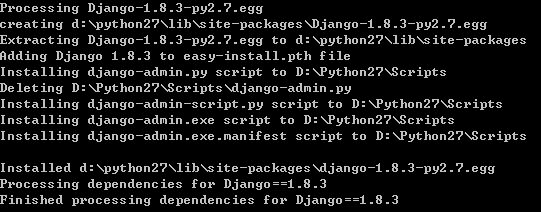
安装django1.8.3

首先安装python-2.7.10.msi，勾选全部；

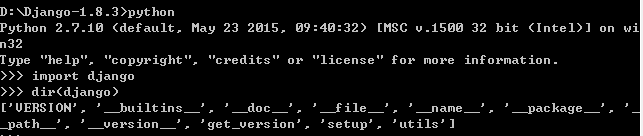
然后下载Django-1.8.3.tar.gz，在linux下tar xf，再传至win8，放到D盘根目录下；

打开cmd，执行“d:”进入D盘根目录，执行“cd Django-1.8.3”进入django命令，执行“python setup.py install”，安装Django-1.8.3



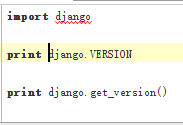


Django安装成功

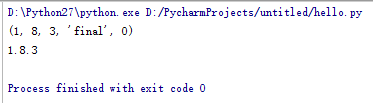


在pycharm新建python file，写入一些代码：

**import** django  
  
**print** django.VERSION  
  
**print** django.get\_version()



按“Ctrl+Shift+F10”,运行结果：

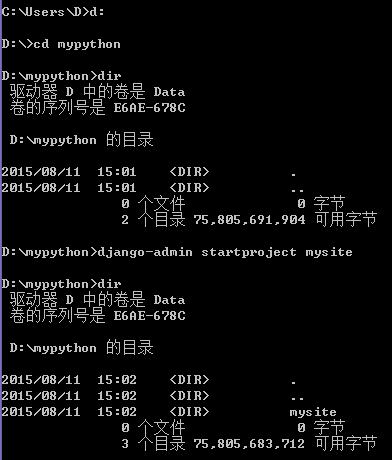


**django.http.HttpResponse**

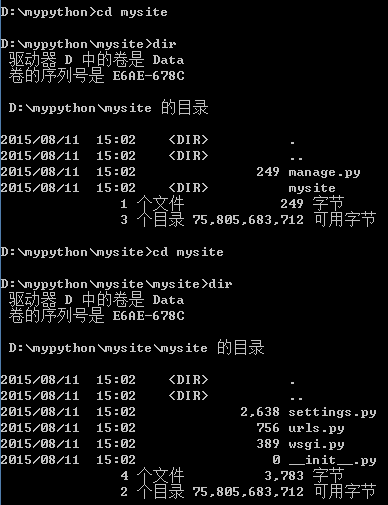
第一个project和app

新建一个project

django-admin startproject mysite



执行命令后，新建了一个 mysite 目录，其中还有一个 mysite 目录，这个子目录 mysite 中是一些项目的设置 settings.py 文件，总的urls配置文件 urls.py 以及部署服务器时用到的 wsgi.py 文件， \_\_init\_\_.py 是python包的目录结构必须的，与调用有关。



我们到外层那个 mysite 目录下(不是mysite中的mysite目录)，新建一个app名称为learn

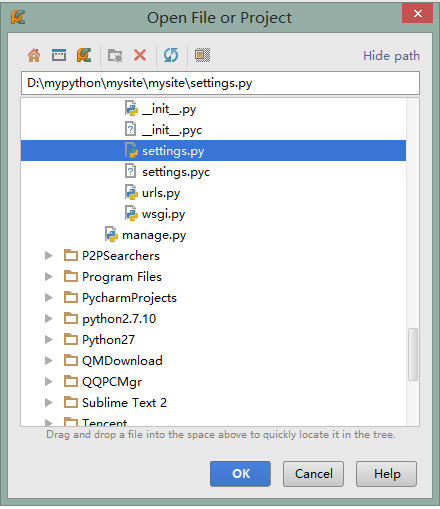
python manage.py startapp learn



看到mysite中多个一个 learn 文件夹，其中有以下文件



把我们新定义的app加到settings.py中的INSTALL\_APPS中



在这里



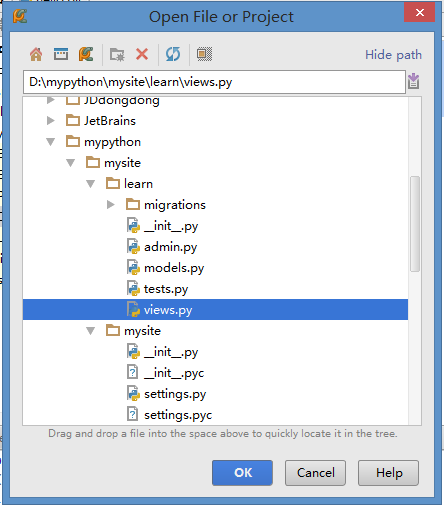
加上”’learn‘,“



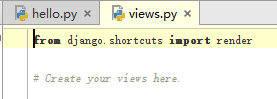
备注,这一步是干什么呢? 新建的 app 如果不加到 INSTALL\_APPS 中的话, django 就不能自动找到app中的模板文件(app-name/templates/下的文件)和静态文件(app-name/static/中的文件) , 后面你会学习到它们分别用来干什么.

**定义视图函数**

我们在learn这个目录中,把views.py打开



默认没有代码的。。。



修改其中的源代码,改成下面的

*#coding:utf-8***from** django.http **import** HttpResponse  
  
**def index**(request)**:  
 return** HttpResponse(u"hello,my name in LTD!")

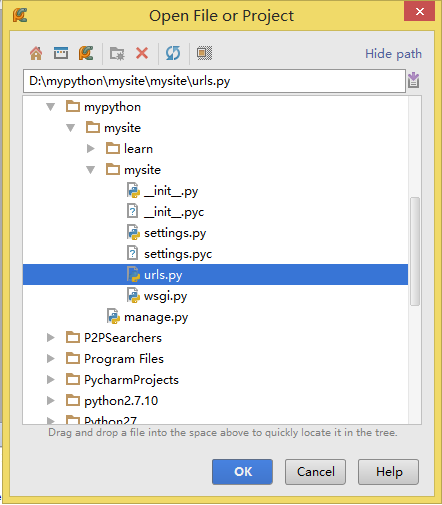


第一行是声明编码为utf-8, 因为我们在代码中用到了中文,如果不声明就报错；第二行引入HttpResponse，它是用来向网页返回内容的，就像Python中的 print 一样，只不过 HttpResponse 是把内容显示到网页上。

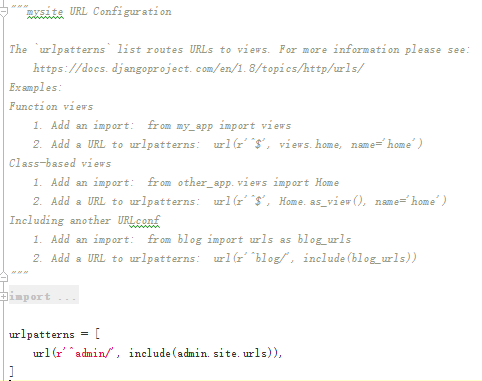
我们定义了一个index()函数，第一个参数必须是request,，与网页发来的请求有关，可以包含get或post的内容,函数返回一行字到网页。

定义视图函数相关的URL(网址)

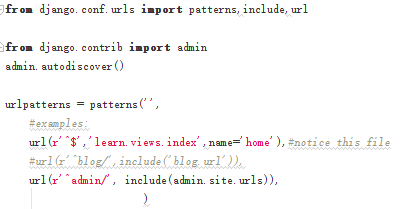
打开 mysite/mysite/urls.py 这个文件



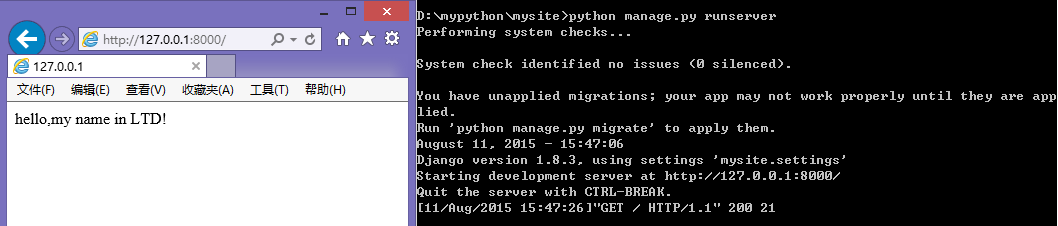
默认就这些code



**from** django.conf.urls **import** patterns,include,url  
  
**from** django.contrib **import** admin  
admin.autodiscover()  
  
urlpatterns **=** patterns('',  
 *#examples:* url(r'^$','learn.views.index',name**=**'home'),*#notice this file  
 #url(r'^blog/',include('blog.url')),* url(r'^admin/', include(admin.site.urls)),  
 )



