

Python-Django 项目入门

参考: <https://www.bilibili.com/video/BV1zt411a7rd?p=1>

目录

Python-Django 项目入门.....	1
安装 Django	1
创建项目	1
数据迁移.....	1
设置超级管理员.....	2
创建 app 应用.....	2
创建数据模型.....	2
设置数据库.....	3
管理后台.....	6
路由配置.....	8
表单.....	11
视图.....	14
模板.....	15

安装 Django

本人使用 Anaconda 在指定环境中安装

创建项目

本人使用 Pycharm, 在 Terminal 中输入命令 `django --admin startproject django_demo`, 其中最后的名称是项目名称;

然后进入该项目, 即输入命令: `cd django_demo`;

启动项目, 输入命令: `python manage.py runserver`, 根据输出的网页进入即可浏览。

Ctrl + C 关闭项目

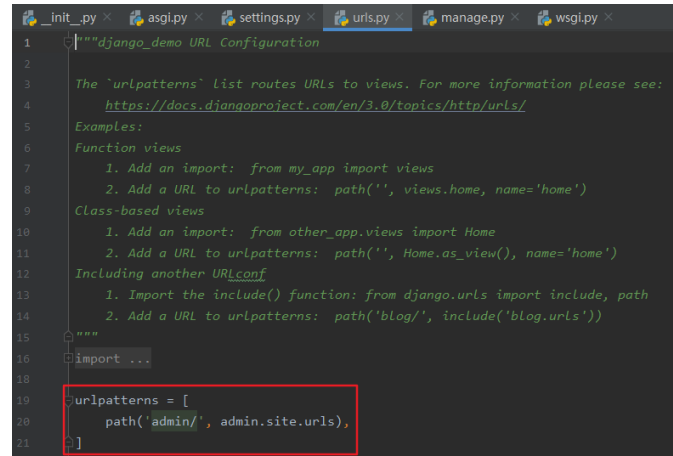
数据迁移

接着输入数据迁移命令: `python manage.py migrate`, 即使用该框架自带的数据库

设置超级管理员

为了方便后台管理，设置超级管理员：`python manage.py createsuperuser`;

然后重启项目，进入网页浏览，同时在网址后面加上`/admin`进入管理员登录页面（路由设置如图文件），输入账户密码登录；

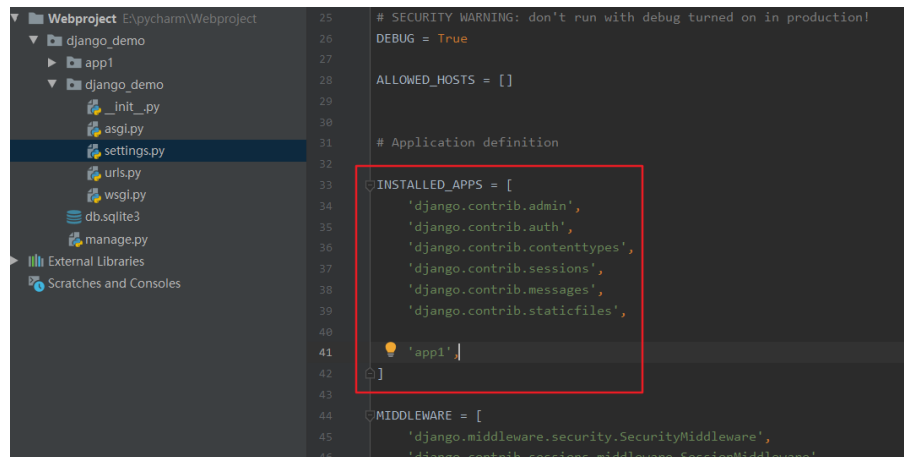


```
1 """django_demo URL Configuration
2
3 The 'urlpatterns' list routes URLs to views. For more information please see:
4     https://docs.djangoproject.com/en/3.0/topics/http/urls/
5 Examples:
6 Function views
7 1. Add an import: from my_app import views
8 2. Add a URL to urlpatterns: path('', views.home, name='home')
9 Class-based views
10 1. Add an import: from other_app.views import Home
11 2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
12 Including another URLconf
13 1. Import the include() function: from django.urls import include, path
14 2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
15 """
16 import ...
17
18 urlpatterns = [
19     path('admin/', admin.site.urls),
20 ]
```

创建 app 应用

这里的 app 应用相当于该项目的子模块。关闭项目创建 app1：
`python manage.py startapp app1`;

然后将创建的子模块配置到该项目中，如下图文件表示。



```
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = True
27
28 ALLOWED_HOSTS = []
29
30 # Application definition
31
32 INSTALLED_APPS = [
33     'django.contrib.admin',
34     'django.contrib.auth',
35     'django.contrib.contenttypes',
36     'django.contrib.sessions',
37     'django.contrib.messages',
38     'django.contrib.staticfiles',
39     'app1',
40 ]
41
42
43 MIDDLEWARE = [
44     'django.middleware.security.SecurityMiddleware',
45     'django.contrib.sessions.middleware.SessionMiddleware',
```

