进行创建，命令如下所示：

```
python manage.py createsuperuser
```

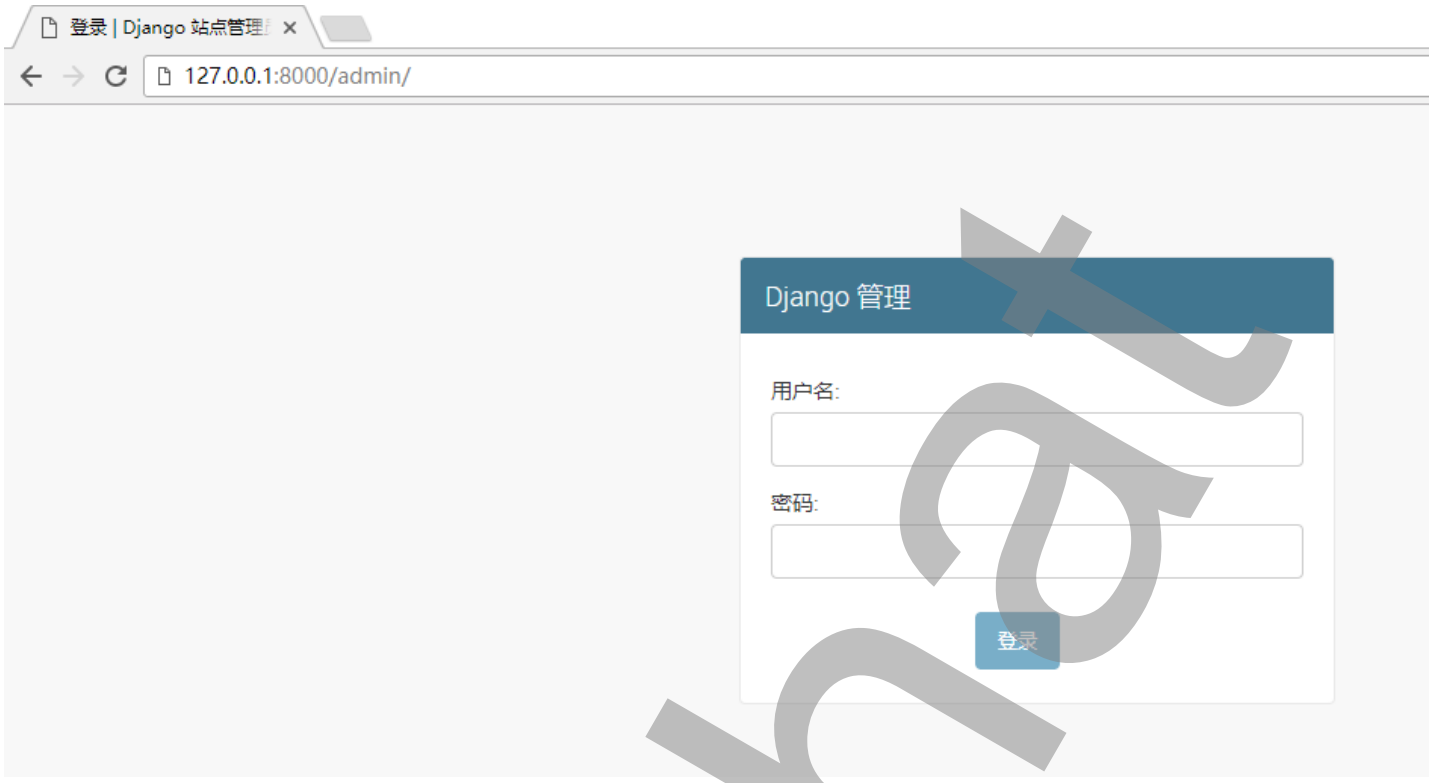
```
(django_env) E:\pythonproject\django_env\videolearn>python manage.py createsuperuser
Username (leave blank to use '[illegible]'): videolearn
Email address: zmister@qq.com
Password:
Password (again):
Superuser created successfully.
```

