# 创建虚拟环境

* 安装虚拟环境命令:

pip install virtualenv

pip install virtualenvwrapper

* 创建名为Dproject的虚拟环境

C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01>**virtualenv.exe Dproject**

Using base prefix 'c:\\users\\lucifer\\appdata\\local\\programs\\python\\python36'

New python executable in C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject\Scripts\python.exe

Installing setuptools, pip, wheel...done.

* 进入虚拟环境目录

C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01>**cd Dproject**

C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject>**cd Scripts**

C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject\Scripts>**activate**

* 从命令行前部已经可以看到进入了虚拟环境

**(Dproject)** C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject\Scripts>

* 虚拟环境中并没有Django ，需要安装Django

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject\Scripts>**pip list**

DEPRECATION: The default format will switch to columns in the future. You can use --format=(legacy|columns) (or define a format=(legacy|columns) in your pip.conf under the [list] section) to disable this warning.

pip (9.0.1)

setuptools (28.8.0)

SQLAlchemy (1.1.13)

wheel (0.30.0)

* 在虚拟环境安装Django

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject\Scripts>**pip install Django==1.11.5**

Collecting Django==1.11.5

Using cached Django-1.11.5-py2.py3-none-any.whl

Collecting pytz (from Django==1.11.5)

Using cached pytz-2017.2-py2.py3-none-any.whl

Installing collected packages: pytz, Django

Successfully installed Django-1.11.5 pytz-2017.2

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject\Scripts>python

Python 3.6.0 (v3.6.0:41df79263a11, Dec 23 2016, 08:06:12) [MSC v.1900 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>> import django

>>> django.VERSION

(1, 11, 5, 'final', 0)

# 创建Django项目

* 以下内容是在windows下的操作，如在linux下应该更简单一些

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject>**python Scripts/django-admin.py startproject Dproject**

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject>**cd Dproject**

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject\Dproject>**tree /F**

卷 Windows8\_OS 的文件夹 PATH 列表

卷序列号为 0000004A 1C6A:8177

C:.

│ manage.py

│

└─Dproject

settings.py

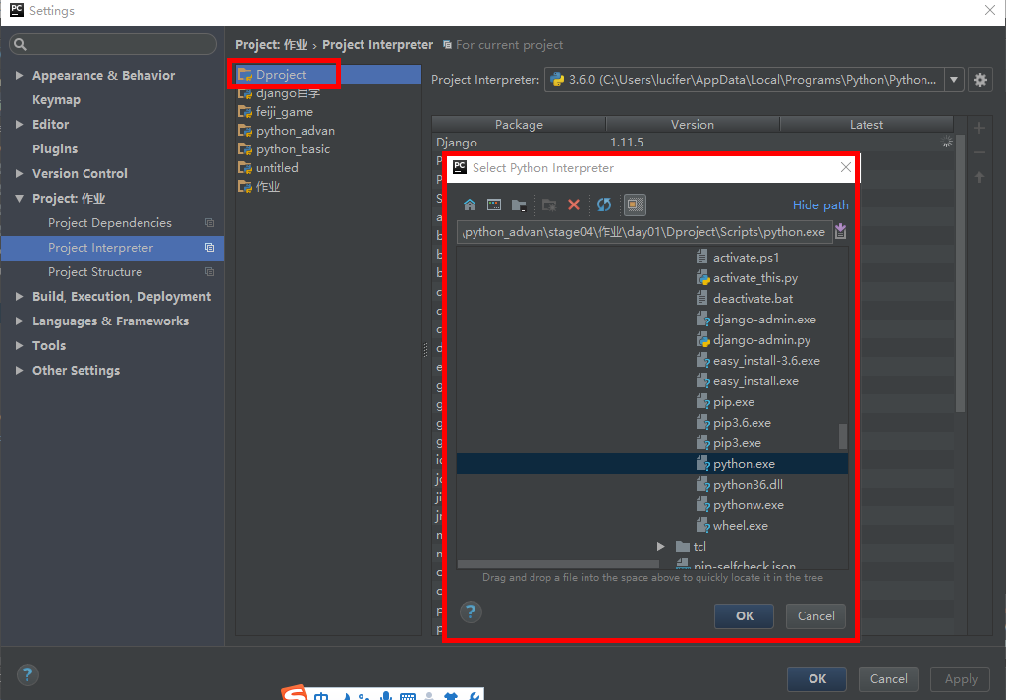
urls.py

wsgi.py

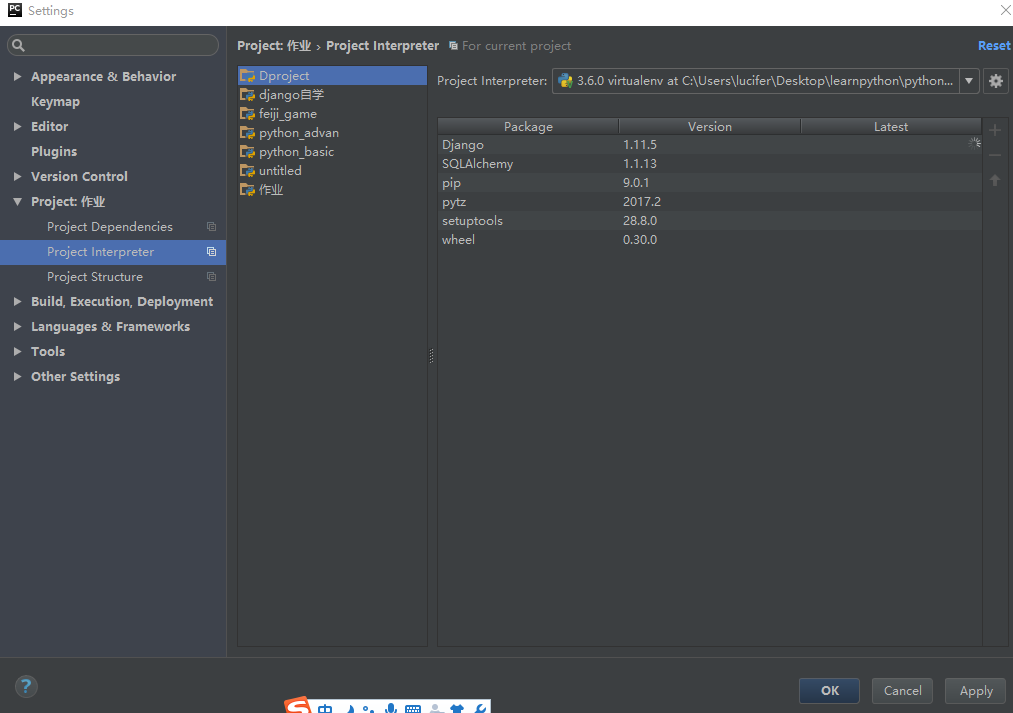
\_\_init\_\_.py

# 在pycharm中打开项目并且调整使用的环境

* 在pycharm的设置中，选中项目和新建的虚拟环境按OK



选择虚拟环境后就可以有多个独立的环境了



# 新建应用

* 新建一个叫 news的新闻应用

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject\Dproject>**python manage.py startapp news**

以下是建立后的目录结构

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject\Dproject>tree /F

卷 Windows8\_OS 的文件夹 PATH 列表

卷序列号为 000000A5 1C6A:8177

C:.

│ manage.py

│

├─.idea

│ │ Dproject.iml

│ │ misc.xml

│ │ modules.xml

│ │ workspace.xml

│ │

│ └─inspectionProfiles

│ profiles\_settings.xml

│

├─Dproject

│ │ settings.py

│ │ urls.py

│ │ wsgi.py

│ │ \_\_init\_\_.py

│ │

│ └─\_\_pycache\_\_

│ settings.cpython-36.pyc

│ \_\_init\_\_.cpython-36.pyc

│

└─**news**

**│ admin.py**

**│ apps.py**

**│ models.py**

**│ tests.py**

**│ views.py**

**│ \_\_init\_\_.py**

**│**

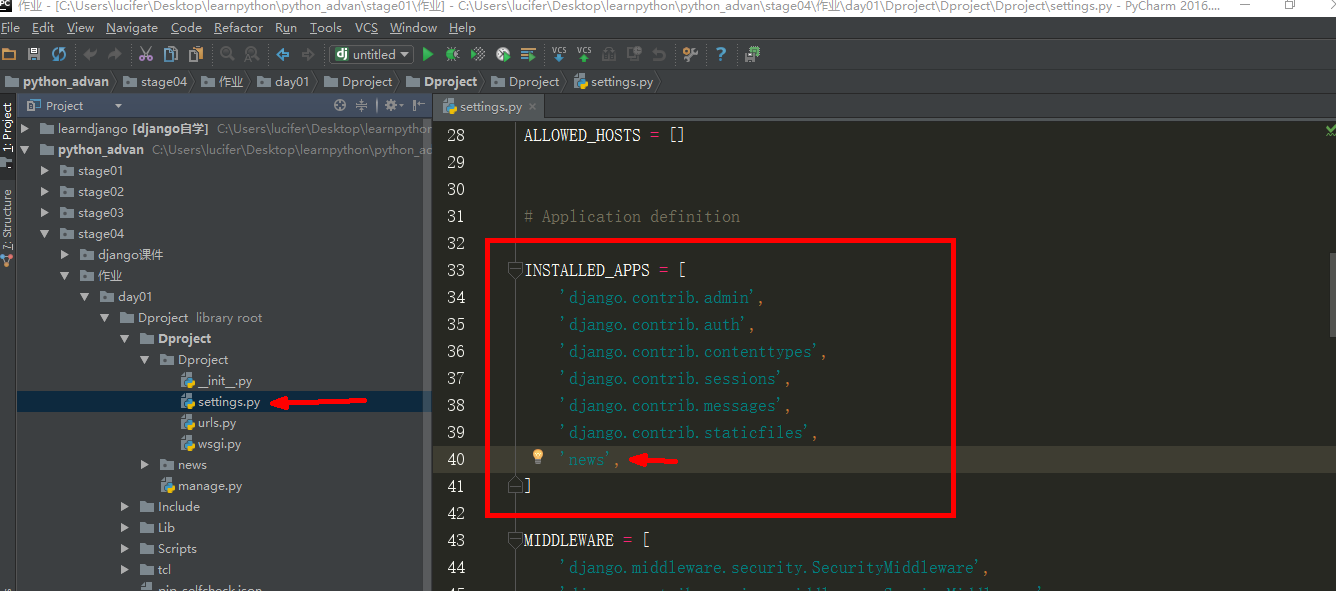
**└─migrations**

**\_\_init\_\_.py**

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject\Dproject>

* 注册应用

在项目的settings.py文件下 注册新建的news应用



* 确认Django服务已经启动，以下红字部分为启动命令 参数可指定IP和端口

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject\Dproject>**python manage.py runserver localhost:8888**

Performing system checks...

System check identified no issues (0 silenced).

You have 13 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.

Run 'python manage.py migrate' to apply them.

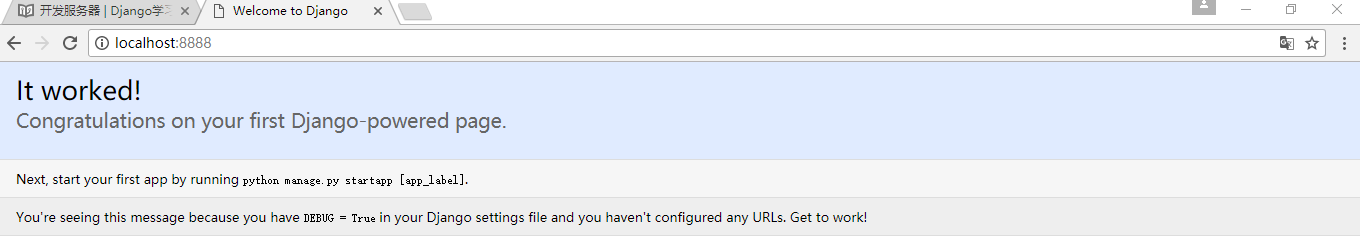
October 06, 2017 - 15:23:01

Django version 1.11.5, using settings 'Dproject.settings'

Starting development server at http://localhost:8888/

Quit the server with CTRL-BREAK.

* 通过浏览器访问确认Django web服务器工作正常



* 按ctrl + c 停止服务进行下一步操作

# 在Django中使用ORM定义模型类

* 在django中进行数据库开发，步骤如下:
* 在models.py模块中定义模型类;
* 根据模型类生成对应SQL, 这一步我们叫做’迁移‘;
* 执行迁移, 这一步我们执行第二步生成的SQL;
* 通过模型类方法完成对数据库的CRUD操作.
* 下面我们创建模型类:

新闻分类表：

表名：NewsCategory

新闻分类：cag\_name

新闻信息表:

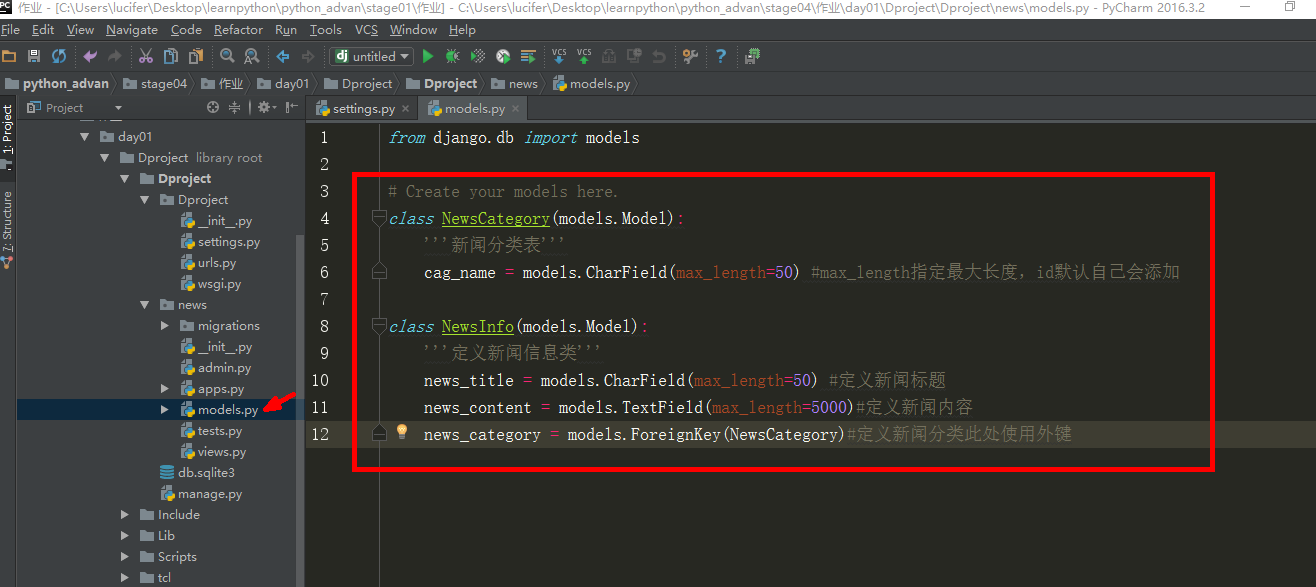
表名：NewsInfo

新闻标题：news\_title

新闻内容：news\_contents

所属分类：news\_cag

* 这里需要注意，新闻分类表中的数据和新闻信息表中的数据关系是一对多关系，也就是一个新闻分类对应多个新闻信息.



* 具体代码如下

*from* django.db *import* models  
  
# Create your models here.  
*class* NewsCategory(models.Model):  
 '''新闻分类表'''  
 cag\_name = models.CharField(max\_length=50) #max\_length指定最大长度，id默认自己会添加  
  
*class* NewsInfo(models.Model):  
 '''定义新闻信息类'''  
 news\_title = models.CharField(max\_length=50) #定义新闻标题  
 news\_content = models.TextField(max\_length=5000)#定义新闻内容  
 news\_category = models.ForeignKey(NewsCategory)#定义新闻分类此处使用外键

* 生成迁移文件

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject\Dproject>**python manage.py makemigrations**

Migrations for 'news':

news\migrations\0001\_initial.py

- Create model NewsCategory

- Create model NewsInfo

* 查看目录可以看到多了0001\_initial.py 文件

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject\Dproject>tree /F

卷 Windows8\_OS 的文件夹 PATH 列表

卷序列号为 00000062 1C6A:8177

C:.

│ db.sqlite3

│ manage.py

│

├─.idea

│ │ Dproject.iml

│ │ misc.xml

│ │ modules.xml

│ │ workspace.xml

│ │

│ └─inspectionProfiles

│ profiles\_settings.xml

│

├─Dproject

│ │ settings.py

│ │ urls.py

│ │ wsgi.py

│ │ \_\_init\_\_.py

│ │

│ └─\_\_pycache\_\_

│ settings.cpython-36.pyc

│ urls.cpython-36.pyc

│ wsgi.cpython-36.pyc

│ \_\_init\_\_.cpython-36.pyc

│

└─news

│ admin.py

│ apps.py

│ models.py

│ tests.py

│ views.py

│ \_\_init\_\_.py

│

├─migrations

│ │ 0001\_initial.py

│ │ \_\_init\_\_.py

│ │

│ └─\_\_pycache\_\_

│ \_\_init\_\_.cpython-36.pyc

│

└─\_\_pycache\_\_

admin.cpython-36.pyc

models.cpython-36.pyc

\_\_init\_\_.cpython-36.pyc

* 查看迁移文件生成的SQL语句 python manage.py sqlmigrate 应用名 文件名

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject\Dproject>**python manage.py sqlmigrate news 0001**

BEGIN;

--

-- Create model NewsCategory

--

CREATE TABLE "news\_newscategory" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "cag\_name" varchar(50) NOT NULL);

--

-- Create model NewsInfo

--

CREATE TABLE "news\_newsinfo" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "news\_title" varchar(50) NOT NULL, "news\_content" text NOT NULL, "news\_category\_id" integer NOT NULL REFERENCES "news\_newscategory" ("id"));

CREATE INDEX "news\_newsinfo\_news\_category\_id\_c18d8327" ON "news\_newsinfo" ("news\_category\_id");

COMMIT;

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject\Dproject>

* 执行迁移

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject\Dproject>**python manage.py migrate**

Operations to perform:

Apply all migrations: admin, auth, contenttypes, news, 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

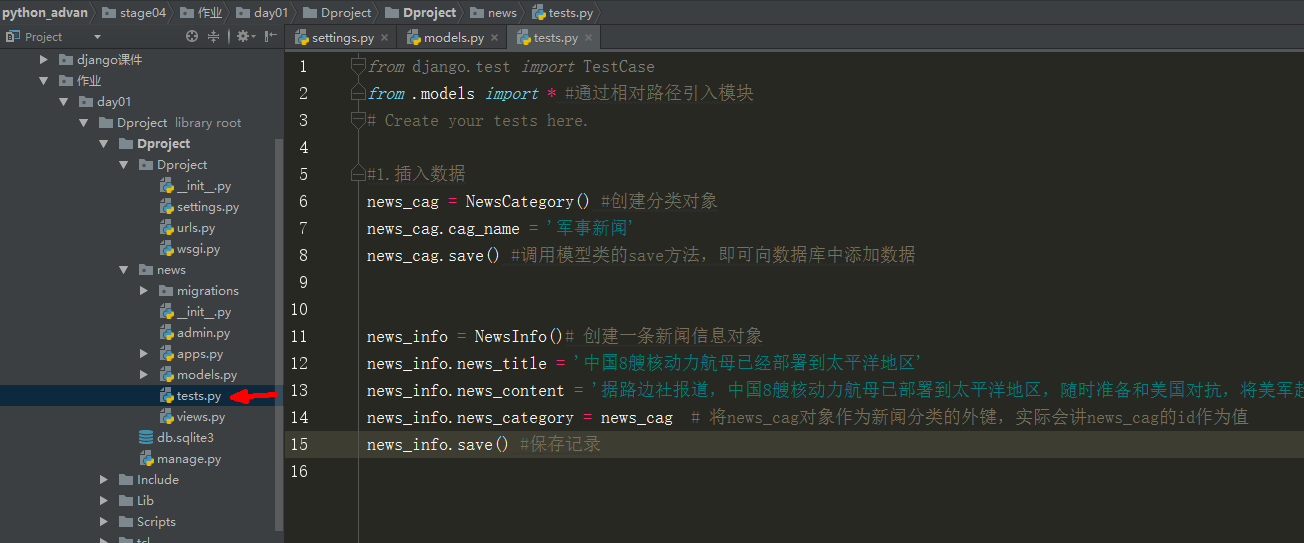
Applying auth.0008\_alter\_user\_username\_max\_length... OK

Applying news.0001\_initial... OK

Applying sessions.0001\_initial... OK

# 测试数据

* 测试数据可以通过2种方法 使用
* python manage.py shell在终端测试，
* 也可以在应用下的 test.py中进行测试
* 以下先在 test.py文件中进行测试。然后再通过shell进行测试



具体代码如下：

*from* django.test *import* TestCase  
*from* .models *import* \* #通过相对路径引入模块  
# Create your tests here.  
  
#1.插入数据  
news\_cag = NewsCategory() #创建分类对象  
news\_cag.cag\_name = '军事新闻'  
news\_cag.save() #调用模型类的save方法，即可向数据库中添加数据  
  
  
news\_info = NewsInfo()# 创建一条新闻信息对象  
news\_info.news\_title = '中国8艘核动力航母已经部署到太平洋地区'  
news\_info.news\_content = '据路边社报道，中国8艘核动力航母已部署到太平洋地区，随时准备和美国对抗，将美军赶出亚太地区'  
news\_info.news\_category = news\_cag # 将news\_cag对象作为新闻分类的外键，实际会讲news\_cag的id作为值  
news\_info.save() #保存记录

* 在命令中执行测试代码命令

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject\Dproject>**python manage.py test news**

Creating test database for alias 'default'...

System check identified no issues (0 silenced).

----------------------------------------------------------------------

Ran 0 tests in 0.000s

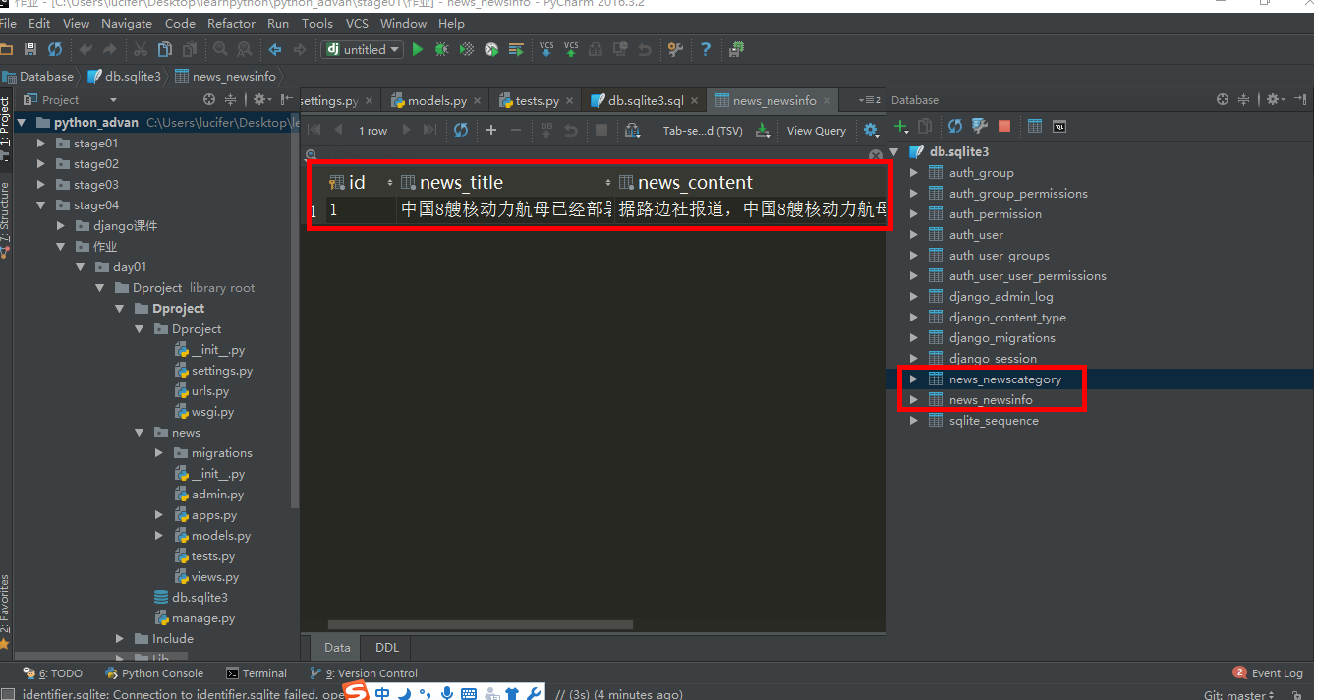
OK

Destroying test database for alias 'default'...

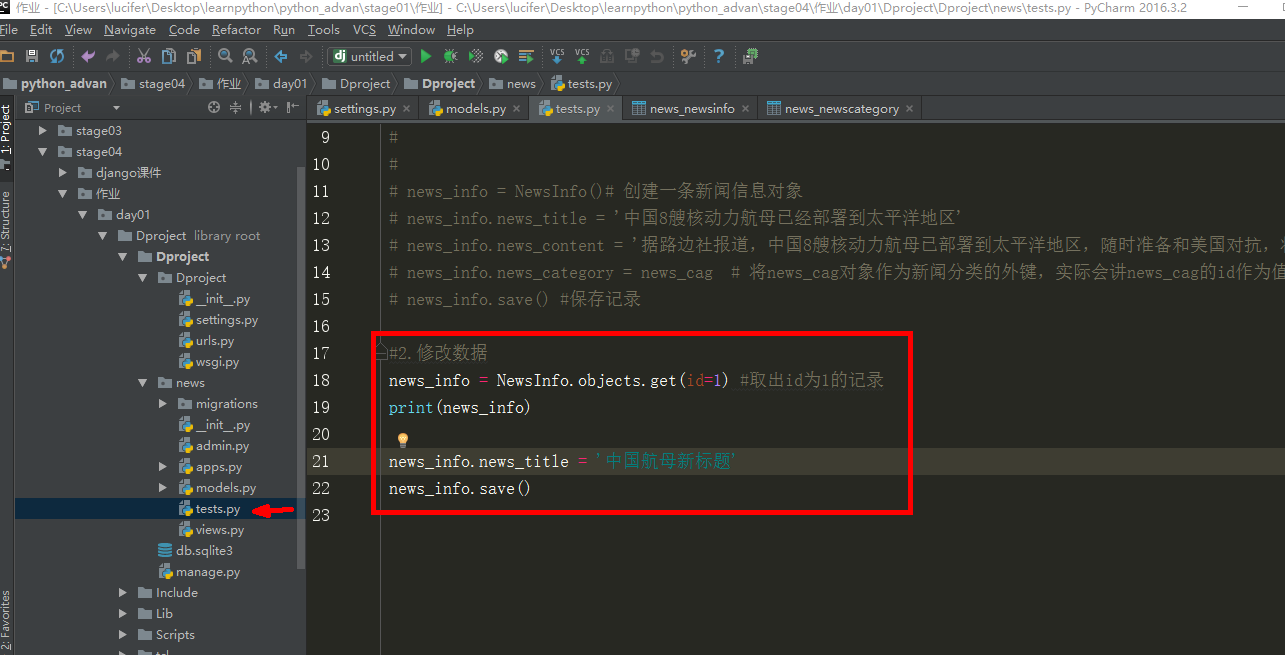
此时数据已插入sqlite数据库，可以通过软件DB Browser for sqlite查看数据

也可以在pycharm集成环境中查看数据，这次试用pycharm在集成环境中查看数据

可以看到2张数据表已经建立吗，并且有数据已经插入。



* 修改数据



* 以下为源代码

#2.修改数据  
news\_info = NewsInfo.objects.get(id=1) #取出id为1的记录  
print(news\_info) #打印该对象，可以定义一个\_\_str\_\_方法改变显示  
  
news\_info.news\_title = '中国航母新标题' #设置新标题  
news\_info.save() #保存修改

* 通过命令行执行修改

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject\Dproject>**python manage.py test news**

**NewsInfo object**

Creating test database for alias 'default'...

