

# 第11章基于web编程

## 主要内容

- Django的安装
- Django编程的重要概念
- 基于Django编程的应用实例

## ===Djangle的安装===

### ● Django的安装

#### ◆ Django 下载地址:

<https://www.djangoproject.com/download/>

#### ◆ 下载 Django 压缩包，解压并和Python安装目录放在同一个根目录，进入 Django 目录，执行 python setup.py install

#### ◆ Django 检测是否安装成功

```
>>> import django  
>>> django.get_version()
```

#### ◆ 在线安装

```
pip install Django
```

# 第11章基于web编程

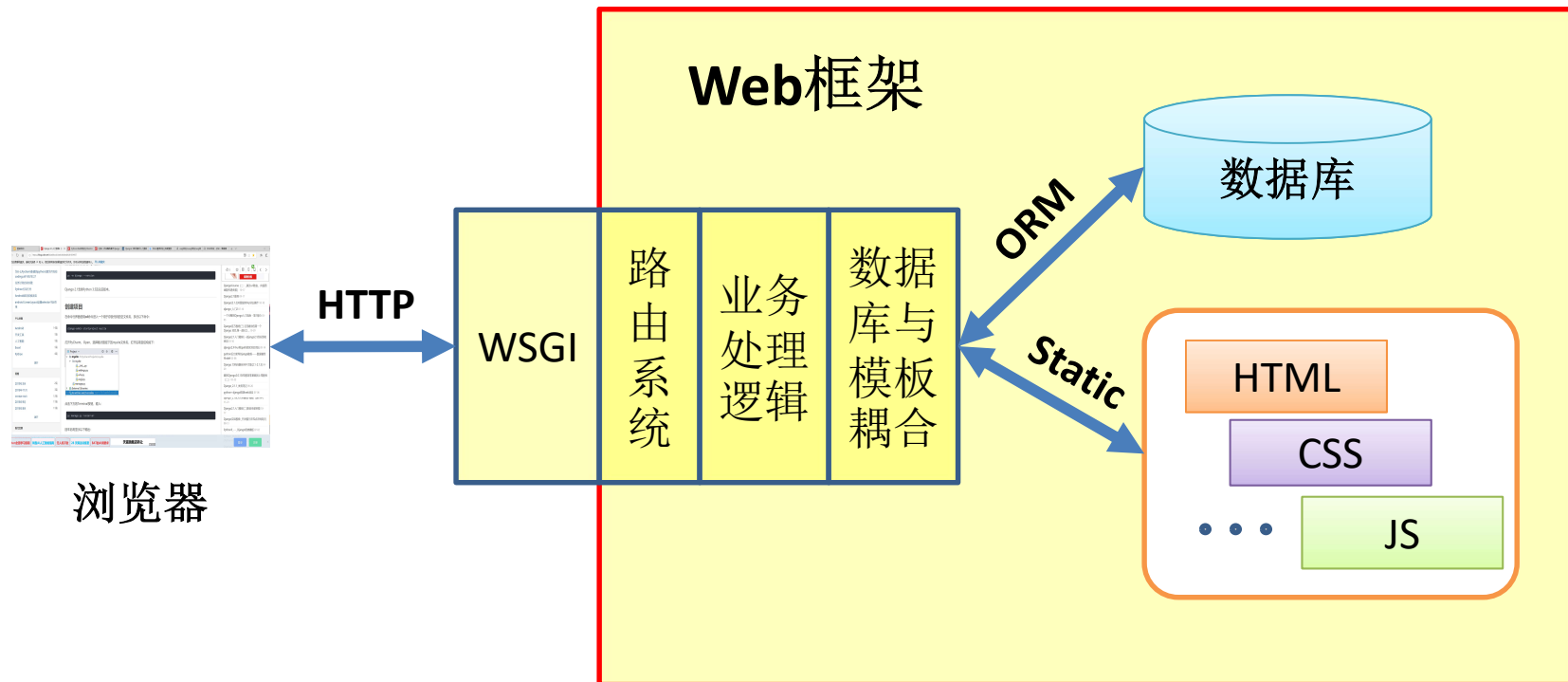
## 主要内容

- Django的安装
- Django编程的重要概念
- 基于Django编程的应用实例

## ===Django 编程的重要概念===

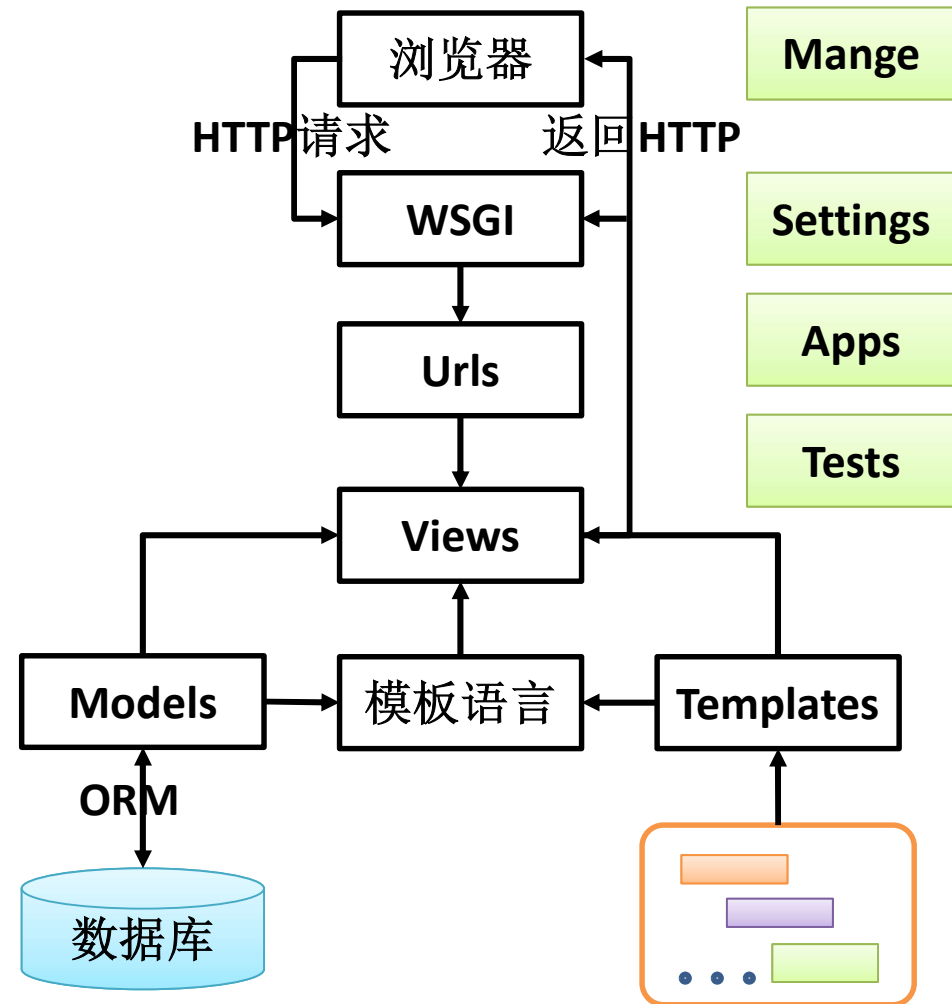
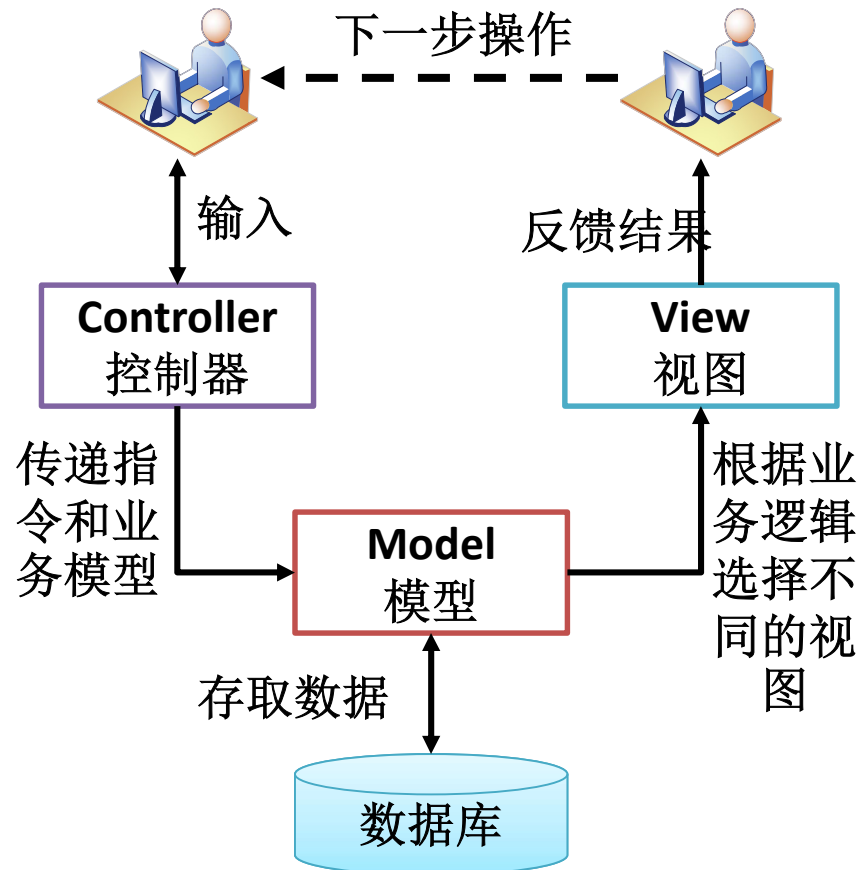
## ● Django的web框架

