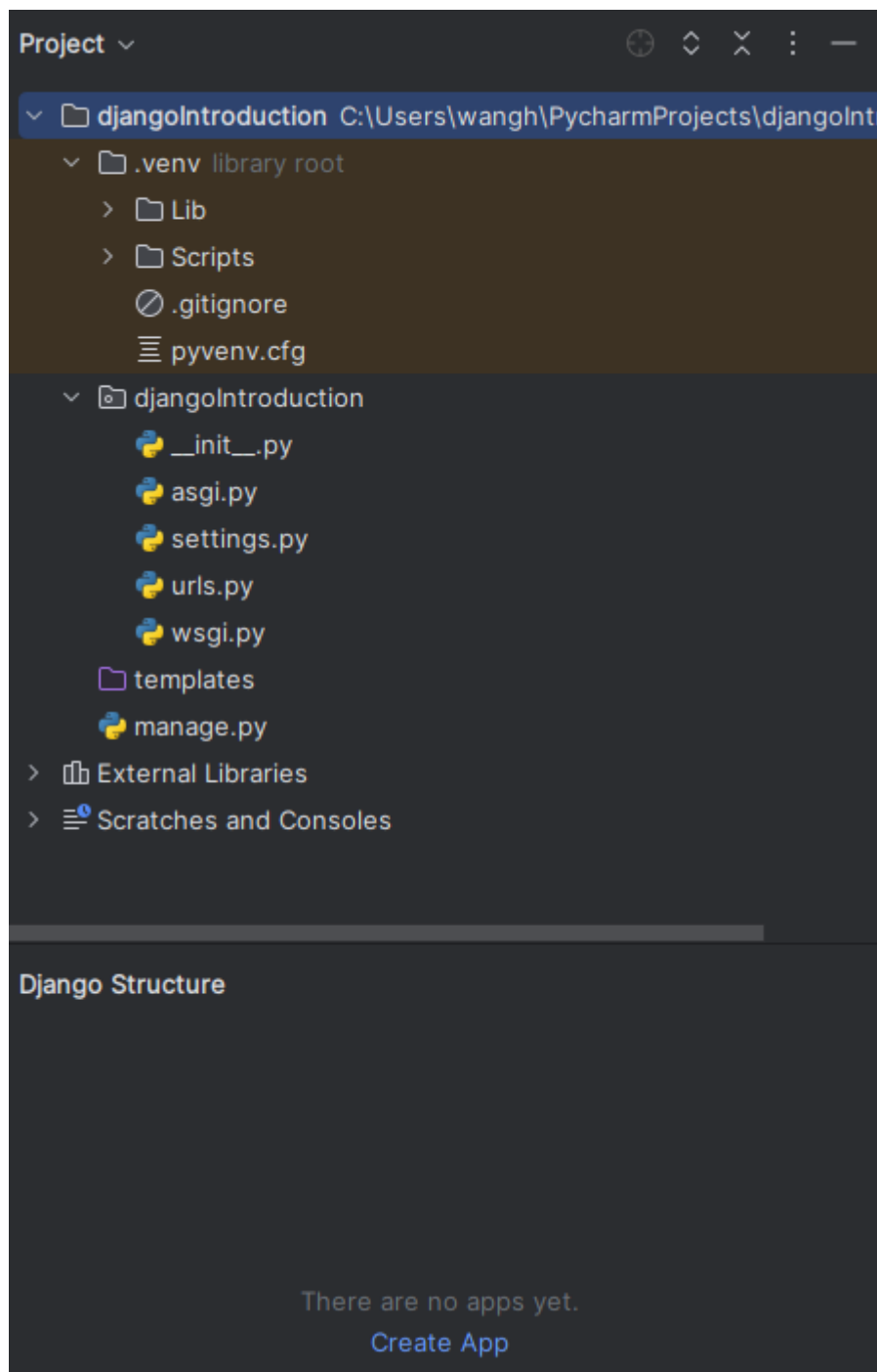
