Django安装与第一个应用

> Django是一个Python定制框架,它能够让开发人员进行高效且快速的开发，基于MVC设计模式。但是Django更关注的是模型(Model),模板(Template)和视图(Views),称为MTV模式，他们的各自职责如下：

层次	职责
模型(Model)，即数据存储层	处理与数据相关的所有事务，如何存取、如何验证有效性，包含哪些行为以及数据之间的关系
模板(Template),表现层	如何与表现相关的决定，如何在页面与其它类型文档中进行显示
视图(View),即业务逻辑层	存储模型以及调取恰当的相关逻辑，模型与模板的桥梁

Django的安装

```
C:\Users\Administrator>pip install Django   # 如果安装可以卸载 pip uninstall Django
Collecting Django
Requirement already satisfied: pytz in d:\anaconda3\lib\site-packages (from Django) (2018.5)
Installing collected packages: Django
Successfully installed Django-2.1.4
```

检测是否安装成功可以通过如下命令：

> 意思是将库中的python模块采用脚本的方式运行

```
C:\Users\Administrator>python --help     // 查看-m参数的含义
C:\Users\Administrator>python -m django --version
2.1.4
```

创建第一个Django项目

> django-admin可以查看更多详细的参数

1. 通过官方提供的模板,来创建项目：django-admin startproject myblog
2. 在自定义的项目目录mylog文件夹中有一个manage.py文件。此文件是项目与交互的命令行工具集的入口

```
C:\Users\Administrator\Desktop\django\myblog>python manage.py
```

```
Type 'manage.py help <subcommand>' for help on a specific subcommand.

Available subcommands:

[auth]
    changepassword
    createsuperuser

[contenttypes]
    remove_stale_contenttypes

[django]
    check
    compilemessages
    createcachetable
```

1. 可以通过python manage.py runserver来启动服务,默认端口是8000可以在后面指定自己的端口

```
C:\Users\Administrator\Desktop\django\myblog>python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).

You have 15 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.
December 11, 2018 - 10:49:46
Django version 2.1.4, using settings 'myblog.settings'
Starting development server at http://127.0.0.1:8000/
```

Django目录下配置文件介绍

> settings.py最重要，具体参考：https://blog.csdn.net/com_ma/article/details/77953936

1. wsgi.py: （Python Web Server Geteway Interface),Python服务器网关接口。Python应用与服务器之前的接口。从代码还可以看出加载了 "project.settings"
2. urls.py: URL配置文件，Django项目中所有的地址(页面) 都需要我们自己去配置
3. settings.py: 这个文件包含了所有关于Django项目的配置信息，均大写。最重要的设置是ROOT_URLCONF,它将作为URLconf告诉Django在这个站点中哪些Python模块将被用到

在项目中创建第一个应用

1. 打开命令行，进入项目中manage.py同级目录
2. 在命令行输入如下命令,如果创建成功则可会在同目录中生成blog

```
C:\Users\Administrator\Desktop\django\myblog>python manage.py
startapp blog

C:\Users\Administrator\Desktop\django\myblog>dir
 驱动器 C 中的卷没有标签。
 卷的序列号是 860E-CA02

 C:\Users\Administrator\Desktop\django\myblog 的目录

2018/12/11  11:20    <DIR>          .
2018/12/11  11:20    <DIR>          ..
2018/12/11  11:15    <DIR>          .idea
2018/12/11  11:20    <DIR>          blog
2018/12/11  10:49                 0 db.sqlite3
2018/12/11  10:43               553 manage.py
2018/12/11  11:04    <DIR>          myblog
```

创建完毕后目录结构如下：

```
C:\Users\Administrator\Desktop\django\myblog\blog 的目录

2018/12/11  11:20    <DIR>          .
2018/12/11  11:20    <DIR>          ..
2018/12/11  11:20                66 admin.py       // 该应用的后台管理
配置
```

```
2018/12/11  11:20                88 apps.py          // 该应用的一些配置在Django1.9以后自动生成
2018/12/11  11:20    <DIR>           migrations      // 数据库迁移模块
2018/12/11  11:20                60 models.py        // 数据模块，支持ORM映射框架，类似MVC设计模式
2018/12/11  11:20                63 tests.py
2018/12/11  11:20                66 views.py         // 执行响应代码所在的模块，代码逻辑处理的主要地点
2018/12/11  11:20                __init__.py         // 拥有此文件，则说明项目本身也可以是一个模块
```

把应用名称添加到settings.py中的INSTALLED_APPS里

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'blog',
]
```

编写第一个Demo

1. 在view.py中编写相应请求的函数

```
from django.shortcuts import render
from django.http import HttpResponse
# Create your views here.

