[Top](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/DEVWEB/DAY03/CASE/01/index.html" \l "page_top_case)

# NSD Devweb DAY03

1. [案例1：安装django](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/DEVWEB/DAY03/CASE/01/index.html" \l "case1)
2. [案例2：创建项目](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/DEVWEB/DAY03/CASE/01/index.html" \l "case2)
3. [案例3：创建应用](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/DEVWEB/DAY03/CASE/01/index.html" \l "case3)
4. [案例4：创建模型](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/DEVWEB/DAY03/CASE/01/index.html" \l "case4)
5. [案例5：生成数据库](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/DEVWEB/DAY03/CASE/01/index.html" \l "case5)
6. [案例6：注册后台](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/DEVWEB/DAY03/CASE/01/index.html" \l "case6)

## 1 案例1：安装django

### 1.1 问题

1. 创建python虚拟环境
2. 激活python虚拟环境
3. 在虚拟环境中安装django
4. 验证django是否安装正确

### 1.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：创建python虚拟环境

虚拟环境是python多版本管理的利器，可以搭建独立的python运行环境，有助于包的管理和防止版本冲突。可以理解为在自己的电脑上安装了一个虚拟电脑。

1)打开终端，创建pyproject目录，使用cd命令，切换到需要创建虚拟环境的目录：

1. [root@localhost ~]# mkdir pyproject
2. [root@localhost ~]# cd pyproject/

2)使用如下命令，在当前目录创建虚拟环境：

1. [root@localhost pyproject]# python3 -m venv django\_env

步骤二：激活python虚拟环境

1. [root@localhost pyproject]# source django\_env/bin/activate

步骤三：在虚拟环境中安装django

1. (django\_env) [root@localhost pyproject]# pip install django==1.11.6

步骤四：验证django是否安装正确

在python解释器输入以下命令检查是否安装成功，如果输出了django的版本号说明安装成功。

1. (django\_env) [root@localhost pyproject]# python
2. Python 3.6.4 (default, Apr 27 2018, 08:26:23)
3. [GCC 4.8.5 20150623 (Red Hat 4.8.5-16)] on linux
4. Type "help", "copyright", "credits" or "license" for more information.
5. >>> import django
6. >>> django.\_\_version\_\_
7. '1.11.6'

## 2 案例2：创建项目

### 2.1 问题

1. 创建名为mysite的项目
2. 生成数据库
3. 创建超级用户
4. 登录后台管理页面

### 2.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：创建名为mysite的项目

1)安装django之后，您现在应该已经有了可用的管理工具django-admin.py。我们可以使用django-admin命令来创建mysite项目：

1. (django\_env) [root@localhost pyproject]# django-admin startproject mysite

2)创建完成后我们可以查看下项目的目录结构：

1. (django\_env) [root@localhost pyproject]# cd mysite
2. (django\_env) [root@localhost mysite]# tree
3. .
4. ├── manage.py
5. └── mysite
6. ├── \_\_init\_\_.py
7. ├── settings.py
8. ├── urls.py
9. └── wsgi.py

目录说明：

mysite：项目的容器。

manage.py：一个使用的命令行工具，可让你以各种方式与该django项目进行交互

mysite/\_\_init\_\_.py：一个空文件，告诉python该目录是一个python包。

mysite/settings.py：该django项目的设置/配置。

mysite/urls.py：该django项目的URL声明；一份由django驱动的网站“目录”。

mysite/wsgi.py：一个WSGI兼容的Web服务器的入口，以便运行你的项目。

3)接下来我们进入mysite目录输入以下命令，启动服务器：

1. (django\_env) [root@localhost mysite]# python manage.py runserver 0.0.0.0:8000

0.0.0.0让其它电脑可连接到开发服务器，8000为端口号，如果不说明，那么端口号默认为8000

注意：django默认使用sqlite3文件数据，如果不能使用，需要安装sqlite-devel：

1. (django\_env) [root@localhost mysite]# yum install sqlite-devel
2. ================================================================================
3. 安装 1 软件包
4. 总下载量：104k
5. 安装大小：366k
6. Is this ok [y/d/N]: y
7. Downloading packages:
8. Running transaction check
9. Running transaction test
10. Transaction test succeeded
11. Running transaction
12. 正在安装 : sqlite-devel-3.7.17-8.el7.x86\_64 1/1
13. 验证中 : sqlite-devel-3.7.17-8.el7.x86\_64 1/1
14. 已安装:
15. sqlite-devel.x86\_64 0:3.7.17-8.el7
16. 完毕！