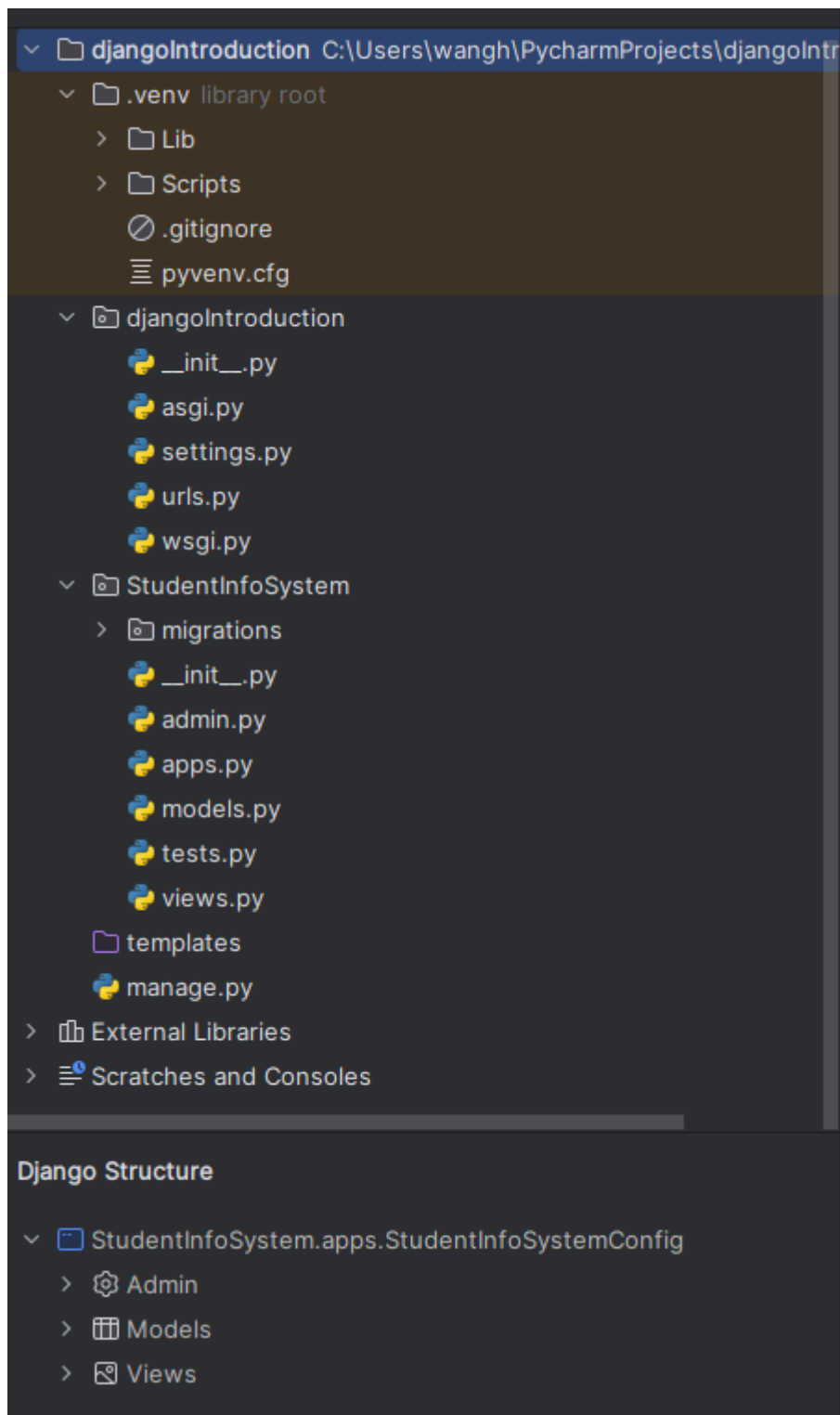