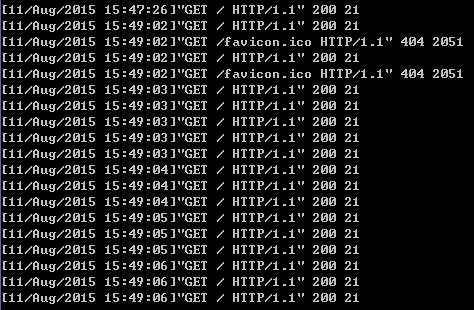
在终端上运行 python manage.py runserver 我们会看到类似下面的信息:



python manage.py runserver 0.0.0.0:8000

监听机器上所有ip 8000端口，访问时用电脑的ip代替 127.0.0.1

在浏览器打开<http://127.0.0.1:8000/>，终端也会有访问记录输出



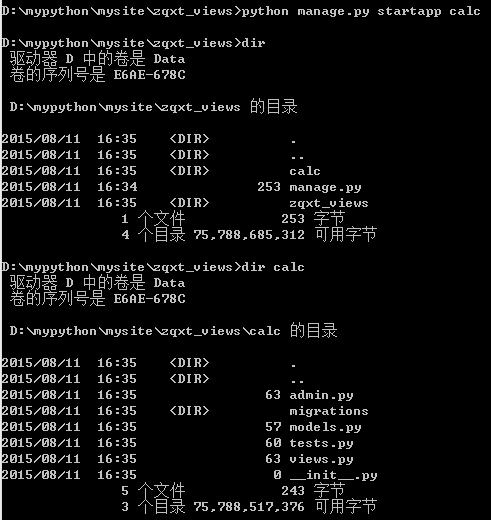
Django中的urls.py用的是正则进行匹配的，如果不熟悉，可以学习[正则表达式](http://www.ziqiangxuetang.com/regexp/regexp-tutorial.html)以及[Python正则表达式](http://www.ziqiangxuetang.com/python/python-reg-expressions.html)。

/add/?a=4&b=5 的GET方法

新建project：zqxt\_views和app：calc

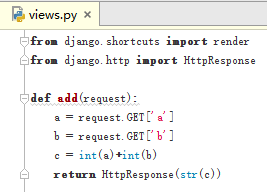
django-admin startproject zqxt\_views

python manage.py startapp calc



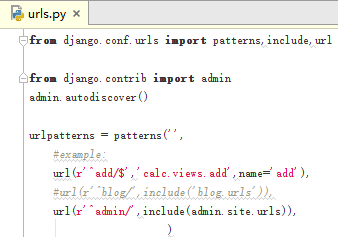
修改zqxt\_views\calc\views.py代码如下：

**from** django.shortcuts **import** render  
**from** django.http **import** HttpResponse  
  
**def add**(request)**:** a **=** request.GET['a']  
 b **=** request.GET['b']  
 c **=** int(a)**+**int(b)  
 **return** HttpResponse(str(c))



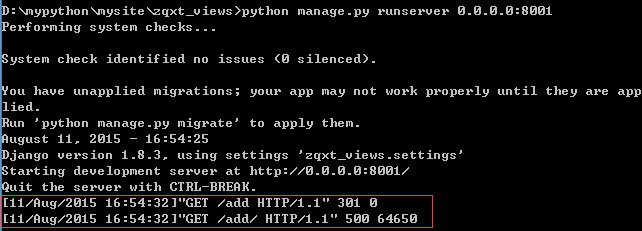
修改zqxt\_views\zqxt\_views\urls.py代码如下:

**from** django.conf.urls **import** patterns,include,url  
  
**from** django.contrib **import** admin  
admin.autodiscover()  
  
urlpatterns **=** patterns('',  
 *#example:* url(r'^add/$','calc.views.add',name**=**'add'),  
 *#url(r'^blog/',include('blog.urls')),* url(r'^admin/',include(admin.site.urls)),  
 )

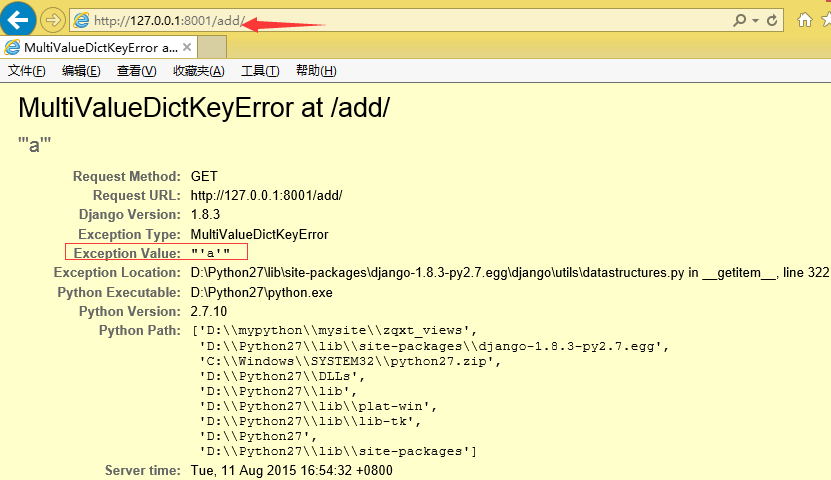


以上添加一个网址来对应我们刚才新建的视图函数，打开开发服务器并访问

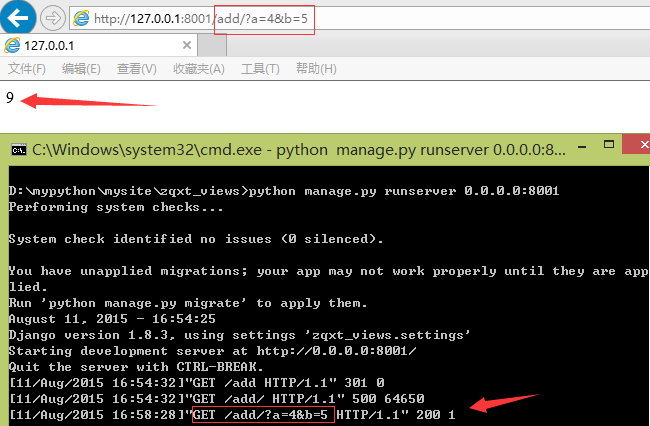
python manage.py runserver 0.0.0.0:8001



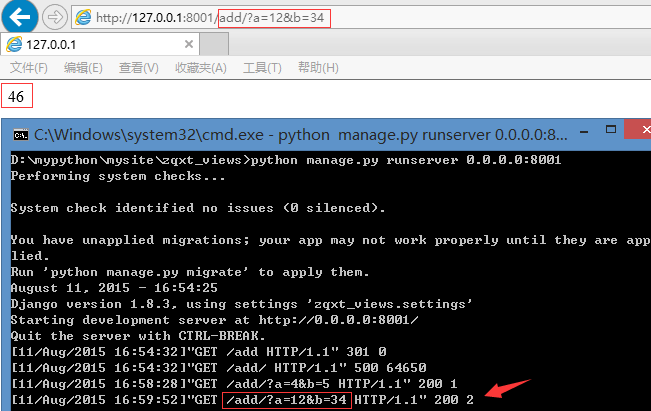
http://127.0.0.1:8001/add/



这是因为我们并没有传值进去，在后面加上 ?a=4&b=5，访问结果如下：



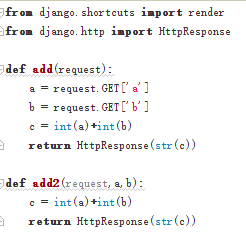
http://127.0.0.1:8001/add/?a=12&b=34



/add/3/4/的GET方式

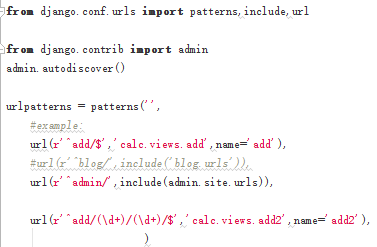
接着修改 calc/views.py文件，再新定义一个add2 函数

**def add2**(request,a,b)**:** c **=** int(a)**+**int(b)  
 **return** HttpResponse(str(c))



接着修改 zqxt\_views/urls.py 文件，再添加一个新的 url

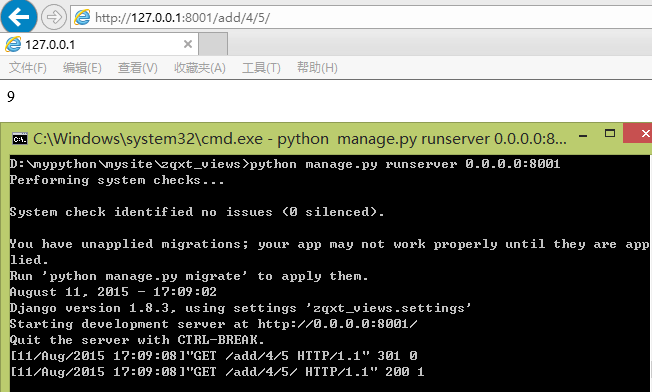
url(r'^add/(\d+)/(\d+)/$','calc.views.add2',name**=**'add2'),

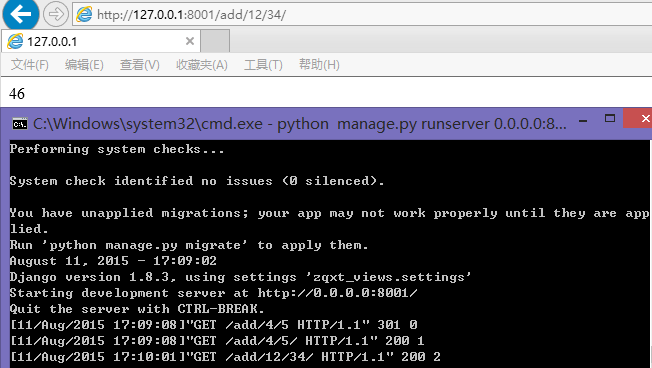


我们可以看到网址中多了 (\d+), 正则表达式中 \d 代表一个数字，+ 代表一个或多个前面的字符，写在一起 \d+ 就是一个或多个数字，用括号括起来的意思是保存为一个子组（更多知识请参见[Python 正则表达式](http://www.ziqiangxuetang.com/python/python-reg-expressions.html)），每一个子组都作为一个参数被 views.py 中的函数接收。