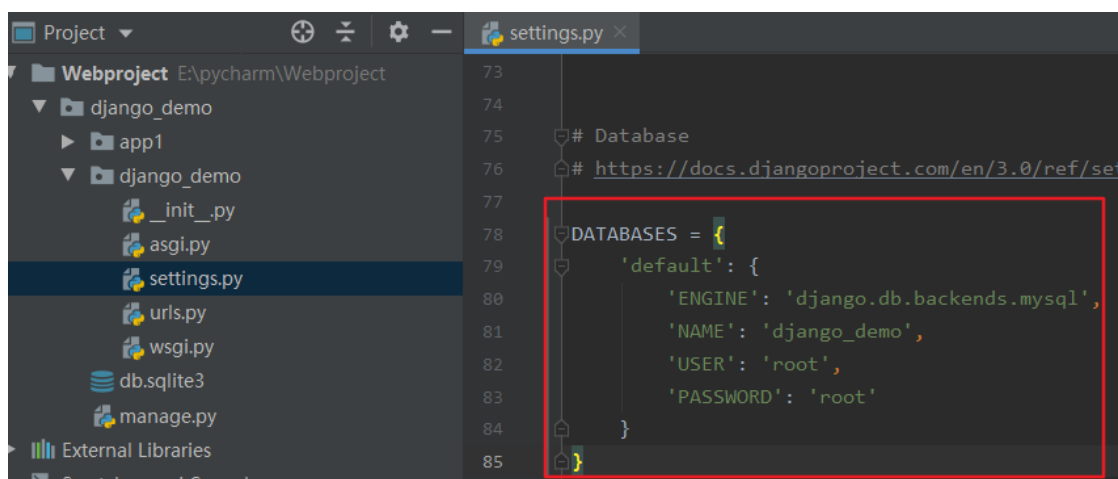
创建数据模型

假如 app1 是商城类，则在 `app1.models.py` 里创建如下类。

```
models.py x
1  from django.db import models
2
3
4  class CreateUpdate(models.Model):
5      created_at = models.DateTimeField(auto_now_add=True)
6      updated_at = models.DateTimeField(auto_now=True)
7
8      class Meta: # 这样设置使得生成数据库表时不会生成该类
9          abstract = True
10
11
12  class Person(CreateUpdate):
13      # first_name, last_name, created_at, updated_at
14      first_name = models.CharField(max_length=30)
15      last_name = models.CharField(max_length=30)
16
17
18  class Order(CreateUpdate):
19      # order_id, order_desc, created_at, updated_at
20      order_id = models.CharField(max_length=30, db_index=True)
21      order_desc = models.CharField(max_length=120)
22
```

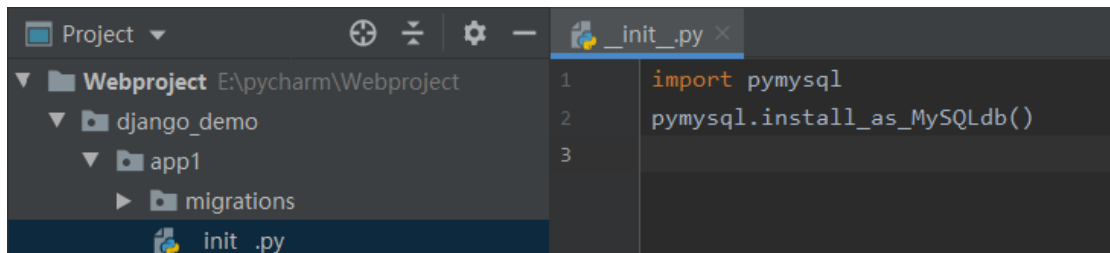
设置数据库

使用 MySQL 数据库，在全局设置文件中修改数据库参数。



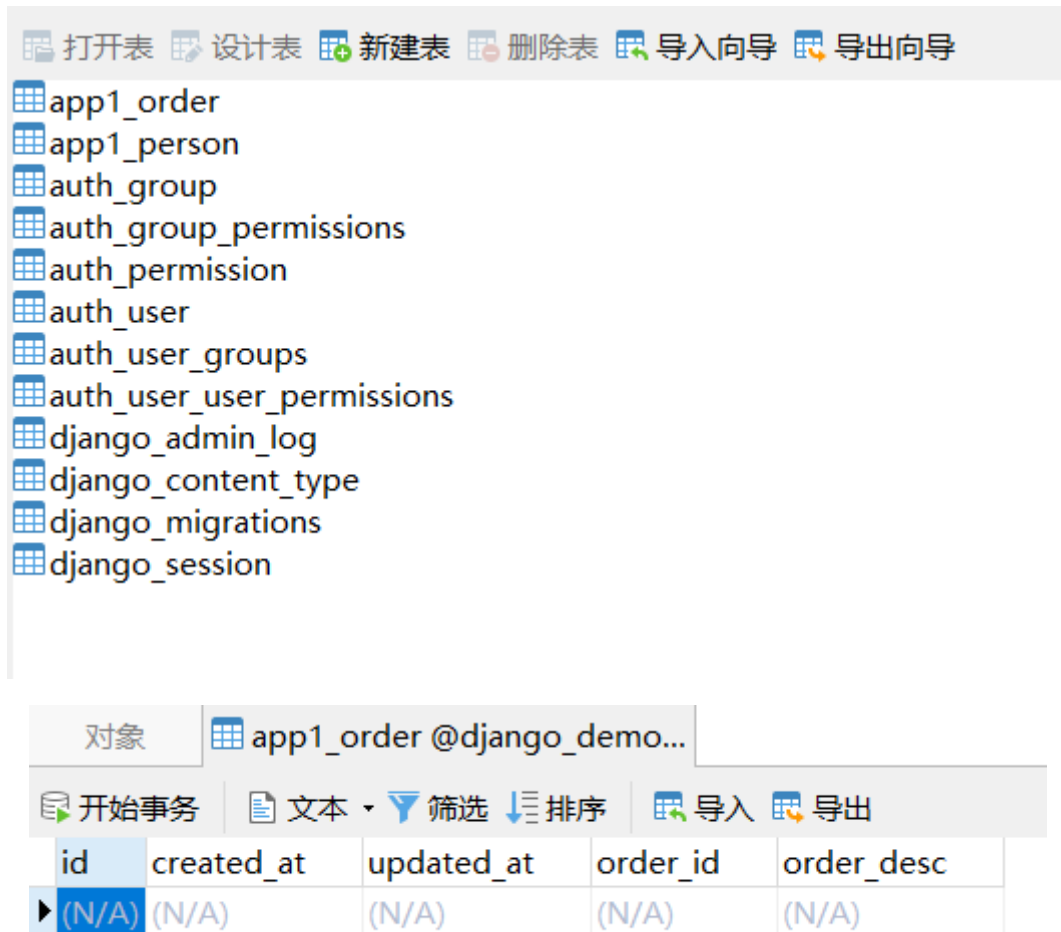
```
settings.py x
73
74
75  # Database
76  # https://docs.djangoproject.com/en/3.0/ref/settings/#databases
77
78  DATABASES = {
79      'default': {
80          'ENGINE': 'django.db.backends.mysql',
81          'NAME': 'django_demo',
82          'USER': 'root',
83          'PASSWORD': 'root'
84      }
85
```