然后重新编译python3，找到python源码包，里面有一个configure可执行文件，过程如下：

1. [root@localhost ~]# cd /opt/Python-3.6.1/
2. [root@localhost Python-3.6.1]# ls
3. aclocal.m4 Include Modules Python
4. build install-sh Objects python-config
5. config.guess Lib Parser python-config.py
6. config.log libpython3.6m.a PC python-gdb.py
7. config.status LICENSE PCbuild README.rst
8. config.sub Mac Programs setup.py
9. configure Makefile pybuilddir.txt Tools
10. configure.ac Makefile.pre pyconfig.h
11. Doc Makefile.pre.in pyconfig.h.in
12. Grammar Misc python
13. [root@localhost Python-3.6.1]# ./configure --prefix=/usr/local/bin    #配置编译
14. [root@localhost Python-3.6.1]# make        #编译源码
15. [root@localhost Python-3.6.1]# make install        #执行安装

4)在浏览器输入你服务器的ip及端口号，如果正常启动，输出信息如图-1所示：

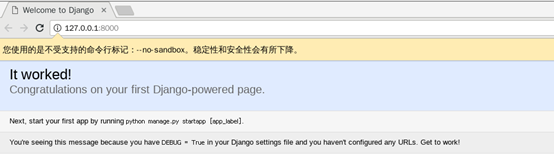


图-1

5)在浏览器输入http://127.0.0.1:8000/admin可访问后台管理界面，输出信息如图-2所示：

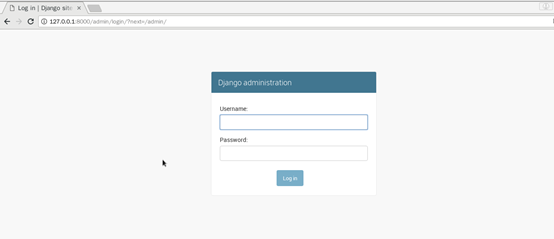


图-2

步骤二：生成数据库：

1)将model层转为迁移文件migration：

1. (django\_env) [root@localhost mysite]# python manage.py makemigrations
2. No changes detected

2)将新版本的迁移文件执行，更新数据库：

1. (django\_env) [root@localhost mysite]# python manage.py migrate
2. Operations to perform:
3. Apply all migrations: admin, auth, contenttypes, sessions
4. Running migrations:
5. Applying contenttypes.0001\_initial... OK
6. Applying auth.0001\_initial... OK
7. Applying admin.0001\_initial... OK
8. Applying admin.0002\_logentry\_remove\_auto\_add... OK
9. Applying contenttypes.0002\_remove\_content\_type\_name... OK
10. Applying auth.0002\_alter\_permission\_name\_max\_length... OK
11. Applying auth.0003\_alter\_user\_email\_max\_length... OK
12. Applying auth.0004\_alter\_user\_username\_opts... OK
13. Applying auth.0005\_alter\_user\_last\_login\_null... OK
14. Applying auth.0006\_require\_contenttypes\_0002... OK
15. Applying auth.0007\_alter\_validators\_add\_error\_messages... OK
16. Applying auth.0008\_alter\_user\_username\_max\_length... OK
17. Applying sessions.0001\_initial... OK

注意：这两种命令调用默认为全局，即对所欲最新更改的model或迁移文件进行操作，如果想对部分app进行操作，就要在其后追加app name。

3)此时项目中会出现一个db.sqlite3文件，这就是默认的数据库文件。输出信息如图-3所示：

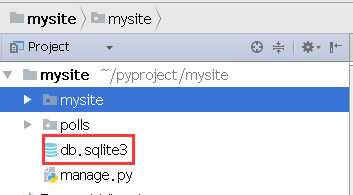


图-3

步骤三：创建超级用户：

输入如下命令，创建后台管理员账号：

1. (django\_env) [root@localhost mysite]# python manage.py createsuperuser
2. Username (leave blank to use 'root'): admin
3. Email address: zzg@tedu.cn
4. Password:
5. Password (again):
6. Superuser created successfully.