**含义：** Web网站模板，利用它的规则，构造或定制所需要的Web网站



## ===Django编程的重要概念===

## ● MVC/MTV



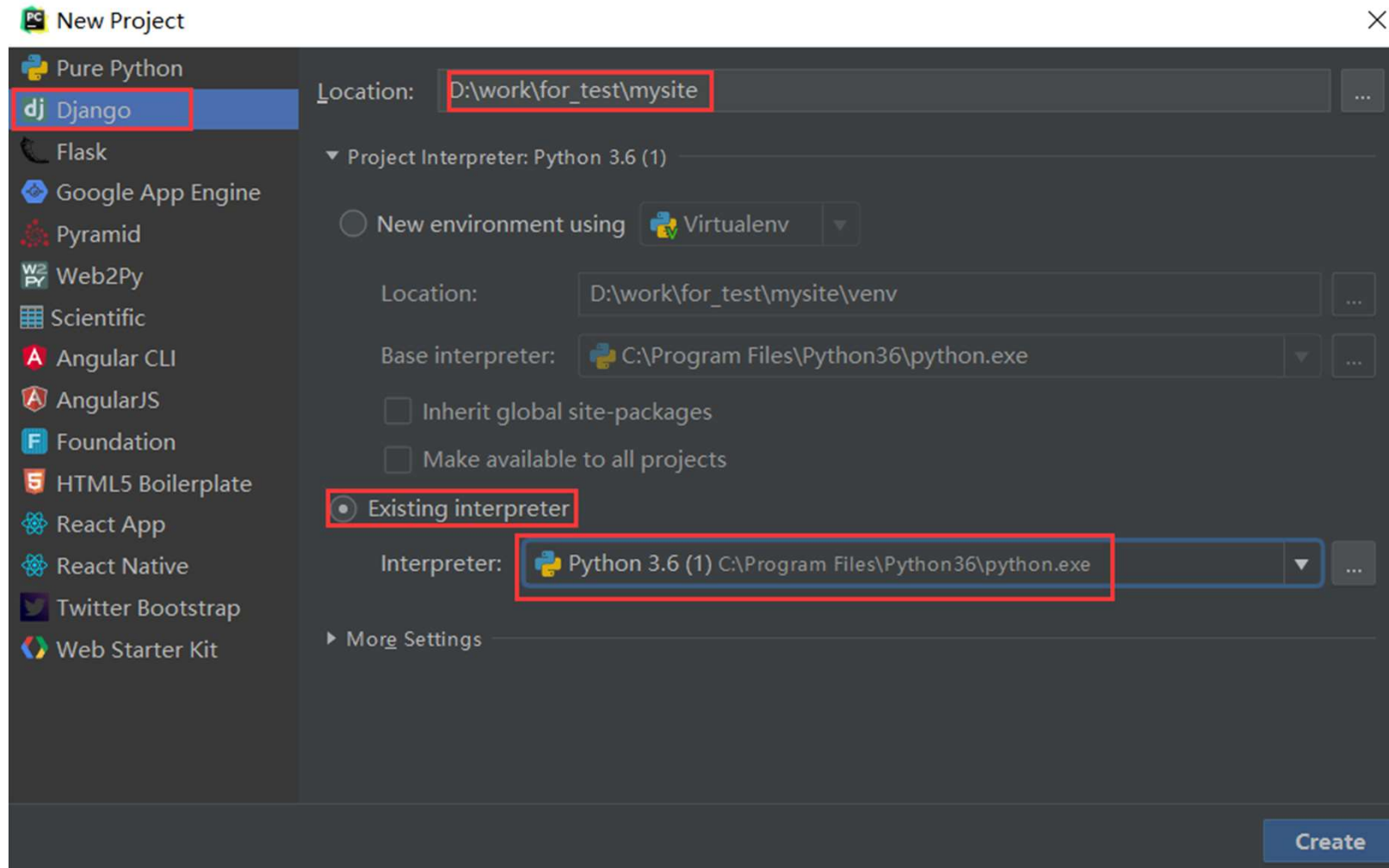
# 第11章基于web编程

## 主要内容

- Django的安装
- Django编程的重要概念
- 基于Django编程的应用实例

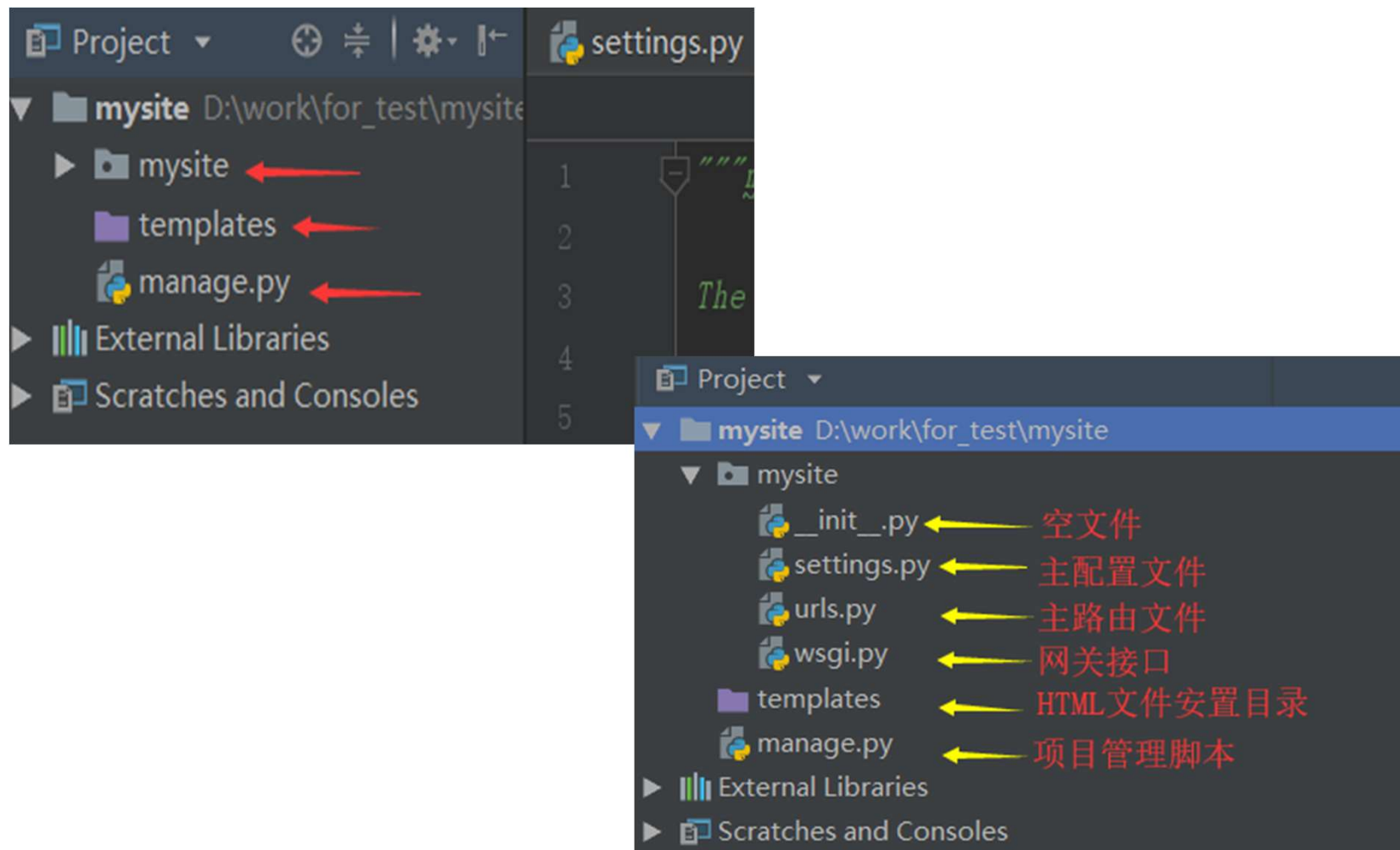
## ===基于Django编程的应用实例===

## ● 创建Web项目



## ===基于Django编程的应用实例===

## ● 基于Django Web的目录结构



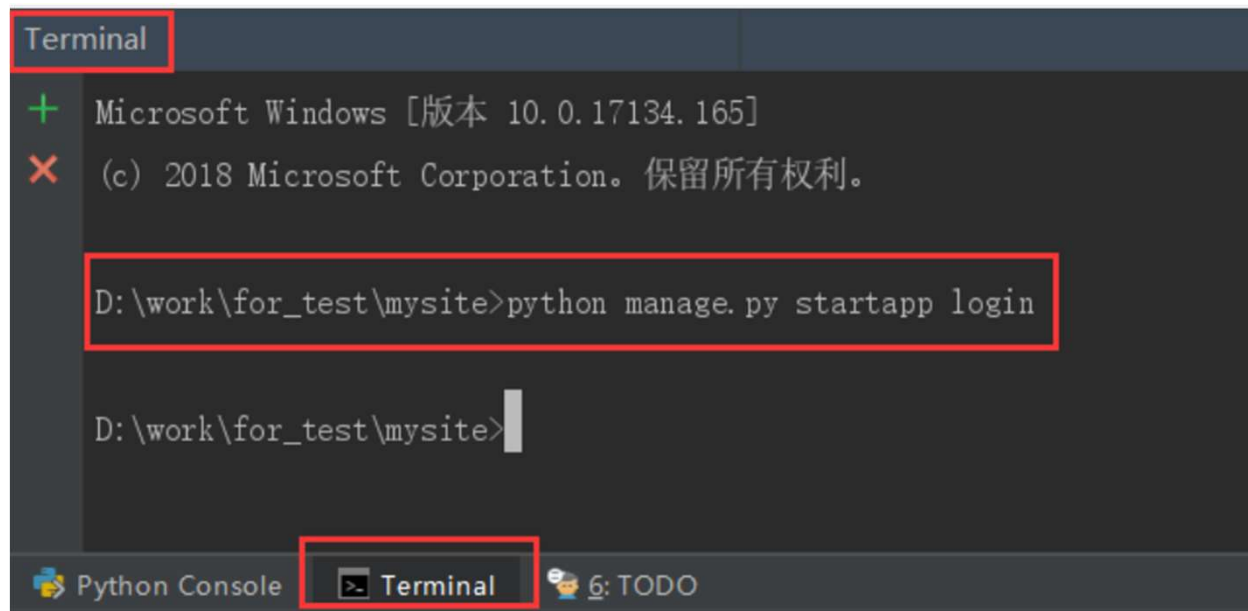