我们再访问 http://127.0.0.1:8000/add/4/5/ 就可以看到和刚才同样的效果，但是这回网址更优雅了

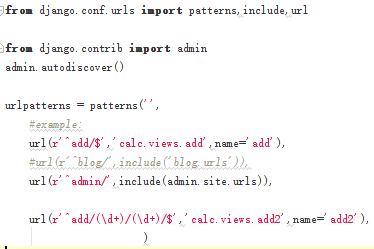
http://127.0.0.1:8001/add/4/5/



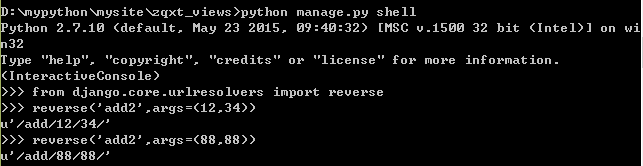


url 中的 name

看一下 urls.py 中的代码



r'^add/(\d+)/(\d+)/$','calc.views.add2',name**=**'add2'指定该url使用calc的views的add2方法，name**=**'add2'的作用是简化内部函数reverse，reverse()通过{% url 'name' 参数 %}获取到网址 /add/4/5/



**Django 结构**

在

django-admin.py startproject project名称

cd project名称

下面我们来看看[startproject](https://docs.djangoproject.com/en/1.8/ref/django-admin/#django-admin-startproject)命令创建了什么：

**mysite/**    项目的根目录，是放project的容器。其命名和Django无关，可以随意修改该文件夹名

**manage.py**  命令行工具-可以通过它与Django project进行互动。[参考](https://docs.djangoproject.com/en/1.8/ref/django-admin/)

**mysite/**  Django project实际的Python代码包（Python package)，目录名称不能改变。

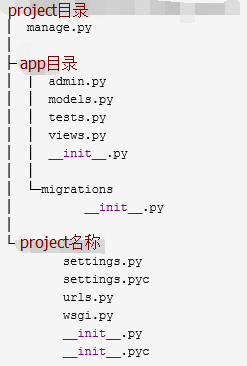
**\_init\_.py**   空文件-通知Python该目录下文件为一个Python Package。

**settings.py**  Django Project的设置文件。[参考](https://docs.djangoproject.com/en/1.8/topics/settings/)

**urls.py**   Django Project的URL声明文件，类似基于Django的站点的“目录”。[参考](https://docs.djangoproject.com/en/1.8/topics/http/urls/)

**wsgi.py**   一个对于WSGI兼容的Web服务器的入口点。[参考](https://docs.djangoproject.com/en/1.8/howto/deployment/wsgi/)

python manage.py startapp app名称



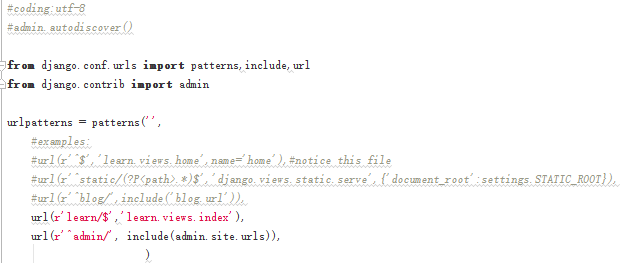
**Django admin**

登录后台

在project/project/urls.py默认就有admin的模块和url设置

**from** django.contrib **import** admin

url(r'^admin/', include(admin.site.urls)),



正确的做法是首先创建登录管理站点的用户，执行如下命令：

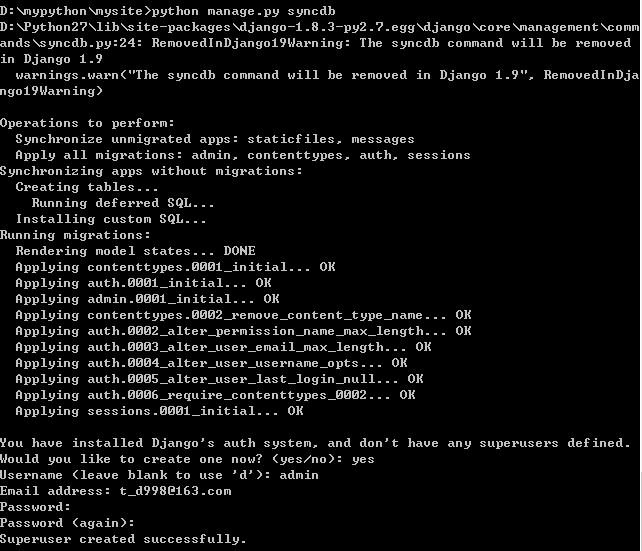
python manage.py createsuperuser



但是我这里首先同步了数据库

python manage.py syncdb

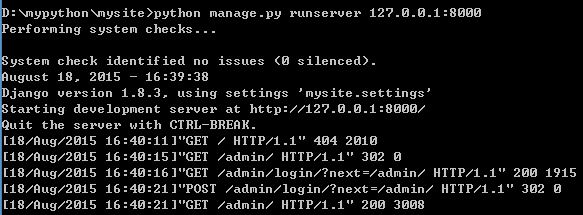
默认使用python自带的sqlite数据库，后面我将使用远程mysql数据库，详情参考Django mysql章节。由于之前我没有建过数据库，所以就现在建了



用户名：admin，密码：123456，邮箱：t\_d998@163.com

启动server

python manage.py runserver 127.0.0.1:8000



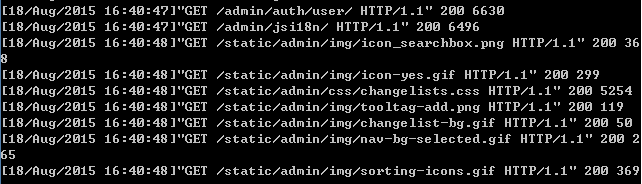
终于看到了期待已久的界面



用户名和密码是在执行”python manager.py syncdb”时创建的，root，123456



探索admin的面板，我们发现所有的css、js、图片等静态文件都是在static目录



在我的电脑上的路径：

D:\Python27\Lib\site-packages\Django-1.8.3-py2.7.egg\django\contrib\admin\static\admin\img

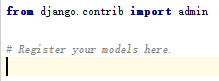
但是我们的app是静态文件目录不是基于此

管理数据库

我们在数据库中创建了Question、Choice两个库，怎么用django的后台管理系统进行可视化操作呢？首先需要告知后台管理系统Question对象的管理接口（have an admin interface）

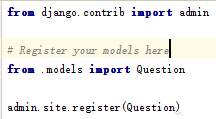
编辑project/app/admin.py文件

它默认就一行：

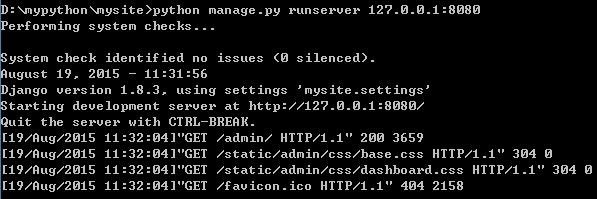


追加如下内容：

**from** .models **import** Question  
  
admin.site.register(Question)



重新启动server，刷新后台界面，看到了我们的数据库









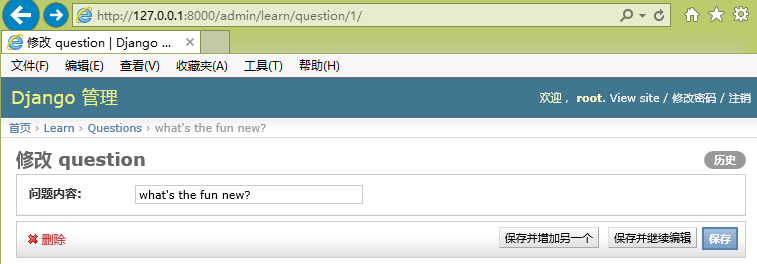
字段的别名

现在我修改下Question库各个字段的属性：

**class** Question(models.Model):  
 question\_text=models.CharField(max\_length=200,verbose\_name=**'问题内容'**)  
 pub\_date=models.DateTimeField(auto\_now=True,verbose\_name=**'提交时间'**)



保存后正在运行的server会自动加载修改的内容，只需要刷新下浏览器，界面发生了变化

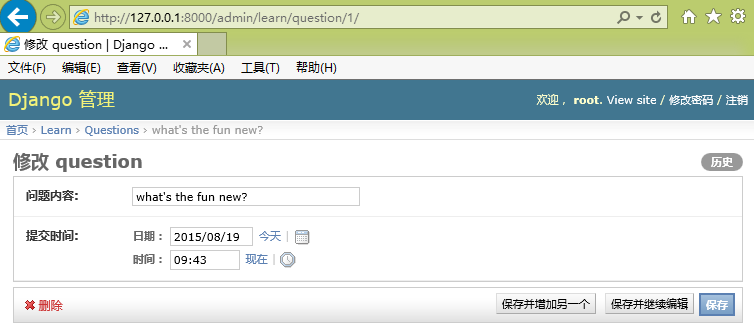


verbose\_name给字段取了别名，auto\_now=True隐藏了修改的时间

我们把auto\_now=True去掉



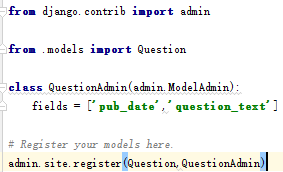
刷新下页面，修改时间回来了



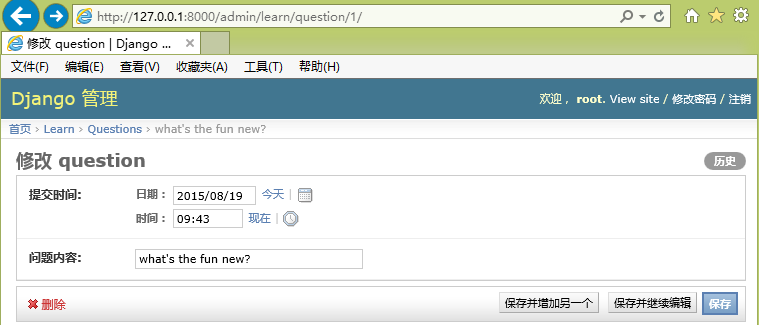
怎样把“提交时间”和“问题内容”两栏互换下位置呢

修改project/app/admin.py，修改QuestionAdmin的fields属性

**class** QuestionAdmin(admin.ModelAdmin):  
 fields = [**'pub\_date'**,**'question\_text'**]  
  
*# Register your models here.*admin.site.register(Question,QuestionAdmin)