然后创建数据库，并在 Terminal 导入 pymysql 模块： `pip install pymysql`；
在 app1 中引入数据库，如图所示。



进行数据表转化: `python manage.py makemigrations`;

最后输入命令: `python manage.py migrate` 得到数据库表如下 (多余的是管理员用户的)



接下来演示 Django 项目的数据 api: `python manage.py shell`

依次输入如下命令, 如插入数据

```

In [1]: from app1.models import Person, Order

In [2]: Person.objects.create(first_name="andy", last_name="Feng")
Out[2]: <Person: Person object (1)>

In [3]: Person.objects.create(first_name="hugo", last_name="Zhang")
Out[3]: <Person: Person object (2)>

In [4]: 

```

开始事务 文本 筛选 排序 导入 导出					
id	created_at	updated_at	first_name	last_name	
1	2020-05-24 10:42:47.193959	2020-05-24 10:42:47.193959	andy	Feng	
2	2020-05-24 10:43:18.912145	2020-05-24 10:43:18.912145	hugo	Zhang	

```

In [4]: p = Person(first_name="xiaoke", last_name="wang")

In [5]: p.save()

```

开始事务 文本 筛选 排序 导入 导出					
id	created_at	updated_at	first_name	last_name	
1	2020-05-24 10:42:47.193959	2020-05-24 10:42:47.193959	andy	Feng	
2	2020-05-24 10:43:18.912145	2020-05-24 10:43:18.912145	hugo	Zhang	
3	2020-05-24 10:45:30.970717	2020-05-24 10:45:30.971715	xiaoke	wang	

如查询数据:

```

In [6]: Person.objects.all()
Out[6]: <QuerySet [<Person: Person object (1)>, <Person: Person object (2)>, <Person: Person object (3)>]>

In [7]: Person.objects.get(first_name="andy")
Out[7]: <Person: Person object (1)>

In [8]: Person.objects.filter(id__gt=1)
Out[8]: <QuerySet [<Person: Person object (2)>, <Person: Person object (3)>]>

In [9]: Person.objects.filter(id__gt=1).order_by('id')
Out[9]: <QuerySet [<Person: Person object (2)>, <Person: Person object (3)>]>

In [10]: 

```

如修改数据:

```

In [10]: p = Person.objects.get(id=2)

In [11]: p.first_name = "Jay"

In [12]: p.save()

In [13]: 

```

开始事务 文本 筛选 排序 导入 导出					
id	created_at	updated_at	first_name	last_name	
1	2020-05-24 10:42:47.193959	2020-05-24 10:42:47.193959	andy	Feng	
2	2020-05-24 10:43:18.912145	2020-05-24 10:51:53.420888	Jay	Zhang	
3	2020-05-24 10:45:30.970717	2020-05-24 10:45:30.971715	xiaoke	wang	

如删除数据：

```
In [13]: p.delete()
Out[13]: (1, {'app1.Person': 1})

In [14]:
```

开始事务 文本 筛选 排序 导入 导出					
id	created_at	updated_at	first_name	last_name	
1	2020-05-24 10:42:47.193959	2020-05-24 10:42:47.193959	andy	Feng	
3	2020-05-24 10:45:30.970717	2020-05-24 10:45:30.971715	xiaoke	wang	

管理后台

将数据库中的信息展示在后台。在 `app1.admin.py` 中进行如下配置（可以设置想要展示的内容），然后重启（若密码错误可以重新创建管理员账户）。

```
admin.py
1 from django.contrib import admin
2 from app1.models import Person, Order
3
4
5 class PersonAdmin(admin.ModelAdmin):
6     list_display = ('first_name',)
7
8
9 admin.site.register(Person, PersonAdmin)
10
```

Django administration

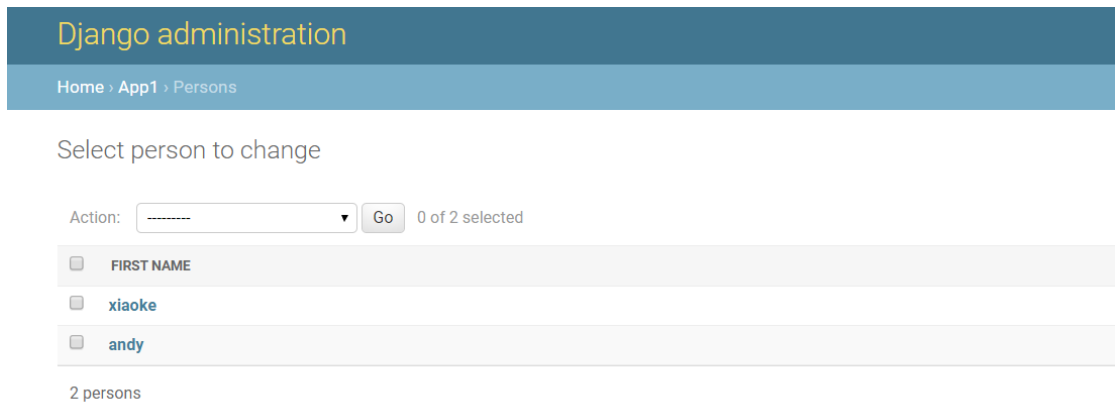
Site administration

APP1	
Persons	+ Add Change
AUTHENTICATION AND AUTHORIZATION	
Groups	+ Add Change
Users	+ Add Change