# 每个请求都会映射到某个函数来处理
def index(request):   # 记得在配置文件中配置当前地址
    return HttpResponse('Hello Django!')
```

1. 在urls.py中配置url请求地址与函数的映射关系

```
from django.contrib import admin
from django.urls import path
import blog.views as bv

```

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('index/', bv.index)
]

```

页面显示效果如下

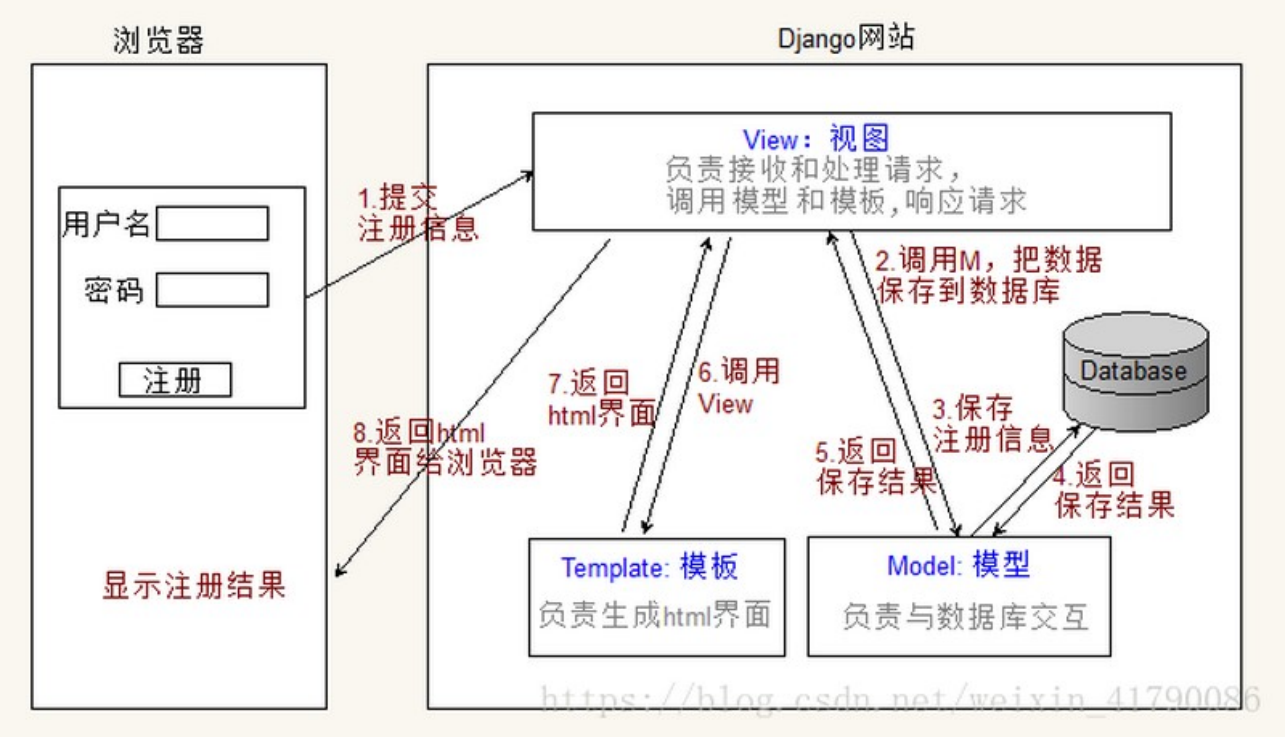
直接输入 http://127.0.0.1:8000/index/ 则可以看到如下结果