效果达到了

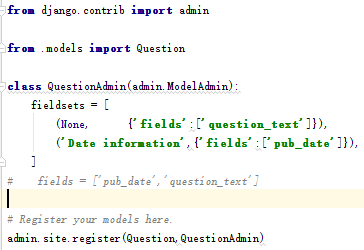


再用两栏分别为两个字段设置标题呢

修改admin.py，修改QuestionAdmin的fieldsets属性

fieldsets = [  
 (None, {**'fields'**:[**'question\_text'**]}),  
 (**'Date information'**,{**'fields'**:[**'pub\_date'**]}),  
]

注释：fieldsets中每个元组的第一个元素是字段集的标题。

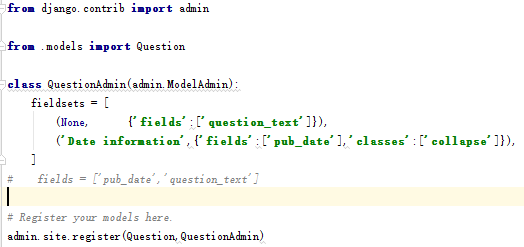


效果如下：

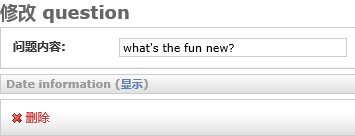


我们还可以把**Date information**缩成按钮

fieldsets = [  
 (None, {**'fields'**:[**'question\_text'**]}),  
 (**'Date information'**,{**'fields'**:[**'pub\_date'**],**'classes'**:[**'collapse'**]}),  
]



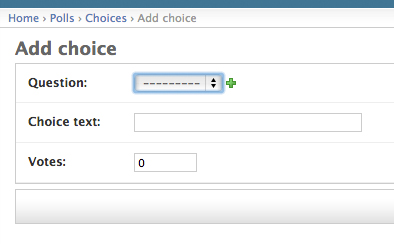
默认以“显示”按钮





关联对象

一个Question有多个Choices，怎样在管理界面中将选项显示出来？最容易相到import Choice, Question，再admin.site.register(Choice)，但是在这个表单中，“Question”字段是一个可选的选项框，包含数据库中所有的Question。 Django知道ForeignKey应该在管理界面中显示为一个<select>选框。类似这样

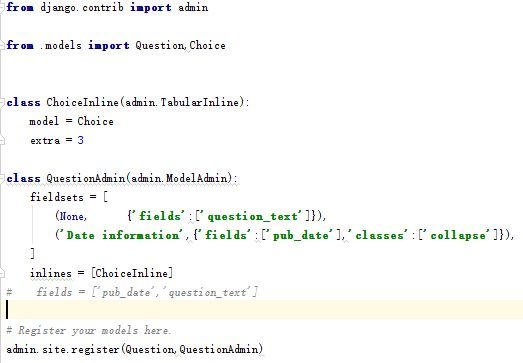


在我们的例子中，目前选框里只有一个Question。另外，注意一下“Question”旁边的“Add Another”链接。 每个与其它对象拥有ForeignKey关系的对象都有一个这种链接。当你点击“Add Another”，你将看到一个带有“Add question”表单的弹出窗口。如果你在这个窗口中添加了一个Question并点击“Save”，Django会将保存这个Question到数据库中，然后动态地将这个对象添加为你正在查看的“Add choice”表单的选择项。但事实上，这是一种不高效的方式来添加Choice对象到系统中。在创建Question对象的同时可以直接添加一组Choice将会更好。让我们实现这个功能。

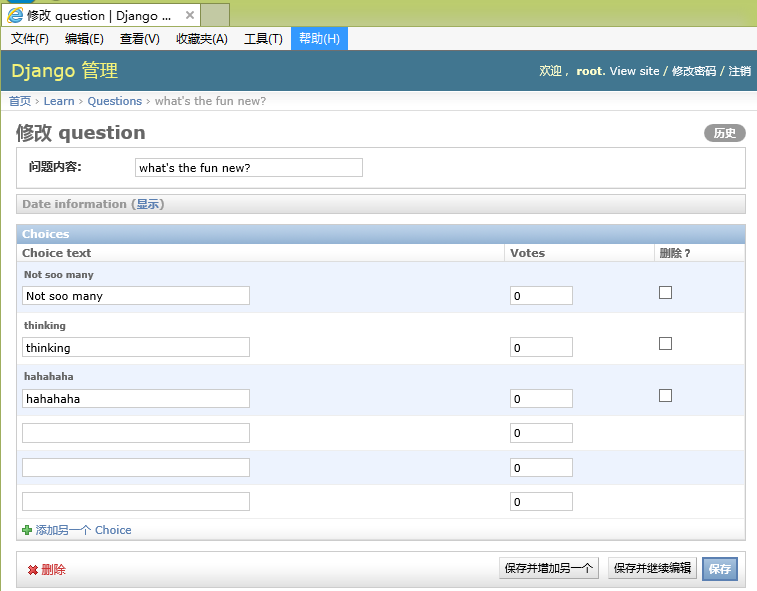
修改admin.py代码如下：

**from** .models **import** Question,Choice  
  
**class** ChoiceInline(admin.TabularInline):  
 model = Choice  
 extra = 3  
  
**class** QuestionAdmin(admin.ModelAdmin):  
 fieldsets = [  
 (None, {**'fields'**:[**'question\_text'**]}),  
 (**'Date information'**,{**'fields'**:[**'pub\_date'**],**'classes'**:[**'collapse'**]}),  
 ]  
 inlines = [ChoiceInline]

注释：TabularInline将以表格的形式显示Choice，不用StackedInline就避免了以表单的“块”显示Choice。extra = 3告诉Django：Choice对象在Question的管理界面中编辑。默认提供足够3个Choice的空间。



效果如下：



试试添加一个question

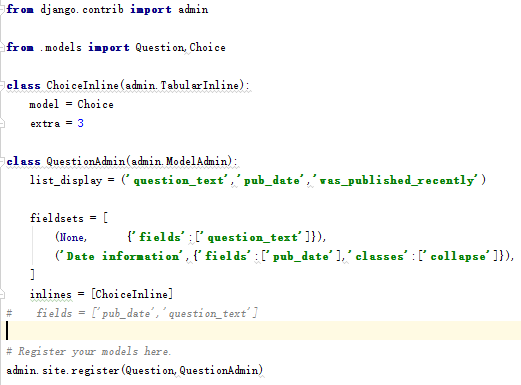


只显示字段名称

默认地，Django显示每个对象的str()返回的内容，list\_display是一个要显示的字段名称的元组，在对象的变更列表页面上作为列显示，只显示每个字段。

修改admin.py，添加如下代码：

list\_display = (**'question\_text'**,**'pub\_date'**,**'was\_published\_recently'**)



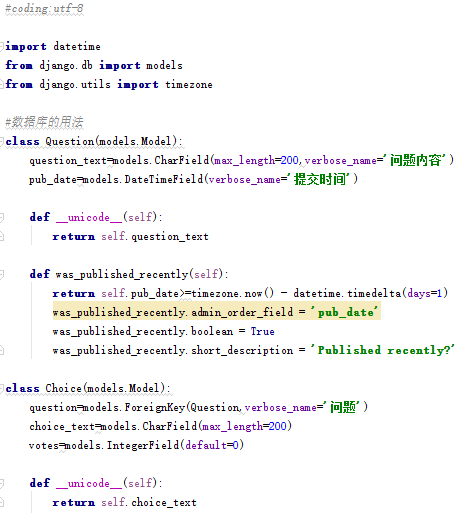
**注释：was\_published\_recently** 是我们在models.py中定义的方法，指明时间是当前时间的前一天



**was\_published\_recently**这列的头部，因为Django不支持按照随便一个方法的输出进行排序。另外注意， **was\_published\_recently**这列的头部默认是这个方法的名字（用空格取代下划线），并且这列的每一项内容都是用返回值的字符串形式表示。我们可以为它添加一些属性，关于这些方法属性的更多信息，请参见 [**list\_display**](http://python.usyiyi.cn/django/ref/contrib/admin/index.html#django.contrib.admin.ModelAdmin.list_display)。

编辑models.py文件，添加如下代码：

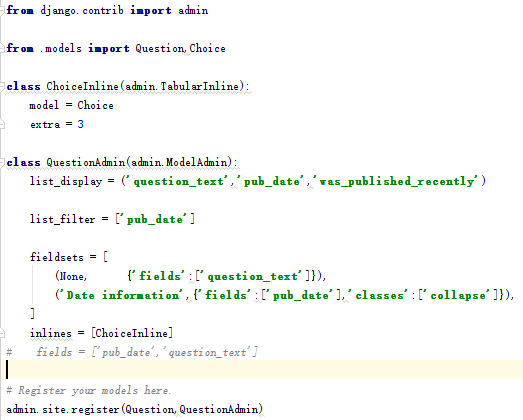
was\_published\_recently.admin\_order\_field = **'pub\_date'**was\_published\_recently.boolean = True  
was\_published\_recently.short\_description = **'Published recently?'**



再编辑admin.py文件，来改进Question变更列表页面：使用list\_filter来添加过滤器。将下面这行添加进QuestionAdmin：

list\_filter = [**'pub\_date'**]