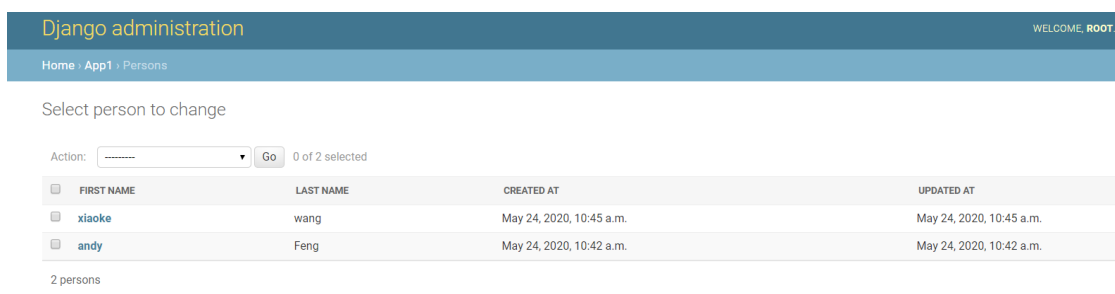
Recent actions

My actions

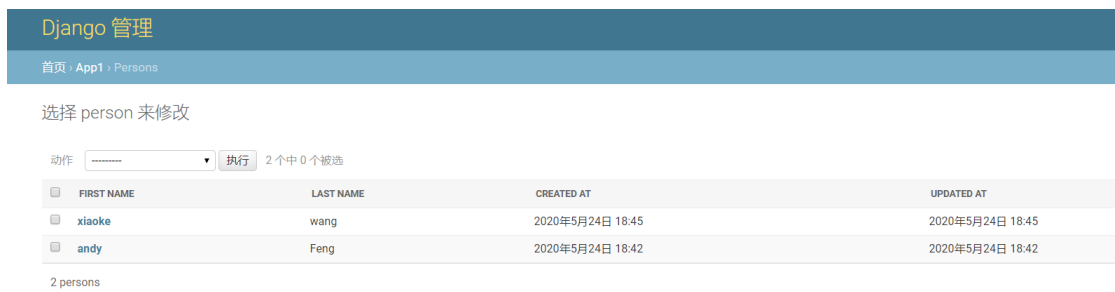
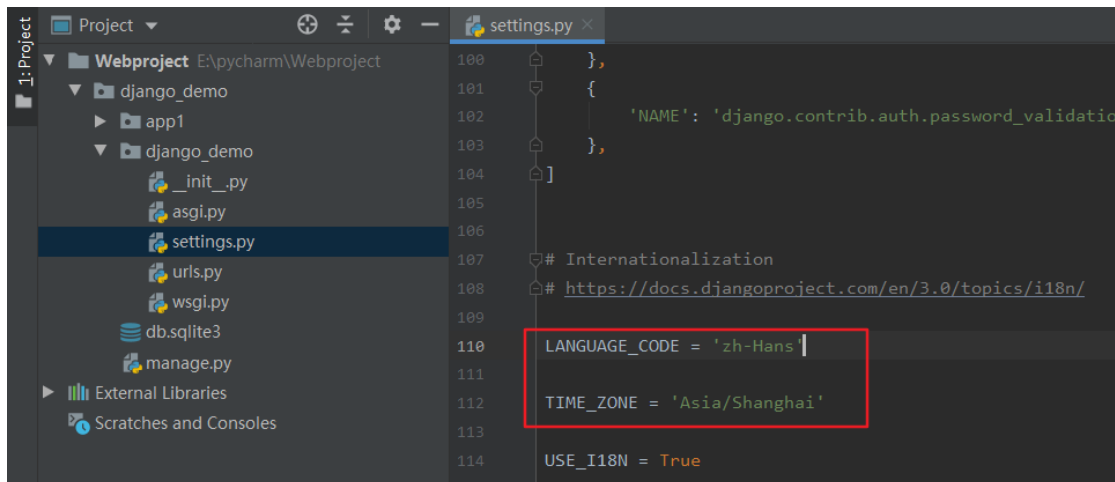
None available



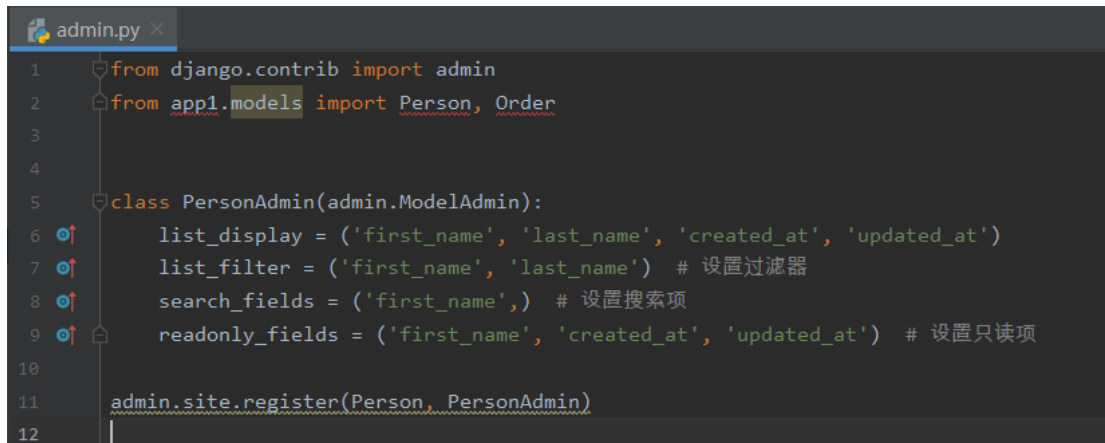
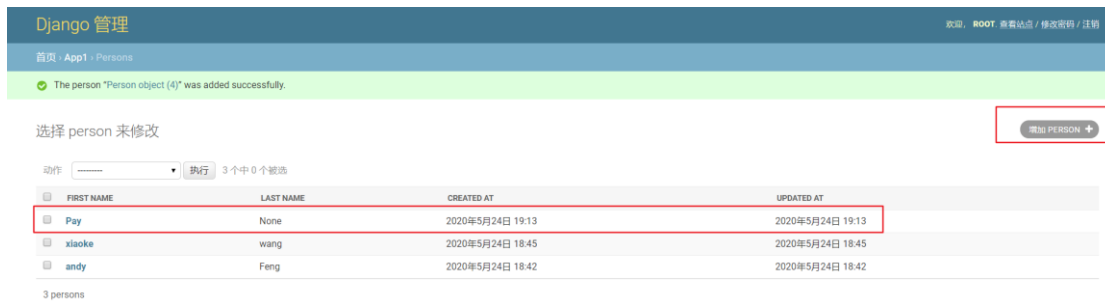
(添加多条信息后)