## ===基于Django编程的应用实例===

## ● 创建App项目

- Django项目：包含多个APP，相互之间比较独立，但也可以有联系。所有的APP共享项目资源。
- App生成指令

`python manage.py startapp login`



```
Terminal
+ Microsoft Windows [版本 10.0.17134.165]
(c) 2018 Microsoft Corporation. 保留所有权利。

D:\work\for_test\mysite>python manage.py startapp login

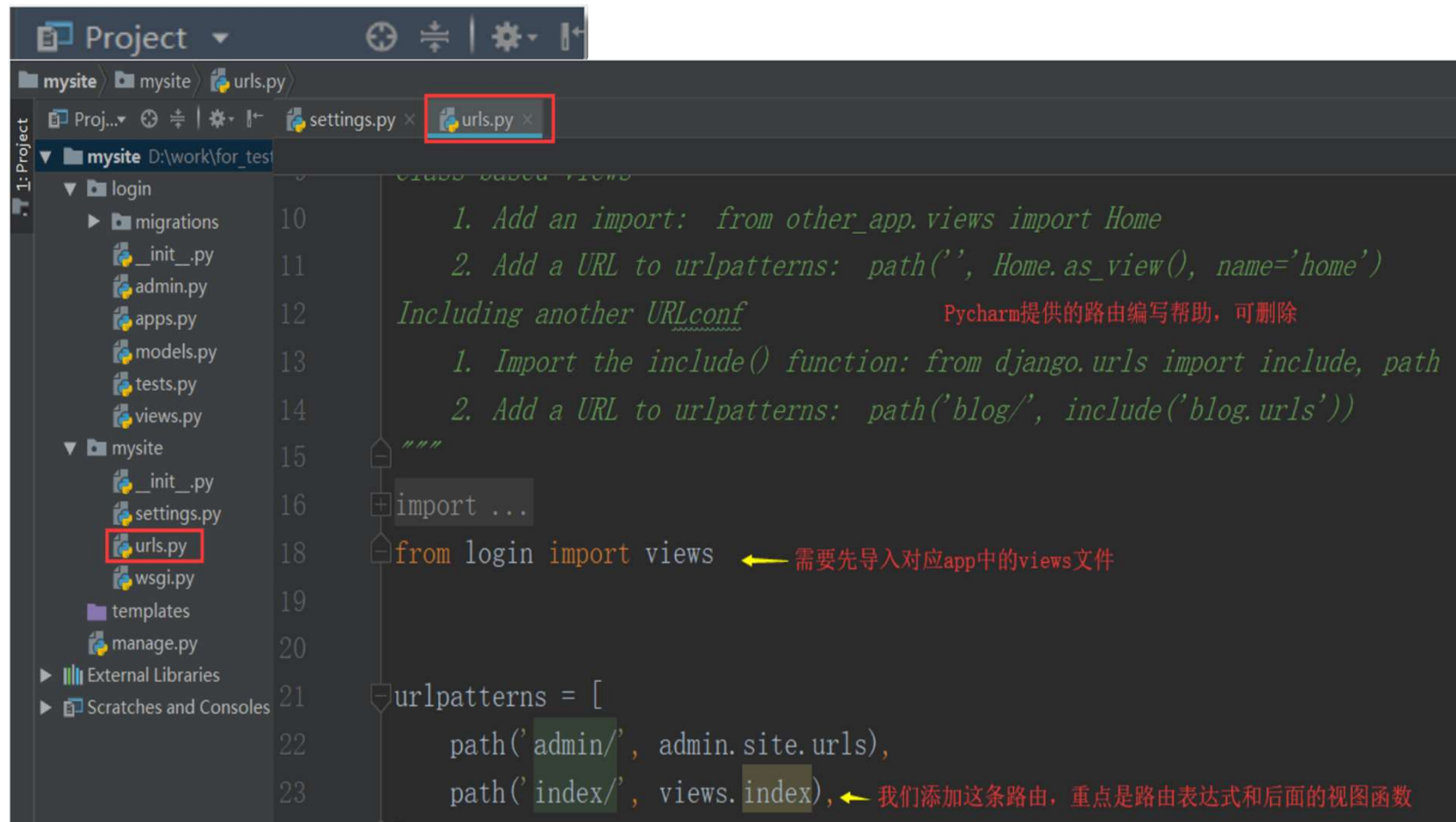
D:\work\for_test\mysite>
```

## ===基于Django编程的应用实例===

## ● 创建App项目

## ➤ App的目录结构

## ➤ 编写路由



## ===基于Django编程的应用实例===

## ● 创建App项目

### ➤ 编写视图函数

```
1 from django.shortcuts import render
2 from django.shortcuts import HttpResponse
3
4 # Create your views here.
5
6
7 def index(request):
8     return HttpResponse('Hello World!')
```

导入这个模块

第一个参数必须是request, 名字可以改, 但是最好不要, 这是潜规则。request参数封装了用户请求的所有内容。

不能直接返回字符串, 必须由这个类封装起来, 才能被HTTP协议识别。

## ===基于Django编程的应用实例===

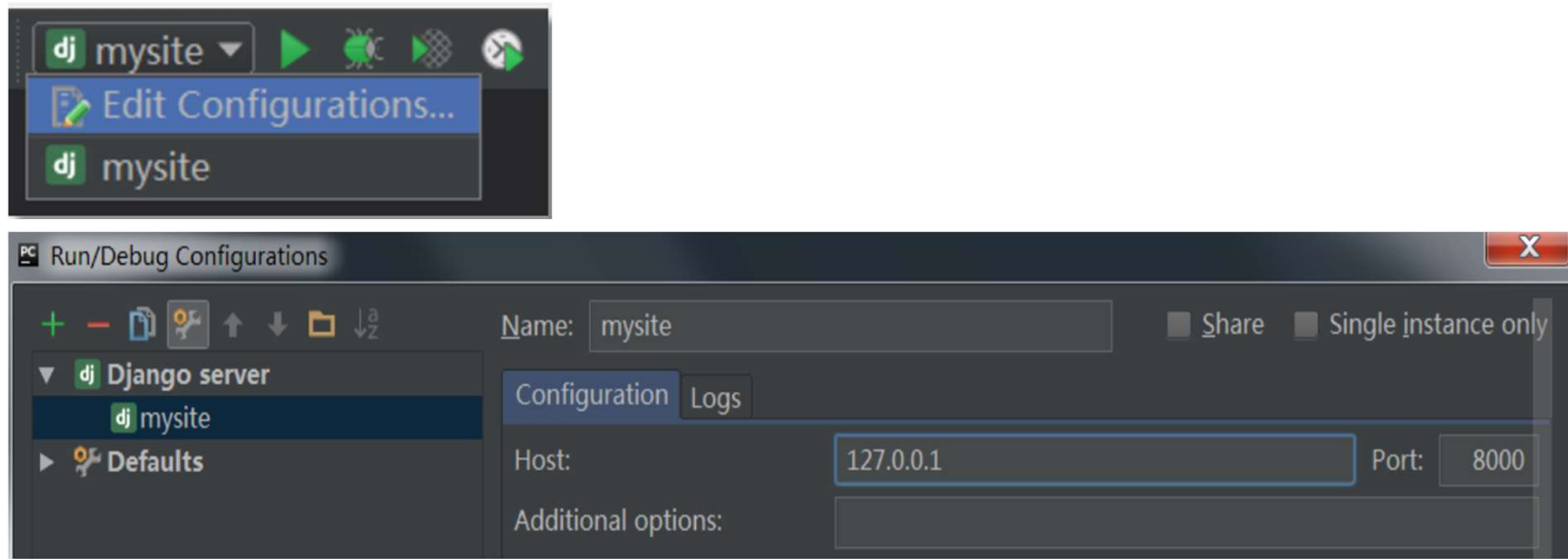
## ● 创建App项目

### ➤ 运行Web服务

#### (1) 命令行的方式

```
python manage.py runserver 127.0.0.1:8020
```

#### (2) 界面的方式



## ===基于Django编程的应用实例===

## ● 创建App项目

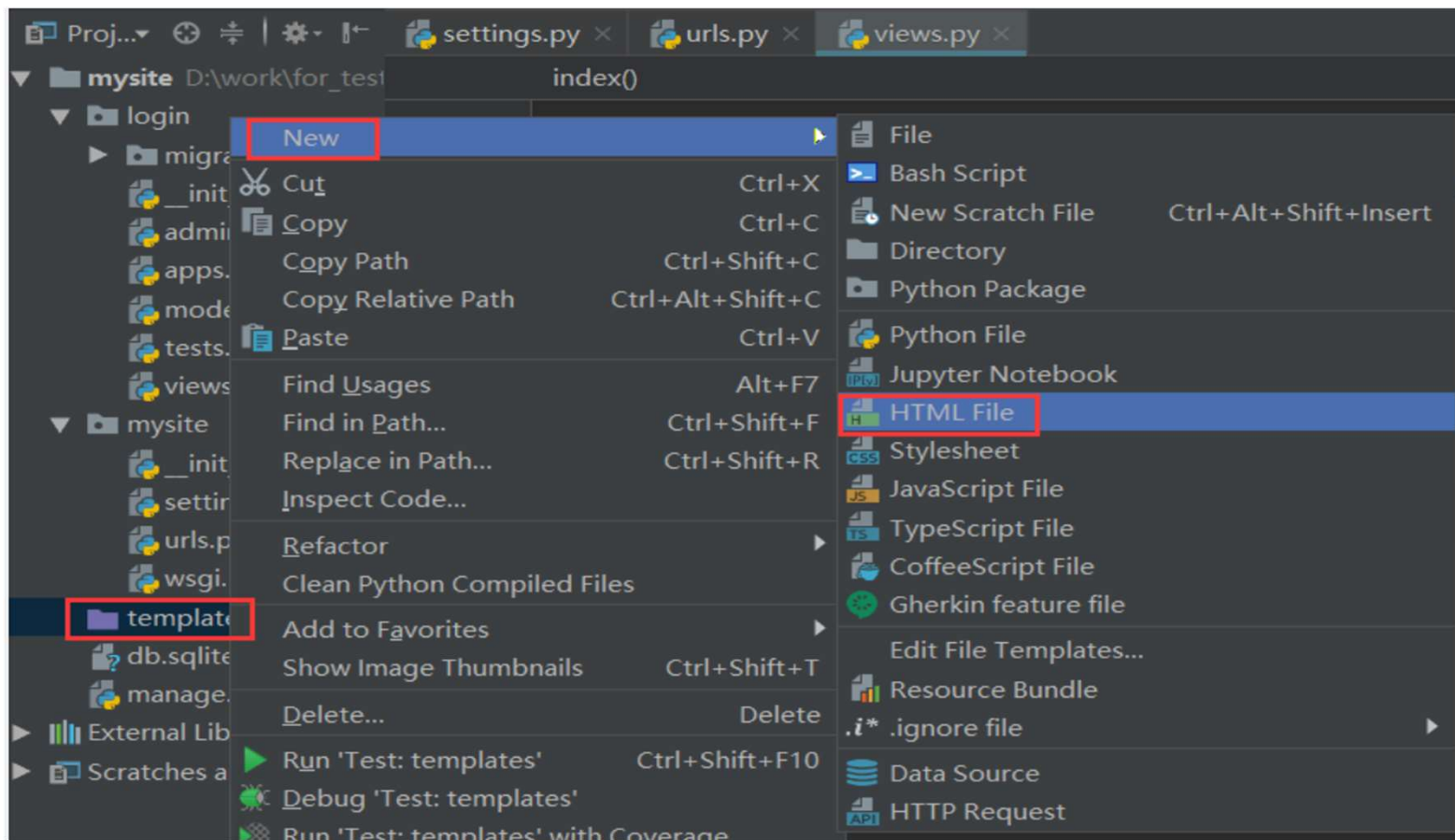
## ➤ Web服务访问



## ===基于Django编程的应用实例===

## ● 创建App项目

## ➤ 返回HTML文件（index.html）





## ===基于Django编程的应用实例===

## ● 创建App项目

### ➤ 返回HTML文件

```
<!DOCTYPE html>
<html lang="en">
  <head> <meta charset="UTF-8">
    <title>test</title>
  </head>
  <body>
    <h1 style="background-color: antiquewhite;color: black">Hello World!</h1>
  </body>
</html>
```

## ===基于Django编程的应用实例===

## ● 创建App项目

## ➤ 返回HTML文件

修改views.py

```
1  from django.shortcuts import render 一般会导入这个模块
2  from django.shortcuts import HttpResponse
3
4  # Create your views here.
5
6
7  def index(request):
8      # return HttpResponse('Hello World!')
9      return render(request, 'index.html')
10
```



## ===基于Django编程的应用实例===

## ● 创建App项目

## ➤ 返回HTML文件

修改settings.py

重启Web服务，再次访问



## ===基于Django编程的应用实例===

## ● 创建App项目

### ➤ 与用户交互

#### (1)修改index.html

```
<!DOCTYPE html>
<html lang="en">
  <head> <meta charset="UTF-8">
    <title>首页</title>
  </head>
  <body>
    <h1>用户输入: </h1>
    <form action="/index/" method="post">
      {% csrf_token %} <!-- 加入这行 -->
      用户名: <input type="text" name="username" /><br />
      密码: <input type="password" name="password" /><br />
      <input type="submit" value="提交" />
    </form>
  </body>
</html>
```

## ===基于Django编程的应用实例===

## ● 创建App项目

## ➤ 与用户交互

## (2)修改views.py

```
from django.shortcuts import render
from django.shortcuts import HttpResponseRedirect
# Create your views here.
def index(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')
        print(username, password)
    return render(request, 'index.html')
```

## ===基于Django编程的应用实例===

## ● 创建App项目

### ➤ 与用户交互—返回动态页面

(1)修改index.html—在from之后，</body>之前加入

```
<h1>用户展示: </h1>
<table border="1">
  <thead> <tr>用户名</tr> <tr>密码</tr> </thead>
  <tbody> {% for item in data %}
    <tr>
      <td>{{ item.user }}</td>
      <td>{{ item.pwd }}</td>
    </tr>
  {% endfor %}
</tbody>
</table>
```

## ===基于Django编程的应用实例===

## ● 创建App项目

## ➤ 与用户交互—返回动态页面

## (2)修改views.py

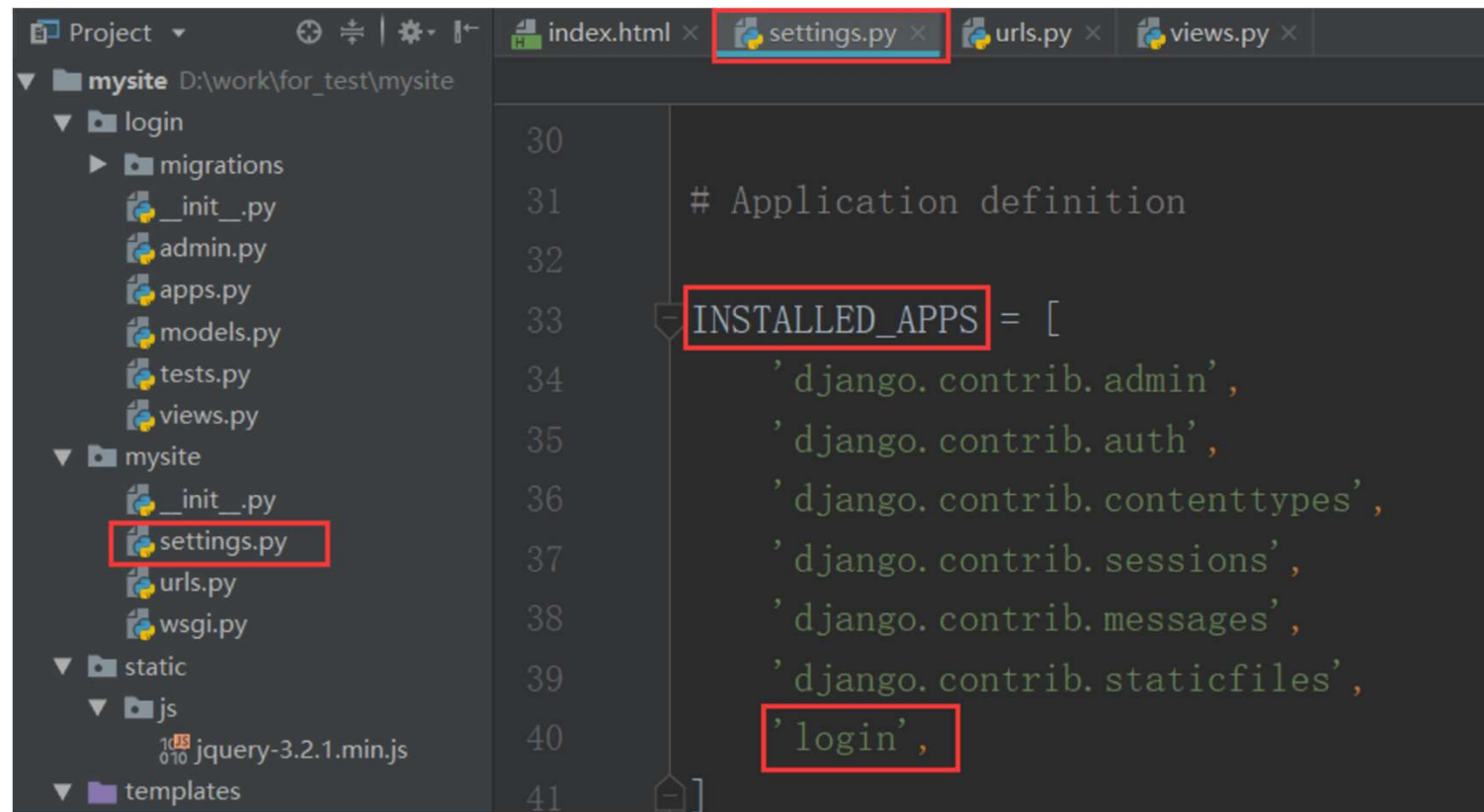
```
user_list = []  
def index(request):  
    if request.method == 'POST':  
        username = request.POST.get('username')  
        password = request.POST.get('password')  
        print(username, password)  
        temp = {'user': username, 'pwd': password}  
        user_list.append(temp)  
    return render(request, 'index.html', {'data': user_list})
```