System check identified no issues (0 silenced).

----------------------------------------------------------------------

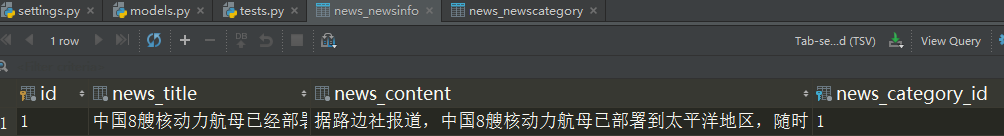
Ran 0 tests in 0.000s

OK

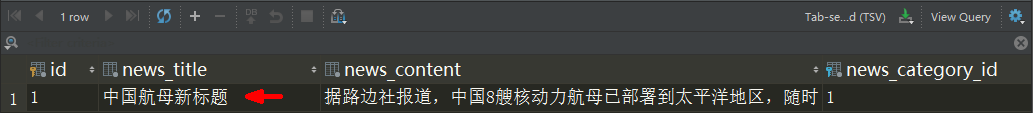
Destroying test database for alias 'default'...

可以看到表的内容改变了

原title



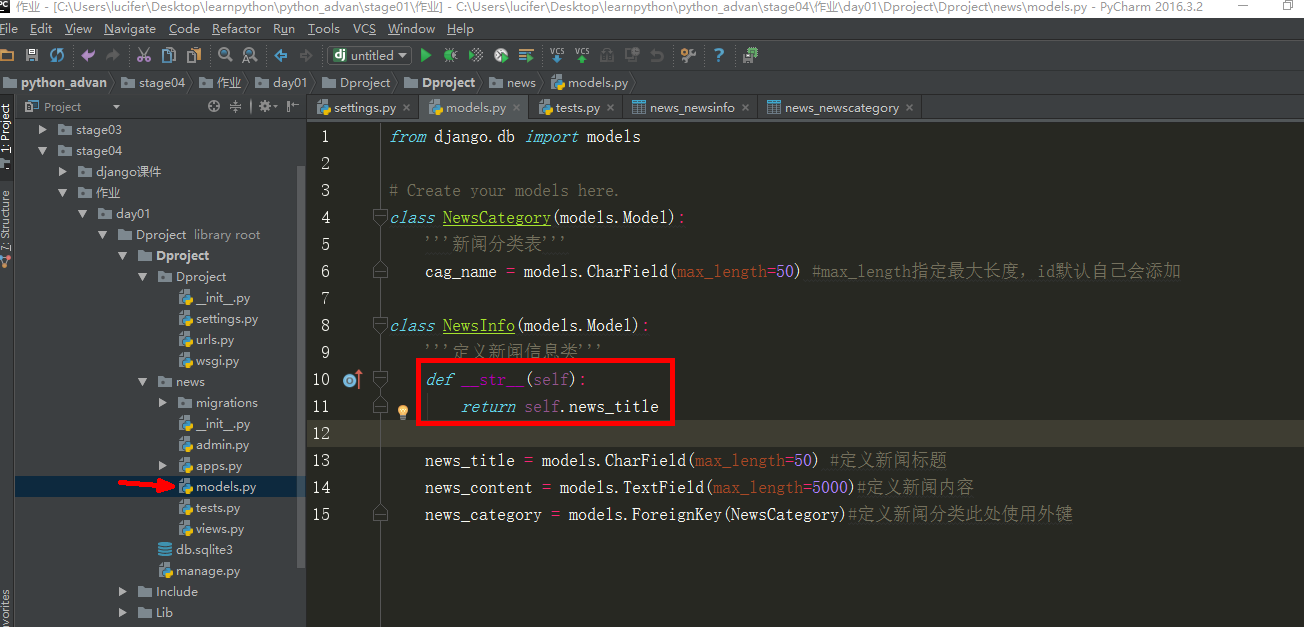
现title



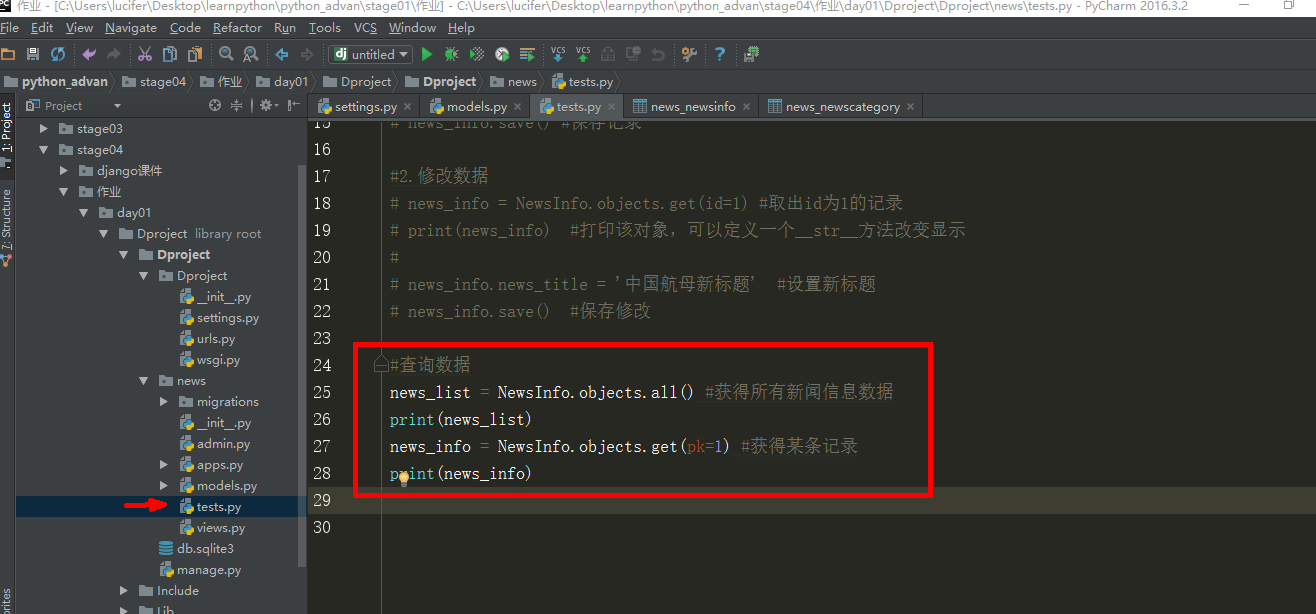
* 查询数据

为了方便查看数据，在models相应的类中增加一个\_\_str\_\_方法，

改方法返回了新闻的标题



回到test.py 测试文件 增加查询的内容



* 源代码如下： 代码略做修改为了更好显示内容

#查询数据  
news\_list = NewsInfo.objects.all() #获得所有新闻信息数据  
print('获取所有新闻信息数据:'+ str(news\_list))  
news\_info = NewsInfo.objects.get(pk=1) #获得某条记录  
print('获得一条新闻数据:'+ str(news\_info))

* 在命令行运行命令执行查询

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject\Dproject>**python manage.py test news**

**获取所有新闻信息数据:<QuerySet [<NewsInfo: 中国航母新标题>]>**

**获得一条新闻数据:中国航母新标题**

Creating test database for alias 'default'...

System check identified no issues (0 silenced).

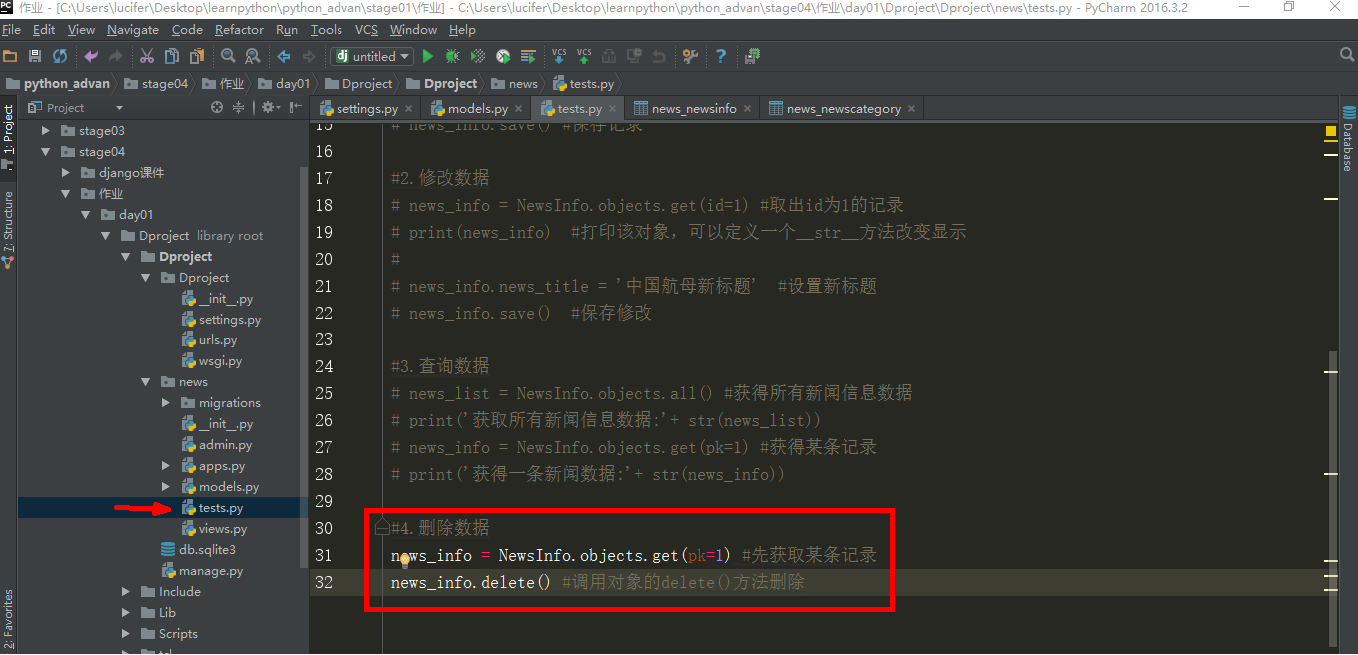
----------------------------------------------------------------------

Ran 0 tests in 0.000s

OK

Destroying test database for alias 'default'...

* 删除数据



* 以下是源代码

#4.删除数据  
news\_info = NewsInfo.objects.get(pk=1) #先获取某条记录  
news\_info.delete() #调用对象的delete()方法删除

* 使用命令删除记录

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject\Dproject>**python manage.py test news**

Creating test database for alias 'default'...

System check identified no issues (0 silenced).

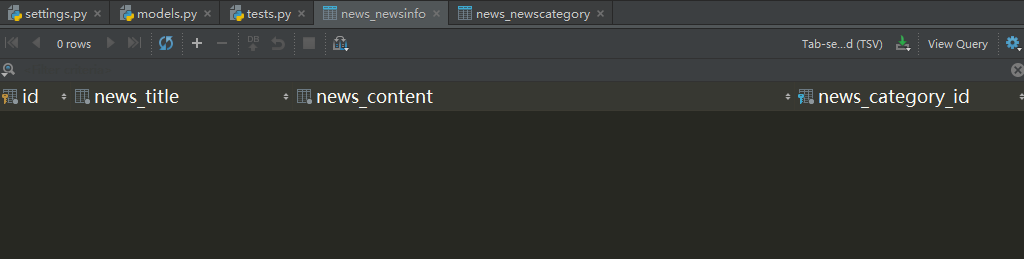
----------------------------------------------------------------------

Ran 0 tests in 0.000s

OK

Destroying test database for alias 'default'...

可以看到表中已经无记录了



接下去简略介绍以下使用shell查询的方法

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject\Dproject>python manage.py shell

Python 3.6.0 (v3.6.0:41df79263a11, Dec 23 2016, 08:06:12) [MSC v.1900 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license" for more information.

(InteractiveConsole)

**>>> from news.models import \***

**>>> nc = NewsCategory.objects.get(pk=1)**

**>>> nc.cag\_name**

'军事新闻'

**>>> ncs = NewsCategory.objects.all()**

**>>> ncs**

<QuerySet [<NewsCategory: NewsCategory object>]>

**>>> nc = NewsCategory()**

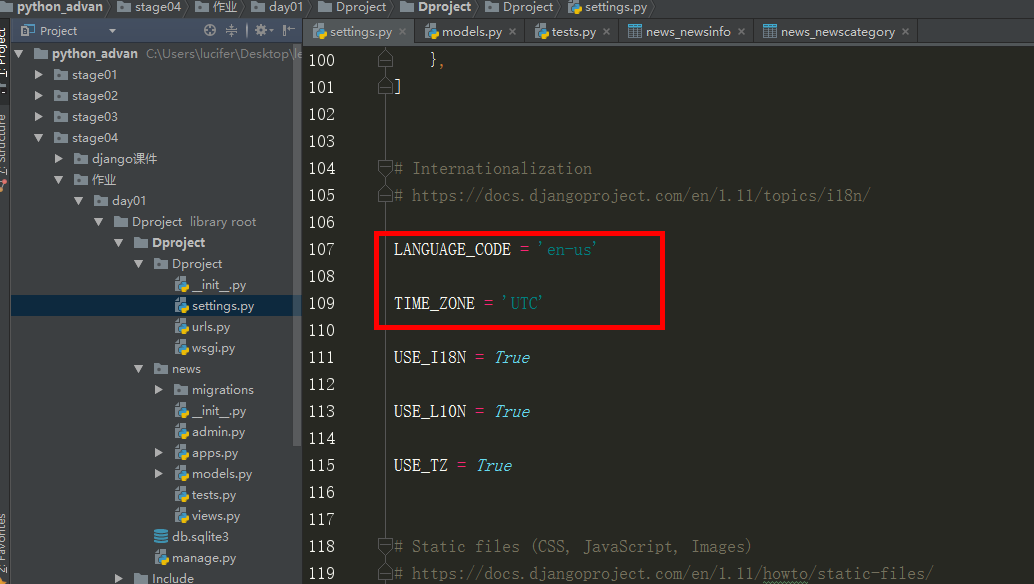
**>>> nc.cag\_name = '娱乐新闻'**

**>>> nc.save()**

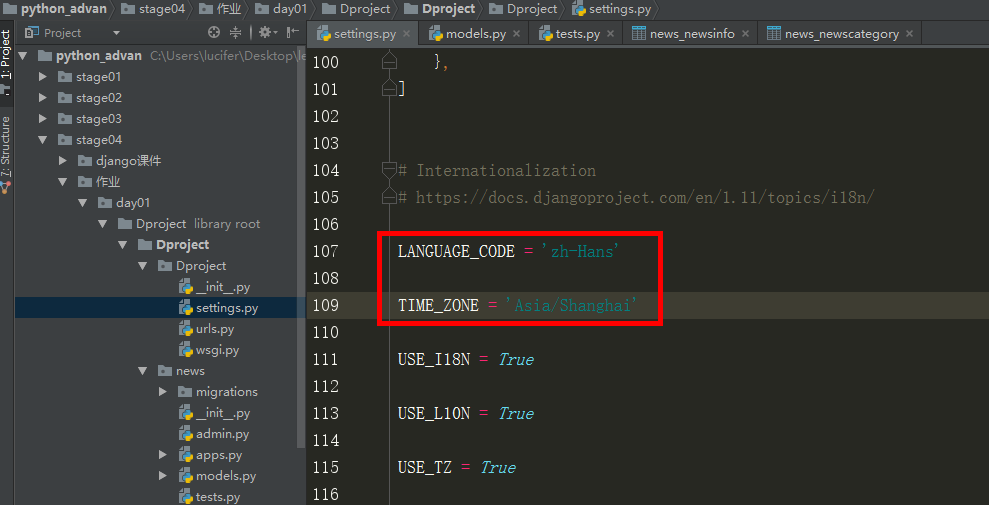
# 后台管理

* 管理界面本地化;
* 创建管理员;
* 注册模型类;
* 自定义管理界面.
* 管理本地化

修改settings.py文件



修改成



源代码如下

LANGUAGE\_CODE = 'zh-Hans'  
  
TIME\_ZONE = 'Asia/Shanghai'

* 创建管理员

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject\Dproject>**python manage.py createsuperuser**

Username (leave blank to use 'lucifer'): **admin**

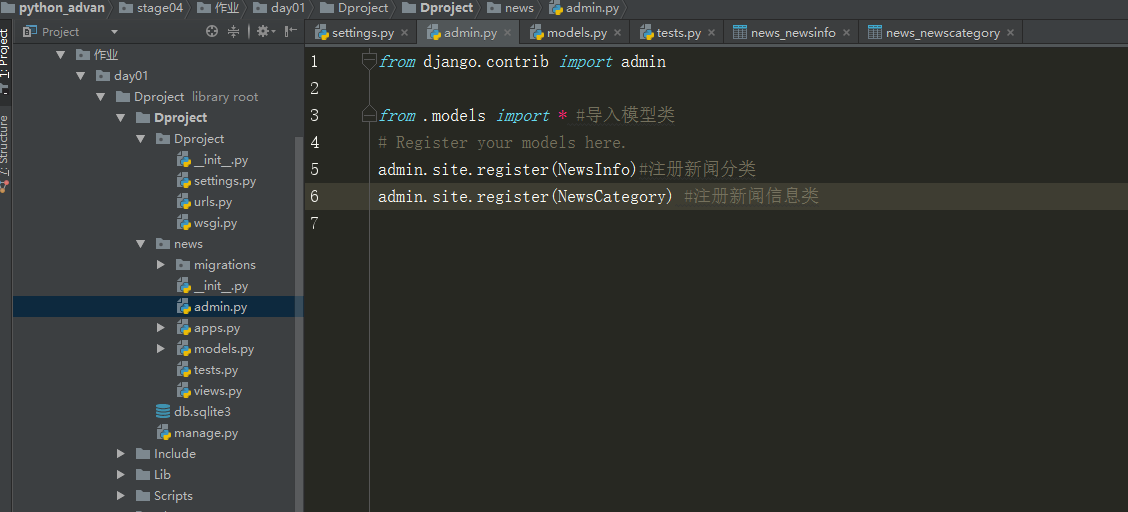
Email address: **admin@Dproject.com**

Password:**abcd1234,**

Password (again):

Superuser created successfully.

* 注册模型类



以下是源代码

*from* .models *import* \* #导入模型类  
# Register your models here.  
admin.site.register(NewsInfo)#注册新闻分类  
admin.site.register(NewsCategory) #注册新闻信息类

* 启动服务器

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject\Dproject>**python manage.py runserver**

Performing system checks...

System check identified no issues (0 silenced).

October 06, 2017 - 18:45:04

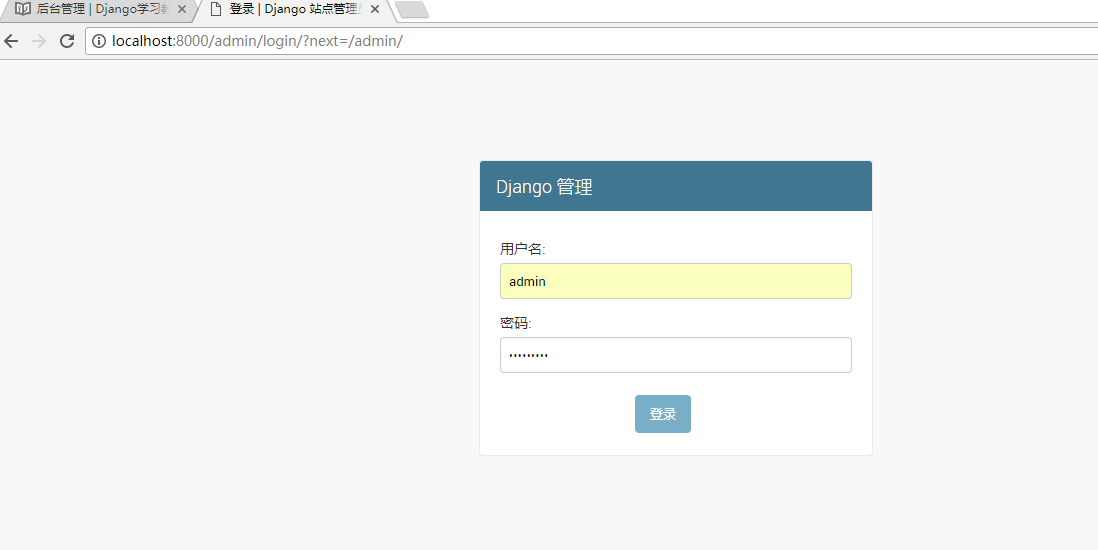
Django version 1.11.5, using settings 'Dproject.settings'

Starting development server at http://127.0.0.1:8000/

Quit the server with CTRL-BREAK.

通过浏览器访问管理站点

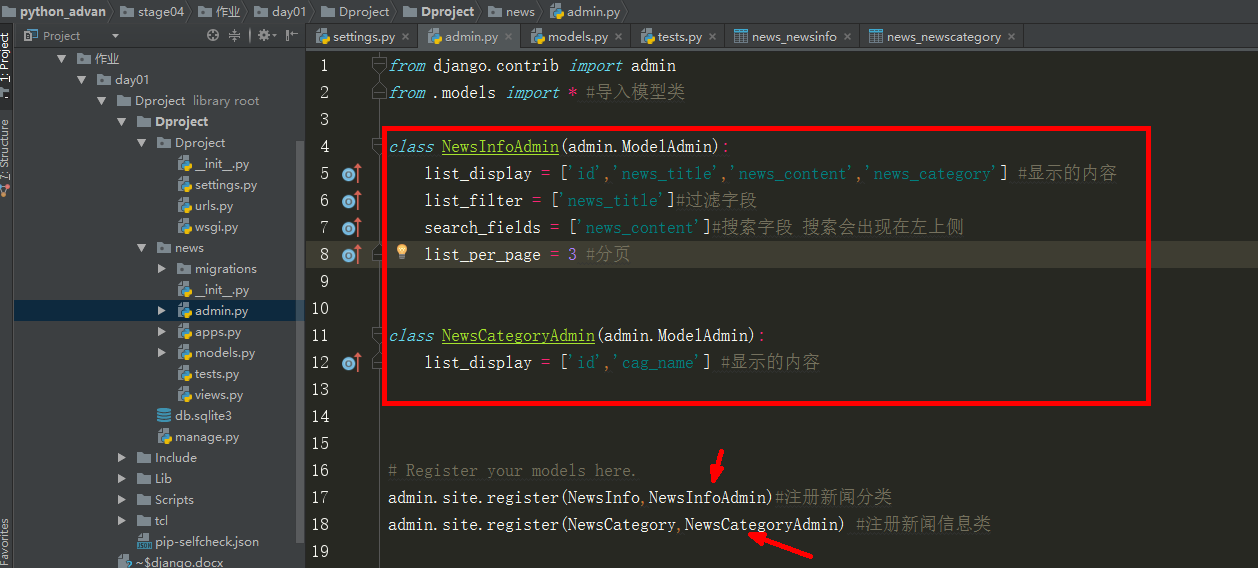
网址是 http://localhost:8000/admin





* 自定义界面管理类

在admin.py中创建连个类并设置 list\_display



以下是源代码

*from* django.contrib *import* admin  
*from* .models *import* \* #导入模型类  
  
*class* NewsInfoAdmin(admin.ModelAdmin):  
 list\_display = ['id','news\_title','news\_content','news\_category'] #显示的内容  
 list\_filter = ['news\_title']#过滤字段  
 search\_fields = ['news\_content']#搜索字段 搜索会出现在左上侧  
 list\_per\_page = 3 #分页  
  
  
*class* NewsCategoryAdmin(admin.ModelAdmin):  
 list\_display = ['id','cag\_name'] #显示的内容  
  
  
  
# Register your models here.  
admin.site.register(NewsInfo,NewsInfoAdmin)#注册新闻分类  
admin.site.register(NewsCategory,NewsCategoryAdmin) #注册新闻信息类

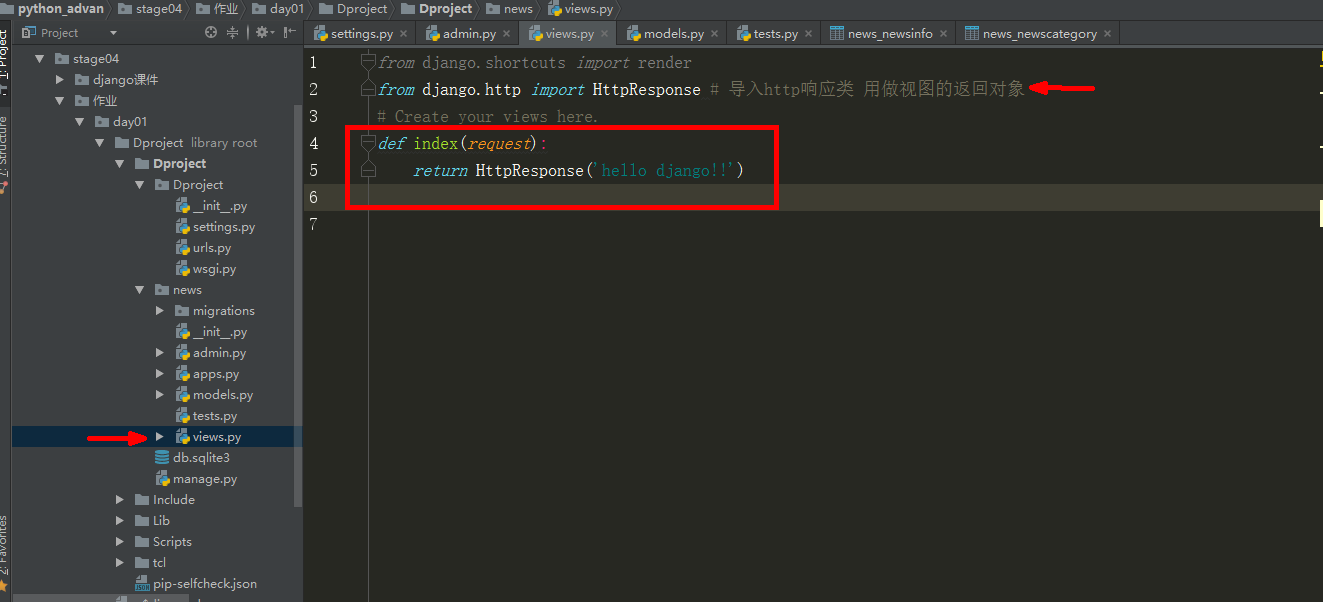
以下是修改后的显示效果



# 视图

* 定义视图;
* 配置URLconf
* 定义视图

先定义一个主页视图 接受request请求参数，并返回最简单的页面 hello django

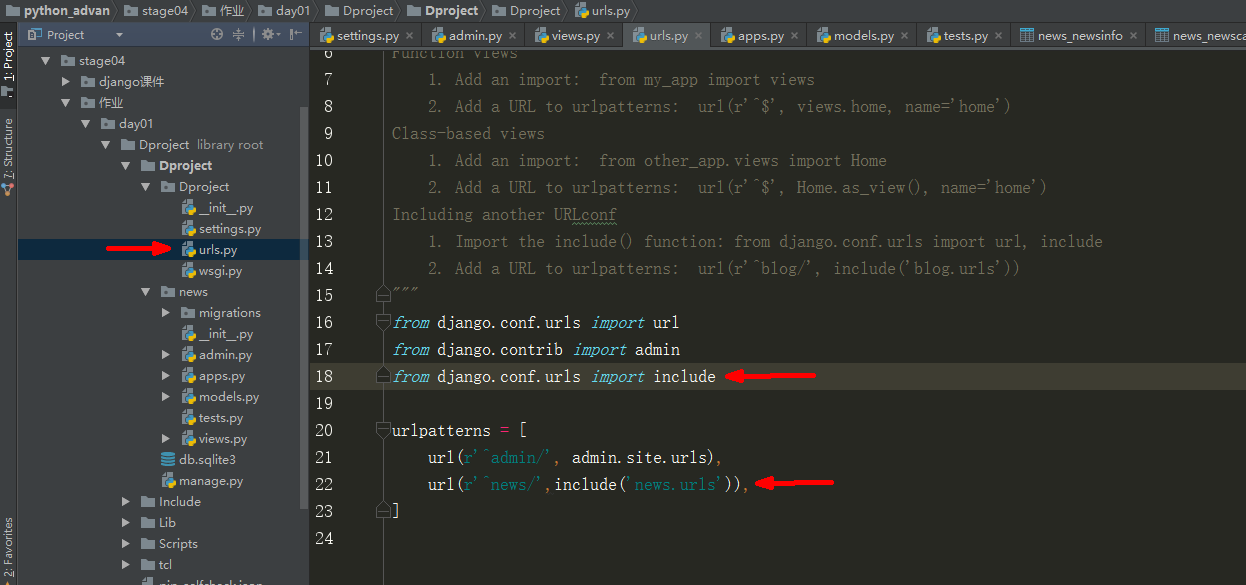


以下是源代码内容

*from* django.shortcuts *import* render  
*from* django.http *import* HttpResponse # 导入http响应类 用做视图的返回对象  
# Create your views here.  
*def* index(*request*):  
 *return* HttpResponse('hello django!!')