根据提示输入用户名、邮箱和密码，一个超级用户就创建好了。我们可以运行项目，访问 /admin 路径看看：

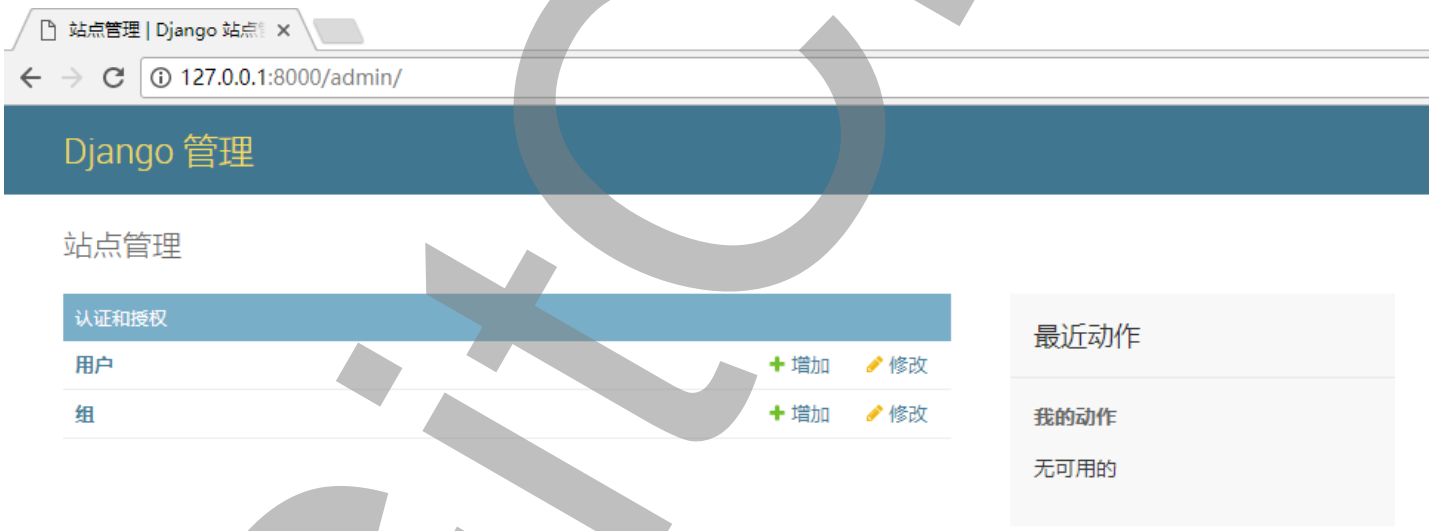
```
(django_env) E:\pythonproject\django_env\videolearn>python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).
December 04, 2017 - 21:02:50
Django version 1.11.7, using settings 'videolearn.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

结果呈现了一个“Django 管理”的登录表单，这就是 Django 自带的后台管理模块：



使用刚刚创建超级用户的用户名和密码进行登录：



进入了传说中功能强大的 Django 自带后台，但是我们并没有添加数据模型进去，所以里面只有之前通过 magrite 命令对应用生成初始化的认证模型（用户和用户组），略显简陋。

你肯定会好奇，为什么打开/admin，就会呈现出来 Django 的管理后台呢?

为了解决这个疑问，我们打开内层 videolearn 文件夹下的 urls.py 文件，里面的内容如下所示：

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

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

在 urls.py 文件中，其引入了两个模块，url 和 admin，然后定义了一个列表 urlpatterns。url 模块用于匹配解析 url 路径，admin 模块就是 Django 的后台模块，名为 urlpatterns 的列表中的内容则是引导 URL 链接——/admin 显示 Django 后台界面的关键。

其中具体的运行机制，在下一篇 Django 的路由和视图中将详细介绍。