Django的MVT设计模式

MVT模式介绍

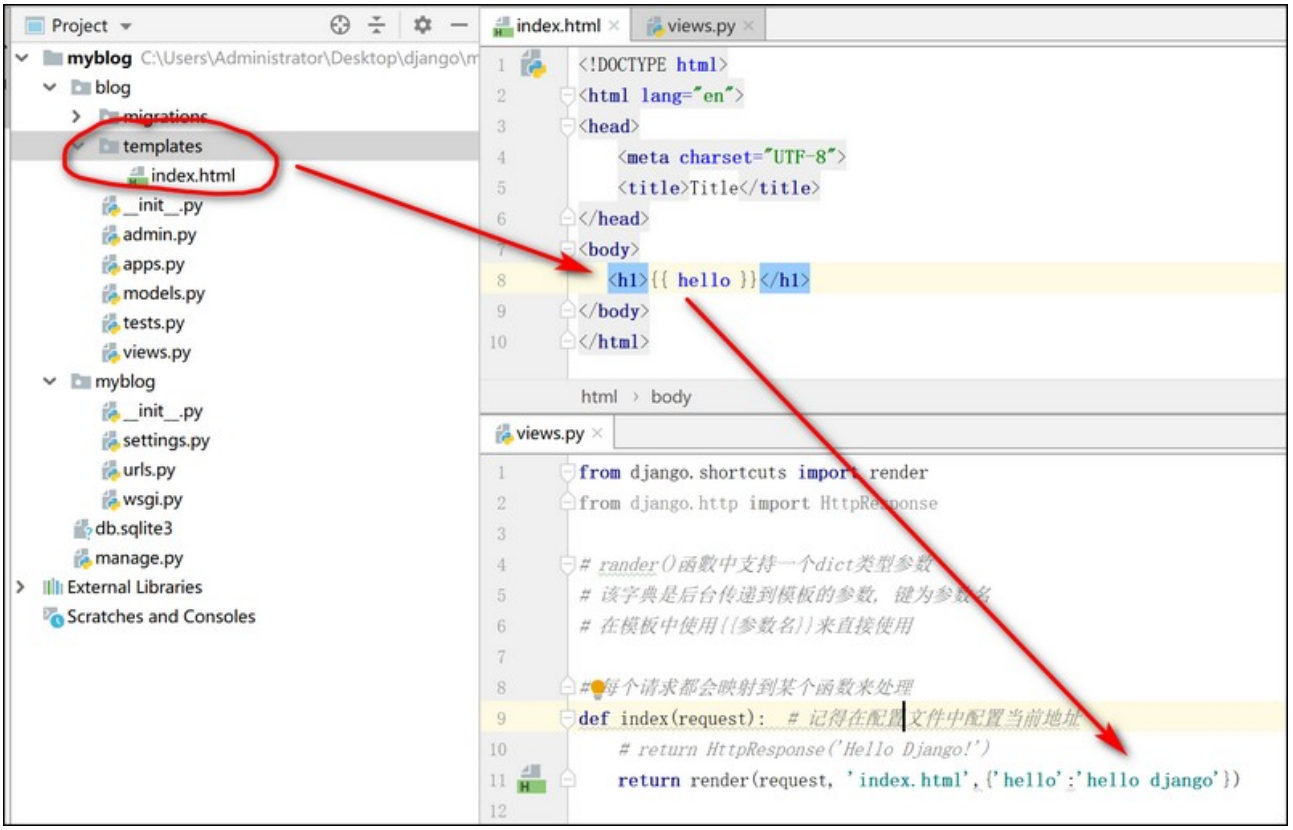
> Django也是MVC框架。 但是，Django框架（内部的URLconf)作为控制器的角色，负责了接收用户请求和转发请求的工作，Django 里更关注的是模型（Model）、模板(Template)和视图（Views），故称之为 Django MVT 模式

1. M：Model,模型与MVC中的M相同，负责对数据的处理,与同步操作
2. V：View,视图与MVC中的C类似，负责处理用户请求，调用M和T，响应请求
3. T：Template, 模板与MVC中的V类似，负责如何显示数据（产生html界面）



Template模板介绍

1. 在App的根目录下创建一个名叫templates的目录
2. 在该目录下创建HTML文件
3. 在view.py中采用render返回存储数据的页面




```
# 每个请求都会映射到某个函数来处理
```

```
def index(request):
    # rander() 函数中支持传递一个context给前端模板(dict类型)
    # def render(request, template_name, context=None)
    return render(request, "index.html", {'hello': 'hello django!!!'})
```

Models与ORM映射实现

> Django中的models通常一个Model对应数据库的一张数据表，Django中Model以类的形式表现，它包含一些基本字段以及数据的一些行为。其实很类似Java的ORM映射框架,它的好处是隐藏了数据访问的细节，不需要编写SQL语句