步骤四：登录后台管理页面

启动django后，可以登录后台管理页面，登录成功输出信息如图-4所示：



图-4

## 3 案例3：创建应用

### 3.1 问题

1. 创建应用polls
2. 注册polls到项目中
3. 配置URLconf，当访问http://127.0.0.1:8000/polls时，由polls进行路由
4. 为polls应用配置主页视图

### 3.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：创建应用

在每个django项目中可以包含多个app，相当于一个大型项目中的分系统、子模块、功能部件等，相互之间比较独立，但也有联系，所有的app共享项目资源，执行如下命令，用startapp命令创建app：

1. (django\_env) [root@localhost mysite]# python manage.py startapp polls

此时项目目录结构如图-5所示：

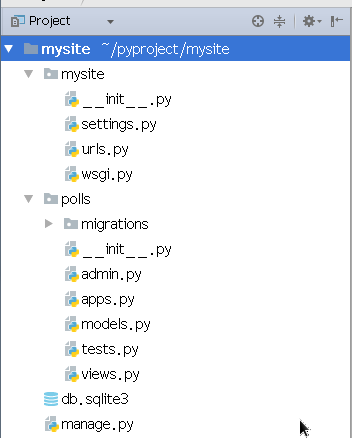


图-5

步骤二：激活应用

为了让django包含创建的app，我们还需要激活app。打开配置文件setting.py，找到INSTALLED\_APPS,然后把我们创建的app配置添加进去，这样django才能使用我们的app。

1. INSTALLED\_APPS = [
2. 'django.contrib.admin',
3. 'django.contrib.auth',
4. 'django.contrib.contenttypes',
5. 'django.contrib.sessions',
6. 'django.contrib.messages',
7. 'django.contrib.staticfiles',
8. 'polls',
9. ]

步骤三：配置URLconf，当访问http://127.0.0.1:8000/polls时，由polls进行路由

1)配置URLconf。在mysite/urls.py文件中添加app中设置的路径，除了admin页面的路径之外，其他路径都应该使用include函数引入

1. from django.conf.urls import url, include
2. from django.contrib import admin
3. urlpatterns = [
4. url(r'^admin/', admin.site.urls),
5. url(r'^polls/', include('polls.urls'))
6. ]

2)创建polls应用的URLconf，即创建模块urls.py

1. (django\_env) [root@localhost mysite]#touch polls/urls.py

3)绑定URL与视图函数，在polls/urls.py中，删除原代码，将一下代码写入urls.py文件中：

1. from django.conf.urls import url
2. from . import views
3. urlpatterns = [
4. url(r'^$', views.index, name='index'),
5. ]

步骤四：为polls应用配置主页视图

1)在app的views.py中添加一个新的视图。创建views.index函数

1. from django.http import HttpResponse
2. def index(request):
3. return HttpResponse(“Hi! 这是polls应用的首页。")

通过上面的步骤，我们将index这个url指向了views里的index（）函数，它接收用户请求，并返回一个“Hi! 这是polls应用的首页。”字符串。

2)验证应用

启动django服务，访问http://127.0.0.1:8000/polls，显示如图-6所示：



图-6

## 4 案例4：创建模型

### 4.1 问题

1. 为polls应用创建模型
2. 一个模型名为Question，用于记录问题
3. 另一个模型名为Choice，每个Choice只能对应一个Question，但是一个Question可以对应多个Choice

### 4.2 方案

Django对各种数据库提供了很好的支持，包括：PostgreSQL、MySQL、SQLite、Oracle。Django为这些数据库提供了统一的调用API。我们可以根据自己业务需求选择不同的数据库。以下我们使用sqlite3默认数据库。

Django支持ORM模型，也就是说我们可以不使用SQL语句对数据进行增删改查。我们要做的就是在模型中指定和数据库的关系。

在这个简单的投票应用中，我们将创建两个模型： Question和Choice

Question对象具有一个question\_text（问题）属性和一个publish\_date（发布时间）属性，用于记录问题

Choice有两个字段：选择的内容和选择的得票统计

每个Choice与一个Question关联

### 4.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：打开配置文件settings.py，找到数据库一行，可以看到如下的配置，目前我们使用默认数据库，暂时不需要对如下配置进行修改。

1. DATABASES = {
2. 'default': {
3. 'ENGINE': 'django.db.backends.sqlite3',
4. 'NAME': os.path.join(BASE\_DIR, 'db.sqlite3'),
5. }
6. }

步骤二：创建模型：

打开polls应用中的models.py文件，然后添加下面两个模型，在polls应用中的models.py文件添加两个模型，每个模型都用一个类表示，该类继承自django.db.models.Model，我们在定义模型的时候指定每一个字段的名字、长度、是否唯一等信息：