1. 创建一个名为 djangoIntroduction 的 Project，它的原始结构应该是这样的。



2. 创建 App: 点击 create App，输入 App 的名称 StudentInfoSystem，回车。目录中多了一个 StudentInfoSystem 文件夹，里面的内容就是我们要 coding 的。



3. 现在点击屏幕上方的绿色按钮，虽然我们什么都没做，但是已经可以访问本地服务器和 admin 窗口了。

如果报错，那么检查 settings.py 下的 INSTALLED_APPS 和 apps.py 两个文件，它们都是 django 在创建 app 时自动生成的，确保两者是一致的（有时会有大小写的问题）。

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    "StudentInfoSystem.apps.StudentinfosystemConfig"
```

```
1 usage
class StudentinfosystemConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'StudentInfoSystem'
```

4. 接下来是 models 层，作用是为 project 添加数据。打开 models.py，创建名为 Student 的类。它只有一个 field，就是 name。每一次修改数据库结构后，都要迁移、更新数据库。

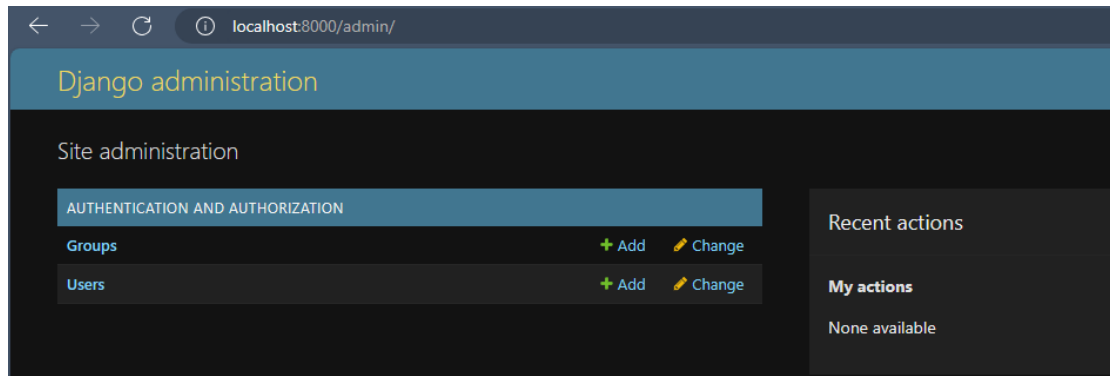
```
class Student(models.Model):
    name = models.CharField(max_length=100)
```

```
manage.py@djangoIntroduction > makemigrations [appname]
```

```
manage.py@djangoIntroduction > makemigrations [appname]
```

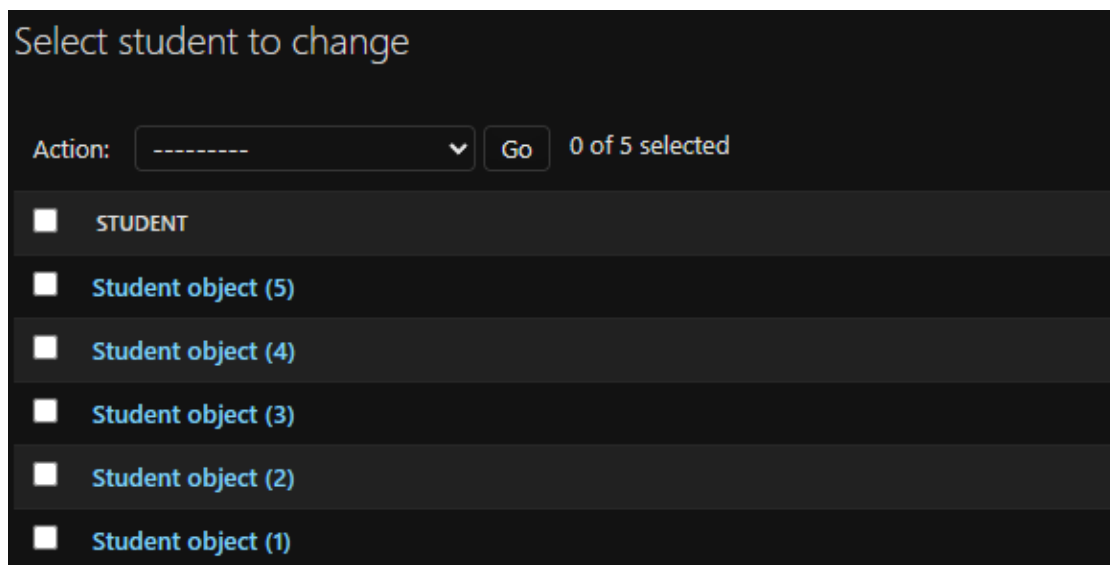
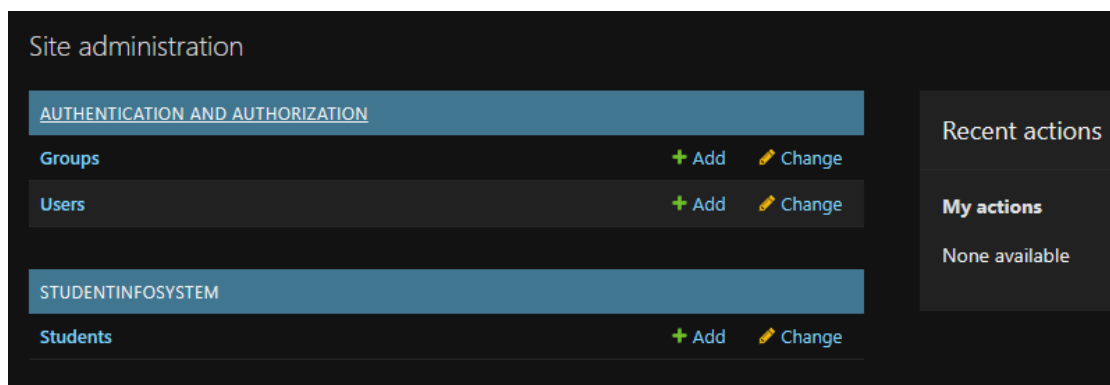
5. 创建管理员账号，自定义用户名和密码。刷新程序，登录 admin 界面，可以进入后台管理系统了