Django官方文档地址：https://docs.djangoproject.com/zh-hans/2.1/

继承models.Model创建模型类

在应用根目录下创建models.py,并引入models模块创建类，继承models.Model,该类即使一张数据表中类的创建字段, 定义的语法为**属性名=models.字段类型**


```
from django.db import models

class Article(models.Model):
    # 如果没有主键,Django会自动生成主键
    title = models.CharField(max_length=32,default='defult title')
    content = models.TextField(null=True)
```

- 在命令中进入manage.py同级目录,执行python manage.py makemigrations blog(项目名可选), 执行完毕后会在blog项目的migrations目录下面会生成一个0001_init文件

```
C:\Users\Administrator\Desktop\django\myblog>python manage.py makemigrations blog
Migrations for 'blog':
  blog\migrations\0001_initial.py
    - Create model Article
```

查看0001.initial.py发现其实就是生成ORM映射的配置文件

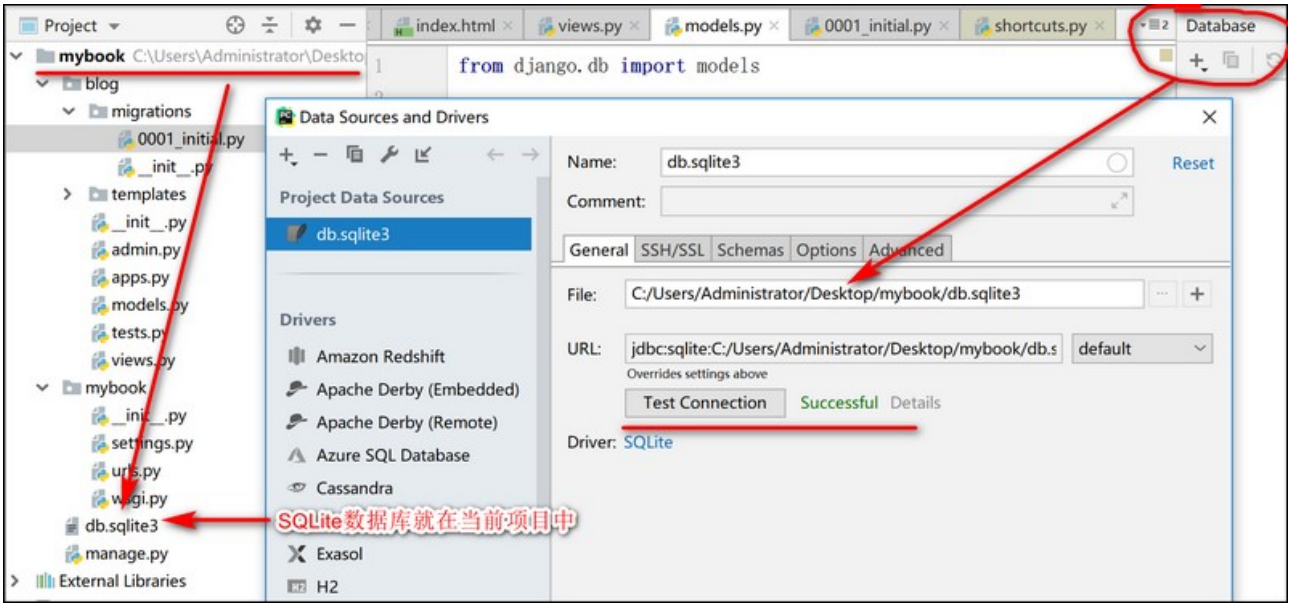
```
operations = [
    migrations.CreateModel(
        name='Article',
        fields=[
            ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
            ('title', models.CharField(default='title', max_length=32)),
            ('content', models.TextField(null=True)),
        ],
    ),
]
```