1. from django.db import models
2. class Question(models.Model):
3. question\_text = models.CharField(max\_length=200)
4. pub\_date = models.DateTimeField('date published')
5. class Choice(models.Model):
6. question = models.ForeignKey(Question, on\_delete=models.CASCADE)
7. choice\_text = models.CharField(max\_length=200)
8. votes = models.IntegerField(default=0)

值得注意的是，每个字段通过Field类的一个实例表示 ， 例如字符字段CharField、日期字段DateTimeField和整数字段IntegerField。 这种方法告诉Django，每个字段中保存着什么类型的数据。Models.ForeignKey用来指定外键约束。

## 5 案例5：生成数据库

### 5.1 问题

1. 执行迁移命令生成数据库
2. 在pycharm中打开数据库
3. 观察数据库表名和字段名，发现与模型的关系

### 5.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：执行迁移命令生成数据库

1)polls应用建立了新的模型，该模型需要反馈到数据库中，通过运行makemigrations命令告诉django，已经对模型做了一些更改，并且会将这些更改记录为迁移文件，迁移文件位于polls/migrations/目录下：

1. (django\_env) [root@localhost mysite]# python manage.py makemigrations
2. Migrations for 'polls':
3. polls/migrations/0001\_initial.py
4. - Create model Choice
5. - Create model Question
6. - Add field question to choice

迁移文件查看如图-7所示：

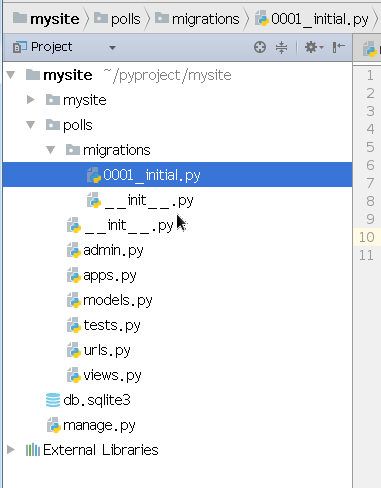


图-7

2)此时只是生成了迁移文件，并没有真正应用到数据库中。如果要引用到数据库，再次使用migrate命令即可：

1. (django\_env) [root@localhost mysite]# python manage.py migrate
2. Operations to perform:
3. Apply all migrations: admin, auth, contenttypes, polls, sessions
4. Running migrations:
5. Applying polls.0001\_initial... OK