由于是英文界面和时间，对于我们都不够友好，可以在全局配置中做如下设置。

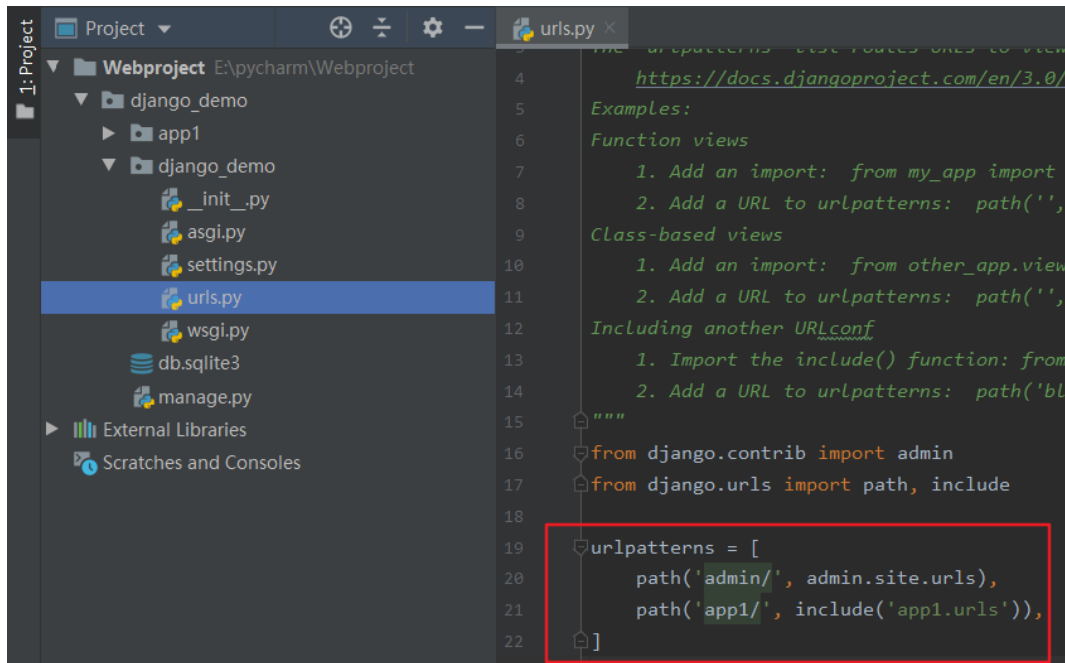


(界面操作后验证功能和时间)

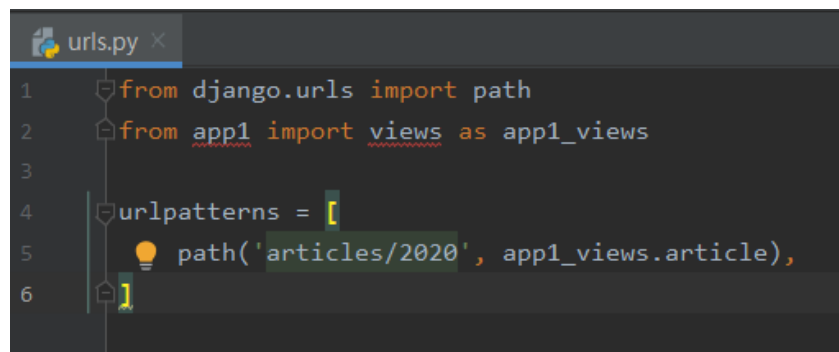


路由配置

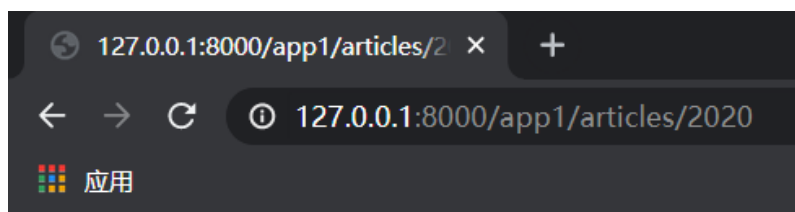
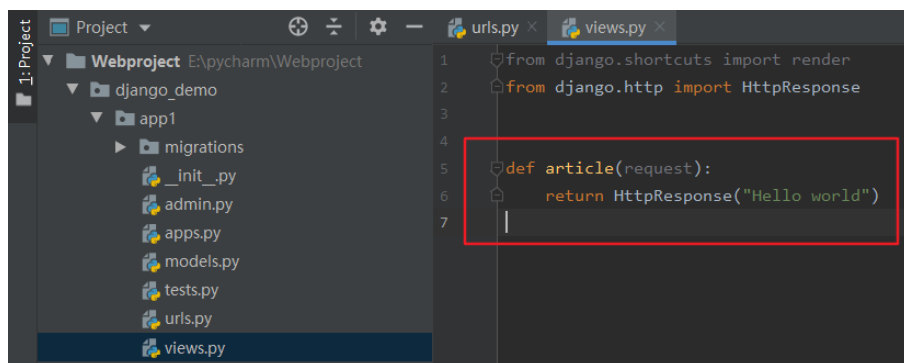
在全局配置中的文件（urls.py）中配置一组路由 **app1**，该组路由是由文件 **app1.urls** 指定细化的，如下。



然后在配置 app1.urls.py 文件。第一种精确路由



然后配置 views.py 文件。



Hello world

第二种格式匹配方式

```
urls.py x views.py x
1 from django.urls import path
2 from app1 import views as app1_views
3
4 urlpatterns = [
5     path('articles/<int:year>', app1_views.article),
6 ]
```

```
urls.py x views.py x
1 from django.shortcuts import render
2 from django.http import HttpResponse
3
4
5 def article(request, year):
6     content = 'the year is %s' % year
7     return HttpResponse(content)
8
```

127.0.0.1:8000/app1/articles/2 x +

← → ↻ 127.0.0.1:8000/app1/articles/2020

应用

the year is 2020

正则表达式方法

```
urls.py x views.py x
1 from django.urls import path, re_path
2 from app1 import views as app1_views
3
4 urlpatterns = [
5     re_path(r'^articles/(?P<year>[0-9]{4})/$', app1_views.article),
6 ]
```