```
manage.py@djangoIntroduction > createsuperuser
```



6. 但是还不能在后台手动添加 Student 表的数据。打开 admin.py，注册我们的 Student 类，重新登录，可以添加 Student 数据了。我们添加 5 个新数据。

```
from django.contrib import admin
# Register your models here.
from .models import Student
admin.site.register(Student)
```



Admin 页面是 django 最具特点的功能，不费力就能有一个后台管理系统，对数据增删改查。后端的工作就此完成，接下来要将数据通过网页展示给用户，这就涉及 views 层。

7. 打开 views.py。这段代码的意思是，当 any_name_1 函数执行时，系统将返回一个名为 template_1 的 HTML 文件，这就是我们要展示给用户看的。

```
from django.shortcuts import render

# Create your views here.

def any_name_1(request):
    return render(request, template_name='template_1.html')
```

8. 在 templates 文件夹中创建这个 template_1.HTML 文件，系统会自动去找。如图，这是标准的 HTML 结构，只有一行段落。

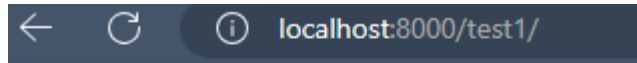
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
  <h1>Hello World</h1>
  <p>This course sucks</p>
</body>
</html>
```

9. 最后在 urls.py 中注册地址 test1/。运行程序

```
from django.contrib import admin
from django.urls import path

from StudentInfoSystem.views import any_name_1

urlpatterns = [
    path('admin/', admin.site.urls),
    path('test1/', any_name_1),
]
```



This course sucks

总结一下，以上 3 个操作的意思是：当用户输入 URL 为 <http://localhost:8000/test1> 时，系统会执行 `views` 层中的 `any_name_1` 函数，该函数会返回一个名为 `template_1` 的 HTML 文件给用户。

目前我们返回的只是一个静态的 HTML，这其实完全可以由 web server 承担，用不到 django。显然，我们还希望这个 HTML 能够与数据库联动，所以我们继续修改。

10. 还是回到 `views.py` 中。想要展示数据，首先要获得该数据。通过 `Student` 类的内置方法（其实是 `models.Model` 类的方法，`Student` 继承了它），我们可以获得当前的 5 项数据，并将其赋值给 `current_students`。

```
def any_name_1(request):  
    current_students = Student.objects.all()  
    return render(request, template_name='template_1.html', context={'current_students_key': current_students})
```

接下来，在 `render` 函数中加一个参数。这个参数的类型必须是一个字典（这是 python 的一种数据结构，类似于数组，只是它的 `index` 不是数字而是字符串）。想要传给 HTML 的所有数据，都必须通过这个字典。目前它只有一组数据，它的 `Key` 是字符串 `current_students_key`（是什么都行，自定），而它的 `value` 是上一行的 `current_students` 变量，它储存了 `Student` 表中的 5 项数据。

11. 修改 `template_1.html`。首先建立一个有序列表。接下来注意两个常见的 django 语法。一个叫标签，`{%.....%}`；一个叫变量，`{{.....}}`。想要在 HTML 中使用 python，我们要在所有 `for/if` 等逻辑判断前加上 `{%.....%}`，同时要在所有变量前加上 `{{.....}}`，这样系统就能将 python 与 HTML 语言区分开。
所以上述代码就是一个很普通的 `foreach` 循环。它遍历了我们从 `views` 函数传过来的字典中 `Key` 值为 `current_students_key` 的 `value`，这个 `value` 包含了 5 项数据，并且将每一项数据的 `name` 值显示出来。重新运行 <http://localhost:8000/test1>，如图。

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
<p>This course sucks</p>

<ol>
  {% for student in current_students_key %}
  <li>{{ student.name }}</li>
  {% endfor %}
</ol>