步骤二：在pycharm中打开数据库

1)通过pycharm的Database窗口查看数据库，如图-8所示：

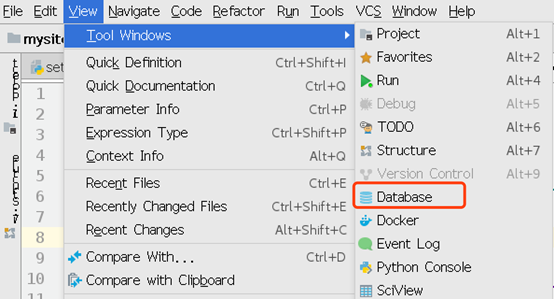


图-8

2)将db.sqlite3文件拖入Database窗口，如图-9所示：

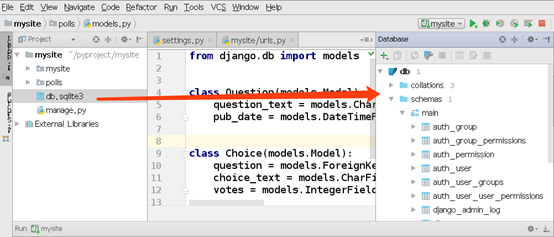


图-9

3)设置Database，如图-10所示：

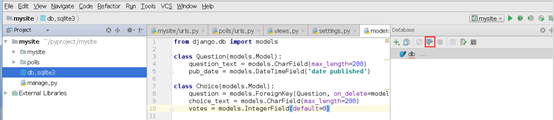


图-10

4)下载丢失的驱动程序文件，如图-11所示：

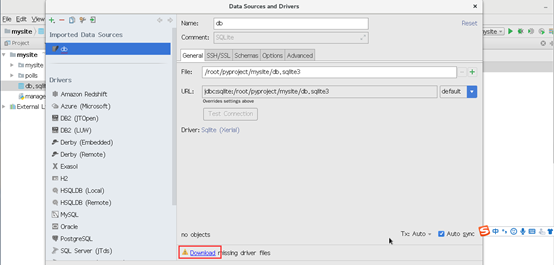


图-11

5)测试连接后，点击/按钮，完成设置，如图-12所示：

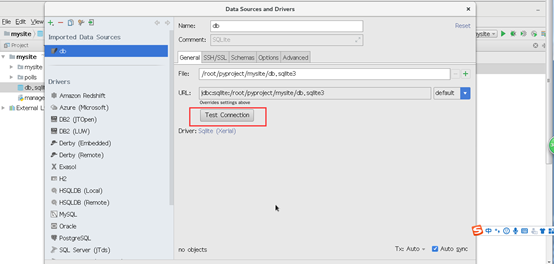


图-12

6)从Database窗口可查前面创建的数据库模型，如图-13所示：

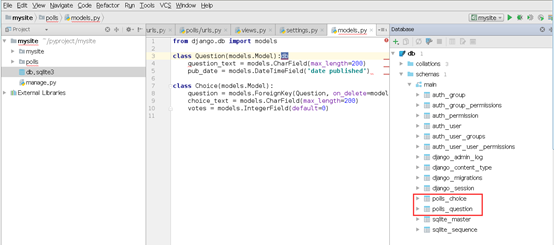


图-13

步骤三：观察数据库表名和字段名

了解与模型的关系，如图-14所示：

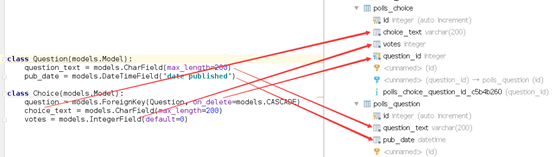


图-14

## 6 案例6：注册后台

### 6.1 问题

1. 将Question和Choice模型添加到后台管理
2. 在后台页中填写几个问题和选项

### 6.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：将Question和Choice模型添加到后台管理

1)启动服务器，在浏览器中访问http://127.0.0.1:8000/admin，进入登录界面，输入创建过的超级用户的用户名密码登录，登录成功后，显示如图-15所示：

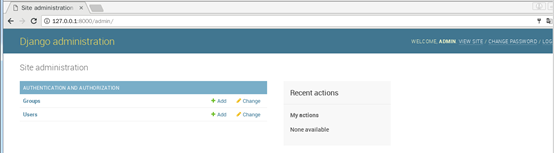


图-15

2)为了让admin界面管理某个数据模型，我们需要先注册该数据模型到admin。比如，我们之前在polls/models.py文件中已经创建了模型Question和Choice，修改polls/admin.py：

1. from django.contrib import admin
2. from .models import Question, Choice
3. admin.site.register(Question)
4. admin.site.register(Choice)

3)刷新后即可看到polls数据表，如图-16所示：

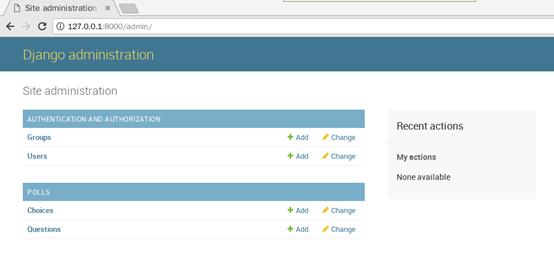


图-16

步骤二：在后台页中填写几个问题和选项

1)添加数据，如图-17所示



图-17

2)添加Question数据，如图-18所示：

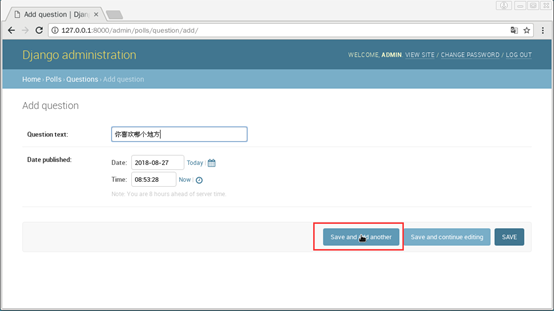


图-18

3)添加Choice数据，如图-19所示：

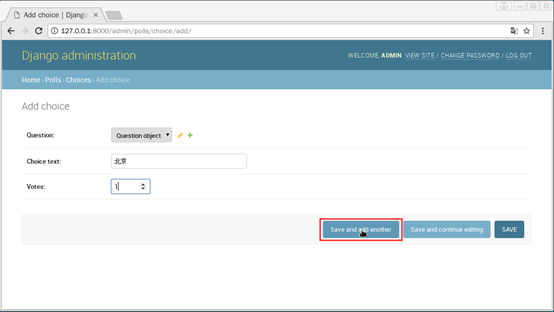


图-19