执行python manage.py migrate 以查看运行的SQL语句

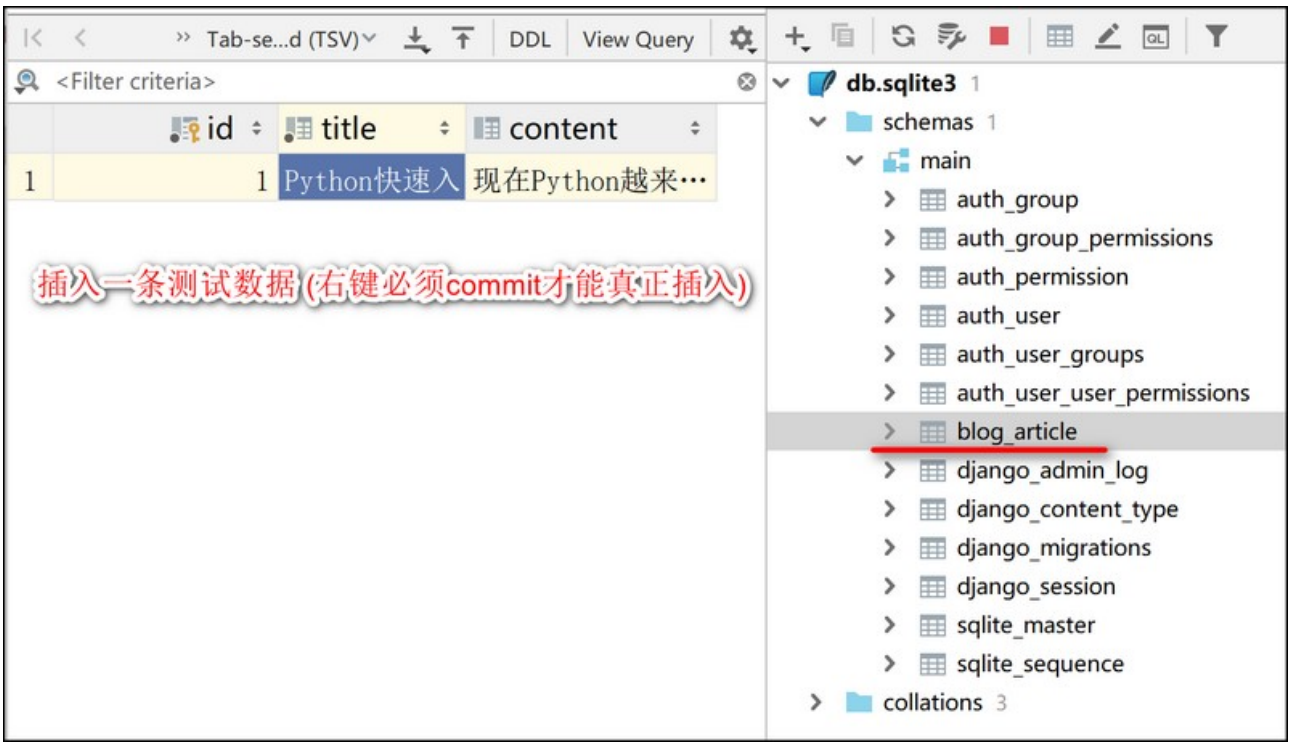
```
C:\Users\Administrator\Desktop\mybook>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, blog, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying 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 blog.0001_initial... OK
  Applying sessions.0001_initial... OK
```

通过PyCharm自带的DataBase连接项目中的SQLite数据库



查看数据库表结构



在views.py中修改rander渲染的数据

```python
from django.shortcuts import render
from django.http import HttpResponse
from . import models

# 每个请求都会映射到某个函数来处理
def index(request):
    # 因为是关键字参数,pk 必须要填写
    article = models.Article.objects.get(pk=1)
    print(article)
    return render(request, "index.html", {'article': article})
```

```html
<body>
   {{ article.id }} | {{ article.title }} | {{ article.content }}
</body>
```

Djaong后台管理功能

> Admin是Django自带的一个功能强大的自动化数据管理界面，被授权的用户可以在后台完成CRUD的基本操作。而且Django提供了许多针对Admin的定制功能。