</body>
</html>

```

This course sucks

1. Henry
2. Min
3. Luo
4. Mark
5. Darren

12. OK。现在可以展示数据库的数据了。但是用户毕竟不是管理员，无法在后台添加数据，所以接下来要用表单（form）。

先搞清楚一个逻辑。用户如何使用表单？顺序是，用户在页面上看到一个空白表单，填写数据，点击提交将数据发送给服务器。全过程是在同一个页面或者说同一个 URL 地址上进行的。这就是说，我们在写表单的时候，要根据用户的操作做一个条件判断。如果用户使用 POST 方法，那么就将表单中的数据传给服务器；如果用户使用 GET 方法，那么就只给他一个空表单等待填写就好。

13. 新建一个 forms.py 文件，定义一个 form 类。注意 import 的内容，容易混淆，别引用错了。这个 StudentForm 继承了 django 的 forms 类，它提供了很多封装好的方法，非常便捷。

这段代码的意思是，StudentForm 根据 model.py 中的 Student 类生成，并且显示 Student 类中的所有 field。由于本例中 Student 类本身只有一个 field，所以区别不大。

```

from django import forms
from StudentInfoSystem.models import Student

class StudentForm(forms.ModelForm):
    class Meta:
        model = Student
        fields = '__all__'

```

14. 回到 views.py，定义一个新函数。我们已经提到了它的逻辑。如果是 POST 方法，就将数据保存到数据库中。如果不是，就给用户一个空白 form 待填写。Print 行的代码，是在控制台输出 success，方便我们纠错。最后一行是说，在数据保存成功后跳转到另一个界面（test1）。比如在实际应用中，注册一个新用户后，跳转到 login in 界面。以上都是可有可无的。

```

def any_name_2(request):
    if request.method == 'POST':
        form = StudentForm(request.POST)
        if form.is_valid():
            print('success')
            form.save()
            return HttpResponseRedirect(reverse('test1'))
    else:
        form = StudentForm()
        return render(request, template_name='template_2.html', context={'form': form})

```

15. 最后编写 template_2.HTML。只有一个 form，都是标准化的内容。第二行是验证数据的真实有效性，相当于 django 封装好的大礼包，永远带着。第三行是说，显示 views 函数传过来的内容（同样是以字典的形式）。最后一行是一个提交数据的按钮。

这里可能有疑惑，我们不是已经在 views.py 中生成 form 吗？为什么在 HTML 中还要再定义 form。事实上，views 生成的是<input>组件，就像 HTML 中的 submit 按钮一样。本例中，它生成的是<input type="textarea">组件，这在我们编写 forms.py 的时候就由 django 自动完成了。


```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    ⚡ <input type="submit" value="Submit">
</form>
</body>
</html>

```

16. 最后，记得在 `urls.py` 中注册地址。这里注意 `path` 的一个 `name` 参数，可有可无。它的作用就是一个代号，方便其他模块的代码引用。比如 `views` 中的那个 `redirect`，引用的就是第一个 `path`，代号是 `test_1`。

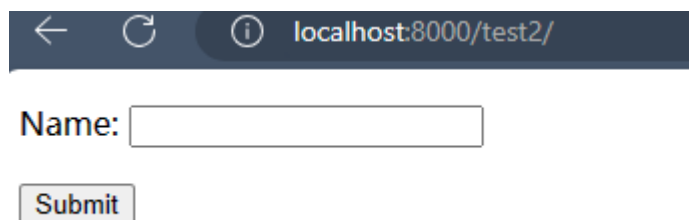
```

from django.contrib import admin
from django.urls import path
from student.views import test_1
from student.views import test_2

urlpatterns = [
    path("", test_1, name='test_1'),
    path('test_2/', test_2, name='test_2'),
    path('admin/', admin.site.urls),
]

```

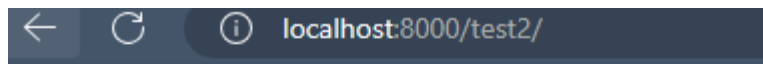
重新运行程序。首先展示一个空表单。输入 Tony，点击 submit，跳转到 `test1` 地址，在列表中看到，Tony 已经成功储存在数据库中了。



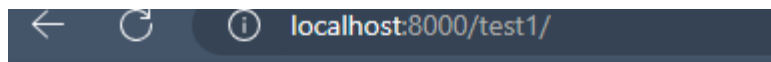
← ↻ ⓘ localhost:8000/test2/

Name:

Submit



Name:



This course sucks

1. Henry
2. Min
3. Luo
4. Mark
5. Darren
6. Tony