注释：这行代码添加一个“Filter”侧边栏，我们可以通过pub\_date字段对变更列表进行过滤



效果如下：

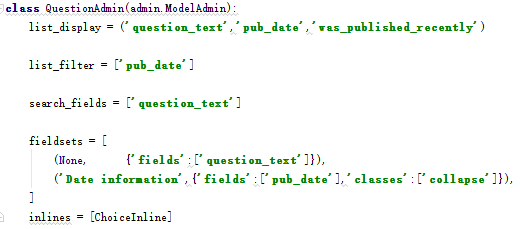


过滤器可以通过提交的时间来检索信息，怎样通过关键字来搜索呢？用search\_fields

在admin.py添加如下代码：

search\_fields = [**'question\_text'**]

注释：这行代码在变更列表的顶部添加了一个搜索框。 当有人将搜索的内容输入搜索框，Django将在question\_text字段中进行搜索。 你可以使用任意数量的字段 —— 但由于它在后台使用LIKE进行查询，限制搜索字段的数量在会使你的数据库查询更容易。



**效果如下：**



变更列表界面提供方便的分页功能。 默认每页显示100条。 [**Change** **list** **pagination**](http://python.usyiyi.cn/django/ref/contrib/admin/index.html#django.contrib.admin.ModelAdmin.list_per_page)、[**search** **boxes**](http://python.usyiyi.cn/django/ref/contrib/admin/index.html#django.contrib.admin.ModelAdmin.search_fields)、[**filters**](http://python.usyiyi.cn/django/ref/contrib/admin/index.html#django.contrib.admin.ModelAdmin.list_filter)、[**date-hierarchies**](http://python.usyiyi.cn/django/ref/contrib/admin/index.html#django.contrib.admin.ModelAdmin.date_hierarchy)和[**column-header-ordering**](http://python.usyiyi.cn/django/ref/contrib/admin/index.html#django.contrib.admin.ModelAdmin.list_display) 都将按照你设想的那样工作。

**Django template**

前面通过HttpResponse直接显示网页内容，现在我们可以把所有内容写在Html文件里，做成”模板“，Django 的模板系统会自动找到app下面的templates文件夹中的模板文件

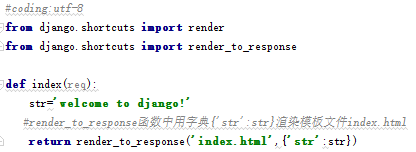
在project/app/views.py文件原来的代码是这样的：



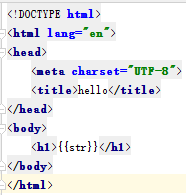
重新定义代码如下：

**from** django.shortcuts **import** render  
**from** django.shortcuts **import** render\_to\_response  
  
**def** index(req):  
 str=**'welcome to django!'** *#render\_to\_response函数中用字典{'str':str}渲染模板文件index.html* **return** render\_to\_response(**'index.html'**,{**'str'**:str})

注释：*render\_to\_response函数中用字典{'str':str}渲染模板文件index.html*



在 APP目录下新建一个 templates 文件夹，里面新建一个 index.html



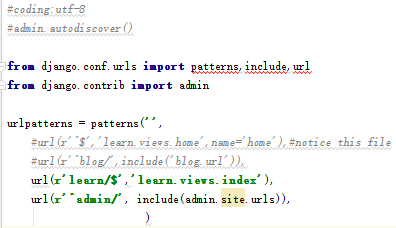


原project/project/urls.py有一行：



改为

url(**r'learn/$'**,**'learn.views.index'**),



*注释：#'learn.views.index'的意思是“到包learn下找到views.py,并执行views.py中的函数index*

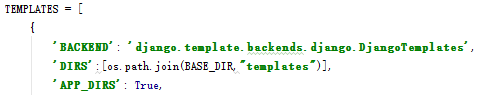
保存后启动server



python manage.py runserver 127.0.0.1:8000



在project/setting.py的“DIRS”模块修改如下：



怎么知道django项目的今天文件的根目录在哪呢？

import django

import sys

import django

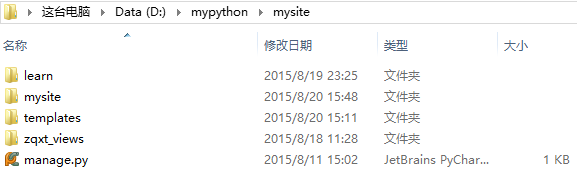
sys.path = sys.path[1:]

print(django.\_\_path\_\_)

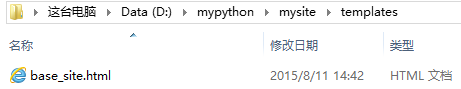


在project/（manager.py的目录）下新建templates目录，将django的静态文件目录下的base\_site.html文件，我的路径是D:\Python27\Lib\site-packages\Django-1.8.3-py2.7.egg\django\contrib\admin\templates\admin\base\_site.html

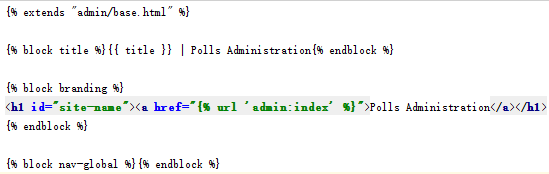
这是项目mysite的根目录



这是base\_site.html的所在目录

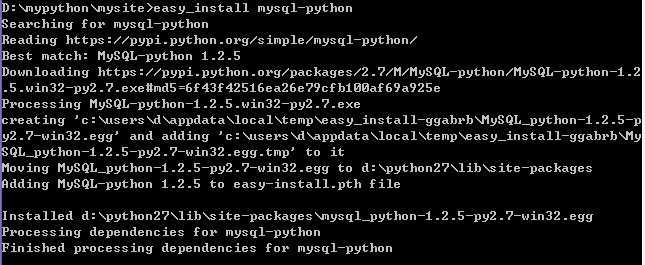


编辑该文件，替换{{ site\_header|default:\_('Django administration') }}为Polls Administration，这里我替换了两处

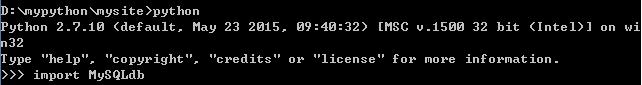


**Django mysql**

easy\_install mysql-python



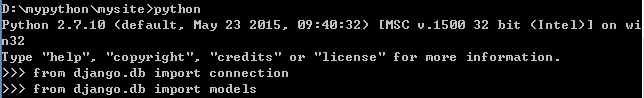
import MySQLdb



在shell可以正常导入django.db包

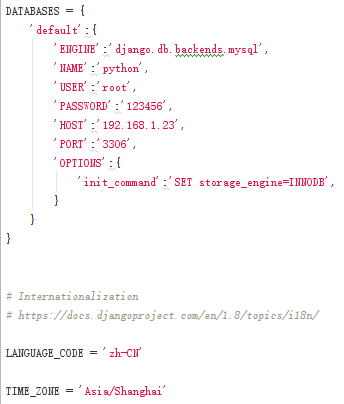
from django.db import connection

from django.db import models



在project/project/setting.py的DATABASES模块写入：

DATABASES **=** {  
 'default'**:**{  
 'ENGINE'**:**'django.db.backends.mysql',  
 'NAME'**:**'python',  
 'USER'**:**'root',  
 'PASSWORD'**:**'123456',  
 'HOST'**:**'192.168.1.23',  
 'PORT'**:**'3306',  
 'OPTIONS'**:**{  
 'init\_command'**:**'SET storage\_engine=INNODB',  
 }  
 }  
}

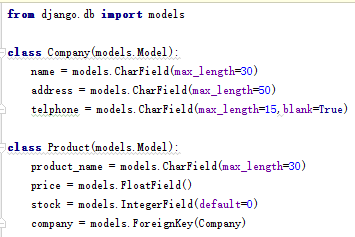


表示连接192.168.1.23的python库

修改app的models

project/app/models.py

**from** django.db **import** models  
  
**class Company**(models.Model)**:** full\_name **=** models.CharField(max\_length**=**30)  
 address **=** models.CharField(max\_length**=**50)  
 telphone **=** models.CharField(max\_length**=**15,blank**=**True)  
  
**class Product**(models.Model)**:** product\_name **=** models.CharField(max\_length**=**30)  
 price **=** models.FloatField()  
 stock **=** models.IntegerField(default=0)  
 company **=** models.ForeignKey(Company)

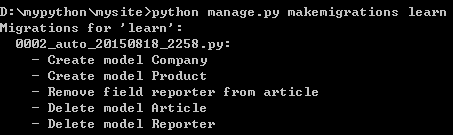


保存后用python manage.py check检查下代码有没有语法上的错误

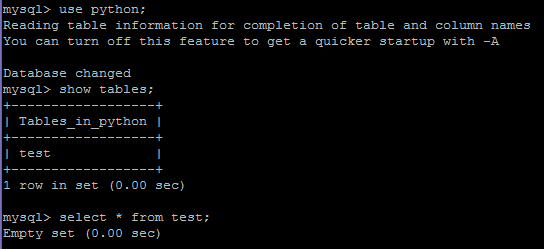


执行python manage.py makemigrations来查看修改models.py前后数据库将要发生的改变（迁移）

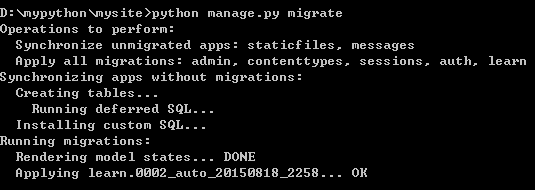
python manage.py makemigrations learn



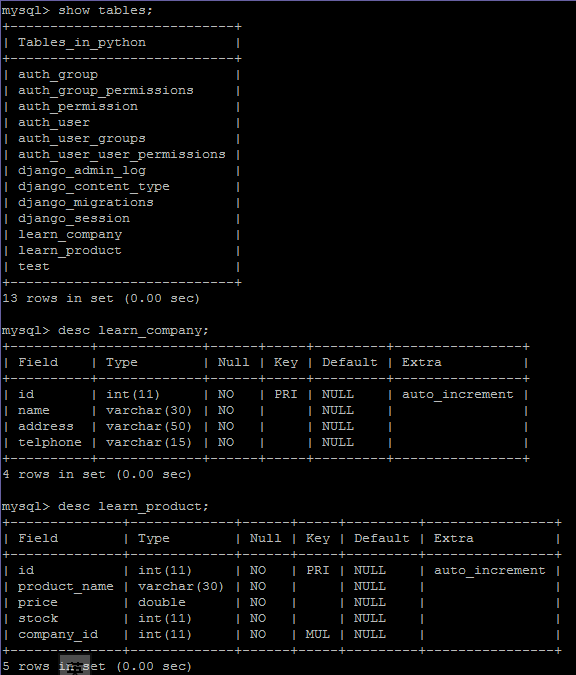
首先看看192.168.1.23的python库的初始状态



执行python manage.py migrate 将这些改变应用到数据库中



可以看到192.168.1.23的python库发生了改变



Ok

对于manage.py的更多功能参考“[django-admin 文档](https://docs.djangoproject.com/en/1.8/ref/django-admin/)”。

现在我们进入Python的交互shell界面并操作Django API

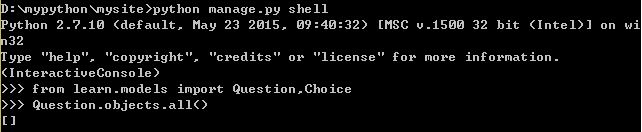
python manage.py shell

进入API后导入我们新建的两个数据库的类

from learn.models import Question,Choice

查看所有记录：

Question.objects.all()



导入时间类

from django.utils import timezone

创建第一条记录

q=Question(question\_text="what's new?",pub\_date=timezone.now())

q.save()

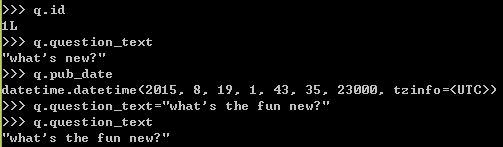


**查看记录**

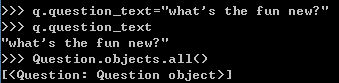
**q.id**

**q.question\_text**

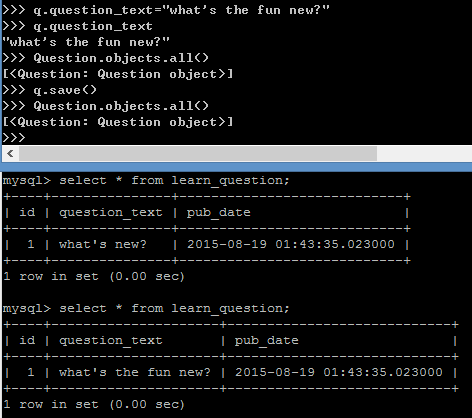
**q.pub\_date**



使用赋值语句，可以修改，不要忘了用save()保存修改



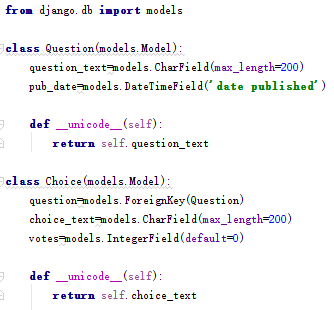
查看后台数据库的表记录



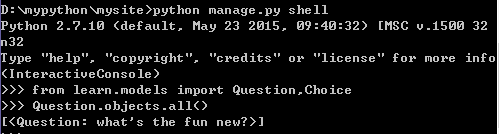
可以看到Question.objects.all()命令返回的信息对我们的帮助不大。我们可以通过修改polls/models.py中的Question和Choice模型来解决这个问题。具体说来就是在Question和Choice类中添加[**\_\_unicode\_\_**](https://docs.djangoproject.com/en/1.8/ref/models/instances/#django.db.models.Model.__unicode__)**函数。对于Python 2来说，你应该定义\_\_unicode\_\_()方法并返回unicode 值。Django 模型具有一个默认的\_\_str\_\_() 方法，它会调用\_\_unicode\_\_()并将结果转换为UTF-8 字节字符串。这意味着unicode(p)将返回一个Unicode 字符串，而str(p)将返回一个字节字符串，其字符以UTF-8编码。Python 的行为则相反：对象的\_\_unicode\_\_方法调用 \_\_str\_\_方法并将结果理解为ASCII 字节字符串。**

**def** \_\_unicode\_\_(self):  
 **return** self.question\_text

**def** \_\_unicode\_\_(self):  
 **return** self.choice\_text



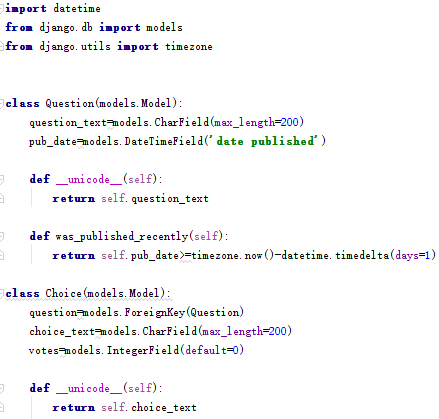
在models.py中添加\_\_unicode\_\_()不仅可以在交互shell界面中提供方便的信息，也在Django自动生成的后台管理时具有重要作用。



在Question类加上这些语句

**import** datetime  
**from** django.utils **import** timezone

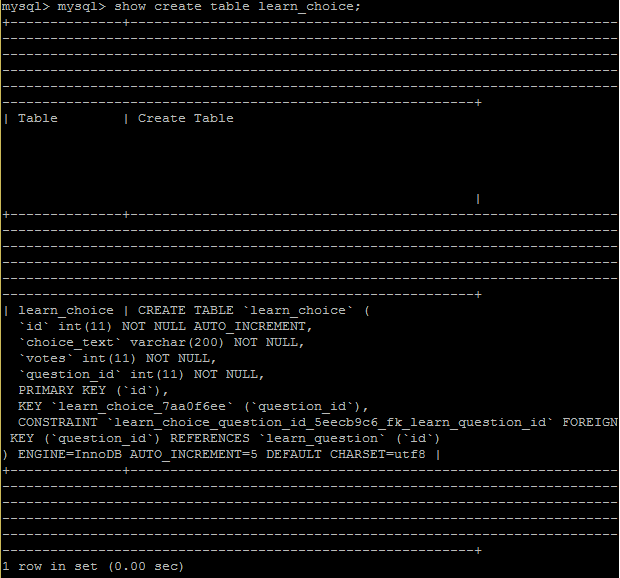
**def** was\_published\_recently(self):  
 **return** self.pub\_date>=timezone.now()-datetime.timedelta(days=1)



关于[datetime](https://docs.python.org/3/library/datetime.html#module-datetime)模块、[django.utils.timezone](https://docs.djangoproject.com/en/1.8/ref/utils/#module-django.utils.timezone)模块想参考官方文档中的相关资料。关于Django中与时区处理相关的内容也可以参考[time zone support doc](https://docs.djangoproject.com/en/1.8/topics/i18n/timezones/)。

我们可以看到在建learn\_question、learn\_choice表时，'init\_command'**:**'SET storage\_engine=INNODB'是生效的：

show create table learn\_choice;

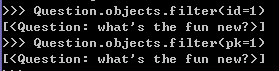


通过id（主键）查询

from learn.models import Question,Choice

Question.objects.filter(id=1)

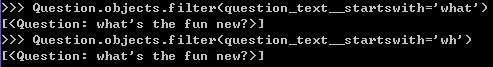
Question.objects.get(pk=1)



通过关键字查询

Question.objects.filter(question\_text\_\_startswith='what')

Question.objects.filter(question\_text\_\_startswith='wh')

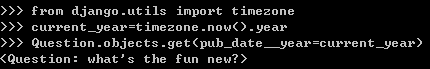


通过时间查询

from django.utils import timezone

current\_year=timezone.now().year

Question.objects.get(pub\_date\_\_year=current\_year)



逻辑判断

我们在models.py定义了was\_published\_recently函数



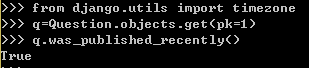
该函数定义了”pub\_date不小于当前时间的其一天“的逻辑

from learn.models import Question,Choice

from django.utils import timezone

q=Question.objects.get(pk=1)

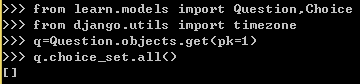
q.was\_published\_recently()



看看Choice模型，Question和Choice通过外键（foreign key）建立关联

q=Question.objects.get(pk=1)

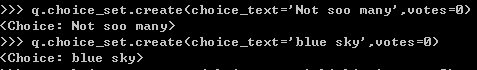
q.choice\_set.all()



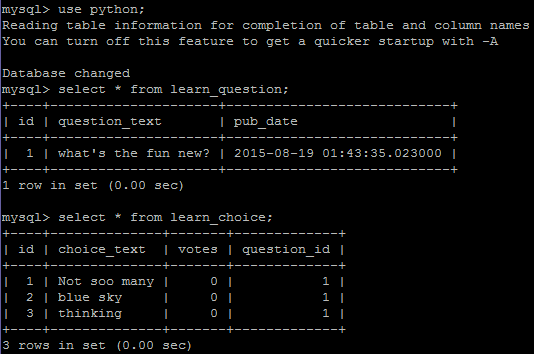
创建3个记录

q.choice\_set.create(choice\_text='Not soo many',votes=0)

q.choice\_set.create(choice\_text='blue sky',votes=0)



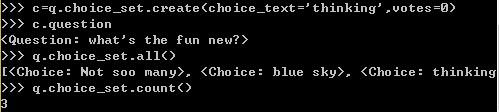
数据库已经插入了记录



由Question对象也可以通过API连接到Choice对象

c=q.choice\_set.create(choice\_text='thinking',votes=0)

q.choice\_set.all()

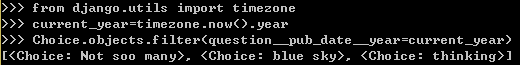


Choice对象直接连接到Question对象

from django.utils import timezone

current\_year=timezone.now().year

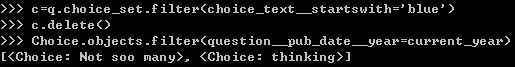
Choice.objects.filter(question\_\_pub\_date\_\_year=current\_year)



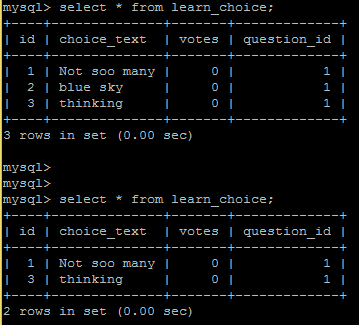
删除操作

c=q.choice\_set.filter(choice\_text\_\_startswith='blue')

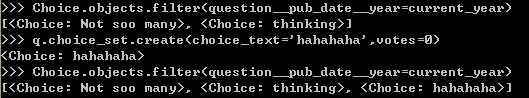
c.delete()

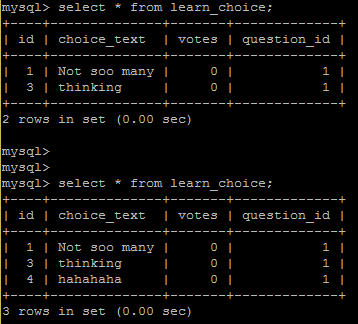


数据库同时也修改了



不过让人奇怪的是，mysql删除一条记录后，自增长的列为什么没有相应的重新排序





**Django Models**

Django是基于MVC的设计模式:

Model（模型）表示应用程序核心（比如数据库记录列表）。

View（视图）显示数据（数据库记录）。

Controller（控制器）处理输入（写入数据库记录）。

准确的说Django是MVT模式，因为C被自动实现了，一般不用程序员做数据库操作，django引入了T，即template，模板。Model就是MVC中的M，代表的数据对象，反映到数据库中就是数据表，Model中的属性是表的一列，Django对Model进行了封装，对Model提供了丰富的查询接口，反映到数据库中，就是提供了丰富的select查询功能，这就是Django的强大之处。

所有的Model都继承自django.db.models.Model类，Model类的每一个属性都继承自django.db.models.Field，这个Field有几个作用：

•决定该Field在数据库中的类型

•决定该Field在前端上如何显示

•做简单的验证

在Django中如何实现关系型数据库的那三种典型关系：多对一，多对多，一对一，参考：<http://blog.csdn.net/hackerain/article/details/39196495>

*#^hello/$是正则表达式，意思是“当输入的网址为http://服务器ip:端口/hello/时”，*

**Django 正则表达式**

向后台的传递参数有两种方式：

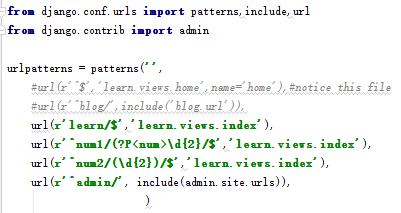
方式一：GET方法（url中传递参数）。例如上面的，?a=4&b=5 ，传递参数a=4,b=5

方式二：POST方法（后续教程介绍）

方式三：正则表达式（url中传递参数）

在urls.py增加如下配置：

url(**r'^num1/(?P<num>\d{2}/$'**,**'learn.views.index'**),  
url(**r'^num2/(\d{2})/$'**,**'learn.views.index'**),



注释：

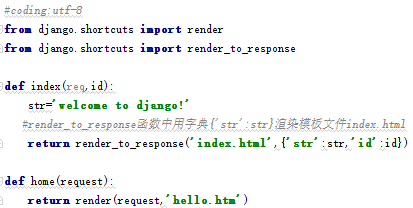
^num1/(?P<id>\d{2})/$是正则表达式，意思是“当输入的url地址为http://服务器ip:端口/num1 /2位数字/时”，参数id的值为url中传递的2位数字

^num2/(\d{2})/$也是正则表达式，和前者不同在于，这里没有显示指出参数名为id，由于在views.py(参看下文第5步)中使用变量id作为接收参数，故此法依然可以向后台的id变量传递参数值,效果与前者一样

在views.py增加如下配置：

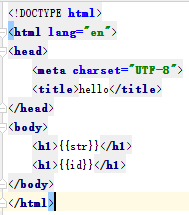
**def** index(req,id):

**'str'**:str,**'id'**:id}



特别说明：index(req,id)函数接受两个参数，分别命名为str和id，不能用原来的index(req)

在templates的index.html中加入如下语句：



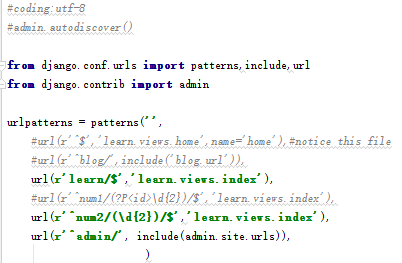
启动app，输入<http://127.0.0.1:8000/num1/23/>、http://127.0.0.1:8000/num1/88





再用第二种正则表达式试

修改urls.py如下



启动app，输入<http://127.0.0.1:8000/num2/66/，http://127.0.0.1:8000/num2/88>，





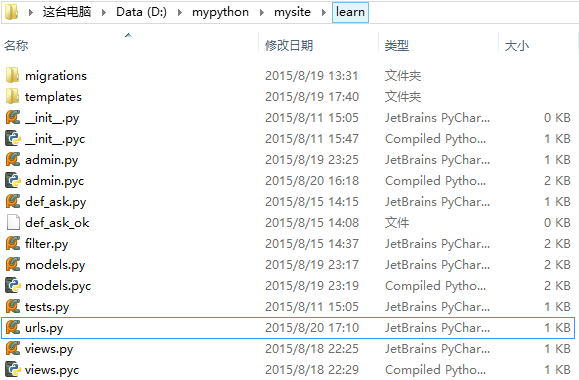
**Django view**

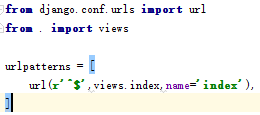
在Django中，网页的页面和其他内容都是由视图来交付（视图对WEB请求进行回应）。 每个视图都是由一个简单的Python函数表示的（对于基于类的视图，一个视图是用一个方法来表示的）。Django使用叫做‘URLconfs’的配置来为URL匹配视图。 一个URLconf负责将URL模式匹配（使用正则表达式）到视图。

在project/app目录下，新建urls.py文件，写入代码：

**from** django.conf.urls **import** url  
**from** . **import** views  
  
urlpatterns = [  
 url(**r'^$'**,views.index,name=**'index'**),  
]

注意:在project目录下才有ursl.py，但是我们可以在app目录下有ursl.py文件

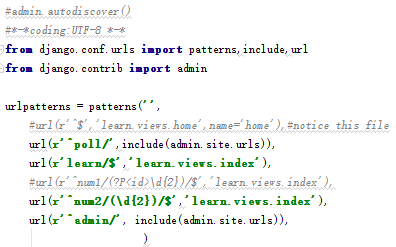




让project的urls.py链接到app的urls.py，修改project/urls.py，在urlpatterns模块加入如下代码：

url(**r'^poll/'**,include(admin.site.urls)),

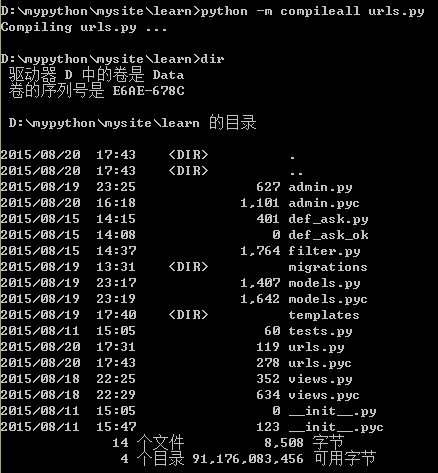
如下：



接下来还需要特别的做法，把新建的urls.py编译成pyc文件

进入project/app、urls.py所在的目录

python -m compileall urls.py



后台 - 我的第一篇博客

在project/app/models.py编写class，创建一个新的数据库

class Post(models.Model):

"""docstring for Post"""

author = models.ForeignKey('auth.User')

title = models.CharField(max\_length=200)

text = models.TextField()

crate\_date = models.DateTimeField(default=timezone.now)

Published\_date = models.DateTimeField(blank=True,null=True)

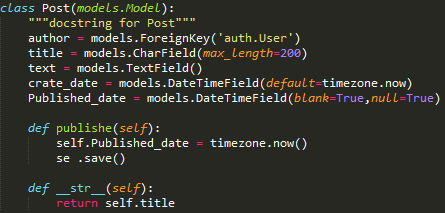
def publishe(self):

self.Published\_date = timezone.now()

se .save()

def \_\_str\_\_(self):

return self.title

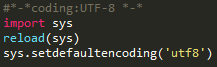


在首部加入

import sys

reload(sys)

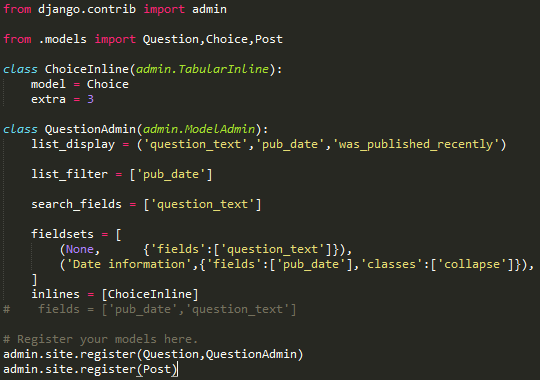
sys.setdefaultencoding('utf8')