127.0.0.1:8000/app1/articles/2 x +

← → ↻ 127.0.0.1:8000/app1/articles/2020

应用

the year is 2020

表单

设置表单，在 `app1.urls.py` 中配置路由。

```
urls.py x
1 from django.urls import path, re_path
2 from app1 import views as app1_views
3
4 urlpatterns = [
5     re_path(r'^articles/(?P<year>[0-9]{4})/$', app1_views.article),
6     path('get_name', app1_views.get_name),
7 ]
```

```
urls.py x views.py x
1 from django.shortcuts import render
2 from django.http import HttpResponse
3
4
5 def article(request, year):
6     content = 'the year is %s' % year
7     return HttpResponse(content)
8
9
10 def get_name(request):
11     return render(request, 'name.html')
```

创建 html 文件内容（包含文件夹），如下。

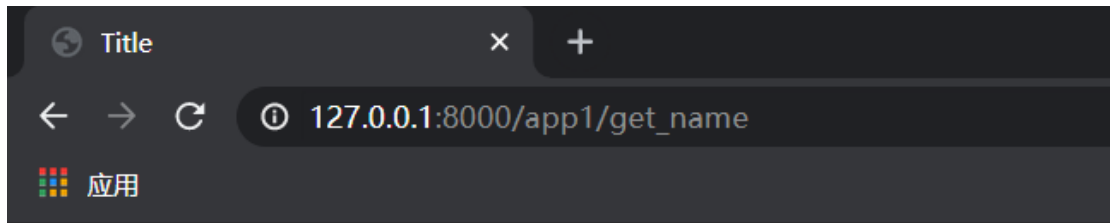
```
Project Webproject E:\pycharm\Webproject
  django_demo
    app1
    django_demo
    templates
      name.html
  db.sqlite3
  manage.py
  External Libraries
  Scratches and Consoles

name.html x
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>Title</title>
6 </head>
7 <body>
8     <form action="/app1/get_name" method="post">
9         <label for="你的名字">你的名字 :</label>
10        <input type="text" id="first_name" name="first_name" maxlength="20" required>
11        <label for="你的姓氏">你的姓氏 :</label>
12        <input type="text" id="last_name" name="last_name" maxlength="20" required>
13    </form>
14 </body>
15 </html>
```

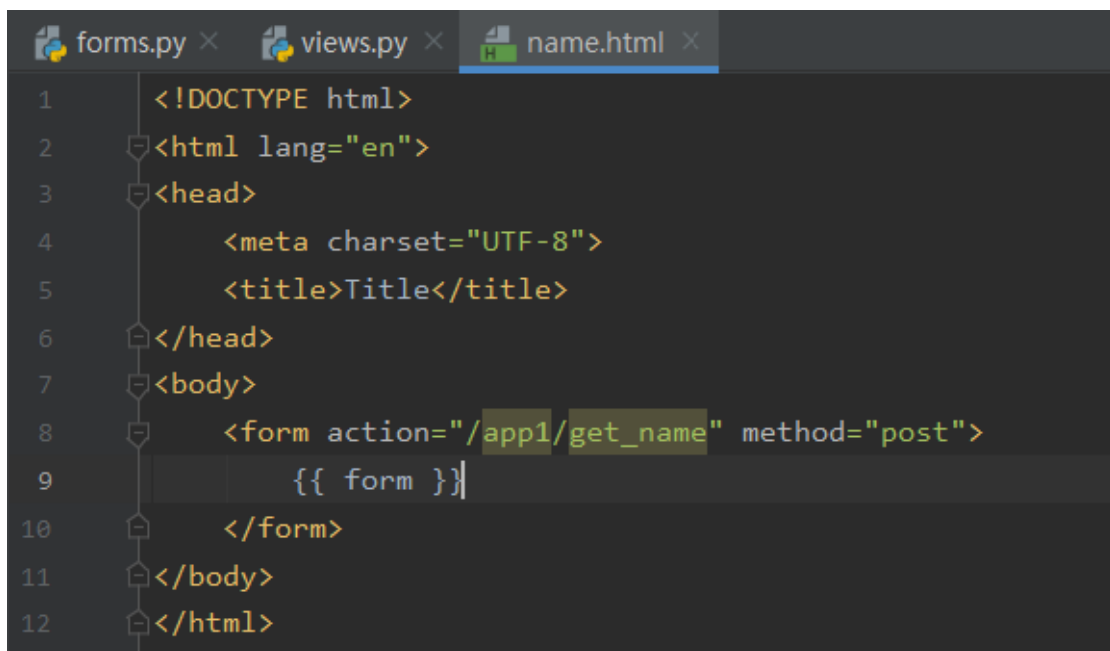
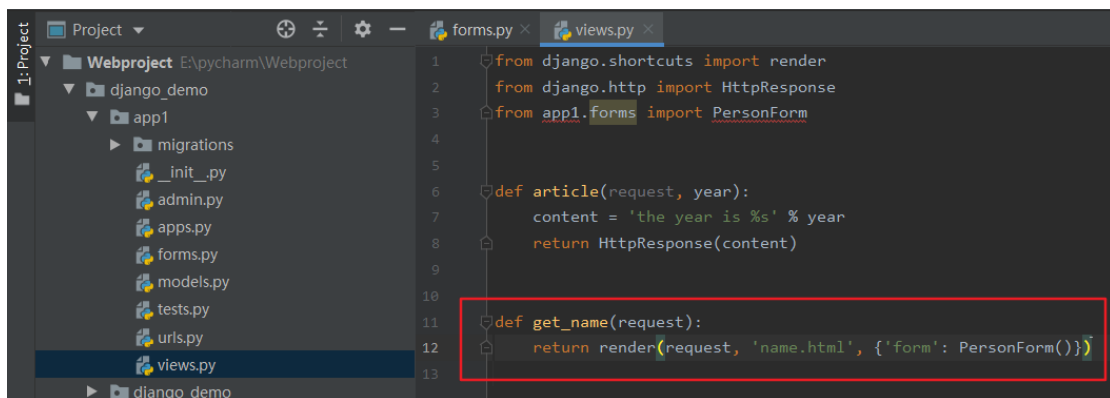
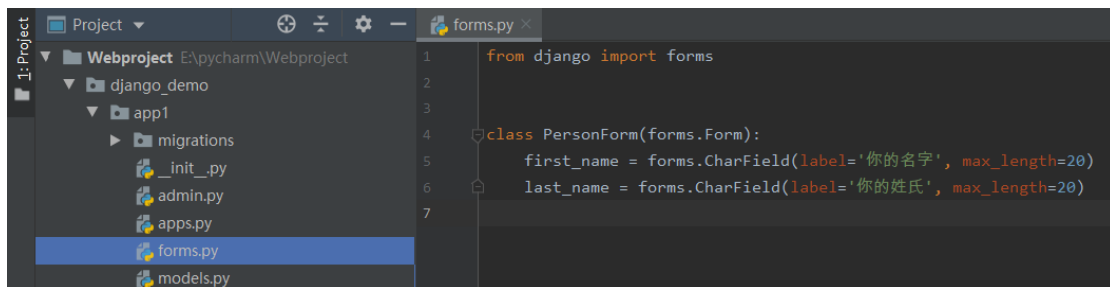
配置 html 文件的关联。