* 配置urlconf

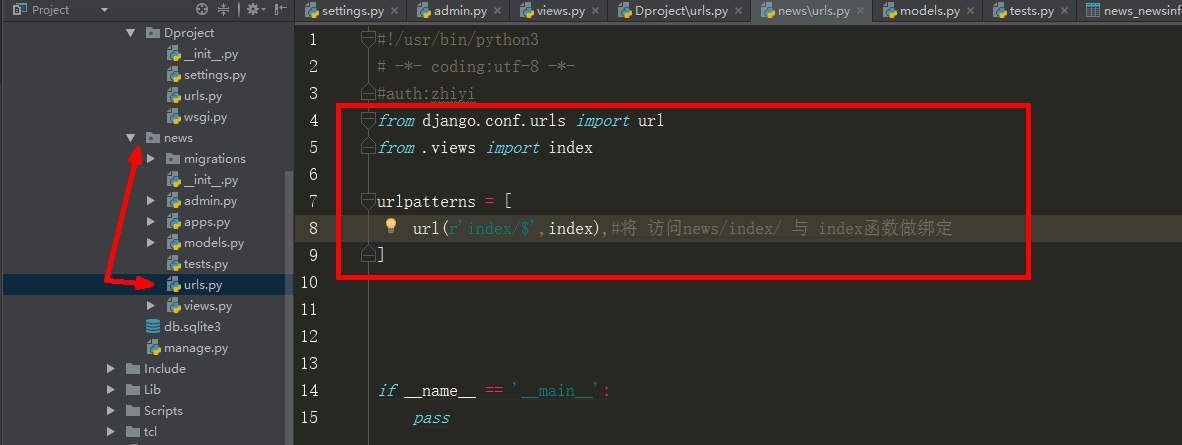
在项目下的urls.py文件修改以下内容



源代码如下：

*from* django.conf.urls *import* url  
*from* django.contrib *import* admin  
*from* django.conf.urls *import* include  
  
urlpatterns = [  
 url(r'^admin/', admin.site.urls),  
 url(r'^news/',include('news.urls')),  
]

然后在news 这个app的文件夹下 再建立一个urls.py 放入如下内容



代码如下

*from* django.conf.urls *import* url  
*from* .views *import* index  
  
urlpatterns = [  
 url(r'index/$',index),#将 访问news/index/ 与 index函数做绑定  
]

* 重新启动服务

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject\Dproject>**python manage.py runserver**

Performing system checks...

System check identified no issues (0 silenced).

October 06, 2017 - 19:41:28

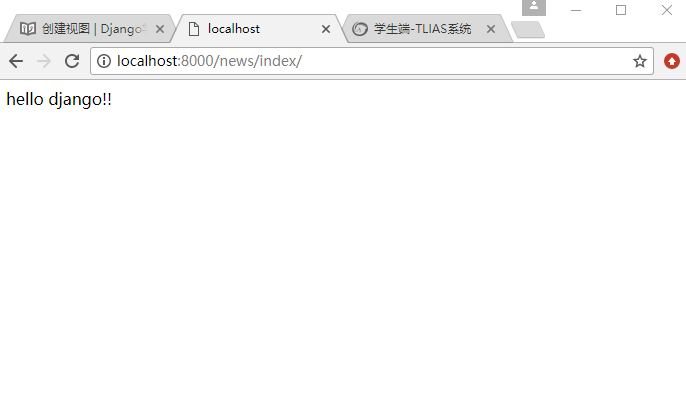
Django version 1.11.5, using settings 'Dproject.settings'

Starting development server at http://127.0.0.1:8000/

Quit the server with CTRL-BREAK.

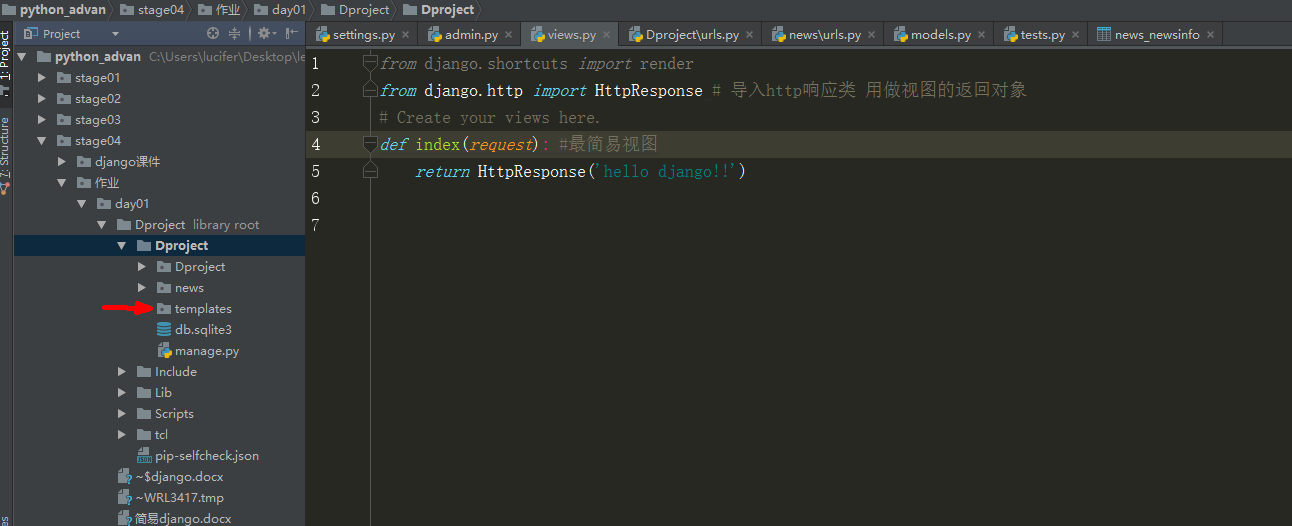
[06/Oct/2017 19:41:31] "GET /news/index/ HTTP/1.1" 200 14

* 访问http://localhost:8000/news/index/

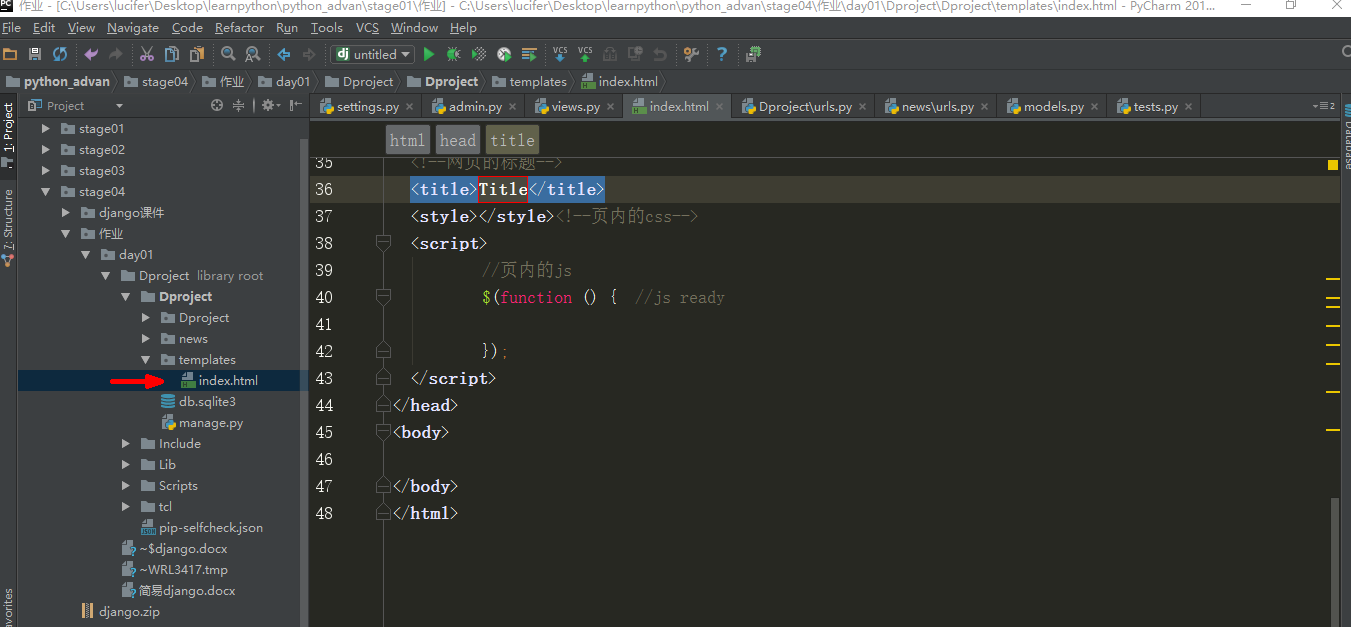


# 使用模板

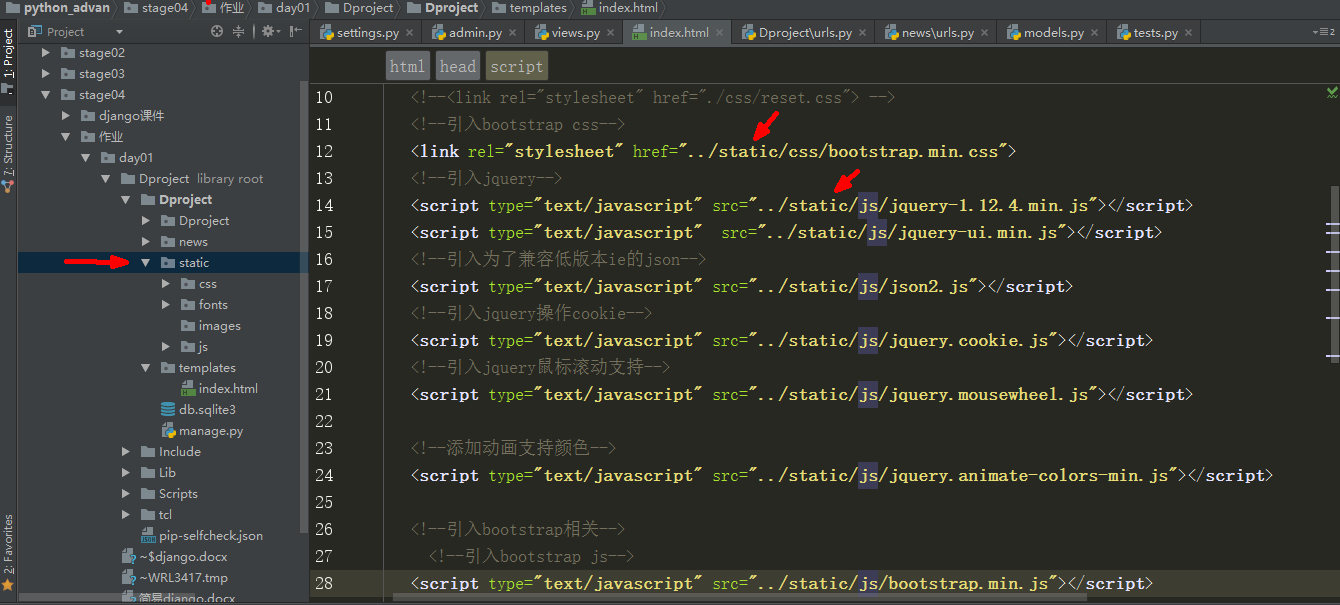
首先在项目目录下新建一个 templates文件夹专门用于存放模板



* 在模板文件夹下创建一个 index.html



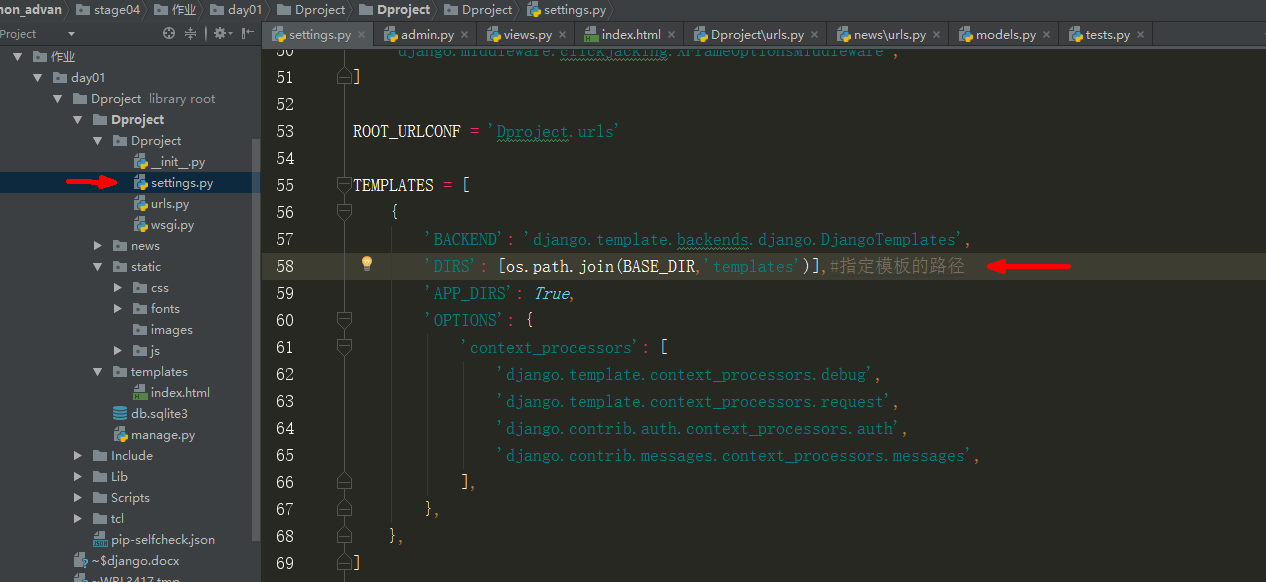
由于我在pycharm中设置了 html的模板会引入css和jquery等js文件 因此特此在建立一个static文件夹，并且拷贝css和js等，并且路径也做了修改



在index.html中增加可替换内容的标记



* 在settings.py中注册模板目录

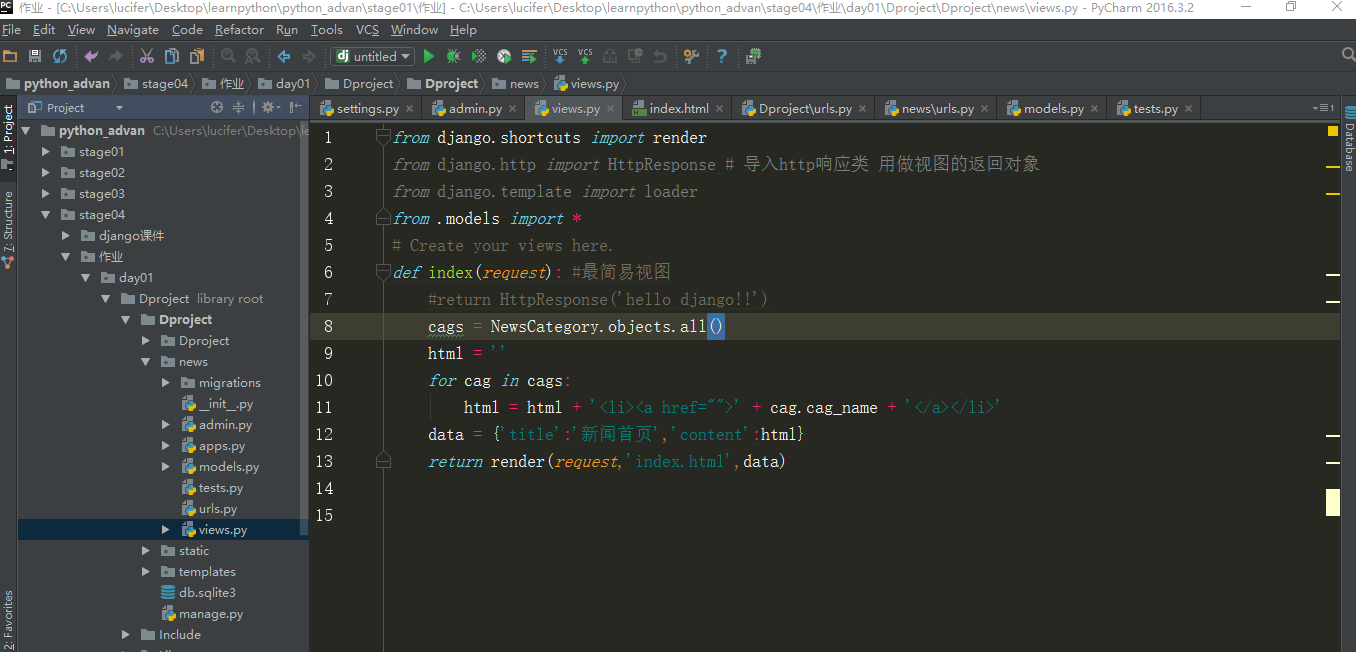


代码如下：

TEMPLATES = [  
 {  
 'BACKEND': 'django.template.backends.django.DjangoTemplates',  
 'DIRS': [os.path.join(BASE\_DIR,'templates')],#指定模板的路径  
 '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',  
 ],  
 },  
 },  
]

* 使用模板

修改views.py下的 index函数



代码如下

*from* django.shortcuts *import* render  
*from* django.http *import* HttpResponse # 导入http响应类 用做视图的返回对象  
*from* django.template *import* loader  
*from* .models *import* \*  
# Create your views here.  
*def* index(*request*): #最简易视图  
 #return HttpResponse('hello django!!')  
 cags = NewsCategory.objects.all()  
 html = ''  
 *for* cag *in* cags:  
 html = html + '<li><a href="">' + cag.cag\_name + '</a></li>'  
 data = {'title':'新闻首页','content':html}  
 *return* render(*request*,'index.html',data)

修改完毕后

访问http://localhost:8000/news/index/



可以看到内容已经成功过替换了 只是内容的外面多了引号。这个问题等待下一次再解决

* 接以上 来解决一下页面显示的问题

修改虚拟环境启动方式

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject>**Scripts\python.exe Dproject\manage.py runserver**

Performing system checks...

System check identified no issues (0 silenced).

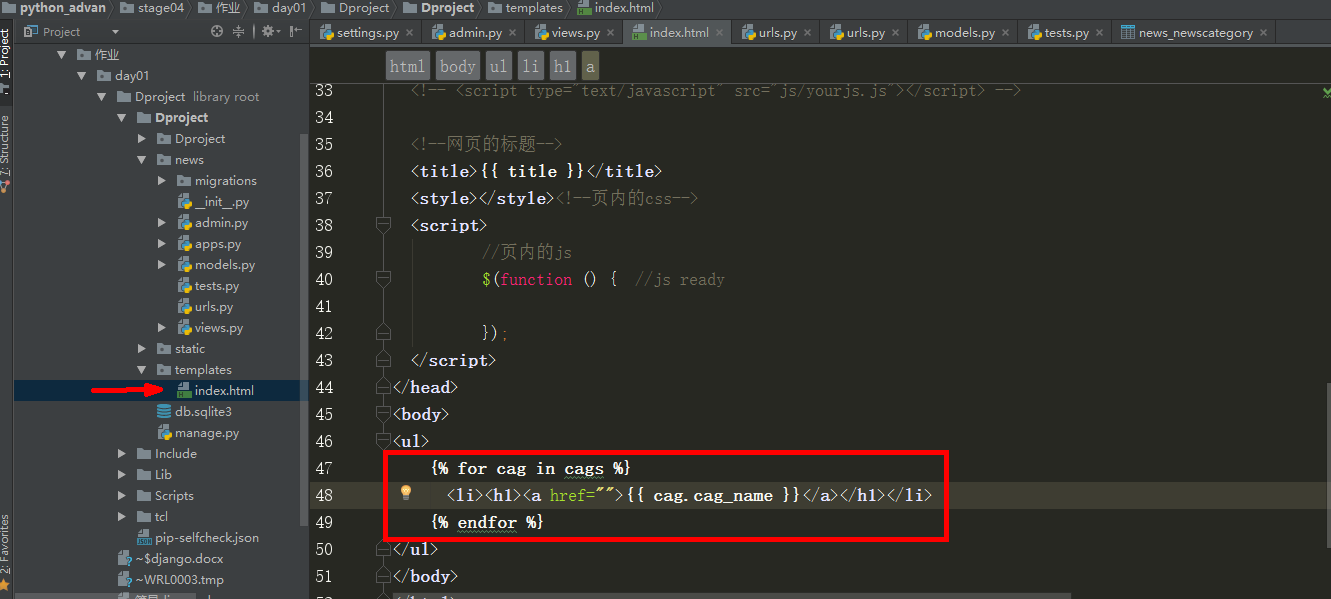
October 07, 2017 - 14:38:28

Django version 1.11.5, using settings 'Dproject.settings'

Starting development server at http://127.0.0.1:8000/

Quit the server with CTRL-BREAK.

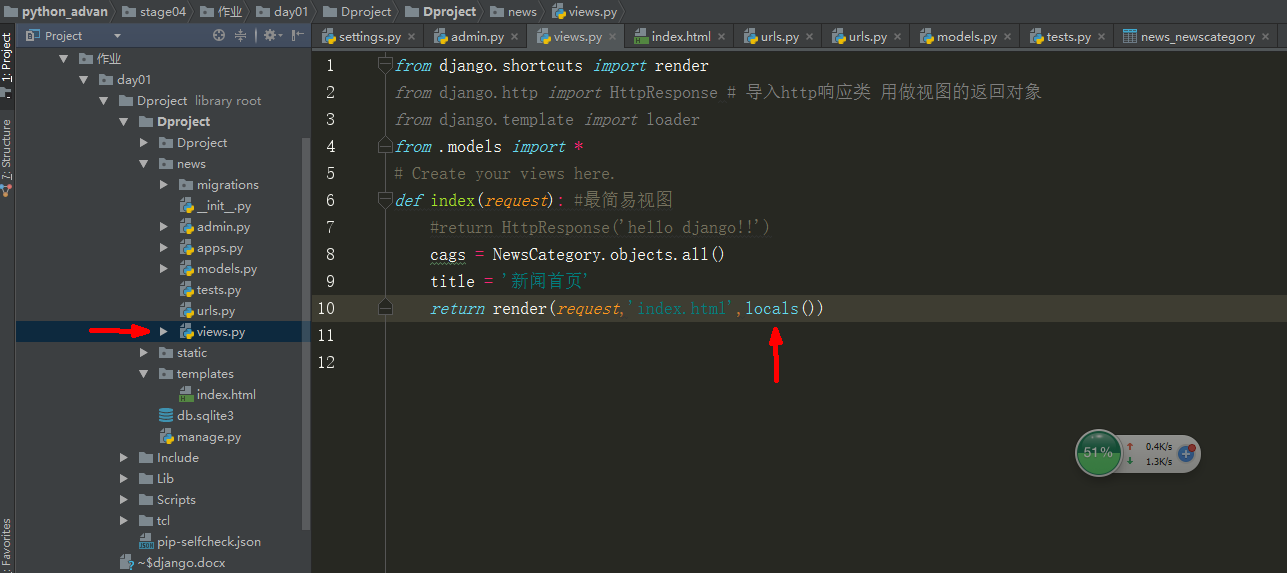
修改index页面里的代码，将for循环写在index里



具体代码如下

<**body**>  
<**ul**>  
 **{% for cag in cags %}** <**li**><**h1**><**a** href=**""**>**{{ cag.cag\_name }}**</**a**></**h1**></**li**>  
 **{% endfor %}**</**ul**>  
</**body**>

修改views里的index函数，更改替换方式，使用locals函数会更为方便，无需在自己组装字典，locals函数会为你组装



以下为源代码：

*from* django.shortcuts *import* render  
*from* django.http *import* HttpResponse # 导入http响应类 用做视图的返回对象  
*from* django.template *import* loader  
*from* .models *import* \*  
# Create your views here.  
*def* index(*request*): #最简易视图  
 #return HttpResponse('hello django!!')  
 cags = NewsCategory.objects.all()  
 title = '新闻首页'  
 *return* render(*request*,'index.html',locals())

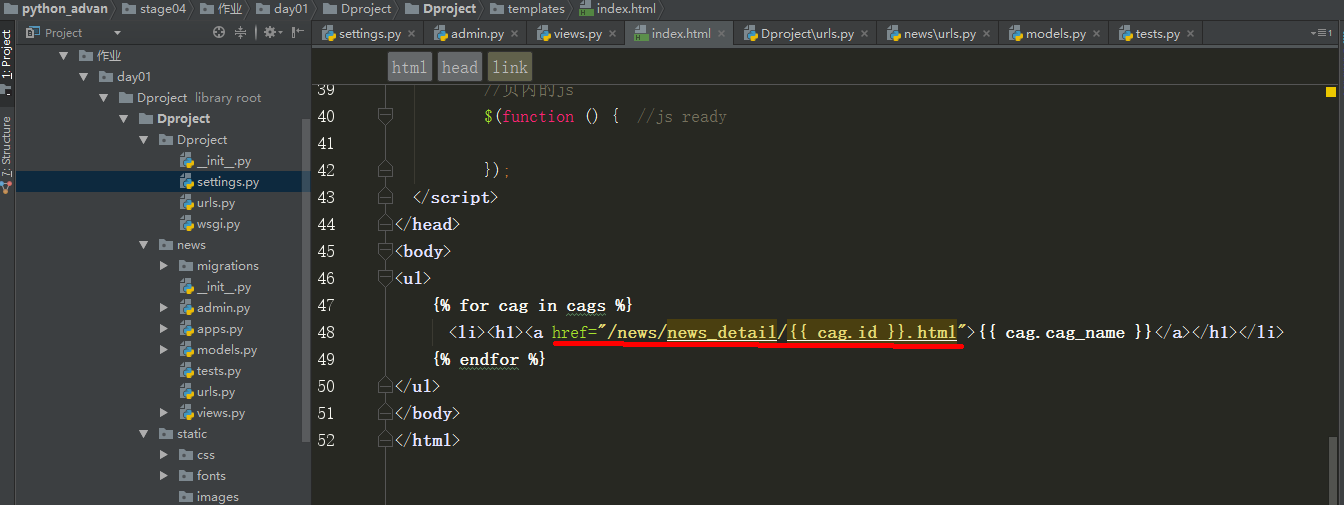
修改完毕之后再访问网页 http://127.0.0.1:8000/news/index/



到此网页已经按照要求已经正确的显示了

但是网页还没添加链接，我们要做到根据点击链接，显示具体新闻的内容

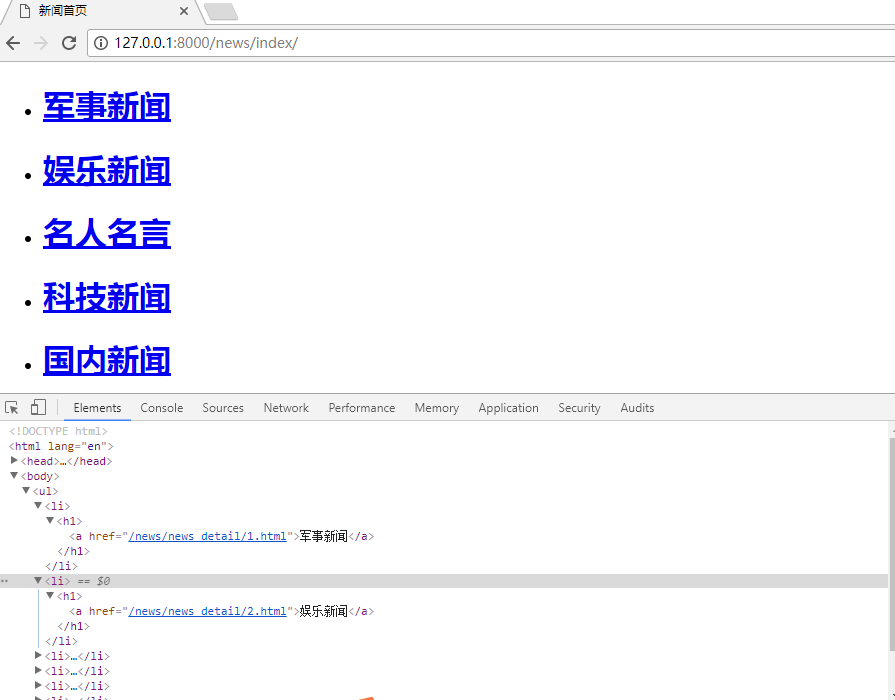
再次修改index.html 增加网页的herf 指向id页



源代码如下：

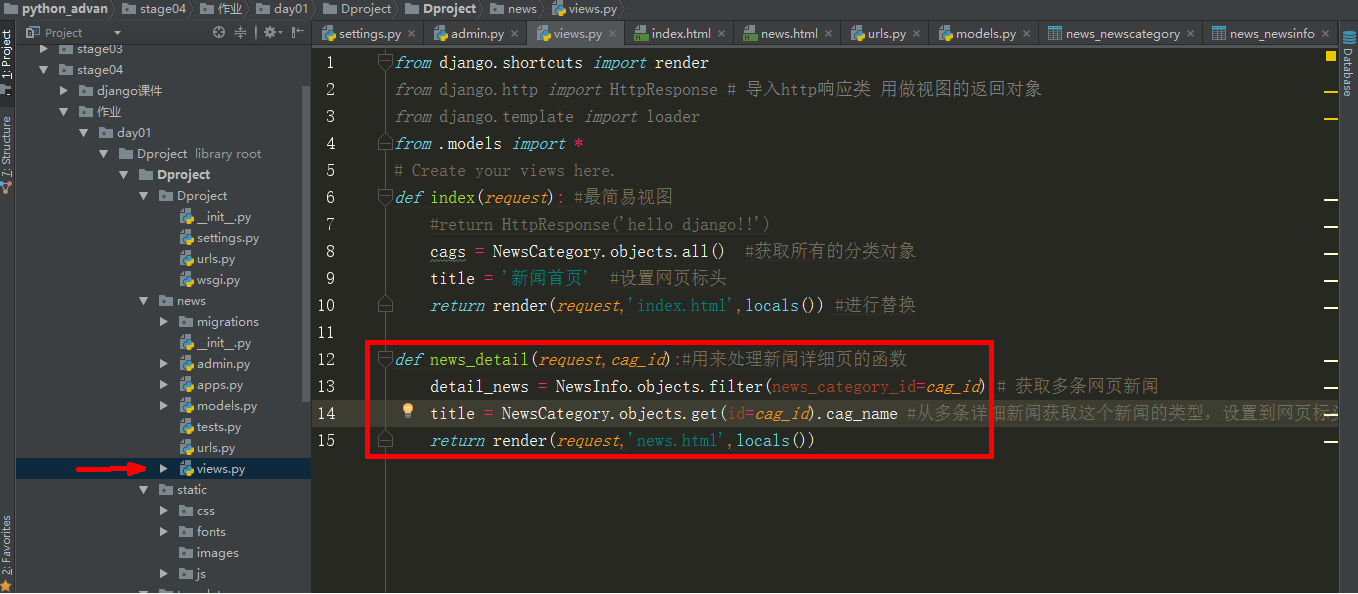
**{% for cag in cags %}** <**li**><**h1**><**a** href=**"/news/news\_detail/{{ cag.id }}.html"**>**{{ cag.cag\_name }}**</**a**></**h1**></**li**>  
**{% endfor %}**

经过这样的修改之后网页的超链接就生成了



可以看到不同的 超链接指向了不同的地址，接下来要做的就是在view中写一个详细新闻的函数，已经对应的模板 来处理这个请求

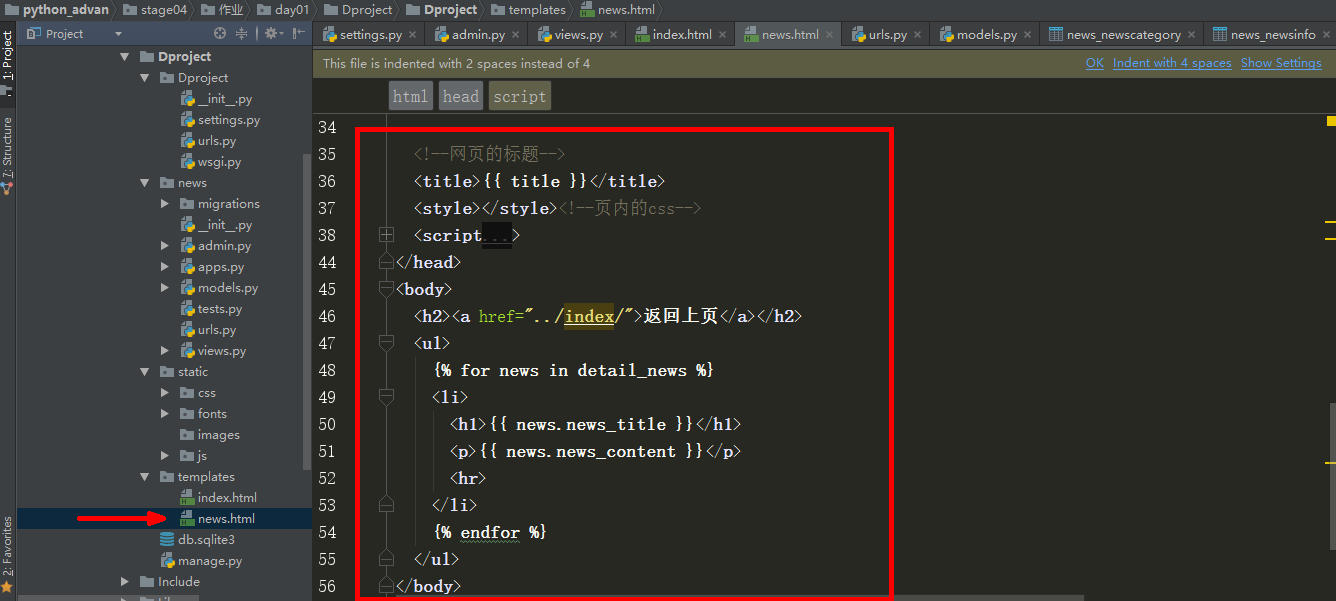
在view中增加一个函数news\_detail 专门用来处理详细新闻页的逻辑



具体代码如下：

*def* news\_detail(*request*,*cag\_id*):#用来处理新闻详细页的函数  
 detail\_news = NewsInfo.objects.filter(news\_category\_id=*cag\_id*) # 获取多条网页新闻  
 title = NewsCategory.objects.get(id=*cag\_id*).cag\_name #从多条详细新闻获取这个新闻的类型，设置到网页标头  
 *return* render(*request*,'news.html',locals())

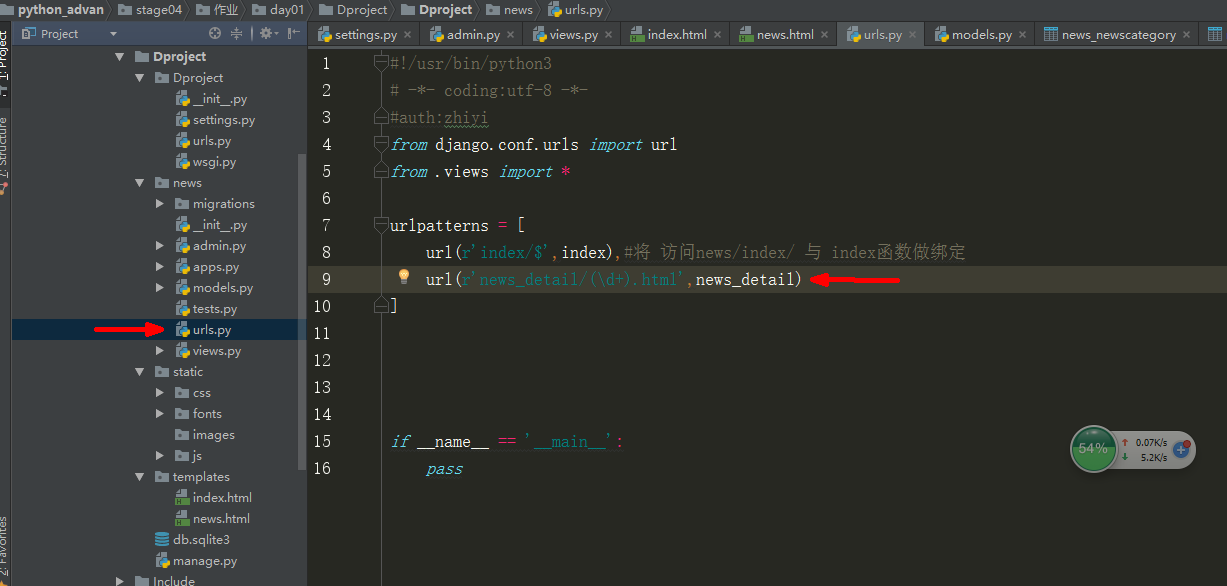
然后增加一个news的模板文件



源代码如下:

<!--网页的标题-->  
 <**title**>**{{ title }}**</**title**>  
 <**style**></**style**><!--页内的css-->  
 <**script**>  
 //页内的js  
 $(function () { //js ready  
   
 });   
 </**script**>  
</**head**>  
<**body**>  
 <**h2**><**a** href=**"../index/"**>**返回上页**</**a**></**h2**>  
 <**ul**>  
 **{% for news in detail\_news %}** <**li**>  
 <**h1**>**{{ news.news\_title }}**</**h1**>  
 <**p**>**{{ news.news\_content }}**</**p**>  
 <**hr**>  
 </**li**>  
 **{% endfor %}** </**ul**>  
</**body**>

要在news项目下的urls.py文件中加上一个路由才能正常工作



正则代码如下：

url(r'news\_detail/(\d+).html',news\_detail)

完成以下工作后 就可以访问网页了







可以看到不同的页面点击后出现的内容是不相同的

# 从sqlite3转向mysql数据库

以下实例使用的python版本为3.6

mysql数据库版本为5.7.17

mysql的安装方法不在本文的讨论范围内因此略去

想要使用mysql数据库，需要安装Pymysql

pip install pymysql

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\

Dproject>Scripts\**pip freeze**

Django==1.11.5

**PyMySQL==0.7.11**

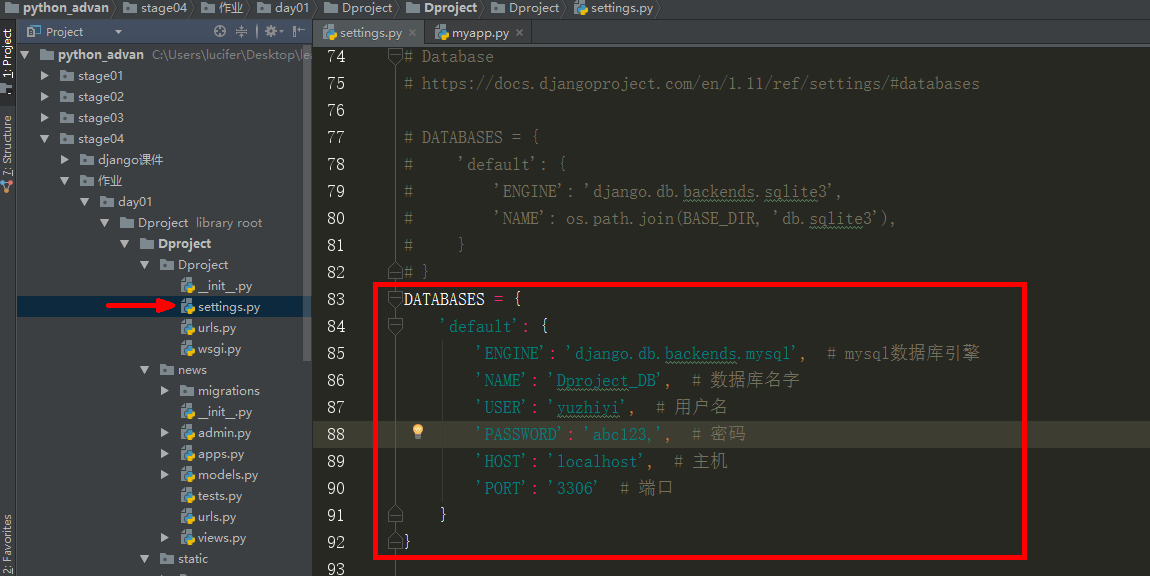
pytz==2017.2

SQLAlchemy==1.1.13

这里我使用的PyMysql是拷贝过去的，由于第一次这样拷贝不确定能不能正常工作

所以这里需要验证以下。（经过验证可以正常工作）

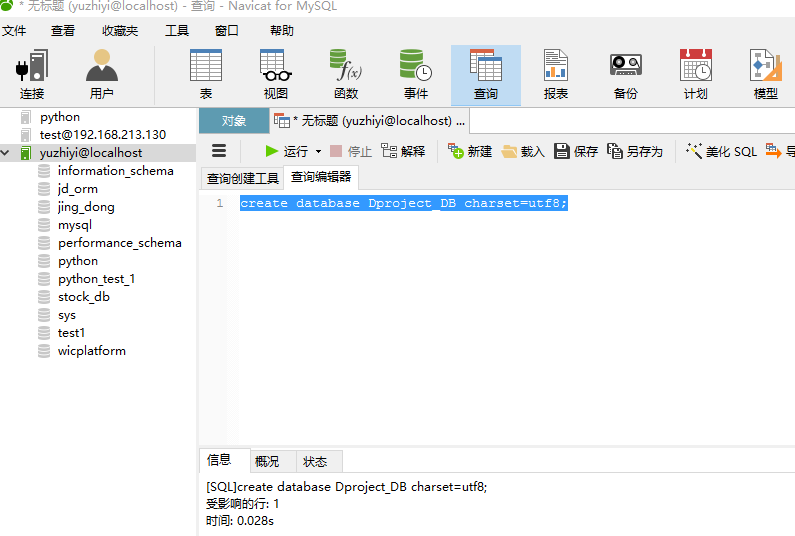
安装完pymysql之后 需要修改 settings.py 下的DATABASES



* 源代码如下

# Database  
# https://docs.djangoproject.com/en/1.11/ref/settings/#databases  
  
# DATABASES = {  
# 'default': {  
# 'ENGINE': 'django.db.backends.sqlite3',  
# 'NAME': os.path.join(BASE\_DIR, 'db.sqlite3'),  
# }  
# }  
DATABASES = {  
 'default': {  
 'ENGINE': 'django.db.backends.mysql', # mysql数据库引擎  
 'NAME': 'Dproject\_DB', # 数据库名字  
 'USER': 'yuzhiyi', # 用户名  
 'PASSWORD': 'abc123,', # 密码  
 'HOST': 'localhost', # 主机  
 'PORT': '3306' # 端口  
 }  
}

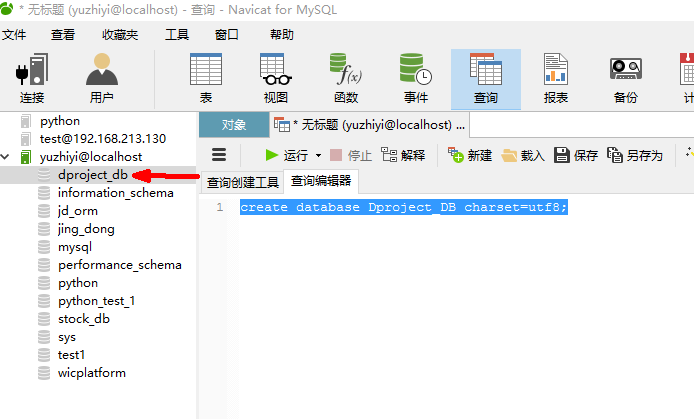
由于库名叫 Dproject\_DB 因此要在mysql下新建一个数据库



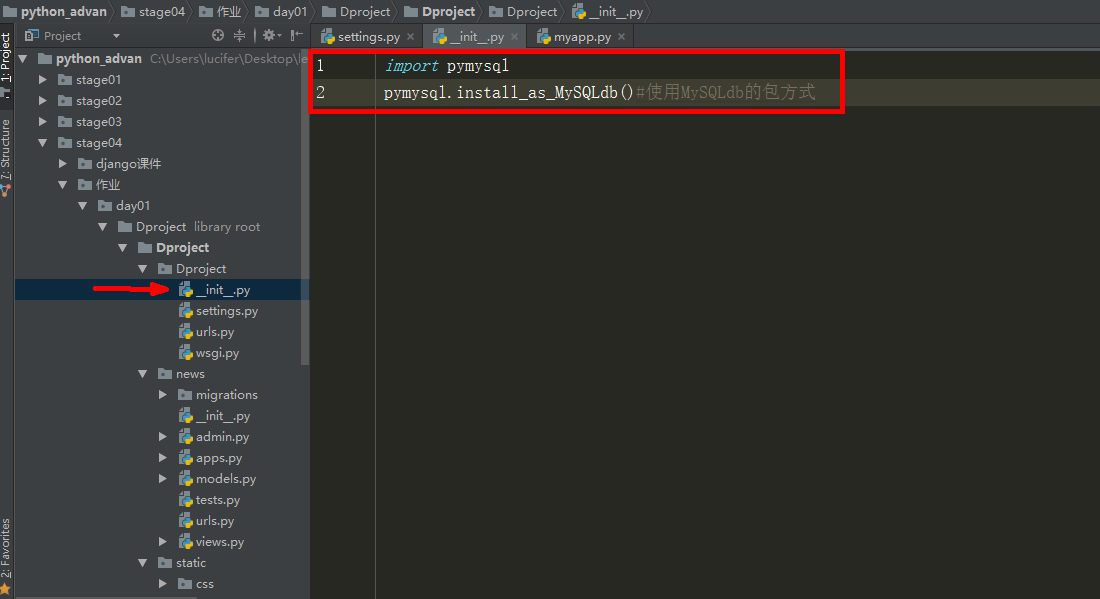
建库语句如下，注意一定要设一下字符集：

**create database Dproject\_DB charset=utf8;**

刷新后可以看到数据库已经建立



接下来要设置使用数据库驱动，要想使用驱动就要更改项目下的\_\_init\_\_.py 已导入pymysql驱动



源代码如下：

*import* pymysql  
pymysql.install\_as\_MySQLdb()#使用MySQLdb的包方式

导入完成之后启动服务器

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\

Dproject>**Scripts\python.exe Dproject\manage.py runserver**

Performing system checks...

System check identified no issues (0 silenced).

**You have 14 unapplied migration(s). Your project may not work properly until you**

**apply the migrations for app(s): admin, auth, contenttypes, news, sessions.**

Run 'python manage.py migrate' to apply them.

October 07, 2017 - 16:57:04

Django version 1.11.5, using settings 'Dproject.settings'

Starting development server at http://127.0.0.1:8000/

Quit the server with CTRL-BREAK.

服务器是启动了 说明我们的pymysql已经配置正确。

注意黑体字提示要做迁移。原因是因为我们已经从sqlite 迁移到了mysql

我们的mysql数据库还是空的呢，因此我们做一下数据迁移

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\

Dproject>**Scripts\python.exe Dproject\manage.py makemigrations**

No changes detected

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject>**Scripts\python.exe Dproject\manage.py migrate**

Operations to perform:

Apply all migrations: admin, auth, contenttypes, news, 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

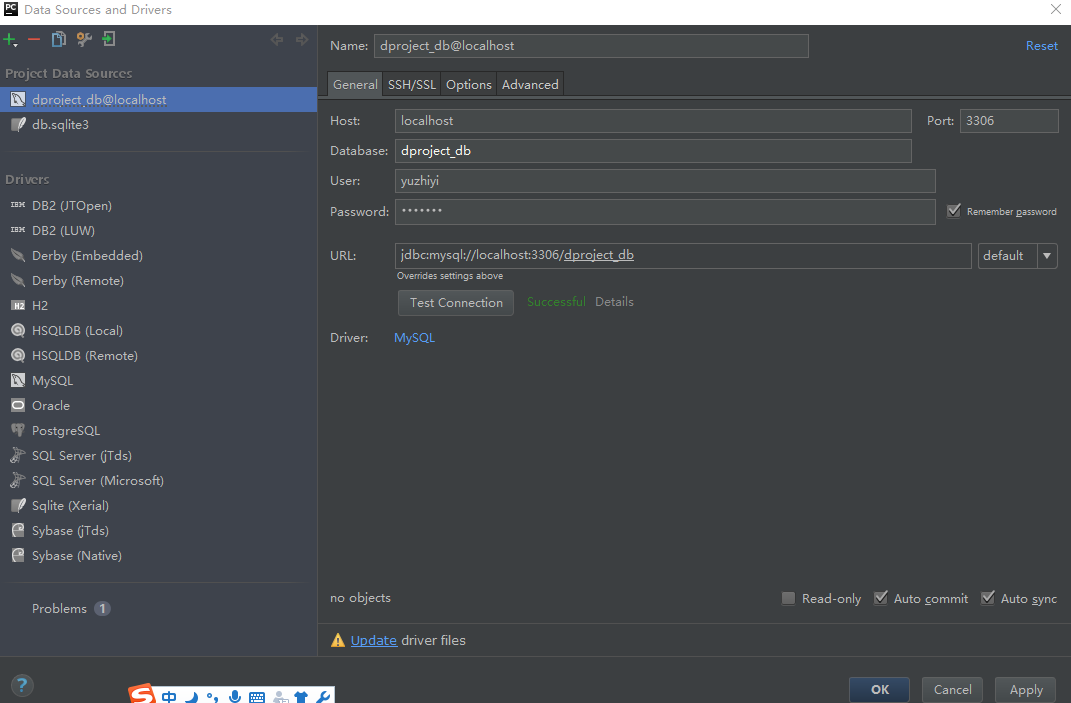
Applying news.0001\_initial... OK

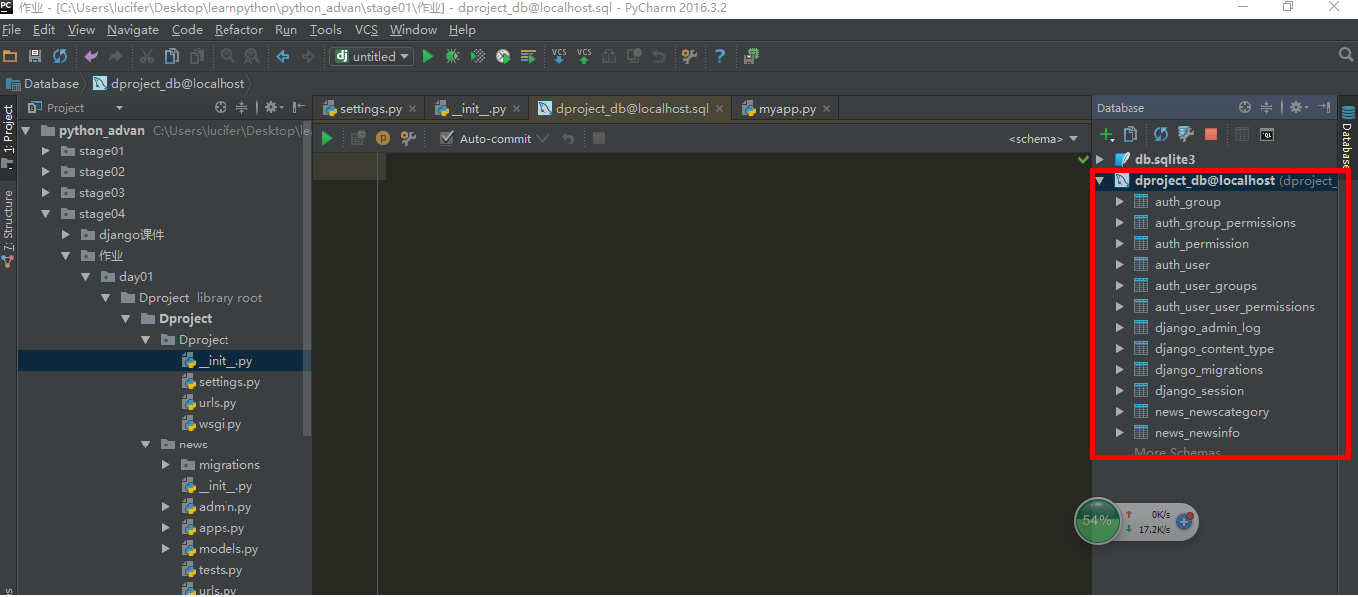
Applying sessions.0001\_initial... OK

一切迁移成功！在navicate中已经可以看到表已经建立了



顺便一提 pycharm也自带mysql管理工具 设置一下就可以用了





再次启动服务器，已经不报错了

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject>**Scripts\python.exe Dproject\manage.py runserver**

Performing system checks...

System check identified no issues (0 silenced).

October 07, 2017 - 17:08:42

Django version 1.11.5, using settings 'Dproject.settings'

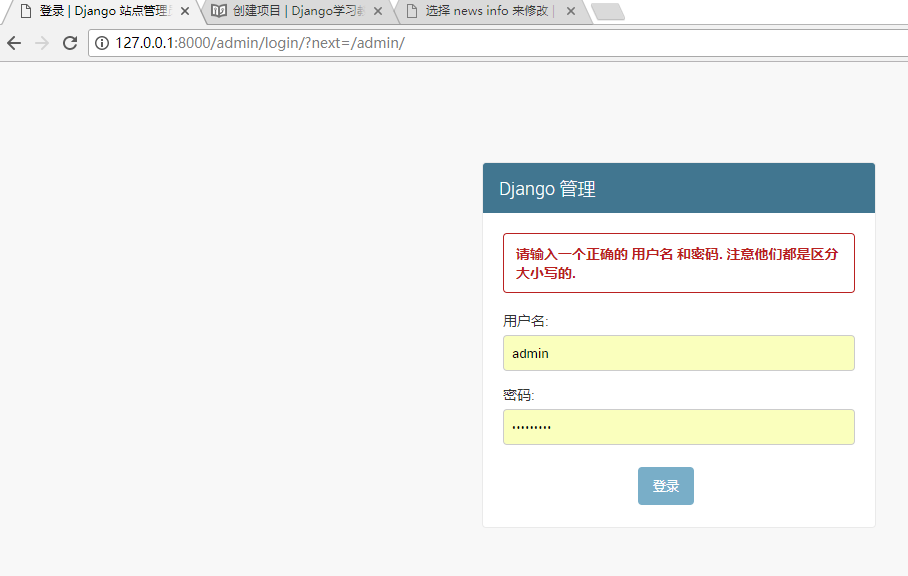
Starting development server at http://127.0.0.1:8000/

Quit the server with CTRL-BREAK.

页面也可以访问了



但是貌似管理员功不能登录了，可见原先的管理员用户名和密码记录在了sqlite里



尝试再建一个管理员吧

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject>**Scripts\python.exe Dproject\manage.py createsuperuser**

Username (leave blank to use 'lucifer'): admin

Email address: admin@Dproject.com

Password:abcd1234，

Password (again):

Superuser created successfully.

再启动服务器

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject>**Scripts\python.exe Dproject\manage.py runserver**

Performing system checks...

System check identified no issues (0 silenced).

October 07, 2017 - 17:15:27

Django version 1.11.5, using settings 'Dproject.settings'

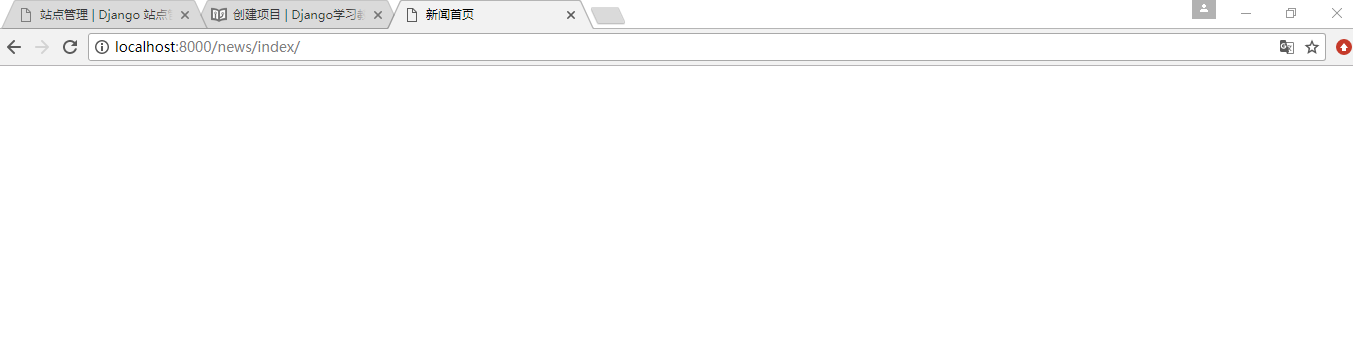
Starting development server at http://127.0.0.1:8000/

Quit the server with CTRL-BREAK.

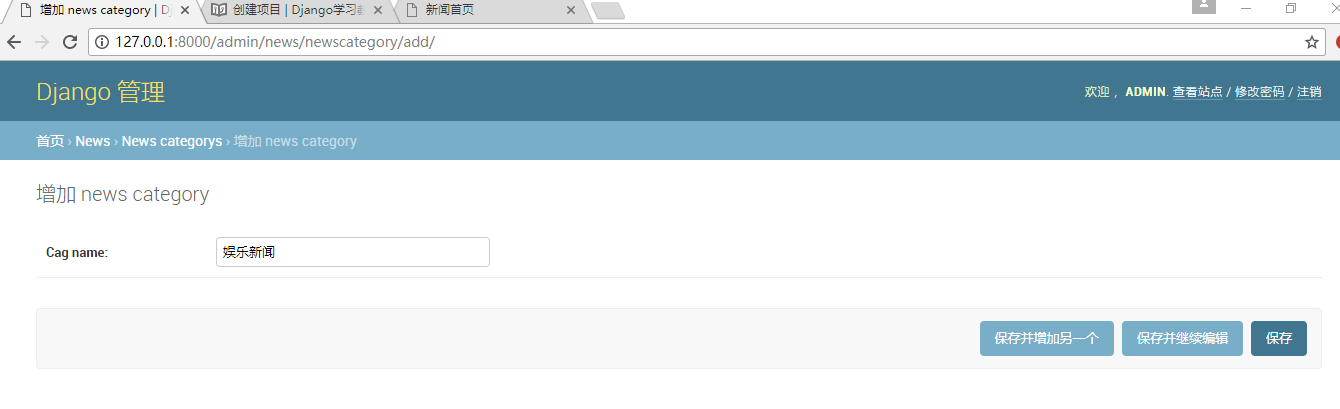
在访问浏览器已经可以顺利登录了



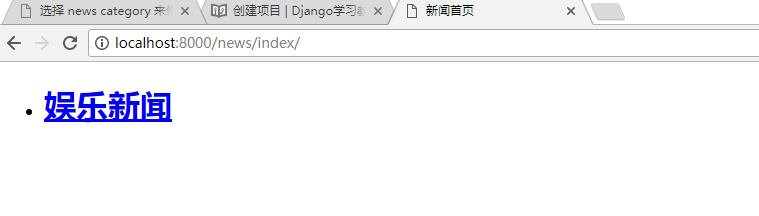
此时因为没有数据，因此新闻页面没有内容，与预期的一样



增加一个新闻分类



数据进入后内容就能正常显示了



由此可以看到一些ORM的优势，更换数据库不必更改代码

# Django模型的进一步研究

在学习过程中遇到了以下困惑：

增加表对原来的表是否有影响？

增加模型的表，是否会对原有表的数据产生影响？

增加字段然后做迁移对原有的数据有没有影响？

连接两个以上的数据库？

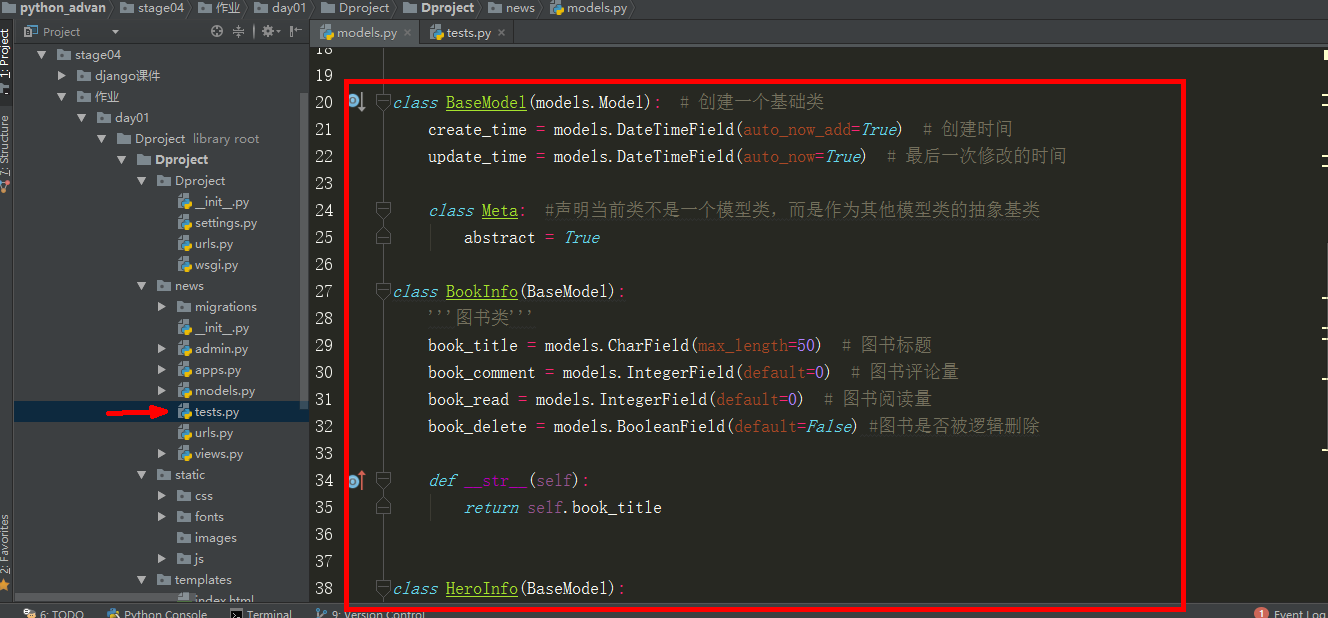
作为测试，不更换项目而是在原有项目的基础上增加表。

然后做迁移便能看出对原有项目中的表是否会产生影响。

这次增加的是小说及小说中的英雄表

为了不影响git的master分支 另外切出一个分支叫hero

在news应用下的models.py中追加小说和英雄表，并提取了一个基类用于记录创建和修改的时间

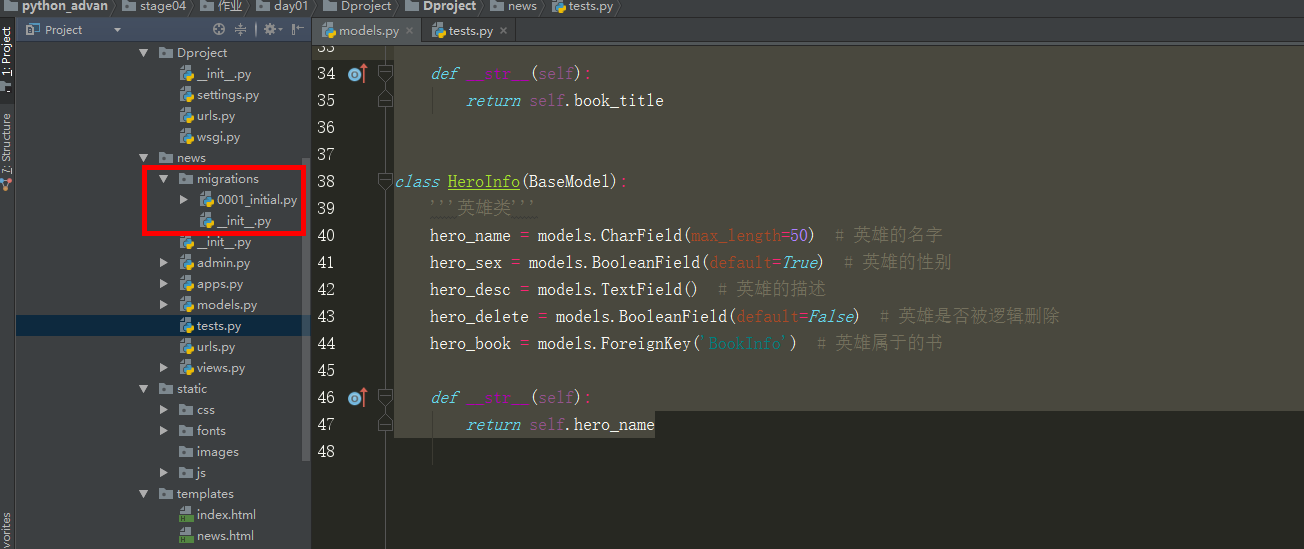


源代码如下：

*class* BaseModel(models.Model): # 创建一个基础类  
 create\_time = models.DateTimeField(auto\_now\_add=*True*) # 创建时间  
 update\_time = models.DateTimeField(auto\_now=*True*) # 最后一次修改的时间  
  
 *class* Meta: #声明当前类不是一个模型类，而是作为其他模型类的抽象基类  
 abstract = *True  
  
class* BookInfo(BaseModel):  
 '''图书类'''  
 book\_title = models.CharField(max\_length=50) # 图书标题  
 book\_comment = models.IntegerField(default=0) # 图书评论量  
 book\_read = models.IntegerField(default=0) # 图书阅读量  
 book\_delete = models.BooleanField(default=*False*) #图书是否被逻辑删除  
  
 *def* \_\_str\_\_(self):  
 *return* self.book\_title  
  
  
*class* HeroInfo(BaseModel):  
 '''英雄类'''  
 hero\_name = models.CharField(max\_length=50) # 英雄的名字  
 hero\_sex = models.BooleanField(default=*True*) # 英雄的性别  
 hero\_desc = models.TextField() # 英雄的描述  
 hero\_delete = models.BooleanField(default=*False*) # 英雄是否被逻辑删除  
 hero\_book = models.ForeignKey('BookInfo') # 英雄属于的书  
  
 *def* \_\_str\_\_(self):  
 *return* self.hero\_name

接下去生成迁移并迁移

这里我们特别关注一下生成迁移前的文件结构



接下来生成迁移

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\

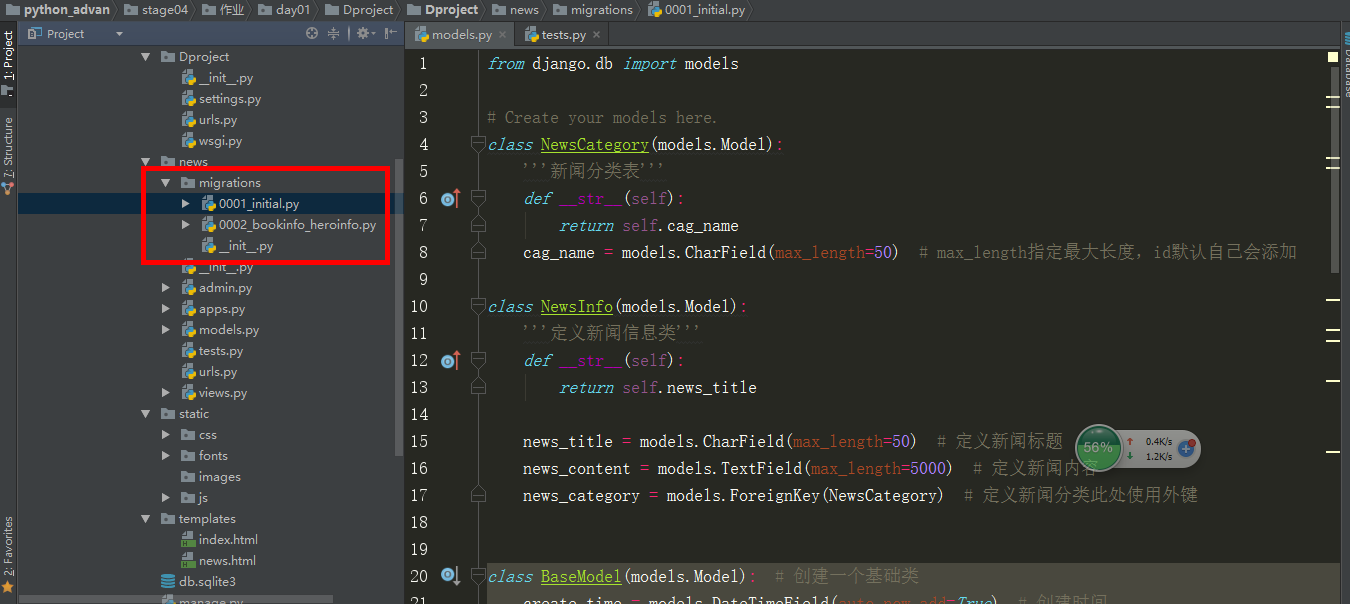
Dproject>**Scripts\python.exe Dproject\manage.py makemigrations**

Migrations for 'news':

Dproject\news\migrations\0002\_bookinfo\_heroinfo.py

- **Create model BookInfo**

- **Create model HeroInfo**



观察发现多了一个0002的文件

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\

Dproject>**Scripts\python.exe Dproject\manage.py sqlmigrate news 0001**

BEGIN;

--

-- Create model NewsCategory

--

CREATE TABLE `news\_newscategory` (`id` integer AUTO\_INCREMENT NOT NULL PRIMARY K

EY, `cag\_name` varchar(50) NOT NULL);

--

-- Create model NewsInfo

--

CREATE TABLE `news\_newsinfo` (`id` integer AUTO\_INCREMENT NOT NULL PRIMARY KEY,

`news\_title` varchar(50) NOT NULL, `news\_content` longtext NOT NULL, `news\_categ

ory\_id` integer NOT NULL);

ALTER TABLE `news\_newsinfo` ADD CONSTRAINT `news\_newsinfo\_news\_category\_id\_c18d8

327\_fk\_news\_newscategory\_id` FOREIGN KEY (`news\_category\_id`) REFERENCES `news\_n

ewscategory` (`id`);

COMMIT;

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\

Dproject>**Scripts\python.exe Dproject\manage.py sqlmigrate news 0002**

BEGIN;

--

-- Create model BookInfo

--

CREATE TABLE `news\_bookinfo` (`id` integer AUTO\_INCREMENT NOT NULL PRIMARY KEY,

`create\_time` datetime(6) NOT NULL, `update\_time` datetime(6) NOT NULL, `book\_ti

tle` varchar(50) NOT NULL, `book\_comment` integer NOT NULL, `book\_read` integer

NOT NULL, `book\_delete` bool NOT NULL);

--

-- Create model HeroInfo

--

CREATE TABLE `news\_heroinfo` (`id` integer AUTO\_INCREMENT NOT NULL PRIMARY KEY,

`create\_time` datetime(6) NOT NULL, `update\_time` datetime(6) NOT NULL, `hero\_na

me` varchar(50) NOT NULL, `hero\_sex` bool NOT NULL, `hero\_desc` longtext NOT NUL

L, `hero\_delete` bool NOT NULL, `hero\_book\_id` integer NOT NULL);

ALTER TABLE `news\_heroinfo` ADD CONSTRAINT `news\_heroinfo\_hero\_book\_id\_9de7d31c\_

fk\_news\_bookinfo\_id` FOREIGN KEY (`hero\_book\_id`) REFERENCES `news\_bookinfo` (`i

d`);

COMMIT;

可以看到 createtime 和 update\_time字段在两张表里都有增加。基类验证成功。

查看两个SQL文件，发现第一个是之前的数据，第2个才是我们要的数据，

那么问题来了，如果我们执行迁移会发生什么样的结果。有待进一步验证

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\

Dproject>**Scripts\python.exe Dproject\manage.py migrate**

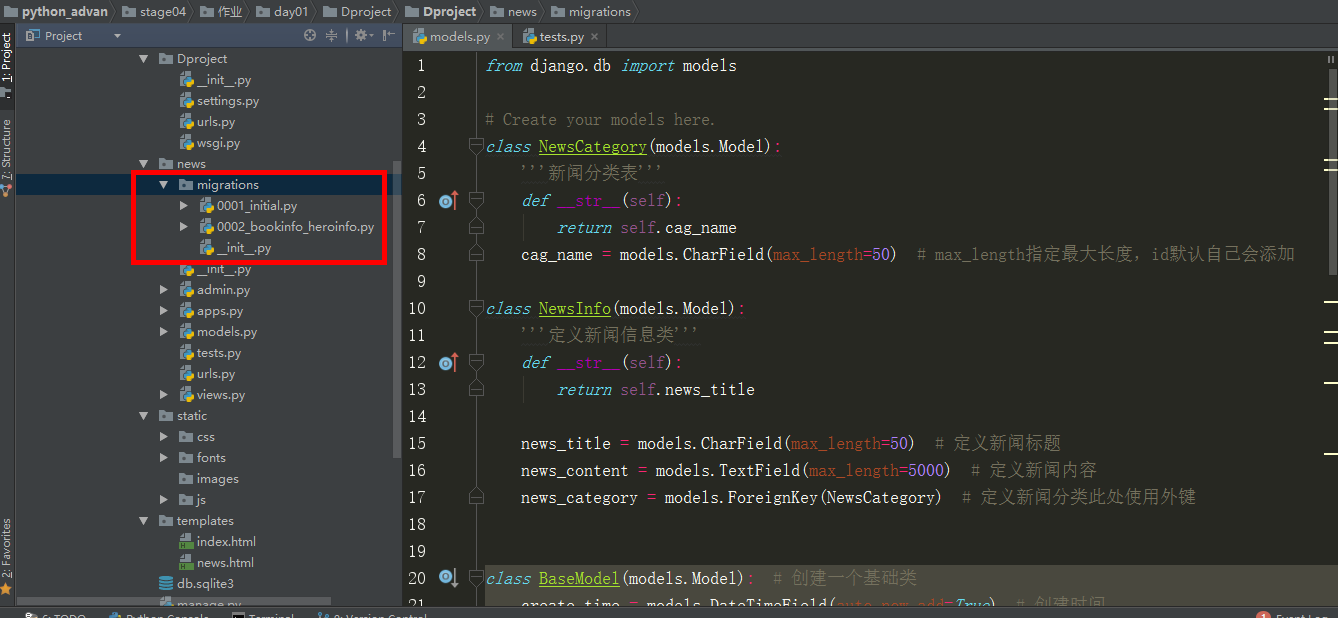
Operations to perform:

Apply all migrations: admin, auth, contenttypes, news, sessions

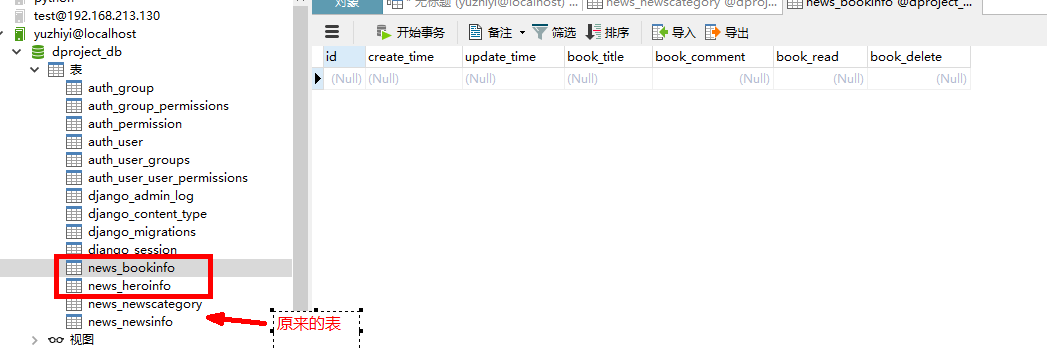
Running migrations:

Applying news.0002\_bookinfo\_heroinfo... OK

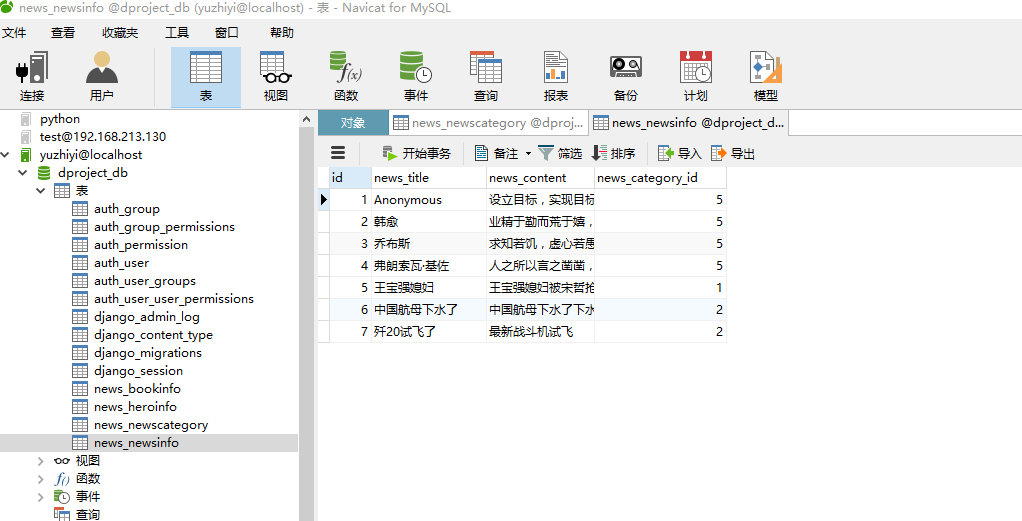
迁移似乎只应用了0002文件，应用完成后 文件还是有2个



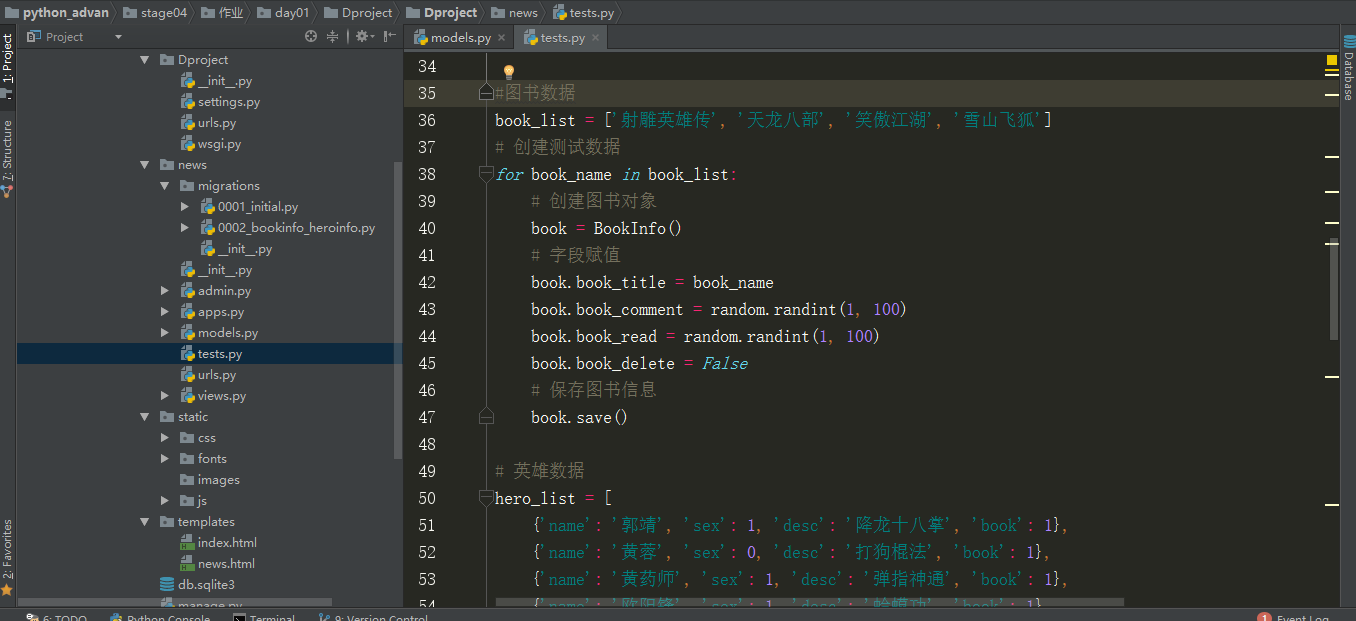
可以看到新表增加了但是对老的表并无影响，表还是在的



那么原来表的数据呢？ 经过查看表的数据也是依然存在的



在tests.py里制造一点数据



源代码如下：

#图书数据  
book\_list = ['射雕英雄传', '天龙八部', '笑傲江湖', '雪山飞狐']  
# 创建测试数据  
*for* book\_name *in* book\_list:  
 # 创建图书对象  
 book = BookInfo()  
 # 字段赋值  
 book.book\_title = book\_name  
 book.book\_comment = random.randint(1, 100)  
 book.book\_read = random.randint(1, 100)  
 book.book\_delete = *False* # 保存图书信息  
 book.save()  
  
# 英雄数据  
hero\_list = [  
 {'name': '郭靖', 'sex': 1, 'desc': '降龙十八掌', 'book': 1},  
 {'name': '黄蓉', 'sex': 0, 'desc': '打狗棍法', 'book': 1},  
 {'name': '黄药师', 'sex': 1, 'desc': '弹指神通', 'book': 1},  
 {'name': '欧阳锋', 'sex': 1, 'desc': '蛤蟆功', 'book': 1},  
 {'name': '梅超风', 'sex': 0, 'desc': '九阴白骨爪', 'book': 1},  
 {'name': '乔峰', 'sex': 1, 'desc': '降龙十八掌', 'book': 2},  
 {'name': '段誉', 'sex': 1, 'desc': '六脉神剑', 'book': 2},  
 {'name': '虚竹', 'sex': 1, 'desc': '天山六阳掌', 'book': 2},  
 {'name': '王语嫣', 'sex': 0, 'desc': '神仙姐姐', 'book': 2},  
 {'name': '令狐冲', 'sex': 1, 'desc': '独孤九剑', 'book': 3},  
 {'name': '任盈盈', 'sex': 0,'desc': '弹琴', 'book': 3},  
 {'name': '岳不群', 'sex': 1, 'desc': '华山剑法', 'book': 3},  
 {'name': '东方不败', 'sex': 1, 'desc': '葵花宝典', 'book': 3},  
 {'name': '胡斐', 'sex': 1, 'desc': '胡家刀法', 'book': 4},  
 {'name': '苗若兰', 'sex': 0, 'desc': '黄衣', 'book': 4},  
 {'name': '程灵素', 'sex': 0, 'desc': '医术', 'book': 4},  
 {'name': '袁紫衣', 'sex': 0, 'desc': '六合拳', 'book': 4},  
]  
  
# 创建测试数据  
*for* hero\_info *in* hero\_list:  
 # 创建英雄类  
 hero = HeroInfo()  
 # 字段赋值  
 hero.hero\_name = hero\_info['name']  
 hero.hero\_desc = hero\_info['desc']  
 hero.hero\_sex = hero\_info['sex']  
 hero.hero\_delete = *False* hero.hero\_book\_id = hero\_info['book']  
 # 保存对象  
 hero.save()

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\

Dproject>**Scripts\python.exe Dproject\manage.py test news**

Creating test database for alias 'default'...

System check identified no issues (0 silenced).

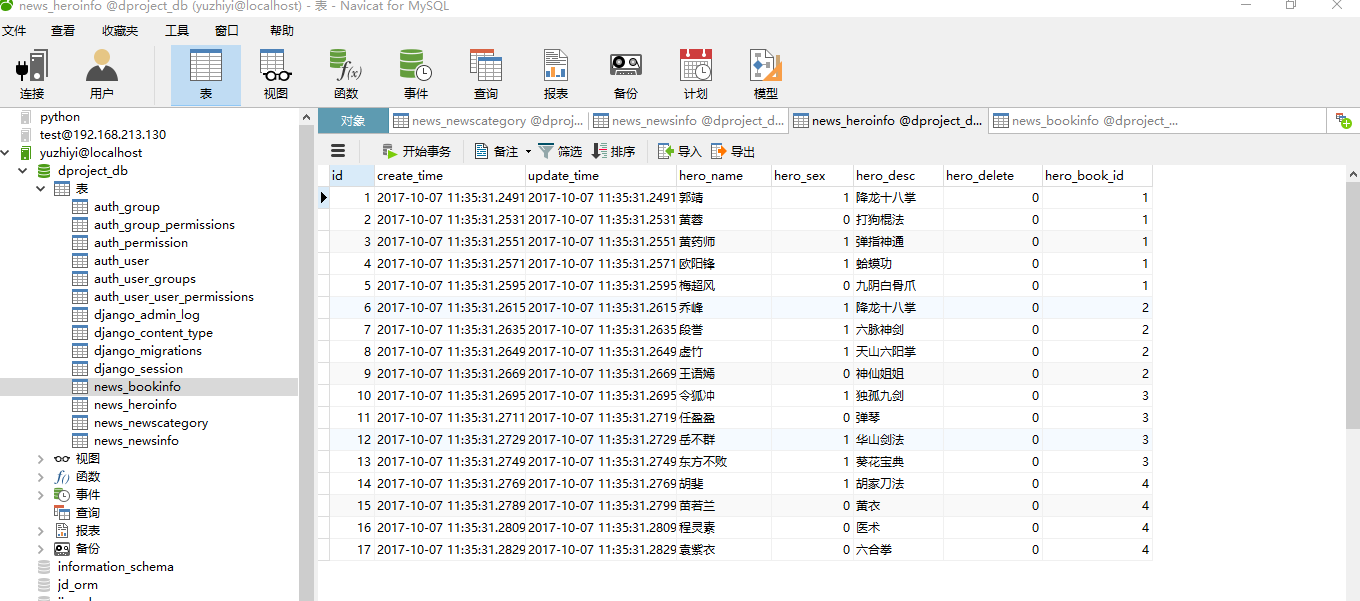
----------------------------------------------------------------------

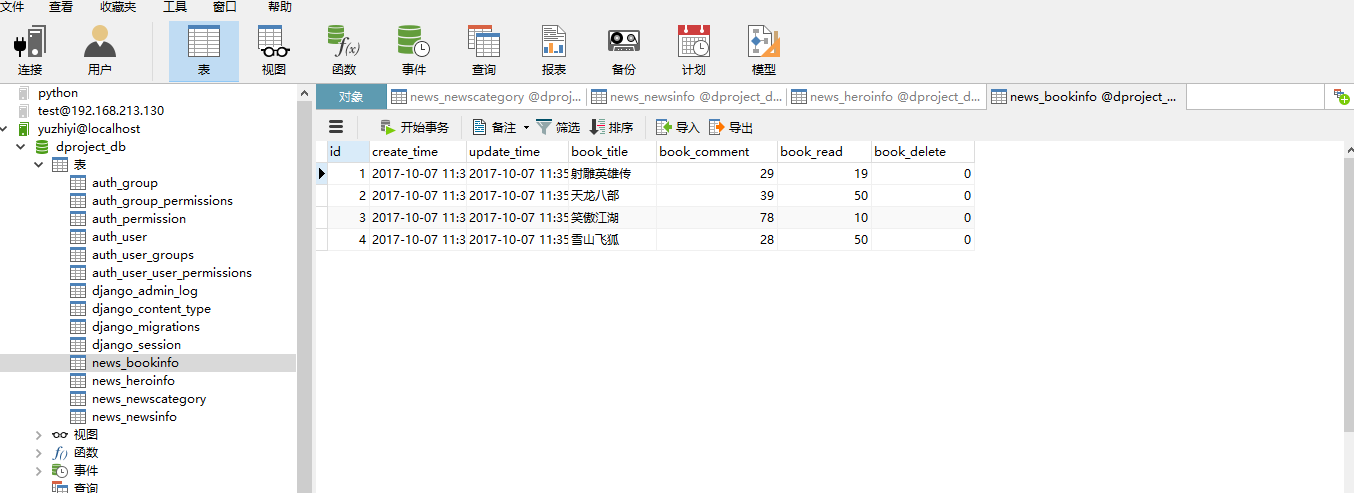
Ran 0 tests in 0.000s

OK

Destroying test database for alias 'default'...

执行了插入测试数据





接下去将做一个级联删除的测试

所谓级联删除就是指将一对多关系中的一删除，多也将跟着一起删除

例如删除笑傲江湖这本书，那么令狐冲、任盈盈、岳不群、东方不败这些英雄人物也将被删除

删除的代码如下

book = BookInfo.objects.get(pk=3) # 获得一本书  
book.delete() # 级联删除

执行测试中的删除语句

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\

Dproject>**Scripts\python.exe Dproject\manage.py test news**

Creating test database for alias 'default'...

System check identified no issues (0 silenced).

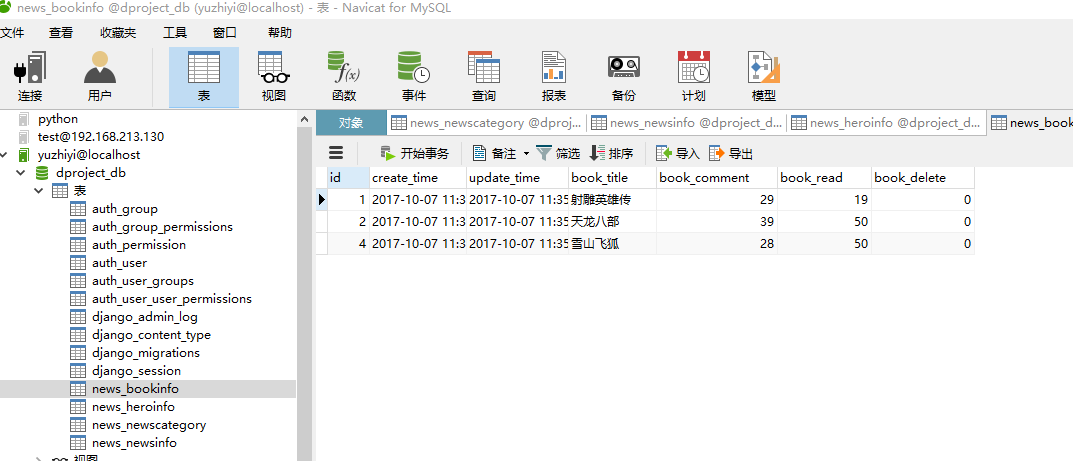
----------------------------------------------------------------------

Ran 0 tests in 0.000s

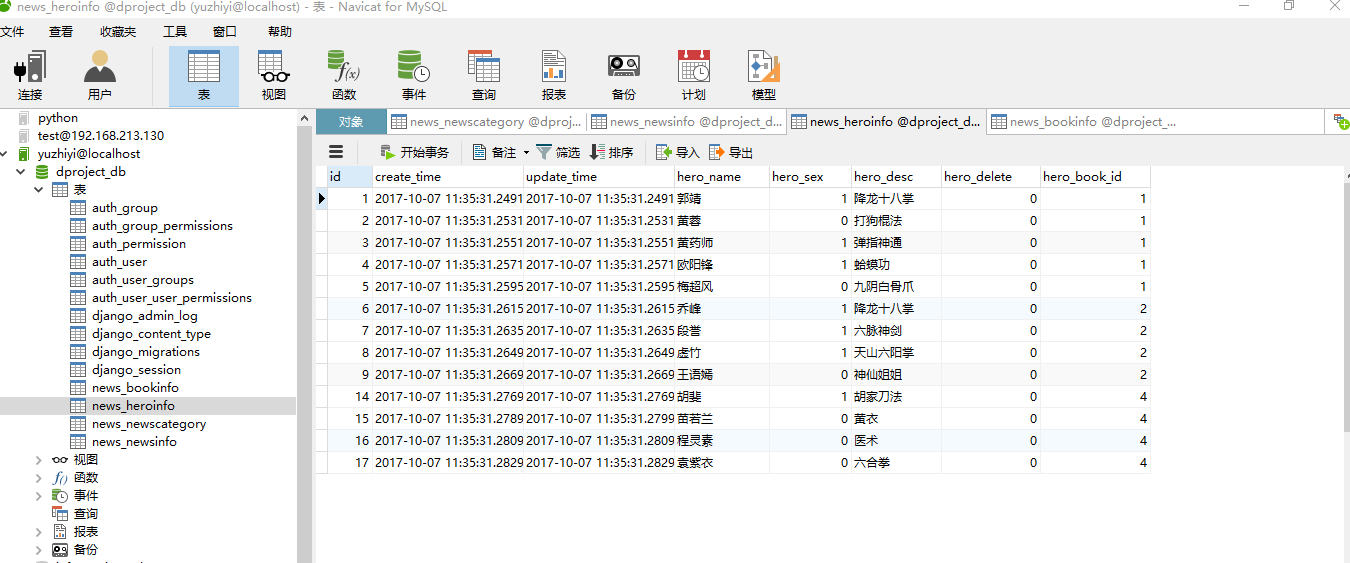
OK

Destroying test database for alias 'default'...

可以看到id为3的笑傲江湖这本书已经被删除了

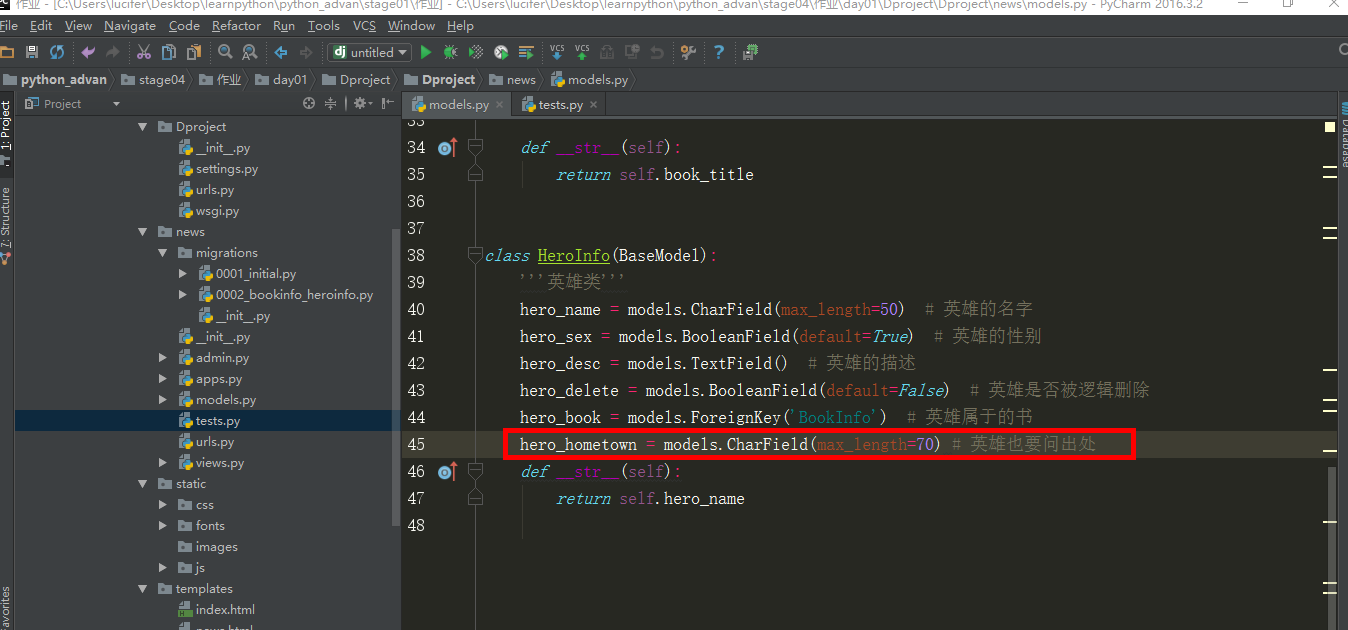


而笑傲江湖小说中的任务也在英雄表中被删除了



* 接下来验证一下增加一个字段对原来的数据有没有影响

我们增加一个hero\_hometown 这样一个字段在heroinfo表里



(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\

Dproject>**Scripts\python.exe Dproject\manage.py makemigrations**

You are trying to add a non-nullable field 'hero\_hometown' to heroinfo without a

default; we can't do that (the database needs something to populate existing ro

ws).

**Please select a fix:**

**1) Provide a one-off default now (will be set on all existing rows with a null**

**value for this column)**

**2) Quit, and let me add a default in models.py**

Select an option: 1

Please enter the default value now, as valid Python

The datetime and django.utils.timezone modules are available, so you can do e.g.

timezone.now

Type 'exit' to exit this prompt

>>> exit

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\

Dproject>**Scripts\python.exe Dproject\manage.py migrate**

Operations to perform:

Apply all migrations: admin, auth, contenttypes, news, sessions

Running migrations:

**No migrations to apply.**

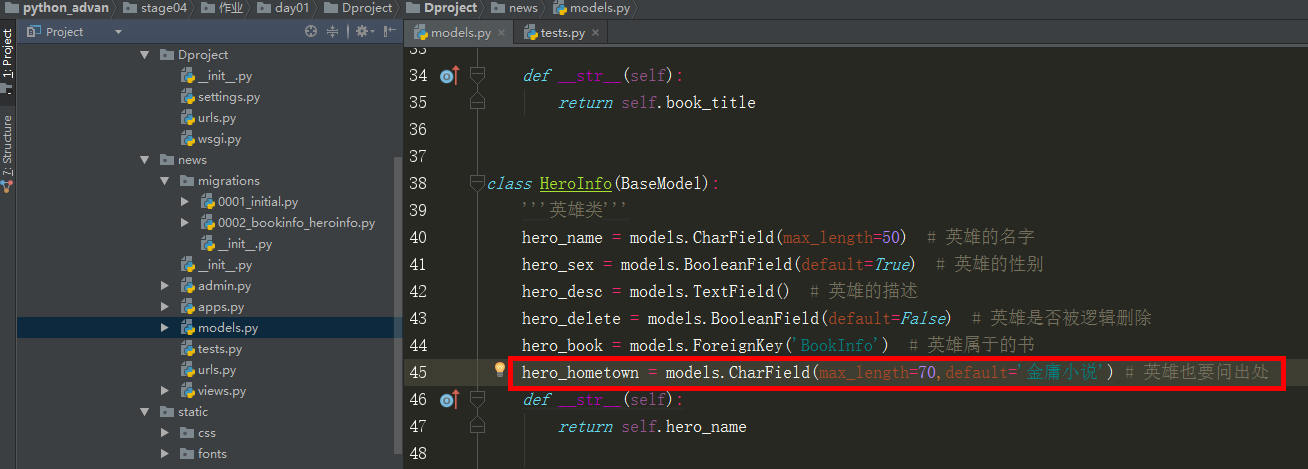
**Your models have changes that are not yet reflected in a migration, and so won**

**'t be applied.**

Run 'manage.py makemigrations' to make new migrations, and then re-run 'manage

.py migrate' to apply them.

以上提示了不允许添加非空字段。看样子这个字段的值要设置成允许非空才行，或者给个默认值。使用一个默认值再试试



源代码如下

hero\_hometown = models.CharField(max\_length=70,default='金庸小说') # 英雄也要问出处

制作迁移文件

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\

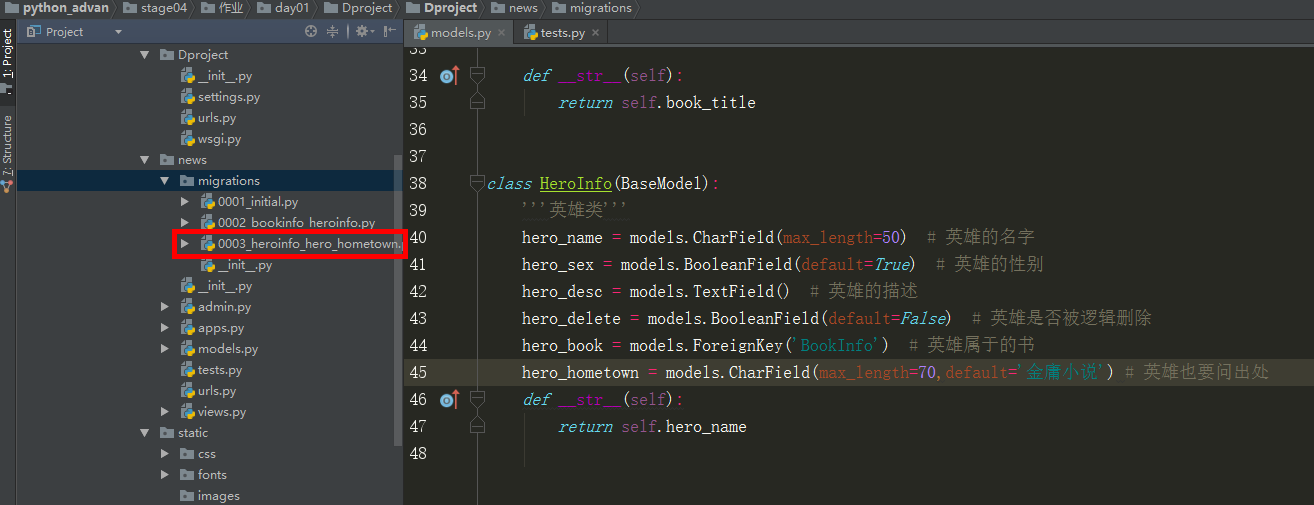
Dproject>**Scripts\python.exe Dproject\manage.py makemigrations**

Migrations for 'news':

**Dproject\news\migrations\0003\_heroinfo\_hero\_hometown.py**

**- Add field hero\_hometown to heroinfo**

多出了一个迁移文件

****

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\

Dproject>**Scripts\python.exe Dproject\manage.py migrate**

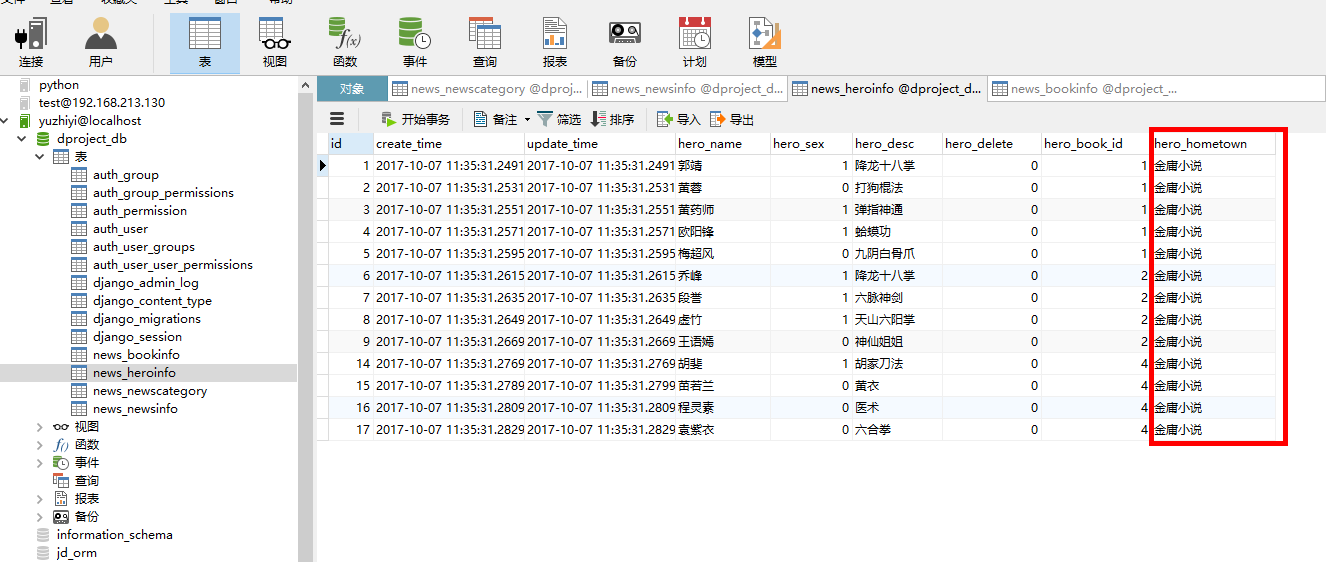
Operations to perform:

Apply all migrations: admin, auth, contenttypes, news, sessions

Running migrations:

Applying news.0003\_heroinfo\_hero\_hometown... OK

设置默认值之后迁移成功了



字段成功添加

为了方便查看数据在admin中将这2张表也注册上



这次使用了装饰器也可以达到相同的效果

源代码如下:

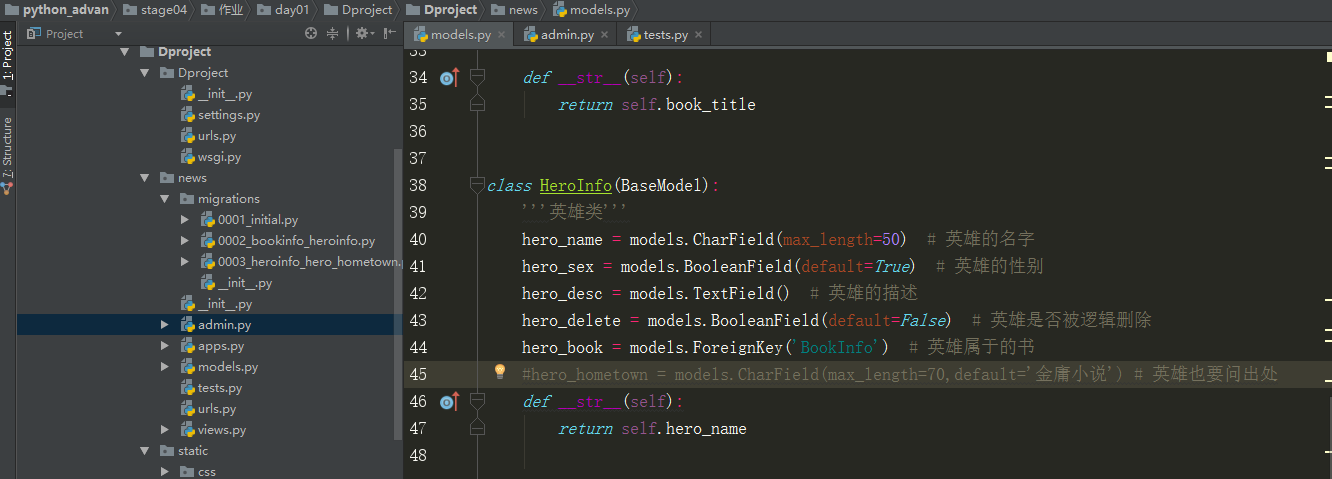
@admin.register(HeroInfo) #另一种注册方法使用装饰器注册  
*class* HeroInfoAdmin(admin.ModelAdmin):  
 list\_display = ['hero\_name','hero\_desc']  
  
@admin.register(BookInfo)  
*class* BookInfoAdmin(admin.ModelAdmin):  
 list\_display = ['book\_title','book\_comment','book\_read']

注册完成后在管理页面就能看到了



接下去要删除一个字段，就用刚才添加的hero\_hometown 做测试

简单的代码里把这个字段注释掉



(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\

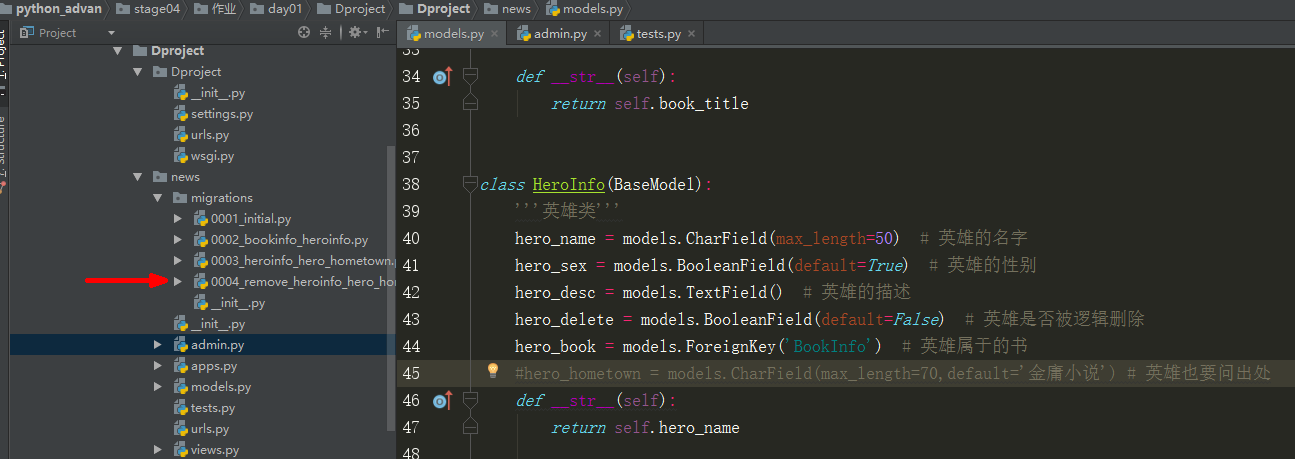
Dproject>**Scripts\python.exe Dproject\manage.py makemigrations**

Migrations for 'news':

**Dproject\news\migrations\0004\_remove\_heroinfo\_hero\_hometown.py**

**- Remove field hero\_hometown from heroinfo**

可以看到生成了一个0004文件

****

文件的内容是 drop column

**(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\**

**Dproject>Scripts\python.exe Dproject\manage.py sqlmigrate news 0004**

**BEGIN;**

**--**

**-- Remove field hero\_hometown from heroinfo**

**--**

**ALTER TABLE `news\_heroinfo` DROP COLUMN `hero\_hometown`;**

**COMMIT;**

进行迁移

**(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\**

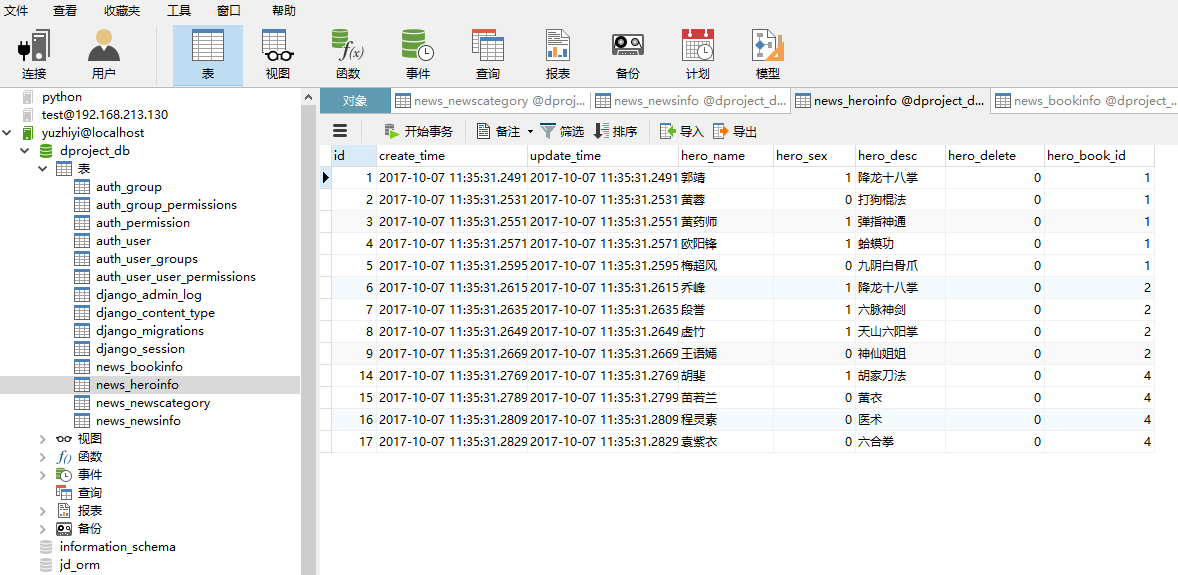
**Dproject>Scripts\python.exe Dproject\manage.py migrate**

**Operations to perform:**

**Apply all migrations: admin, auth, contenttypes, news, sessions**

**Running migrations:**

**Applying news.0004\_remove\_heroinfo\_hero\_hometown... OK**

****

可以看到该列数据已经被删除，并且没有影响其他的列。

在做其他测试时发生了创建测试数据库报错。

然后尝试了删除和hero\_hometown有关的 migrations下的文件0003 0004之后才正常

具体原因不是很清楚，也不清楚为什么会建立测试数据库

有待进一步查证(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject>**Scripts\python.exe Dproject\manage.py test news**

<QuerySet [<HeroInfo: 郭靖>, <HeroInfo: 黄药师>, <HeroInfo: 欧阳锋>, <HeroInfo: 乔峰>, <HeroInfo: 段誉>, <HeroInfo: 虚竹>, <HeroInfo: 胡斐>]>

Creating test database for alias 'default'...

Got an error creating the test database: (1007, "Can't create database 'test\_dproject\_db'; database exists")

Type 'yes' if you would like to try deleting the test database 'test\_Dproject\_DB', or 'no' to cancel: no

Tests cancelled.

。

。

。

first\_packet = self.connection.\_read\_packet()

File "C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject\lib\site-packages\pymysql\connections.py", line 1014, in \_read\_packet

packet.check\_error()

File "C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject\lib\site-packages\pymysql\connections.py", line 393, in check\_error

err.raise\_mysql\_exception(self.\_data)

File "C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject\lib\site-packages\pymysql\err.py", line 107, in raise\_mysql\_exception

raise errorclass(errno, errval)

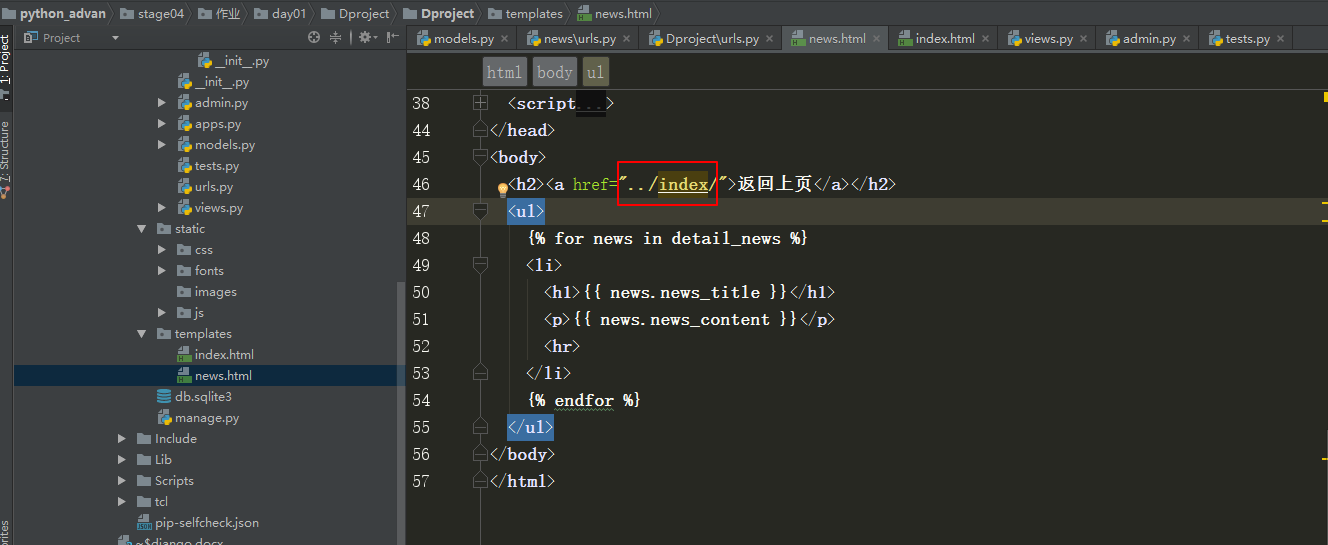
**django.db.utils.InternalError: (1067, "Invalid default value for 'hero\_hometown'")**

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject>

# 反向URL解析的测试

在课堂中老师讲解了反向url解析的概念，但是对这个概念依然很模糊

因此打算做一个实现来测试



可以看到在新闻详细页里的返回主页的连接里，采用了硬编码，写死了url 。

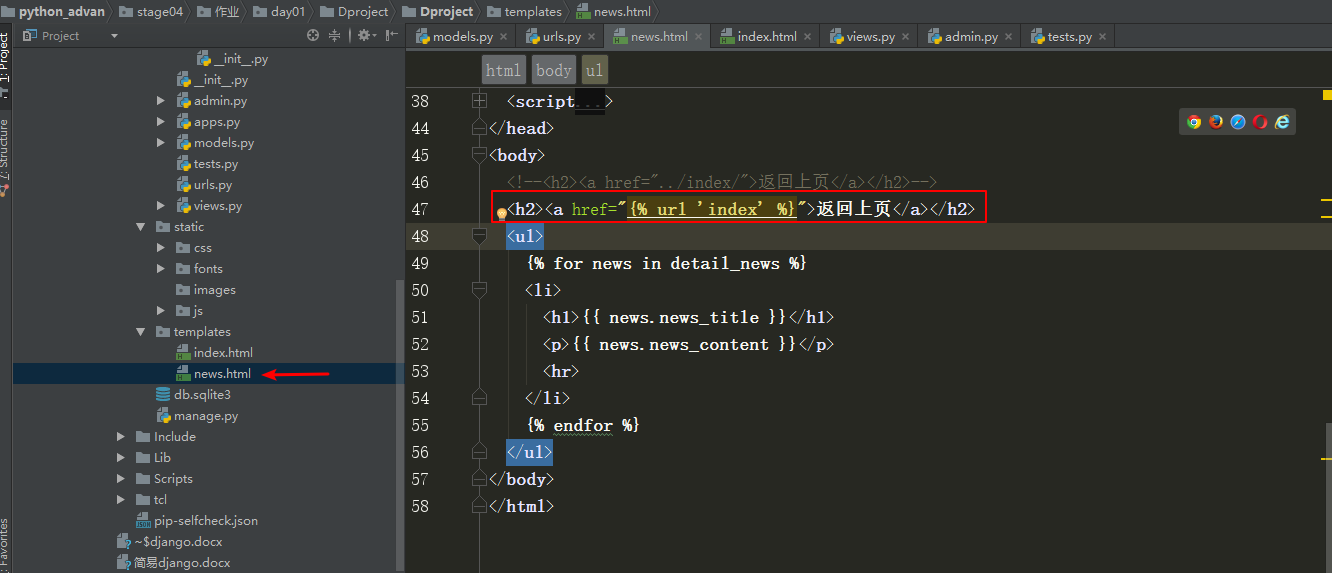
因此如果修改urlpattern的匹配方式的话，这个连接返回就会失效。

因此尝试反向url解析

想达到的效果是：如果urlpattern 里的匹配规则从 /new/index/ 变为 /new/index.html

而模板是不需要修改任何代码的。

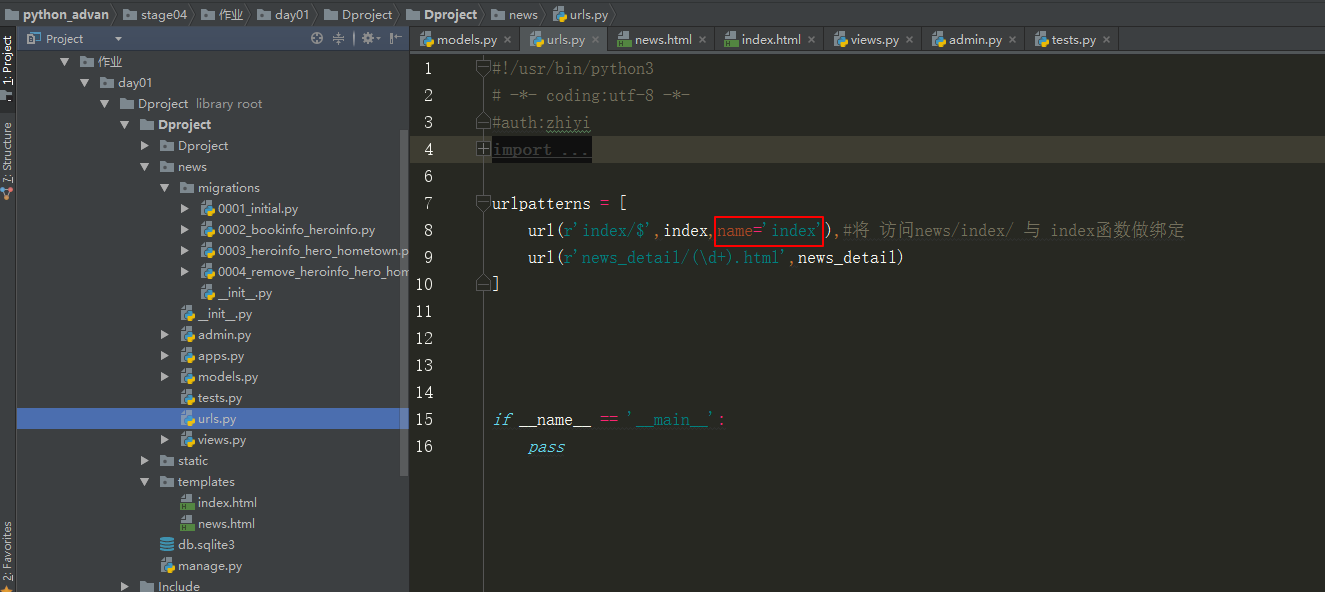
首先修改news.html页面



源代码如下:

<**h2**><**a** href=**"{% url 'index' %}"**>**返回上页**</**a**></**h2**>

在urls.py中为相应的页面设置一个name=index的参数 和 模板页面中的名字匹配



源代码如下：

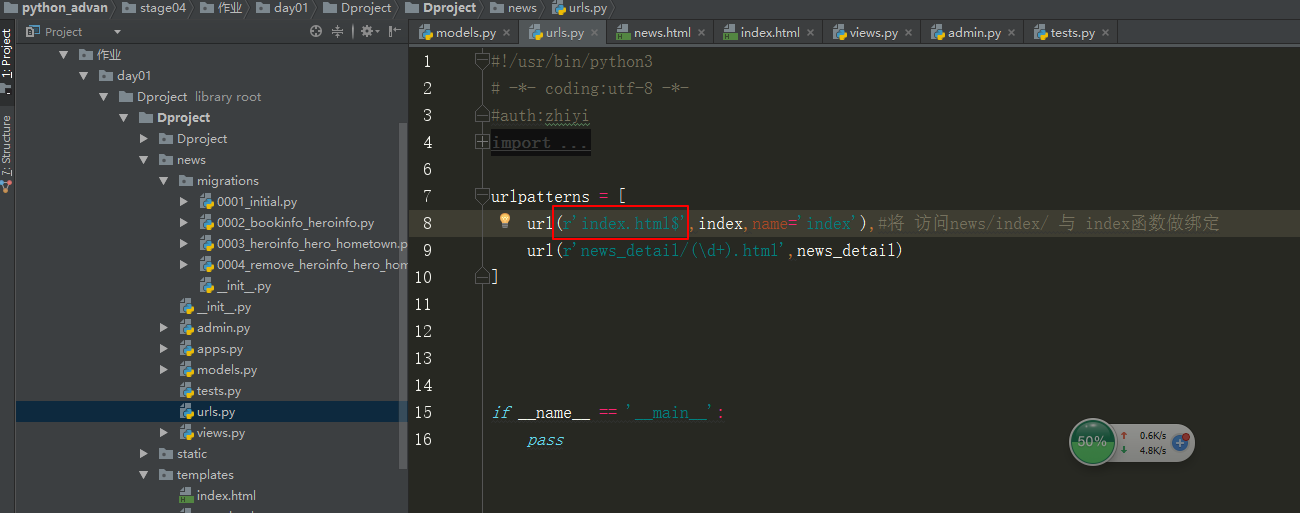
url(r'index/$',index,name='index'),





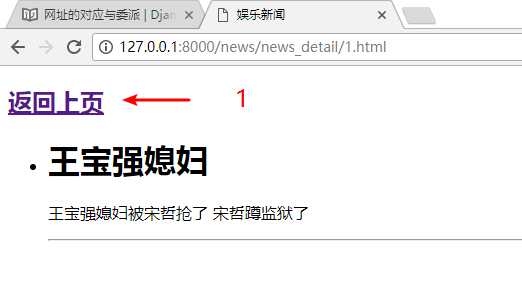
此时网页返回的功能完全正常。

接下来将 /news/index/ 改成 /news.index.html 看看会发生什么变化



源代码如下：

url(r'index.html$',index,name='index'),



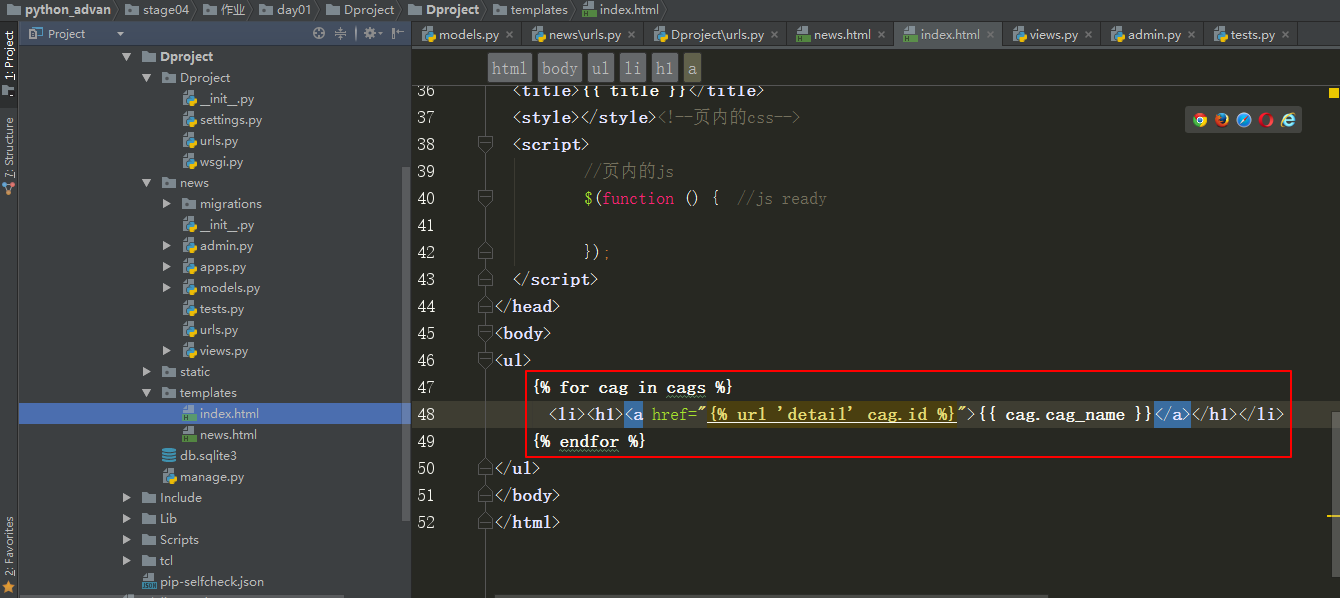


此时可以看到 回到首页的url地址已经从 news/index/ 变成了 news/index.html

而模板文件因为使用了名字，所以没有更改还是使用

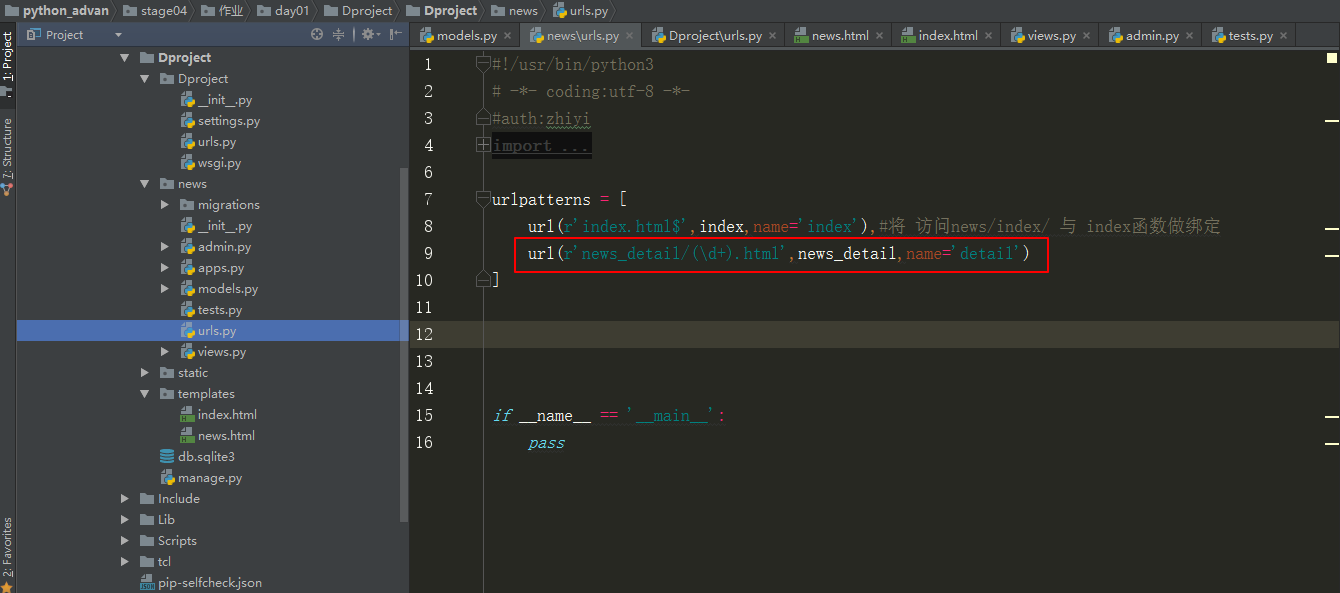
<**h2**><**a** href=**"{% url 'index' %}"**>**返回上页**</**a**></**h2**>

进一步做了主页上生成页面的反向解析



源代码如下:

<**ul**>  
 **{% for cag in cags %}** <**li**><**h1**><**a** href=**"{% url 'detail' cag.id %}"**>**{{ cag.cag\_name }}**</**a**></**h1**></**li**>  
 **{% endfor %}**</**ul**>



源代码如下：

urlpatterns = [  
 url(r'index.html$',index,name='index'),#将 访问news/index/ 与 index函数做绑定  
 url(r'news\_detail/(\d+).html',news\_detail,name='detail')  
]

更改完成后尝试更改路由

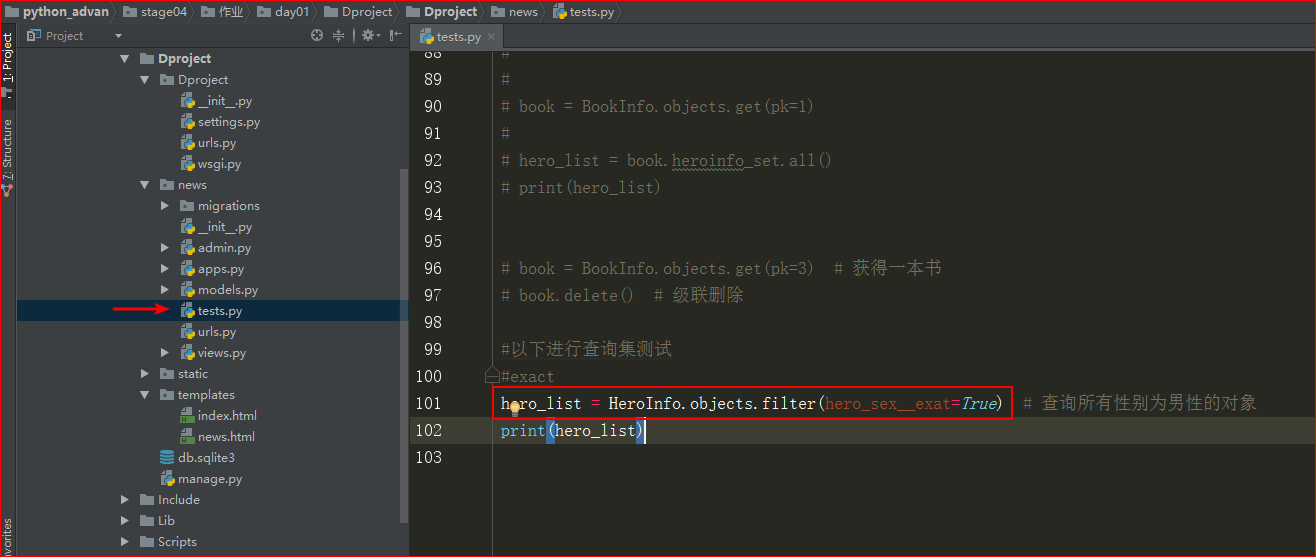




可以看到只要一改变路由，网站的网址也跟着一起改变了

# 查询集

以下的测试代码都将建立在 test.py中



* exact

#exact  
hero\_list = HeroInfo.objects.filter(hero\_sex\_\_exat=*True*) # 查询所有性别为男性的对象  
print(hero\_list)

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject>**Scripts\python.exe Dproject\manage.py test news**

**<QuerySet [<HeroInfo: 郭靖>, <HeroInfo: 黄药师>, <HeroInfo: 欧阳锋>, <HeroInfo: 乔峰>, <HeroInfo: 段誉>, <HeroInfo: 虚竹>, <HeroInfo: 胡斐>]>**

Creating test database for alias 'default'...

Got an error creating the test database: (1007, "Can't create database 'test\_dproject\_db'; database exists")

Type 'yes' if you would like to try deleting the test database 'test\_Dproject\_DB', or 'no' to cancel: no

Tests cancelled.

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject>



可以看到与查询结果一致

* contains

#contains  
hero\_list = HeroInfo.objects.filter(hero\_desc\_\_contains='八') #查询描述中带有8的英雄  
print(hero\_list)

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject>Scripts\python.exe Dproject\manage.py test news

<QuerySet [<HeroInfo: 郭靖>, <HeroInfo: 乔峰>]>

Creating test database for alias 'default'...

System check identified no issues (0 silenced).

----------------------------------------------------------------------

Ran 0 tests in 0.001s

OK

Destroying test database for alias 'default'...

* startswith endswith

#startswith endswith  
hero\_list = HeroInfo.objects.filter(hero\_desc\_\_startswith='六') # 查询以六开头的英雄描述  
print([ hero.hero\_desc *for* hero *in* hero\_list])

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject>Scripts\python.exe Dproject\manage.py test news

['六脉神剑', '六合拳']

* in

#in  
hero\_list = HeroInfo.objects.filter(pk\_\_in=[1,3,5])  
print(hero\_list)

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject>Scripts\python.exe Dproject\manage.py test news

<QuerySet [<HeroInfo: 郭靖>, <HeroInfo: 黄药师>, <HeroInfo: 梅超风>]>

* gt gte lt lte

# gt gte lt lte  
hero\_list = HeroInfo.objects.filter(pk\_\_gt=3)  
print(hero\_list)

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject>Scripts\python.exe Dproject\manage.py test news

<QuerySet [<HeroInfo: 欧阳锋>, <HeroInfo: 梅超风>, <HeroInfo: 乔峰>, <HeroInfo: 段誉>, <HeroInfo: 虚竹>, <HeroInfo: 王语嫣>, <HeroInfo: 胡斐>, <HeroInfo: 苗若兰>, <HeroInfo: 程灵素>, <HeroInfo: 袁紫衣>]>

* exclude

# exclude　按照条件反方向查找  
hero\_list = HeroInfo.objects.exclude(pk\_\_gt=3)  
print(hero\_list)

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject>Scripts\python.exe Dproject\manage.py test news

<QuerySet [<HeroInfo: 郭靖>, <HeroInfo: 黄蓉>, <HeroInfo: 黄药师>]>

* get

# get  
*try*:  
 book = BookInfo.objects.get(pk=20)  
*except* BookInfo.DoesNotExist:  
 book = BookInfo()  
 book.id = 20  
 book.book\_read = random.randint(1, 100)  
 book.book\_comment = random.randint(1, 100)  
 book.book\_title = '大话西游'  
 book.save()

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject>Scripts\python.exe Dproject\manage.py test news

Creating test database for alias 'default'...

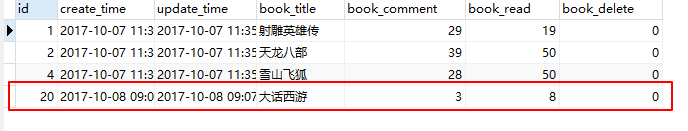
System check identified no issues (0 silenced).

----------------------------------------------------------------------

Ran 0 tests in 0.000s

OK

Destroying test database for alias 'default'...



* F对象 进行同一条数据间不同字段的比较

# F对象  
book\_list = BookInfo.objects.filter(book\_comment\_\_gte=models.F('book\_read'))  
print(book\_list) # 查询评论数比阅读量大的书  
  
  
book\_list = BookInfo.objects.filter(book\_comment\_\_gte=models.F('book\_read')\*2)  
print(book\_list) # 查询评论数大于阅读量两倍书

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject>Scripts\python.exe Dproject\manage.py test news

<QuerySet [<BookInfo: 射雕英雄传>]>

<QuerySet []>

* Q对象

# Q对象  
book\_list = BookInfo.objects.filter(book\_read\_\_gt=20).filter(pk\_\_gt=2)  
print(book\_list) #查询阅读量大于20且主键大于2的书

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject>Scripts\python.exe Dproject\manage.py test news

<QuerySet [<BookInfo: 雪山飞狐>]>

# Ｑ对象支持　&(与) |(或)  
book\_list = BookInfo.objects.filter( models.Q(book\_read\_\_gt=20) & models.Q(pk\_\_gt=2) )  
print(book\_list) #查询图书的阅读量大于20并且id大于2的书  
  
book\_list = BookInfo.objects.filter( models.Q(book\_read\_\_gt=20) | models.Q(pk\_\_gt=2) )  
print(book\_list) #查询图书的阅读量大于20或者id大于2的书

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject>Scripts\python.exe Dproject\manage.py test news

<QuerySet [<BookInfo: 雪山飞狐>]>

<QuerySet [<BookInfo: 天龙八部>, <BookInfo: 雪山飞狐>, <BookInfo: 大话西游>]>

* 关联查询

# 通过模型类实现关联查询  
book\_list = BookInfo.objects.filter(heroinfo\_\_hero\_desc\_\_contains='八')  
print(book\_list) # 查询英雄的描述中含有八的图书记录

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject>Scripts\python.exe Dproject\manage.py test news

<QuerySet [<BookInfo: 射雕英雄传>, <BookInfo: 天龙八部>]>

book\_list = HeroInfo.objects.filter(hero\_book\_\_book\_title='天龙八部')  
print(book\_list) #查询属于天龙八部这本书的所有英雄

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject>Scripts\python.exe Dproject\manage.py test news

<QuerySet [<HeroInfo: 乔峰>, <HeroInfo: 段誉>, <HeroInfo: 虚竹>, <HeroInfo: 王语嫣>]>

* 结果集排序

# 结果集排序  
hero\_list = HeroInfo.objects.order\_by('-id')  
print(hero\_list) # 按照ID倒序排列结果

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject>Scripts\python.exe Dproject\manage.py test news

<QuerySet [<HeroInfo: 袁紫衣>, <HeroInfo: 程灵素>, <HeroInfo: 苗若兰>, <HeroInfo: 胡斐>, <HeroInfo: 王语嫣>, <HeroInfo: 虚竹>, <HeroInfo: 段誉>, <HeroInfo: 乔峰>, <HeroInfo: 梅超风>, <HeroInfo: 欧阳锋>, <HeroInfo: 黄药师>, <HeroInfo: 黄蓉>, <HeroInfo: 郭靖>]>

hero\_list = HeroInfo.objects.filter(id\_\_gt=2).order\_by('-id')  
print(hero\_list) #设置筛选条件的倒序

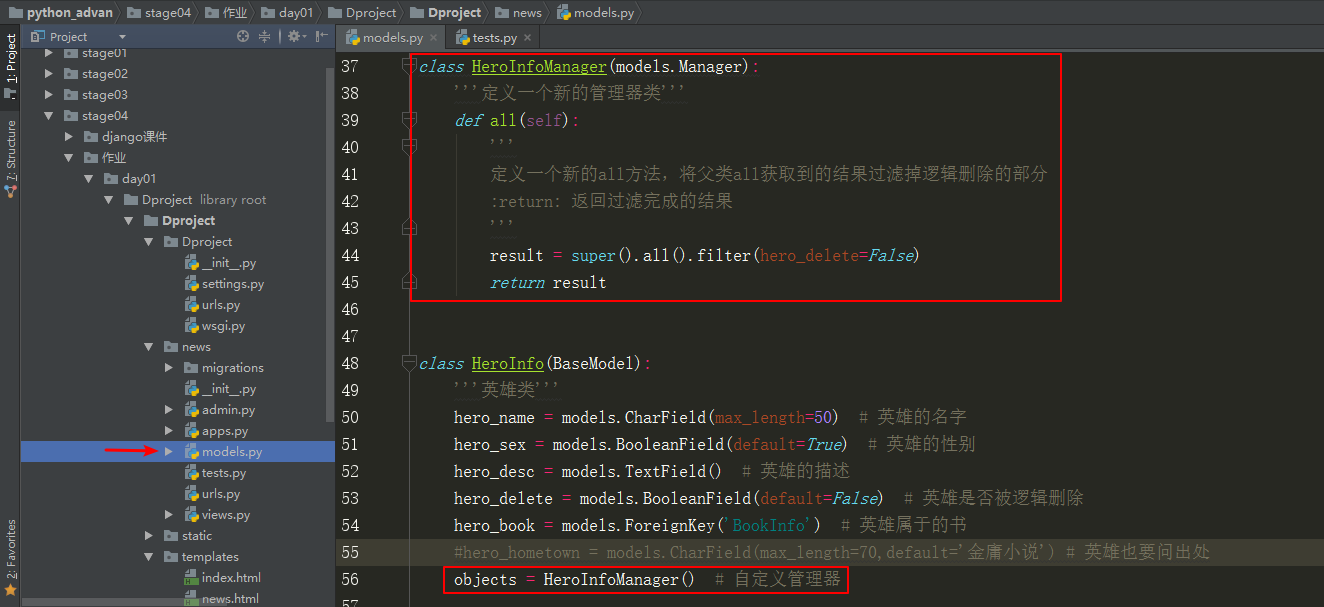
(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject>Scripts\python.exe Dproject\manage.py test news

<QuerySet [<HeroInfo: 袁紫衣>, <HeroInfo: 程灵素>, <HeroInfo: 苗若兰>, <HeroInfo: 胡斐>, <HeroInfo: 王语嫣>, <HeroInfo: 虚竹>, <HeroInfo: 段誉>, <HeroInfo: 乔峰>, <HeroInfo: 梅超风>, <HeroInfo: 欧阳锋>, <HeroInfo: 黄药师>]>

# 自定义管理器

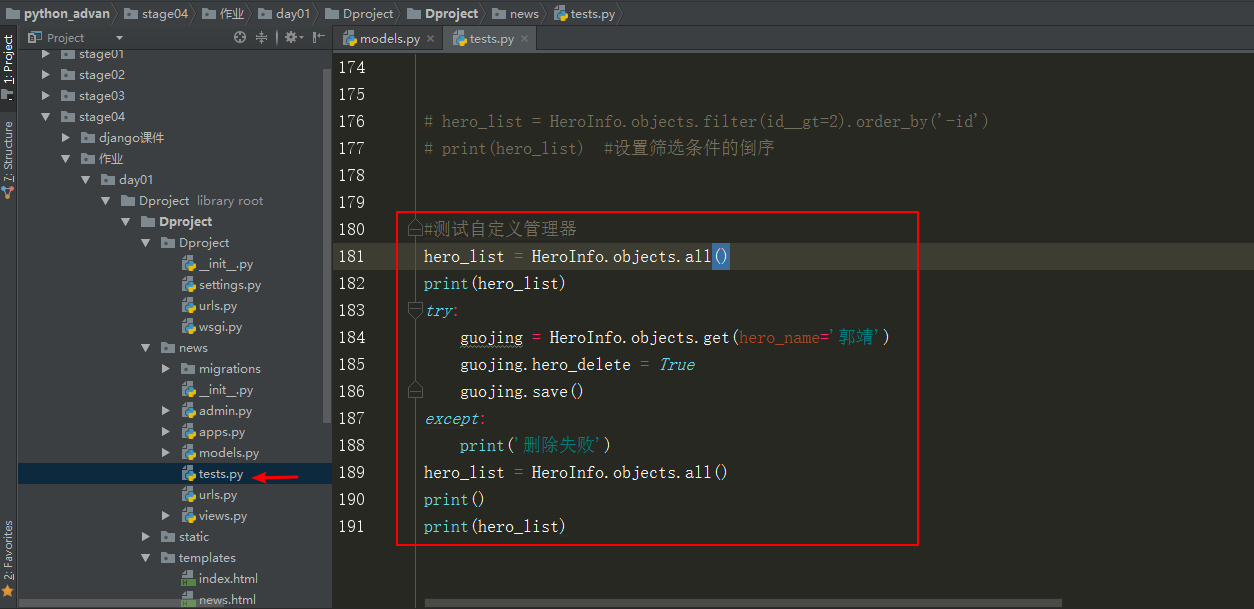
在models.py 中新定义一个HeroInfoManager类，该类集成models.Manager类

并在HeroInfo中 指定自定义管理器类



源代码如下：

*class* HeroInfoManager(models.Manager):  
 '''定义一个新的管理器类'''  
 *def* all(self):  
 '''  
 定义一个新的all方法，将父类all获取到的结果过滤掉逻辑删除的部分  
 :return: 返回过滤完成的结果  
 '''  
 result = super().all().filter(hero\_delete=*False*)  
 *return* result  
  
  
*class* HeroInfo(BaseModel):  
 '''英雄类'''  
 hero\_name = models.CharField(max\_length=50) # 英雄的名字  
 hero\_sex = models.BooleanField(default=*True*) # 英雄的性别  
 hero\_desc = models.TextField() # 英雄的描述  
 hero\_delete = models.BooleanField(default=*False*) # 英雄是否被逻辑删除  
 hero\_book = models.ForeignKey('BookInfo') # 英雄属于的书  
 #hero\_hometown = models.CharField(max\_length=70,default='金庸小说') # 英雄也要问出处  
 objects = HeroInfoManager() # 自定义管理器

在测试文件中 编写一个测试，先打印出所有英雄，然后获取郭靖这个英雄，设置英雄的

hero\_delete为True，此时只是在数据库的hero\_delete字段的值从0改变到1

并没有再数据库中删除记录



(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject>Scripts\python.exe Dproject\manage.py test news

<QuerySet [**<HeroInfo: 郭靖>**, <HeroInfo: 黄蓉>, <HeroInfo: 黄药师>, <HeroInfo: 欧阳锋>, <HeroInfo: 梅超风>, <HeroInfo: 乔峰>, <HeroInfo: 段誉>, <HeroInfo: 虚竹>, <HeroInfo: 王语嫣>, <HeroInfo: 胡斐>, <HeroInfo: 苗若兰>, <HeroInfo: 程灵素>, <HeroInfo: 袁紫衣>]>

<QuerySet [<HeroInfo: 黄蓉>, <HeroInfo: 黄药师>, <HeroInfo: 欧阳锋>, <HeroInfo: 梅超风>, <HeroInfo: 乔峰>, <HeroInfo: 段誉>, <HeroInfo: 虚竹>, <HeroInfo: 王语嫣>, <HeroInfo: 胡斐>, <HeroInfo: 苗若兰>, <HeroInfo: 程灵素>, <HeroInfo: 袁紫衣>]>

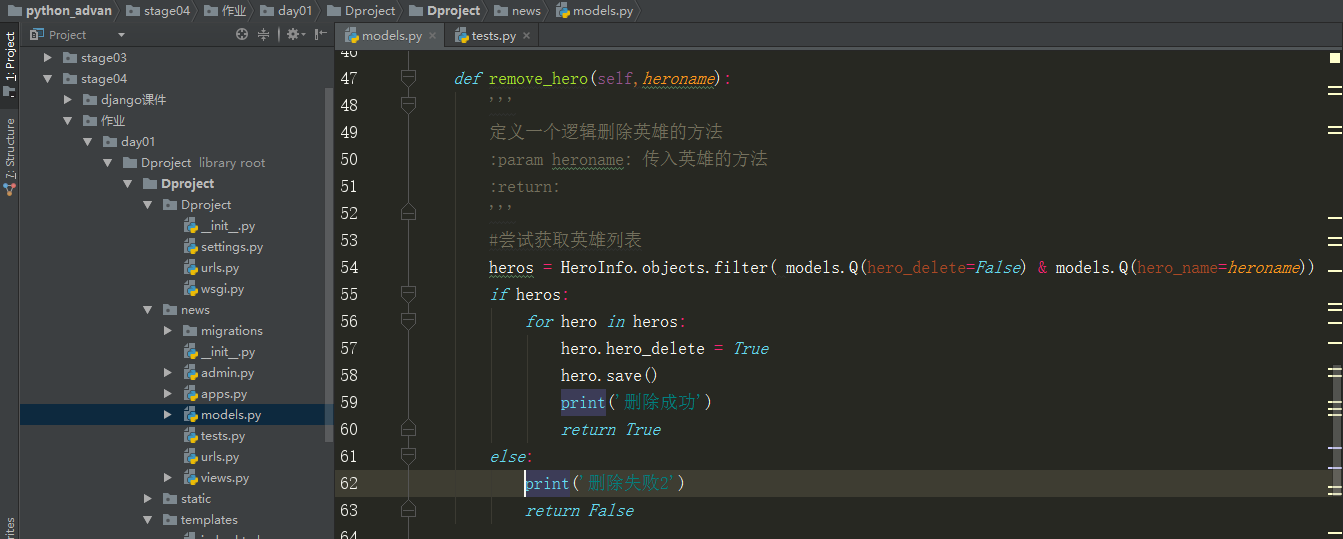
Creating test database for alias 'default'...

System check identified no issues (0 silenced).

但是由于使用了新的管理器，并改写了all方法。可以看到删除前查询郭靖还是存在的

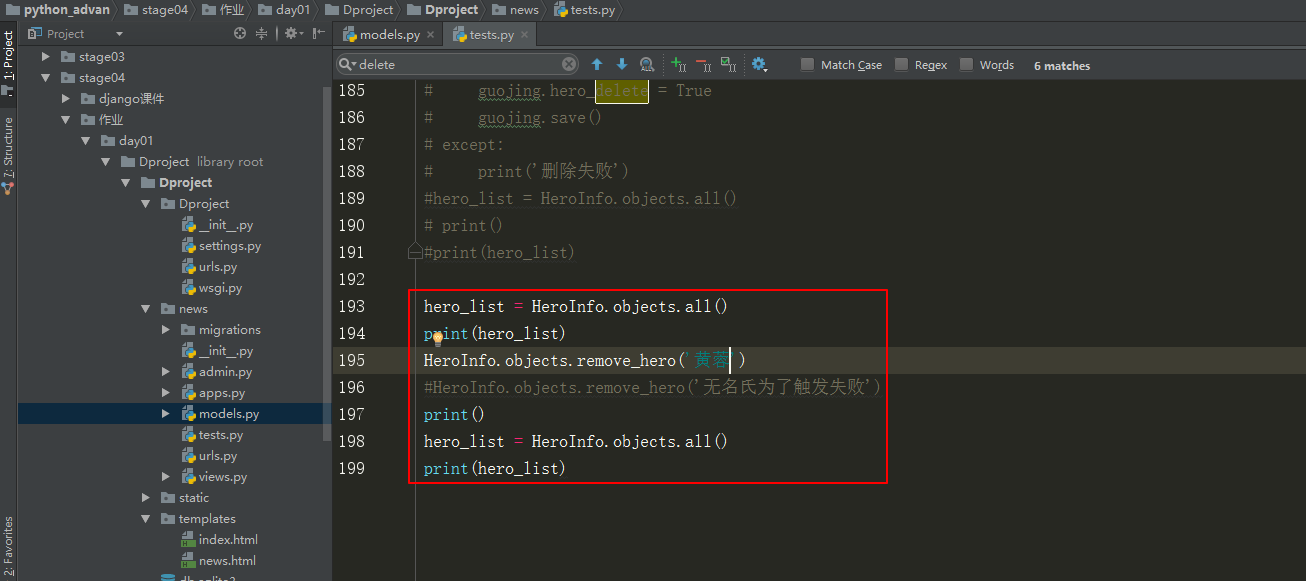
执行删除后 ，记录里就没有郭靖了 黄蓉变成了第一个

进一步改写HeroInfoManager的delete方法，以改变删除时的行为



源代码如下：

*def* remove\_hero(self,*heroname*):  
 '''  
 定义一个逻辑删除英雄的方法  
 :param heroname: 传入英雄的方法  
 :return:  
 '''  
 #尝试获取英雄列表  
 heros = HeroInfo.objects.filter( models.Q(hero\_delete=*False*) & models.Q(hero\_name=*heroname*))  
 *if* heros:  
 *for* hero *in* heros:  
 hero.hero\_delete = *True* hero.save()  
 print('删除成功')  
 *return True  
 else*:  
 print('删除失败2')  
 *return False*



测试代码如下：

hero\_list = HeroInfo.objects.all()  
print(hero\_list)  
HeroInfo.objects.remove\_hero('黄蓉')  
#HeroInfo.objects.remove\_hero('无名氏为了触发失败')  
print()  
hero\_list = HeroInfo.objects.all()  
print(hero\_list)

结果如下

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject>Scripts\python.exe Dproject\manage.py test news

<QuerySet [<HeroInfo: 郭靖>, **<HeroInfo: 黄蓉>**, <HeroInfo: 黄药师>, <HeroInfo: 欧阳锋>, <HeroInfo: 梅超风>, <HeroInfo: 乔峰>, <HeroInfo: 段誉>, <HeroInfo: 虚竹>, <HeroInfo: 王语嫣>, <HeroInfo: 胡斐>, <HeroInfo: 苗若兰>, <HeroInfo: 程灵素>, <HeroInfo: 袁紫衣>]>

**删除成功**

<QuerySet [<HeroInfo: 郭靖>, <HeroInfo: 黄药师>, <HeroInfo: 欧阳锋>, <HeroInfo: 梅超风>, <HeroInfo: 乔峰>, <HeroInfo: 段誉>, <HeroInfo: 虚竹>, <HeroInfo: 王语嫣>, <HeroInfo: 胡斐>, <HeroInfo: 苗若兰>, <HeroInfo: 程灵素>, <HeroInfo: 袁紫衣>]>

Creating test database for alias 'default'...

System check identified no issues (0 silenced).

----------------------------------------------------------------------

Ran 0 tests in 0.000s

OK

Destroying test database for alias 'default'...

(Dproject) C:\Users\lucifer\Desktop\learnpython\python\_advan\stage04\作业\day01\Dproject>Scripts\python.exe Dproject\manage.py test news

<QuerySet [<HeroInfo: 郭靖>, <HeroInfo: 黄药师>, <HeroInfo: 欧阳锋>, <HeroInfo: 梅超风>, <HeroInfo: 乔峰>, <HeroInfo: 段誉>, <HeroInfo: 虚竹>, <HeroInfo: 王语嫣>, <HeroInfo: 胡斐>, <HeroInfo: 苗若兰>, <HeroInfo: 程灵素>, <HeroInfo: 袁紫衣>]>

**删除失败2**

<QuerySet [<HeroInfo: 郭靖>, <HeroInfo: 黄药师>, <HeroInfo: 欧阳锋>, <HeroInfo: 梅超风>, <HeroInfo: 乔峰>, <HeroInfo: 段誉>, <HeroInfo: 虚竹>, <HeroInfo: 王语嫣>, <HeroInfo: 胡斐>, <HeroInfo: 苗若兰>, <HeroInfo: 程灵素>, <HeroInfo: 袁紫衣>]>

Creating test database for alias 'default'...

System check identified no issues (0 silenced).

可以看到第一次删除成功，如果在删除一次就因为已经删除而失败