在project/app/admin.py注册新建的数据库

from .models import Question,Choice,Post

admin.site.register(Post)

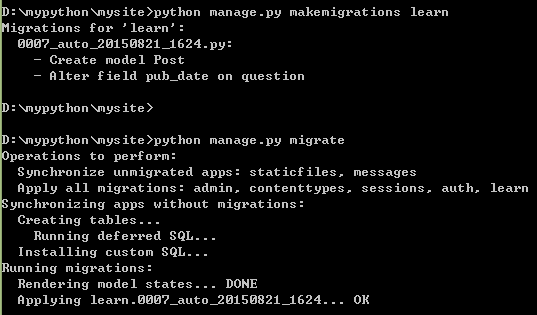


检查语法

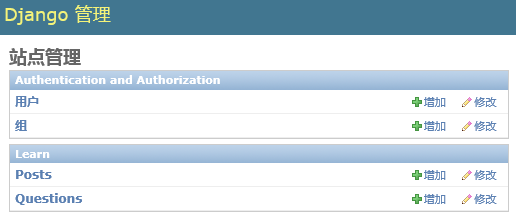


同步数据库

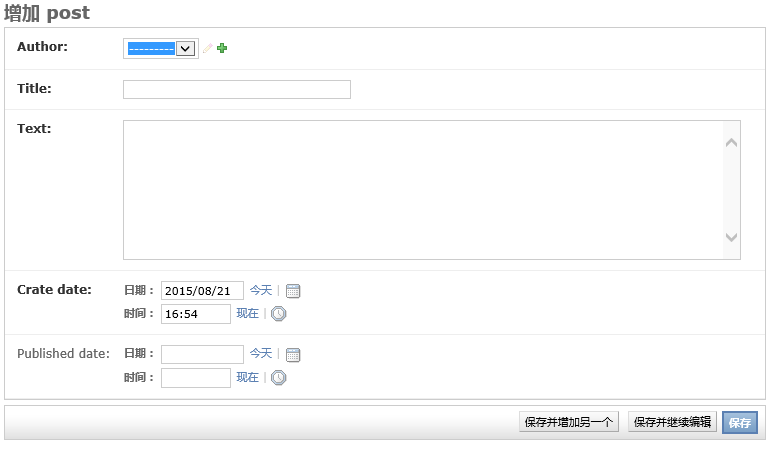
python manage.py migrate



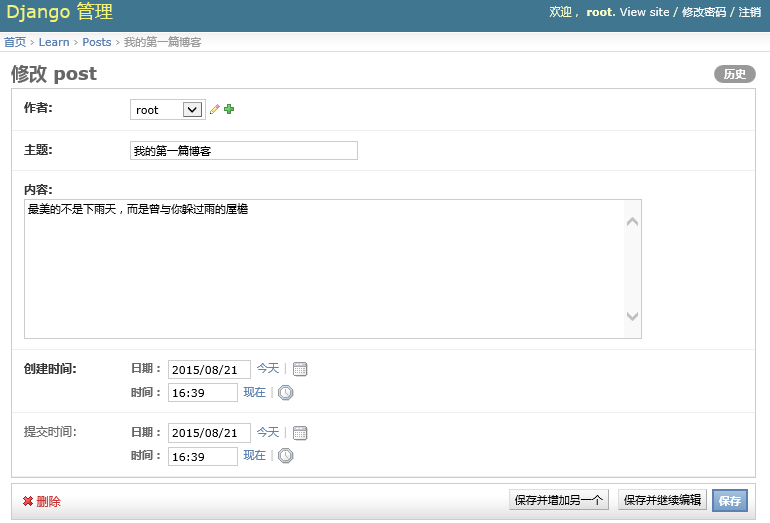
启动server，登录后台，可以看到新建的Posts数据库



可以有哪些操作？



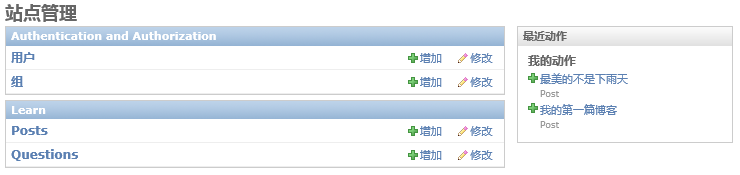
我的第一篇博客



第二篇



OK

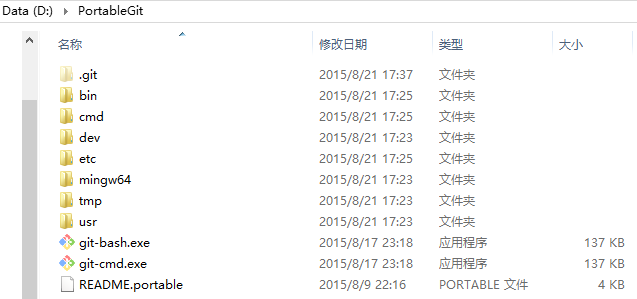




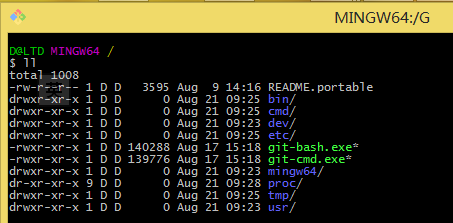
git github

参考：<http://www.bootcss.com/p/git-guide/>

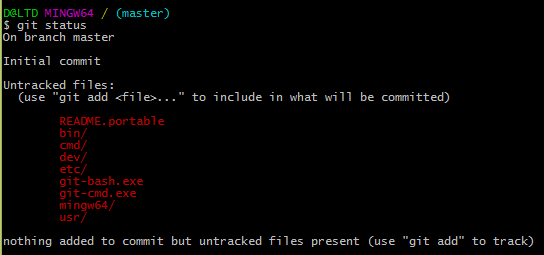
在<http://git-scm.com/download/win>下载PortableGit-2.5.0-64-bit.7z.exe，解压至D盘，无安装过程



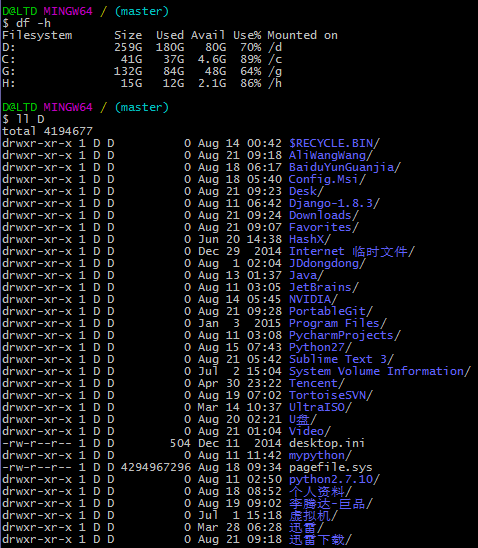
执行“git-bash.exe”，打开GIT的bash环境



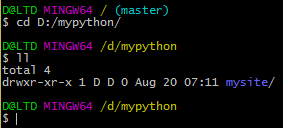
查询状态git status



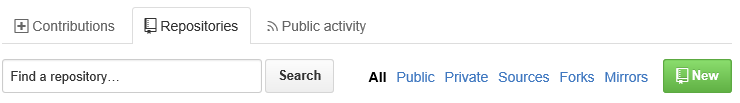
查看硬盘信息



进入project项目目录



在github.com注册，用户名litdd，[邮箱t\_d998@163.com](mailto:邮箱t_d998@163.com)，密码123qweasd，新建repositories



安装说明进行

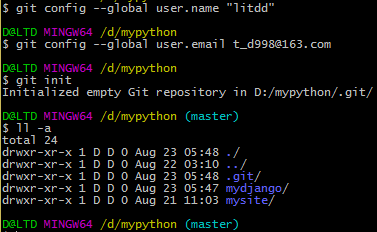


初始化git环境

git config --global user.name "litdd"

git config --global user.email t\_d998@163.com

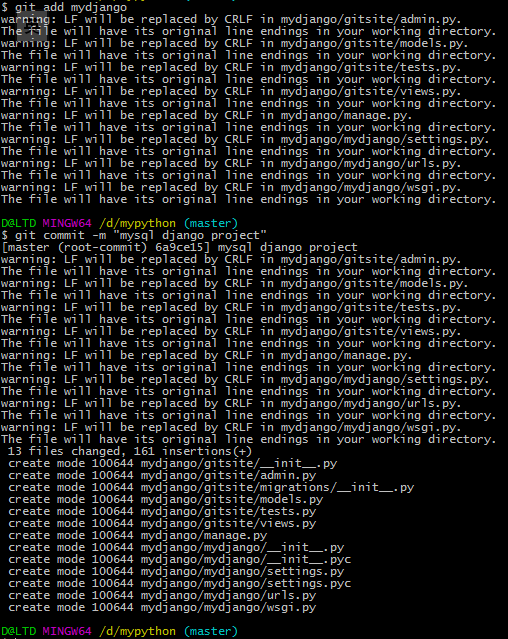
git init



把当前目录mydjango添加到缓存区

git add mydjango

git commit -m "mysql django project"

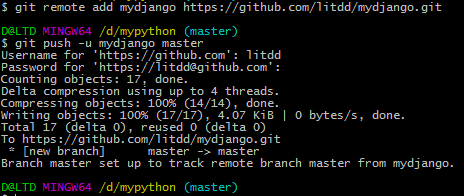


以上的改动现在已经在本地仓库的 HEAD 中了，先将仓库连接到github，再将这些改动提交到github

git remote add mydjango https://github.com/litdd/ mydjango.git

git push -u mydjango master

注意：“add mydjango”、“-u mydjango”是针对github的项目的名称，不能复用，每次remote add时必须更换



在github已经可以看到上传的文件