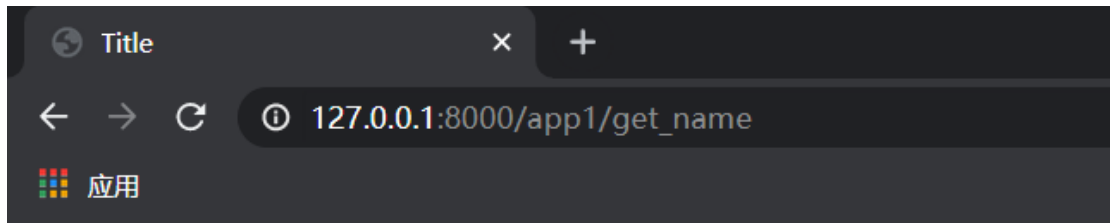
```
Project Webproject E:\pycharm\Webproject
  django_demo
    app1
    django_demo
      _init_.py
      asgi.py
      settings.py
      urls.py
      wsgi.py
    templates
      db.sqlite3
      manage.py
    External Libraries
    Scratches and Consoles

settings.py x
47 'django.middleware.common.CommonMiddleware'
48 'django.middleware.csrf.CsrfViewMiddleware'
49 'django.contrib.auth.middleware.AuthenticationMiddleware'
50 'django.contrib.messages.middleware.MessageMiddleware'
51 'django.middleware.clickjacking.XFrameOptionsMiddleware'
52 ]
53
54 ROOT_URLCONF = 'django_demo.urls'
55
56 TEMPLATES = [
57     {
58         'BACKEND': 'django.template.backends.django.DjangoTemplates',
59         'DIRS': [BASE_DIR + '/templates'],
60         'APP_DIRS': True,
61         'OPTIONS': {
```



更高级的设置，创建表单类





你的名字: 你的姓氏:

增加提交按钮，以及增强安全措施，防止跨域攻击

```
</head>
<body>
    <form action="/app1/get_name" method="post">
        {% csrf_token %}
        {{ form }}
        <button type="submit">提交</button>
    </form>
</body>
</html>
```

处理提交

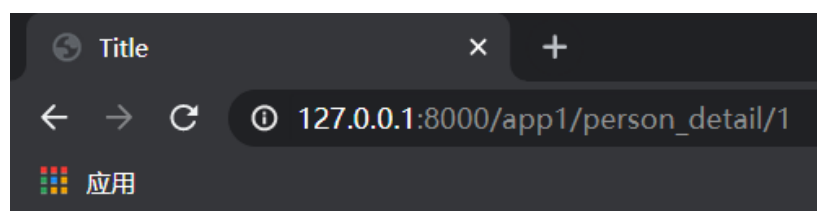
```
forms.py x views.py x name.html x
1 from django.shortcuts import render
2 from django.http import HttpResponse, HttpResponseRedirect
3 from app1.forms import PersonForm
4
5
6 def article(request, year):
7     content = 'the year is %s' % year
8     return HttpResponse(content)
9
10
11 def get_name(request):
12     if request.method == "POST":
13         form = PersonForm(request.POST)
14         if form.is_valid():
15             first_name = form.cleaned_data['first_name']
16             last_name = form.cleaned_data['last_name']
17             return HttpResponse(first_name + ' ' + last_name)
18         else:
19             return HttpResponseRedirect('/error/')
20     return render(request, 'name.html', {'form': PersonForm()})
21
```

视图

```
views.py | person_detail.html | urls.py
1 from django.shortcuts import render
2 from django.http import HttpResponse, HttpResponseRedirect, Http404
3 from app1.forms import PersonForm
4 from app1.models import Person
5
6
7 def article(request, year):
8     content = 'the year is %s' % year
9     return HttpResponse(content)
10
11
12 def person_detail(request, pk):
13     try:
14         p = Person.objects.get(pk=pk)
15     except Person.DoesNotExist:
16         raise Http404('Person Does Not Exist')
17     return render(request, 'person_detail.html', {'person': p})
18
```

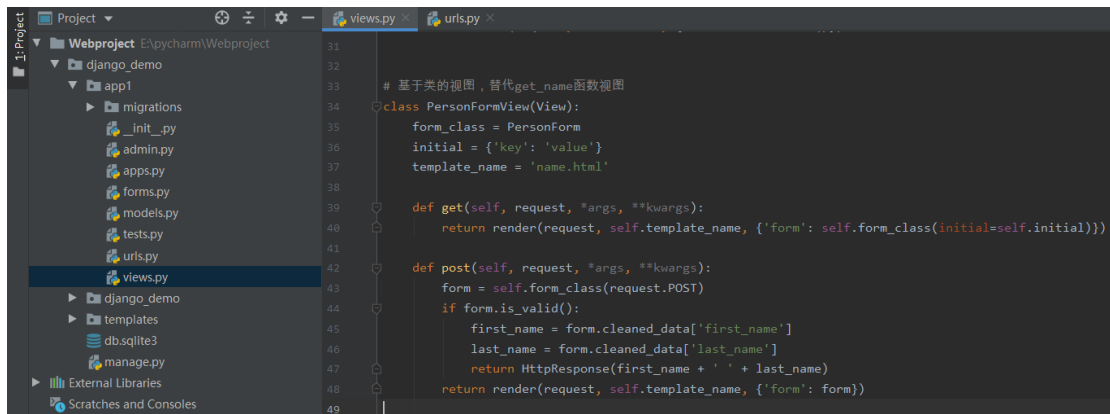
```
Project | views.py | person_detail.html | urls.py
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>Title</title>
6 </head>
7 <body>
8     {{ person.first_name }}
9 </body>
10 </html>
```

```
views.py | person_detail.html | urls.py
1 import ...
2
3
4 urlpatterns = [
5     re_path(r'^articles/(?P<year>[0-9]{4})/$', app1_views.article),
6     path('get_name', app1_views.get_name),
7     path('person_detail/<int:pk>', app1_views.person_detail),
8 ]
```

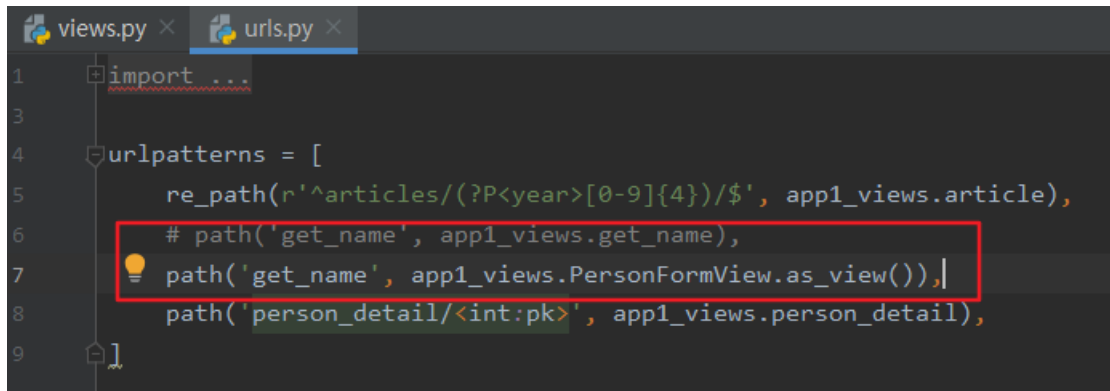


andy

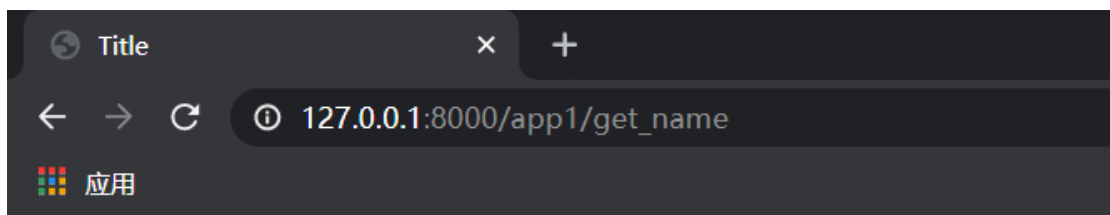
以上所有视图都是基于函数的，下面演示基于类的视图。



```
31
32
33 # 基于类的视图，替代get_name函数视图
34 class PersonFormView(View):
35     form_class = PersonForm
36     initial = {'key': 'value'}
37     template_name = 'name.html'
38
39     def get(self, request, *args, **kwargs):
40         return render(request, self.template_name, {'form': self.form_class(initial=self.initial)})
41
42     def post(self, request, *args, **kwargs):
43         form = self.form_class(request.POST)
44         if form.is_valid():
45             first_name = form.cleaned_data['first_name']
46             last_name = form.cleaned_data['last_name']
47             return HttpResponse(first_name + ' ' + last_name)
48         return render(request, self.template_name, {'form': form})
49
```

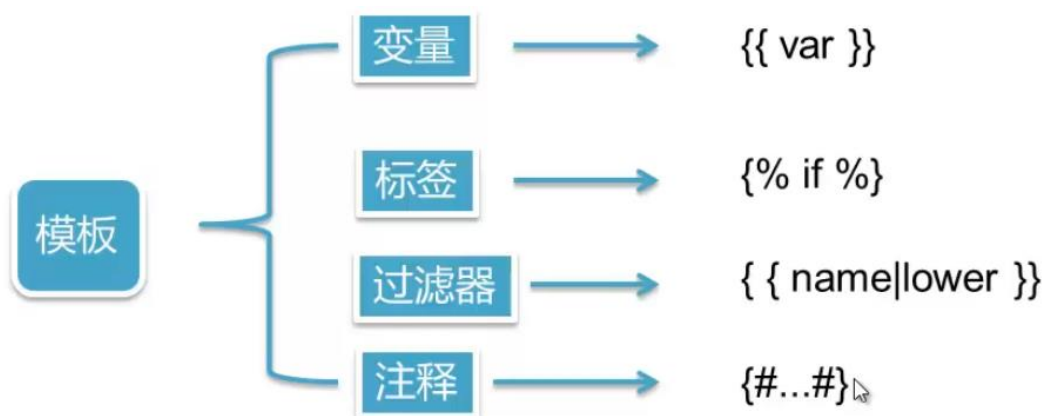


```
1 import ...
2
3
4 urlpatterns = [
5     re_path(r'^articles/(?P<year>[0-9]{4})/$', app1_views.article),
6     # path('get_name', app1_views.get_name),
7     path('get_name', app1_views.PersonFormView.as_view()),
8     path('person_detail/<int:pk>', app1_views.person_detail),
9 ]
```



你的名字: 你的姓氏:

模板



```
person_detail.html ×
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>Title</title>
6 </head>
7 <body>
8     {{ person.first_name }}
9
10    {% if person.first_name == 'andy' %}
11    你好，andy
12    {% else %}
13    你好， {# 陌生人（被注释，且审查元素也看不到） #}
14    {% endif %}
15
16    {{ person.first_name | upper }}
17
18    <!-- 我是注释，在审查元素时可以看到 -->
19 </body>
20 </html>
```