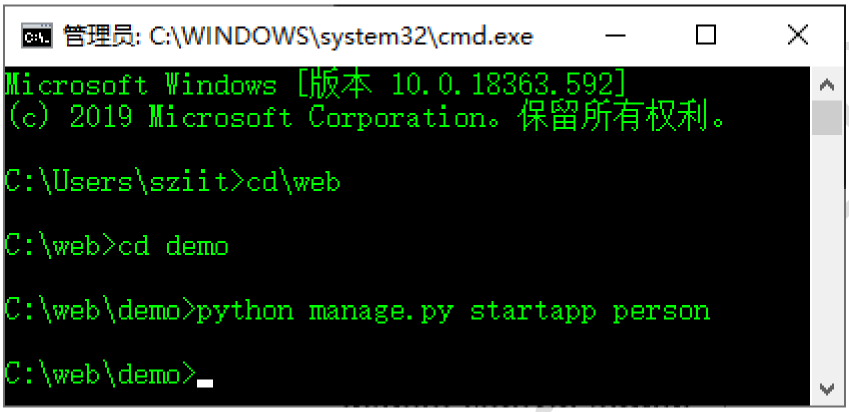
模块 显示数据

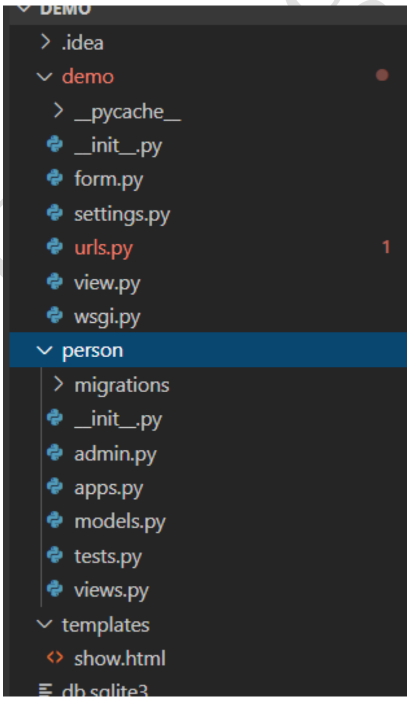
(*) models 不能在根目录使用，必须在一个 App 中，例如 person 中

(*) 创建 person 的 App

python manage.py startapp person



```
Microsoft Windows [版本 10.0.18363.592]
(c) 2019 Microsoft Corporation。保留所有权利。

C:\Users\sziit>cd\web

C:\web>cd demo

C:\web\demo>python manage.py startapp person

C:\web\demo>
```



```
DEMO
  > .idea
  v demo
      > __pycache__
      __init__.py
      form.py
      settings.py
      urls.py            1
      view.py
      wsgi.py
  v person
      > migrations
      __init__.py
      admin.py
      apps.py
      models.py
      tests.py
      views.py
  v templates
      show.html
  db.sqlite3
```

(*) 在 settings.py 中加入 person

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

(*)**Setting the connection to Sqlite3 as:**

```

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'persons.db'),
    }
}

```

Note the persons.db will be created automatically after executing "phthon manage.py makemigrations".

(*) templates\show.html

```
<html>
<body>
{% for person in persons %}
{{person}}<br>
{{person.Name}},{{person.Sex}},{{person.Age}}<br>
{% endfor %}
</body>
</html>

```

(*) person\models.py

```
from django.db import models

# Create your models here.
class PersonModel(models.Model):
    Name=models.CharField(primary_key=True,max_length=30)
    Sex = models.CharField(max_length=30)
    Age=models.IntegerField()

    def __str__(self):
        return self.Name

```

(*) view.py (not person's views.py)

```python
from django.shortcuts import render
from person.models import PersonModel

def show(request):
    persons=PersonModel.objects.all()
    PersonModel.objects.all().delete()
    PersonModel.objects.create(Name="a",Sex="male",Age=20)
    PersonModel.objects.create(Name="b", Sex="female", Age=19)
    persons=PersonModel.objects.all()
    print(type(persons))
    print(type(persons[0]))

    return render(request,"show.html",locals())
```

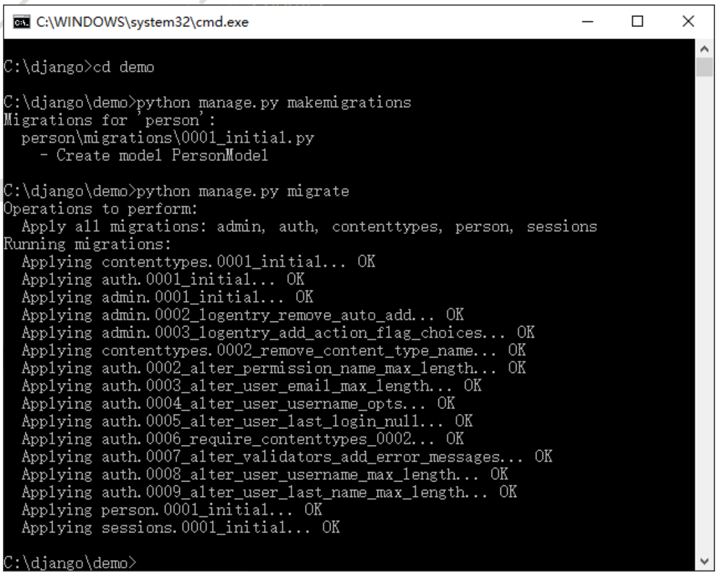
(*) urls.py

```python
from django.conf.urls import url
from . import view

urlpatterns = [
    url(r'^$', view.show),
]
```

(*) Executing

python manage.py makemigrations

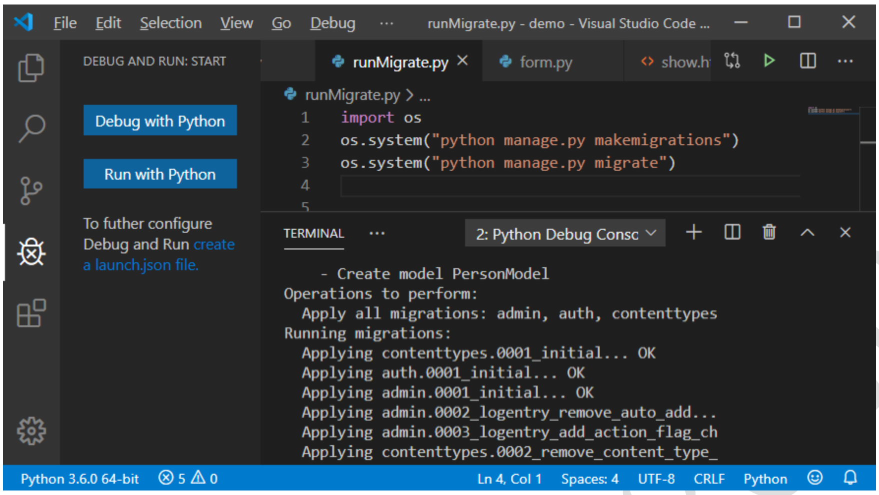
python manage.py migrate



```
C:\django>cd demo

C:\django\demo>python manage.py makemigrations
Migrations for 'person':
  person\migrations\0001_initial.py
    - Create model PersonModel

C:\django\demo>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, person, 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 admin.0003_logentry_add_action_flag_choices... 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 auth.0009_alter_user_last_name_max_length... OK
  Applying person.0001_initial... OK
  Applying sessions.0001_initial... OK

C:\django\demo>
```

（*）runMigrate.py

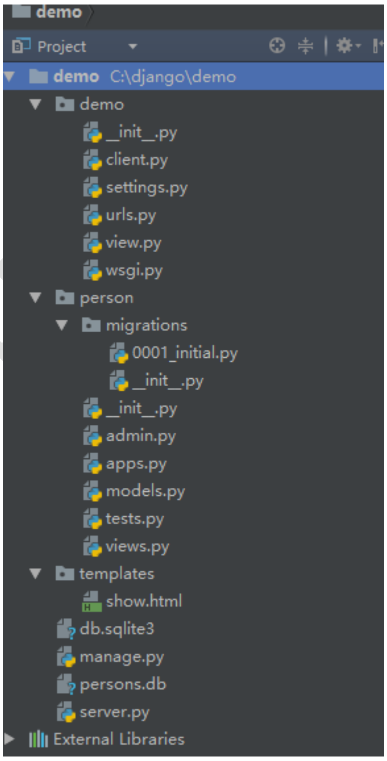
```
import os
os.system("python manage.py makemigrations")
os.system("python manage.py migrate")
```


File Edit Selection View Go Debug ··· runMigrate.py - demo - Visual Studio Code ...
DEBUG AND RUN: START
Debug with Python
Run with Python
To futher configure Debug and Run create a launch.json file.
runMigrate.py
form.py
show.h
TERMINAL
2: Python Debug Consc
Python 3.6.0 64-bit ⊗ 5 ⚠ 0
Ln 4, Col 1 Spaces: 4 UTF-8 CRLF Python

```
import os
os.system("python manage.py makemigrations")
os.system("python manage.py migrate")
```

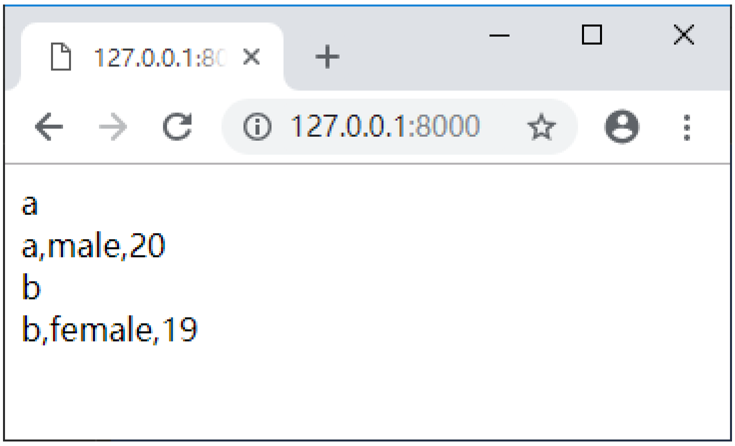
```
    - Create model PersonModel
Operations to perform:
  Apply all migrations: admin, auth, contenttypes
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add...
  Applying admin.0003_logentry_add_action_flag_ch
  Applying contenttypes.0002_remove_content_type_
```

(*)



```
demo  C:\django\demo
  demo
    __init__.py
    client.py
    settings.py
    urls.py
    view.py
    wsgi.py
  person
    migrations
      0001_initial.py
      __init__.py
    __init__.py
    admin.py
    apps.py
    models.py
    tests.py
    views.py
  templates
    show.html
  db.sqlite3
  manage.py
  persons.db
  server.py
External Libraries
```

(*)



```
a
a,male,20
b
b,female,19
```

```
<class 'django.db.models.query.QuerySet'>
<class 'person.models.PersonModel'>
```

即 **persons** 是一个 **QuerySet** 列表对象，列表的每个元素是一个 **PersonModel** 对象。

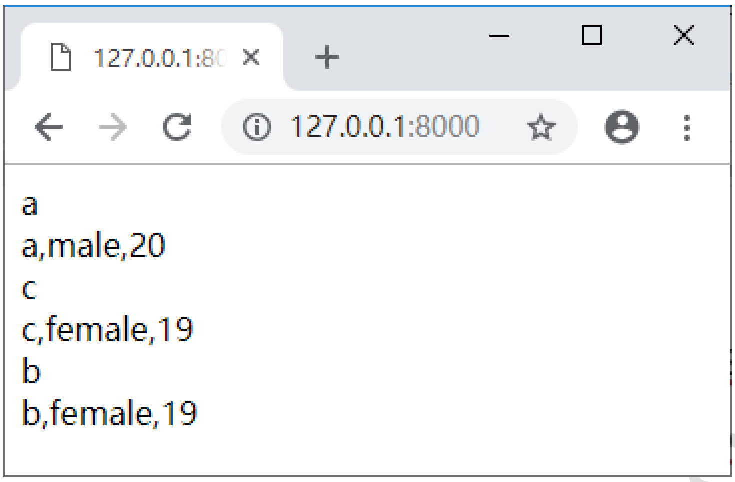
(*) 我们可以把程序编写在 **person\views.py**

```python
from django.shortcuts import render
from person.models import PersonModel

def show(request):
    persons=PersonModel.objects.all()
    PersonModel.objects.all().delete()
    PersonModel.objects.create(Name="a",Sex="male",Age=20)
    PersonModel.objects.create(Name="c", Sex="female", Age=19)
    PersonModel.objects.create(Name="b", Sex="female", Age=19)
    persons=PersonModel.objects.all()
    print(type(persons))
    print(type(persons[0]))
    return render(request,"show.html",locals())
```

(*) urls.py

```python
from django.conf.urls import url
import person.views

urlpatterns = [
    url(r'^$', person.views.show),
]
```

结果：



```
a
a,male,20
c
c,female,19
b
b,female,19
```

(*)

由此可见 demo 目录的 view.py 与 person 中的 views.py 都能一样的组织数据。

重要：无论程序在什么目录下，所有的引用都是从根目录（即 c:\web\demo)往下看的，例如 demo\view.py 与 person\views.py 都使用：

```
from person.models import PersonModel
```

因为 PersonModel 即在从根目录开始的 person 目录的 models.py 中，而无论 views.py 在哪里，在引用时都是从根目录开始的，因此都是 from person.models import PersonModel。

(*)urls.py

使用 person\views.py 时使用：

```
from django.conf.urls import url
import person.views
urlpatterns = [
    url(r'^$', person.views.show),
]
```

即：import person.views

(*)urls.py

使用 demo 目录 views.py 时使用：

```
from django.conf.urls import url
import demo.view
urlpatterns = [
    url(r'^$', demo.view.show),
]
```

但是由于 urls.py 在 demo 目录中，与 demo\view.py 同一个目录，因此也可以写出：

```
from django.conf.urls import url
from . import view
urlpatterns = [
    url(r'^$', view.show),
]
```

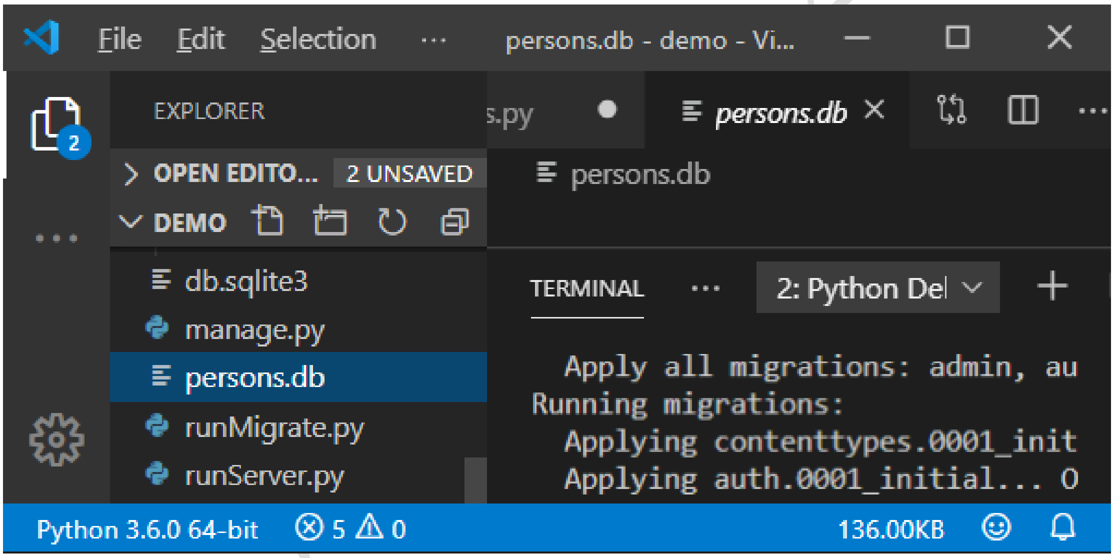
其中 from . import view 表示从与 urls.py 相同的目录（即 demo)中引入 view.py，效果与下列一样：

```
from django.conf.urls import url
from demo import views
urlpatterns = [
    url(r'^$', views.show),
]
```

(*) 最后结果：

(1) 按 settings.py 确定的那样创建了 persons.db 数据库

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'persons.db'),
    }
}
```



```
  Apply all migrations: admin, au
Running migrations:
  Applying contenttypes.0001_init
  Applying auth.0001_initial... O
```

(2) 并按 PersonModel 确定的那样创建表格，表格有(Name,Sex,Age)

```
class PersonModel(models.Model):
    Name=models.CharField(primary_key=True,max_length=30)
    Sex = models.CharField(max_length=30)
    Age=models.IntegerField()
```

(3) 创建了三条记录

```
def show(request):
    persons=PersonModel.objects.all()
    PersonModel.objects.all().delete()
    PersonModel.objects.create(Name="a",Sex="male",Age=20)
    PersonModel.objects.create(Name="c", Sex="female", Age=19)
    PersonModel.objects.create(Name="b", Sex="female", Age=19)
    persons=PersonModel.objects.all()
    return render(request,"show.html",locals())
```

由此可见

(1) Django 通过定义 PersonModel，在 PersonModel 中定义字段。

(2) 通过执行：

python manage.py makemigrations

python manage.py migrate

那么会在数据库中确定一张与该 **PersonModel** 对应的数据库表 **person_personmodel**。

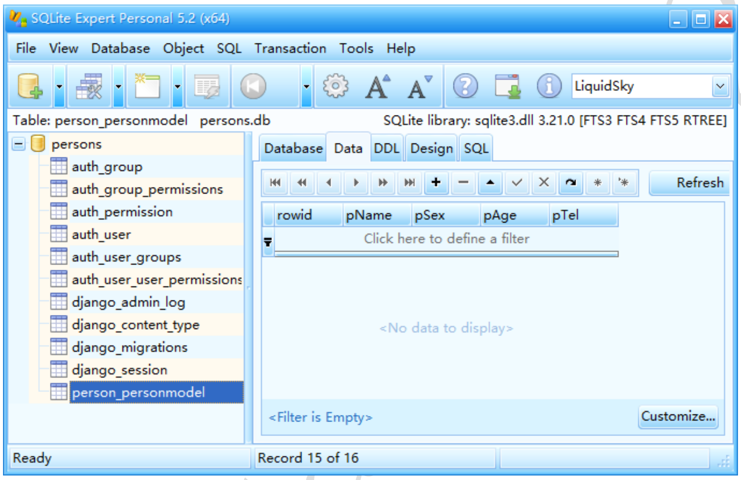

SQLite Expert Personal 5.2 (x64)
File View Database Object SQL Transaction Tools Help
LiquidSky
Database Data DDL Design SQL
Refresh
Customize...

Table: person_personmodel persons.db SQLite library: sqlite3.dll 3.21.0 [FTS3 FTS4 FTS5 RTREE]

- persons
  - auth_group
  - auth_group_permissions
  - auth_permission
  - auth_user
  - auth_user_groups
  - auth_user_user_permissions
  - django_admin_log
  - django_content_type
  - django_migrations
  - django_session
  - person_personmodel

rowid	pName	pSex	pAge	pTel

Click here to define a filter

<No data to display>

<Filter is Empty>

Ready Record 15 of 16

(3) 如果改变表的结构，那么要求：

(a) 删除 django_migrations 表中 app 值为 person 的所有行；

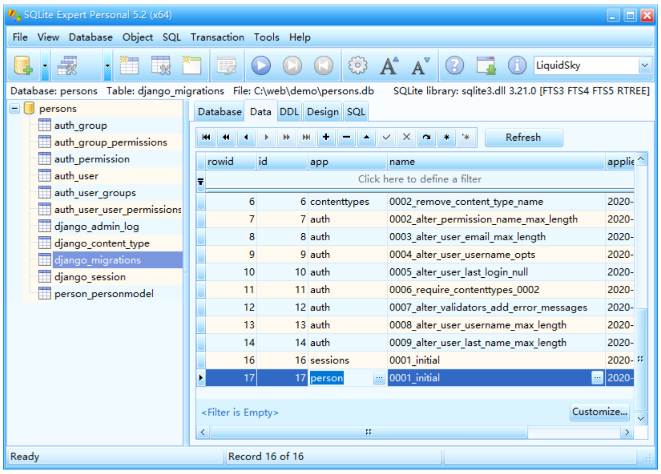
(b) 删除 person_personmodel 表;

(c) 删除 person\migrations 中的 001_XXX、002_XXX、....等所有文件;

(d) 再次执行这两个命令。

python manage.py makemigrations

python manage.py migrate


SQLite Expert Personal 5.2 (x64)
File View Database Object SQL Transaction Tools Help
LiquidSky
Database Data DDL Design SQL
Refresh
Customize...

Database: persons Table: django_migrations File: C:\web\demo\persons.db SQLite library: sqlite3.dll 3.21.0 [FTS3 FTS4 FTS5 RTREE]

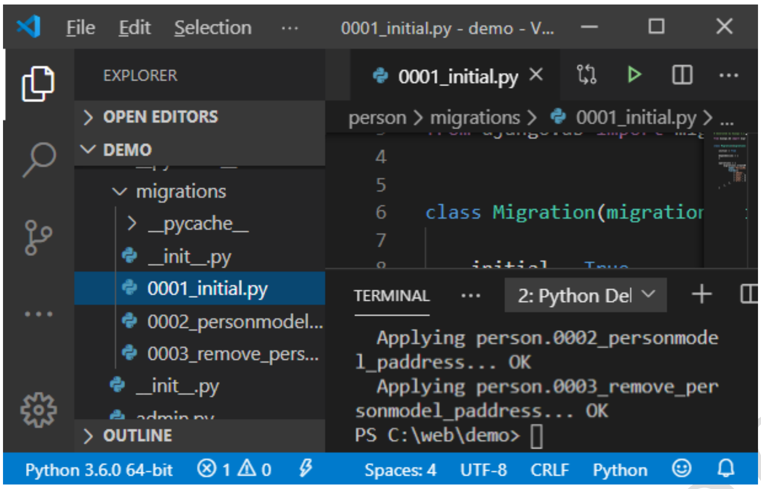
- persons
  - auth_group
  - auth_group_permissions
  - auth_permission
  - auth_user
  - auth_user_groups
  - auth_user_user_permissions
  - django_admin_log
  - django_content_type
  - django_migrations
  - django_session
  - person_personmodel

Click here to define a filter

rowid	id	app	name	applie
6	6	contenttypes	0002_remove_content_type_name	2020-
7	7	auth	0002_alter_permission_name_max_length	2020-
8	8	auth	0003_alter_user_email_max_length	2020-
9	9	auth	0004_alter_user_username_opts	2020-
10	10	auth	0005_alter_user_last_login_null	2020-
11	11	auth	0006_require_contenttypes_0002	2020-
12	12	auth	0007_alter_validators_add_error_messages	2020-
13	13	auth	0008_alter_user_username_max_length	2020-
14	14	auth	0009_alter_user_last_name_max_length	2020-
16	16	sessions	0001_initial	2020-
17	17	person	0001_initial	2020-

<Filter is Empty>

Ready Record 16 of 16



```
class Migration(migratior
```

```
  Applying person.0002_personmode
l_paddress... OK
  Applying person.0003_remove_per
sonmodel_paddress... OK
PS C:\web\demo> 
```

(3) Django 定义了一套通过：

PersonMode.objects.method(....)

操作数据库的方法，实现 **select**、**update**、**delete**、**insert** 等操作，避免我们直接取操作数据库。

例如获取所有记录：

persons=PersonModel.objects.all()

例如增加记录：

PersonModel.objects.create(Name="c", Sex="female", Age=19)