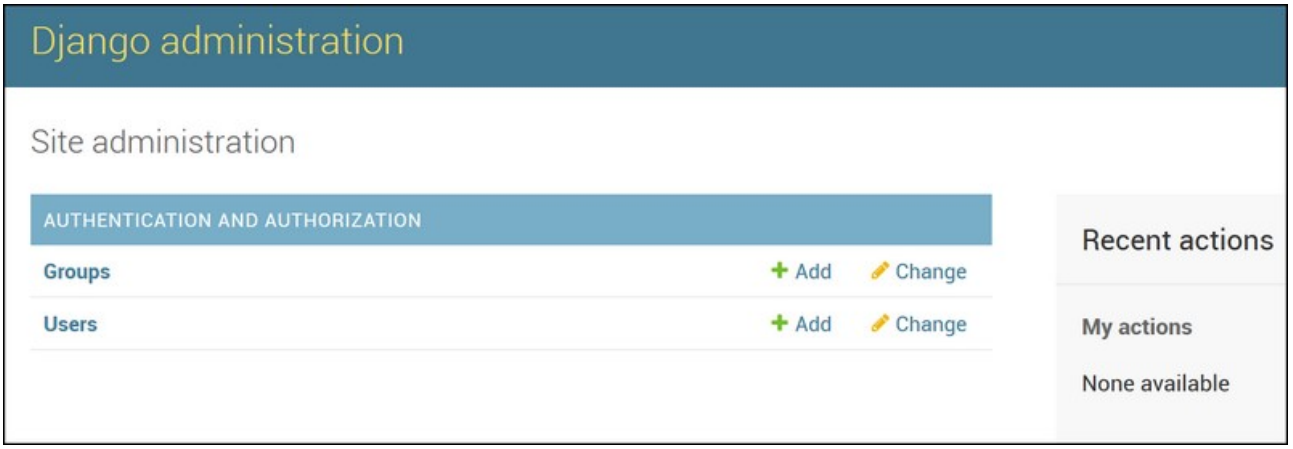
指定Admin的账号、邮箱、密码

```
C:\Users\Administrator\Desktop\django\myblog>python manage.py
createsuperuser
Username (leave blank to use 'administrator'): admin    // 不建议当前
系统登录名作为用户名
Email address: 574231288@qq.com
Password:       // 自己设置密码,密码必须大于8位
Password (again):
Superuser created successfully.
```

访问系统已经配置好的管理界面（urls.py）

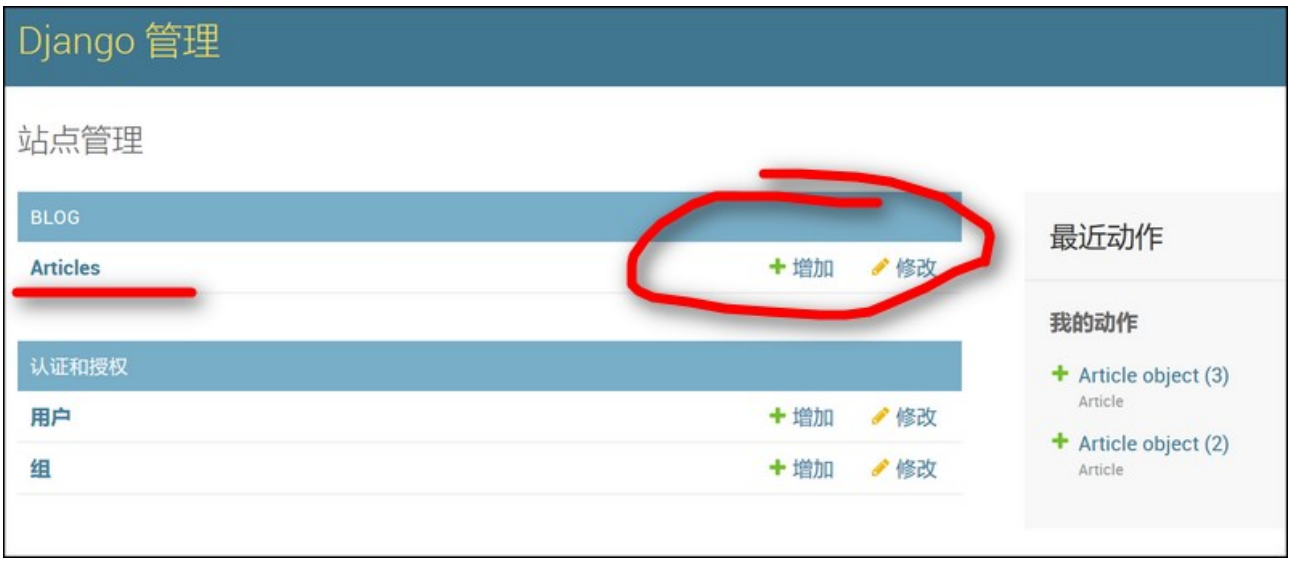
```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('index/', bv.index),
]
```

登录成功可以进入到后台管理界面,后台也可以通过配置文件，把界面修改成中文(settings.py中的LANGUAGE_CODE = 'zh_Hans')



admin.py配置要管理的模块

```
from blog.models import Article
admin.site.register(Article)
```



默认显示一个object对象,如果我们想看对象的某些属性信息, 可以进行如下配置

```
class ArticleAdmin(admin.ModelAdmin):
    list_display = ['id','title','content']

admin.site.register(Article,ArticleAdmin)
```

显示效果如下: