

Django学习笔记

以下为Windows环境

0. 下载Django

```
pip install Django
```

1. Hello World：在VScode上创建并运行一个Django项目

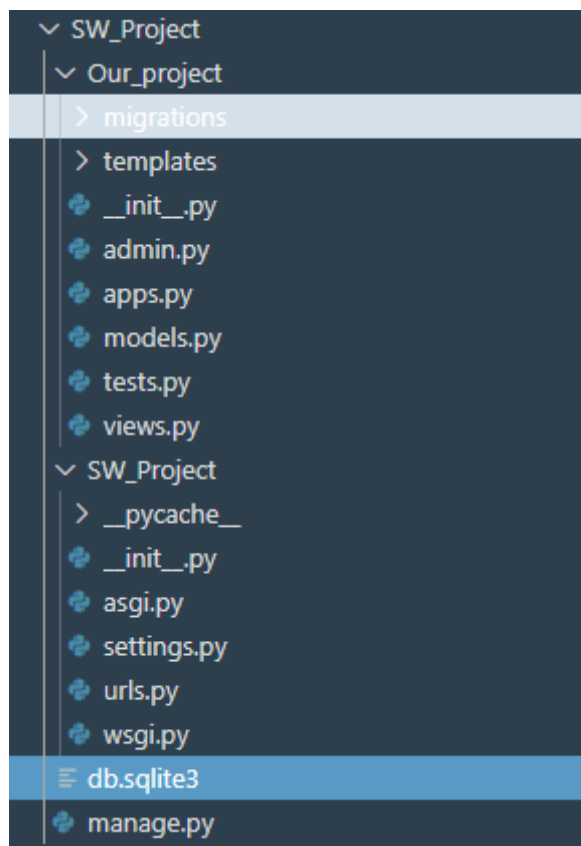
- 创建一个文件夹，并在VScode中打开文件夹，在VSC中打开terminal
- 创建一个项目：

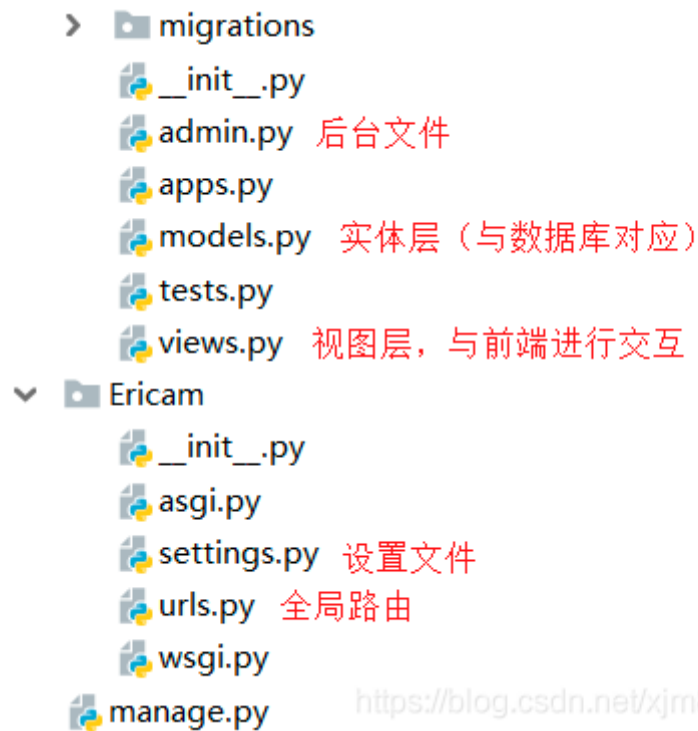
```
django-admin startproject SW_Project  
cd SW_Project
```

创建一个app:

```
python manage.py startapp Our_project  
cd Our_project  
mkdir templates //在项目建立一个新的文件夹
```

会看到这样的项目结构：





(图源：见水印)

- 项目运行：

```
python manage.py runserver
```

```
PS C:\Users\LENOVO\Desktop\4001_SW_Project\SW_Project> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 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.
February 23, 2022 - 19:06:23
Django version 3.2.3, using settings 'SW_Project.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

若运行成功会返回一个url（我这里返回的是<http://127.0.0.1:8000/>），证明这个项目在本地的服务器上跑起来了，打开这个即可看到页面。

3. 建一个简单的网站

- (1) 配置一下setting

setting.py中加入刚刚创的app来告诉框架加入了这样一个新的app

```
settings.py ×
SW_Project > SW_Project > settings.py > ...
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'Our_project'
41 ]
```

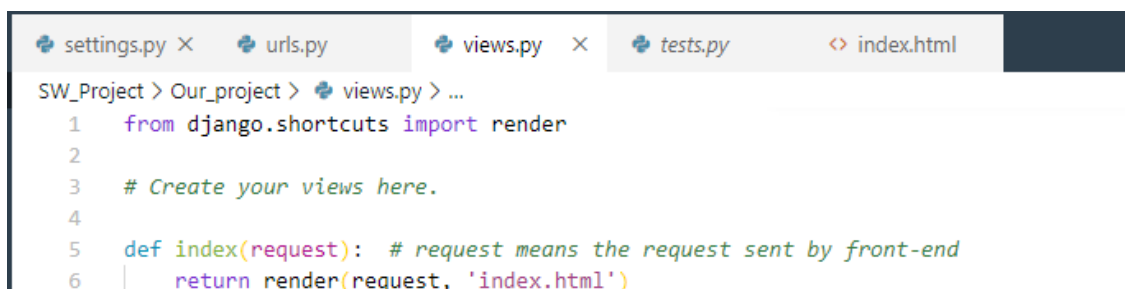
(2) 上手

- 首先在/Our_Project/templates中创建一个html文件作为网页要展示的面



```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>The Page!!</title>
5   </head>
6   <body>
7     <h1>This is a Heading</h1>
8     <p>This is a paragraph.</p>
9   </body>
10 </html>
```

接着在/Our_project/views中写一个函数，这个函数会调用这个html文件：



```
1 from django.shortcuts import render
2
3 # Create your views here.
4
5 def index(request): # request means the request sent by front-end
6     return render(request, 'index.html')
```

- 这里的参数request代表从前端发送过来的请求（之后会用到）

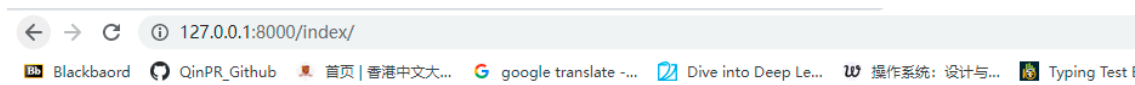
这个函数又会被url调用，所以去/SW_Project/urls加上，这样当我们在浏览器中输入对应的url，就会自动调用index这个函数，接着这个函数又会返回index.html给网页



```
14 2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
15 """
16 from django.contrib import admin
17 from django.urls import path
18 from Our_project import views
19
20 urlpatterns = [
21     path('admin/', admin.site.urls),
22     path('index/', views.index)
23 ]
24
```

(记得import views)

最后，在浏览器中输入对应的url (localhost:8000/index) 就能看到效果：



This is a Heading

This is a paragraph.

4. 与前端交互

刚刚做的仅仅是后端在自娱自乐，并没有和前端的交互，一般情况下前端会发送数据（比如输入用户名和密码）给后端，后端做出处理后再返回结果给前端。

这里，简单地做一个登陆页面来展示后端如何处理前端数据并与前端交互。

(1) 登陆页面：

- 修改刚刚的Our_project/templates/index.html



```

SW_Project > Our_project > templates > <> index.html > html
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>The Page!!</title>
5      </head>
6      <body>
7          <h1>This is a Heading</h1>
8          <p>This is a paragraph.</p>
9          <form method="POST" action="/login/">
10             {% csrf_token %}
11             STUDENT ID: <input type="number" name="stuid">
12             PASSCODE: <input type="number" name="passcode">
13             <input type="submit" name="login" >
14         </form>
15     </body>
16 </html>

```

这里前端会以stuid和passcode两个变量得到用户输入的两个变量，然后把url变为localhost:8000/login，然后用POST的方式发送给后端。

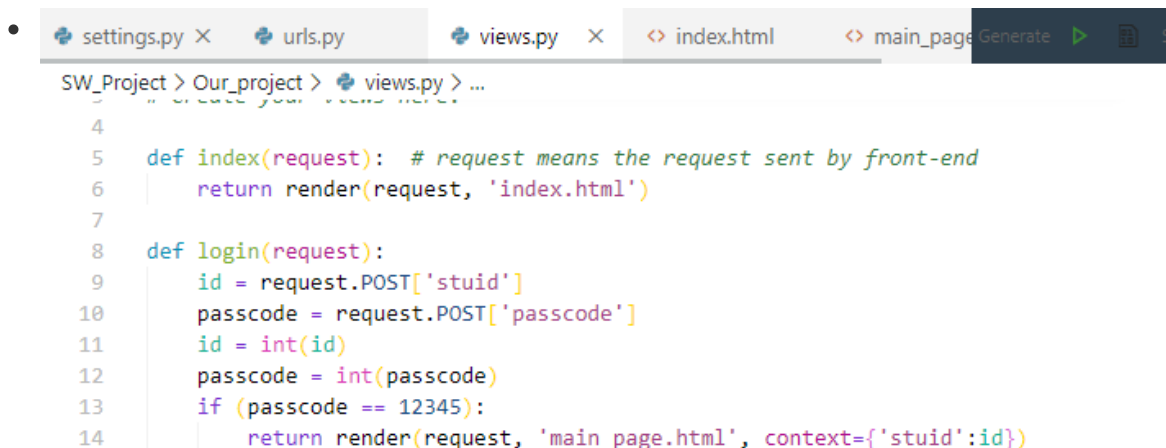
- 后端的处理从url开始，先在/SW_Project/urls.py添加localhost:8000/login和views.py中函数的连接：

```

urlpatterns = [
    path('admin/', admin.site.urls),
    path('index/', views.index),
    path('login/', views.login)
]

```

前端的请求将会由views.py中的login函数处理，所以接下来写login函数。



```

SW_Project > Our_project > views.py > ...
4
5  def index(request): # request means the request sent by front-end
6      return render(request, 'index.html')
7
8  def login(request):
9      id = request.POST['stuid']
10     passcode = request.POST['passcode']
11     id = int(id)
12     passcode = int(passcode)
13     if (passcode == 12345):
14         return render(request, 'main_page.html', context={'stuid':id})

```

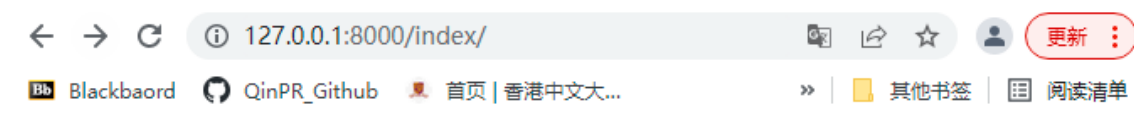
用request.POST['XXX']得到前端发送过来的请求中的变量。判断密码是否为12345，如果是则返回登陆成功的主页面main_page.html，并带上变量'stuid'(变量值为id)。

- 接着在templates/下写个main_page.html的页面：

```
settings.py  urls.py  views.py  index.html  main_page.html x
SW_Project > Our_project > templates > main_page.html > html
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>The Page!!</title>
5      </head>
6      <body>
7          <h1>Login success with id: {{stuid}}</h1>
8      </body>
9  </html>
```

- 注意，这里用{{变量名}}的方法接收后端发送回来的变量。
- 整个流程可以概括为：
index.html中接受用户输入，并改变url，发送数据 -> urls.py将对应url映射到views.py的login函数上 -> login函数做出处理，返回新的页面main_page.html,并传回变量stuid -> main_page.html接受stuid并打印

- 最终效果是：

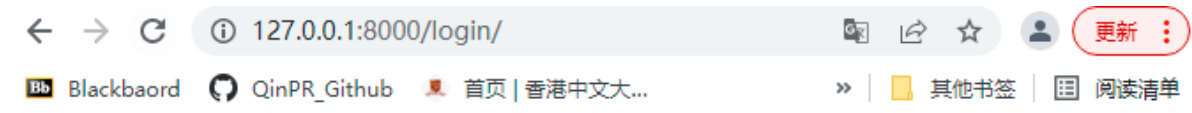


This is a Heading

This is a paragraph.

STUDENT ID: PASSCODE:

点提交后跳转到：



Login success with id: 119010249

5. 与数据库交互

在这里演示如何跟连接MySQL并进行数据库操作

- 先在mysql中创建一个空的schema
- 首先在settings.py中将DATABASES改成要连接的数据库
 - 'NAME'为schema的名称
 - 'USER', 'PASSWORD'对应数据库的用户密码
 - 'HOST', 'PORT'是数据库的ip和端口号，如果是本地的数据库就是127.0.0.1(mysql数据库默认用3306端口)

```
urls.py views.py models.py settings.py X _init_.py
SW_Project > SW_Project > settings.py > ...
72
73
74 # Database
75 # https://docs.djangoproject.com/en/3.2/ref/settings/#databases
76
77 DATABASES = {
78     'default': {
79         'ENGINE': 'django.db.backends.mysql',
80         'NAME': 'sw_db',
81         'USER': 'root',
82         'PASSWORD': 'Q@pr294118',
83         'HOST': '127.0.0.1',
84         'PORT': '3306',
85     }
86 }
```

- 在Our_project/_init_.py中加东西:

```
urls.py views.py models.py settings.py _init_.py ...\SW_Project _init_.py ...\Our_project X
SW_Project > Our_project > _init_.py > ...
1 import pymysql
2
3 pymysql.install_as_MySQLdb
```

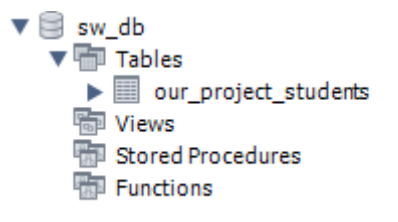
- 接着最重要的一步: 在/Our_projects/models.py中加入想加入数据库的表, 比如我想加一个students的表, 表里包括id和密码:

```
urls.py views.py models.py X settings.py _init_.py ...\SW_Project _init_.py
SW_Project > Our_project > models.py > ...
1 from django.db import models
2
3 # Create your models here.
4
5 # describe the Database
6 class students(models.Model):
7     user_id = models.CharField(max_length=10)
8     passcode = models.CharField(max_length=20)
```

- 接着我们将这个表migrate到mysql数据库中

```
python manage.py makemigration
python manage.py migrate
```

将会看到mysql中原本sw_db的tables里原本是空的, 现在多了Our_project_students这个table:



- 成功与数据库建立连接并加入了表, 接下来进行一个简单的往表里加一条信息的操作, 基于上面views.py里修改一下, 让它在登陆的时候, 将id和passcode加入到数据库student的表中:
 - 先把表import进来:

```
urls.py views.py X models.py settings.py _init_.py ...\SW_Project
SW_Project > Our_project > views.py > ...
1 from django.shortcuts import render
2 from Our_project.models import students
```

- 接着在login函数中加一行create的代码:

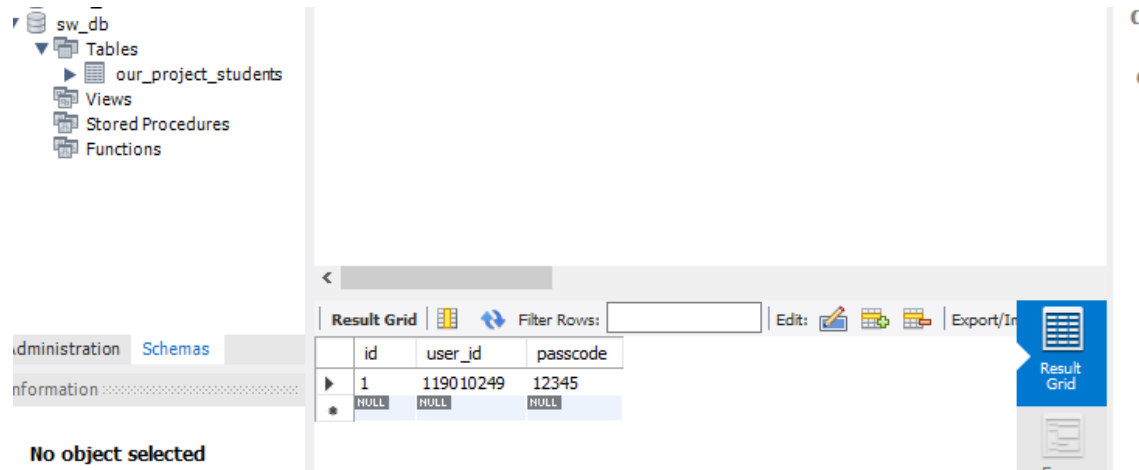
```
def login(request):
    id = request.POST['stuid']
    passcode = request.POST['passcode']

    students.objects.create(user_id = id, passcode = passcode)

    id = int(id)
    passcode = int(passcode)

    if (passcode == 12345):
        return render(request, 'main_page.html', context={'stuid':id})
```

- 效果是，当用户输入id, passcode登陆后，可以在数据库中看到多了一个条目：



The screenshot shows a database management interface. On the left, a tree view displays the database structure: 'sw_db' contains 'Tables', 'Views', 'Stored Procedures', and 'Functions'. Under 'Tables', 'our_project_students' is selected. The main area shows a 'Result Grid' with the following data:

	id	user_id	passcode
▶	1	119010249	12345
*	NULL	NULL	NULL

Below the table, it says 'No object selected'. The interface also includes a 'Filter Rows' field and buttons for 'Edit', 'Export/Import', and 'Result Grid'.