## ===基于Django编程的应用实例===

## ● 创建App项目

### ➤ 与用户交互—使用数据库

#### (1)需要注册app



## ===基于Django编程的应用实例===

## ● 创建App项目

## ➤ 与用户交互—使用数据库

## (2)配置数据库相关的参数

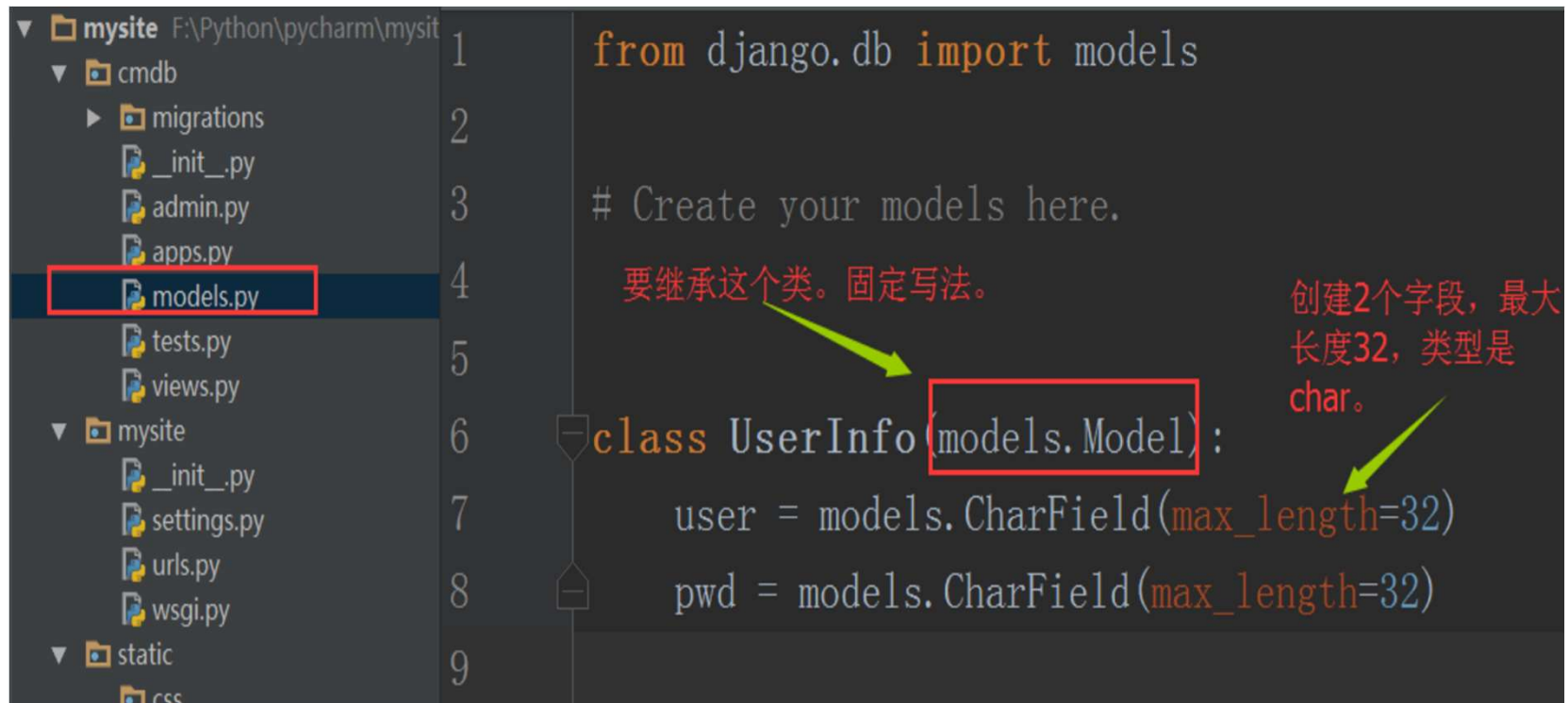


## ===基于Django编程的应用实例===

## ● 创建App项目

➤ 与用户交互—使用数据库

(3)编辑models.py文件, MTV--M





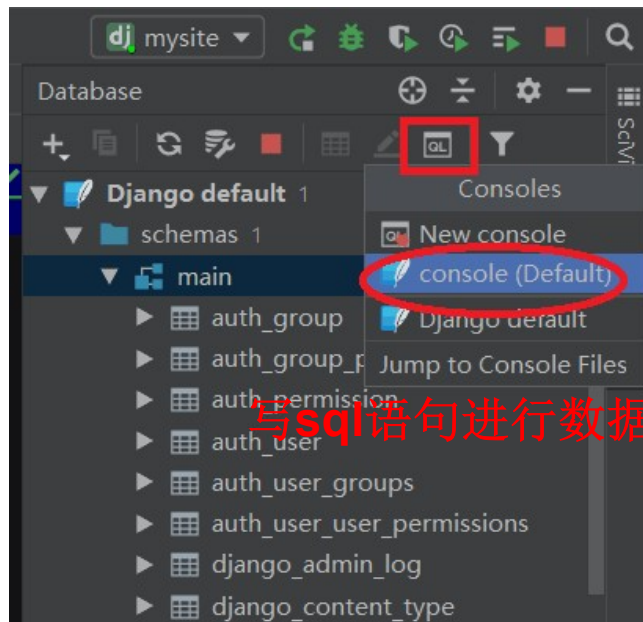
## ===基于Django编程的应用实例===

## ● 创建App项目

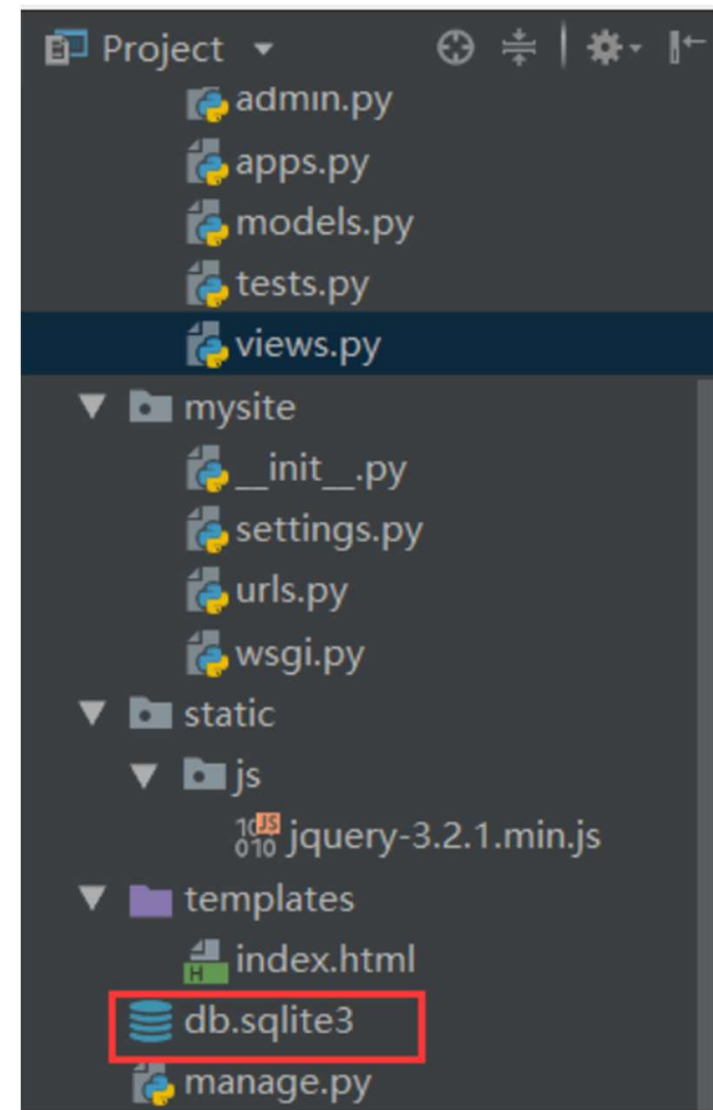
➤ 与用户交互—使用数据库

(4)创建数据库表

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



与sql语句进行数据库检查



## ===基于Django编程的应用实例===

## ● 创建App项目

➤ 与用户交互—使用数据库

## (4)创建数据库表

The screenshot displays a Django application interface. On the left, a table named 'login\_userinfo' is shown with two columns: 'user' and 'pwd'. The table contains two rows of data:

	user	pwd
1	20016009	123
2	I am father	321

A red arrow points to the table. On the right, a 'Database' sidebar shows the 'Django default' database schema. The 'main' schema is expanded, and the 'login\_userinfo' table is highlighted with a red circle. The 'Database' sidebar is also highlighted with a red box.

## ===基于Django编程的应用实例===

## ● 创建App项目

## ➤ 与用户交互—使用数据库

## (5)修改views.py

```
def index(request):  
    if request.method == 'POST':  
        username = request.POST.get('username')  
        password = request.POST.get('password') #  
            将数据保存到数据库  
        models.UserInfo.objects.create(  
            user=username, pwd=password)  
    # 从数据库中读取所有数据，注意缩进  
    user_list = models.UserInfo.objects.all()  
    return render(request, 'index.html', {'data': user_list})
```

# 本章小结

- Django的安装
- Django编程的重要概念
- 基于Django编程的应用实例