使用 python manage.py shell 可以进入Django的调试界面，测试语句的效果

一．创建工程，创建并设置APP

1.创建工程：django-admin startproject HelloWorld

2.创建应用：到项目根目录下 python manage.py startapp xxx

3.在setting.py里添加app名

4.配置URL：在app里新建一个url文件，并配置总url文件与app的url文件 /blog/index/

5.运行网站：python manage.py runserver

1. 配置APP
2. 先写models，将需要传入的数据写好

①一个class就是一个数据表

class Article(models.Model):

title = models.CharField(max\_length=32, default='xixixi')

content = models.TextField(null=True)

pub\_time = models.DateTimeField(auto\_now=True) 带这个参数则不可修改，是创建时的时间

②写完后cmd进行两步数据迁移

python manage.py makemigrations app名(可选) 在app/migrations目录下生成移植文件

python manage.py migrate 进行数据迁移

（用python manage.py sqlmigrate app名 文件id 查看SQL，文件id为app/migrations目录里的id）

1. 如何更改数据库中的内容？

①用数据库软件写；②在admin里改

1. admin怎么用？

①写一个Model对应的admin配置类，来控制admin界面

class ArticleAdmin(admin.ModelAdmin):

list\_display = ('title', 'content')

list\_filter = ('pub\_time',) #注意，元组一个元素时定要加上，

②之后注册Model时，绑定配置类 admin.site.register(Article, ArticleAdmin)

③小问题：

创建网站管理账户：python manage.py createsuperuser

改admin的字，在setting里 改'ch-hans'

想让管理网站的类成员显示正确名称，在app的models里，在类里加方法：

def \_\_str\_\_(self):

return Article.title

1. 先写html文件，根据前端需求，设置后端传递

①在app目录下创建templates文件夹,再创一个与app同名的文件夹，html文件放在里面

②语法：

在html文件中，调用变量用 {{article.title}}

变量使用过滤器（变量处理函数）：{{ value | filter }} 例如：{{ article.id | defult:'0' }} 作用是当article.id不存在时，返回0（不然Django默认返回空字符串）

配置url <a href="{% url 'blog:article\_page' article.id %}">{{article.title}}</a>

For语法：{% for article in articles %}

<a href="">{{article.title}}</a>

<br/>

{% endfor %}

If语法：{% if article %}

{% else %}

{% endif %}

③网页中有提交，需要写form的action及method，<form action="" method="post" or "get">

post处理短数据，以及提交的东西中带有<a>不行 若用到post 要在form下加上 {% csrf\_token %}

<input type="hidden" name="article\_id" value="0">

1. 写响应，响应与html文件有两种关系：①向html传入类；②用request接收html传出的form。

例子：

①def article\_page(request, article\_id):  
 article = models.Article.objects.get(pk=article\_id)  
 return render(request, 'blog/article\_page.html', {'article': article})

②可多传个参数（url要配合）：

def article\_page(request, article\_id):

article = models.Article.objects.get(pk=article\_id)

return render(request, 'blog/article\_page.html', {'article': article})

③接收form传回来的数据表：

def edit\_action(request):

title = request.POST.get('title', '无')

content = request.POST.get('content', '无')

models.Article.objects.create(title=title, content=content)

articles = models.Article.objects.all()

return render(request, 'blog/index.html', {'articles': articles})

④修改对象：

article = models.Article.objects.get(pk=article\_id)

article.title = title

article.content = content

article.save()

1. 写url

①path('article/<int:article\_id>', views.article\_page), <int:article\_id>为响应中函数的第二个参数article\_id

②html里需要用到{% url 'blog:article\_page' article.id %}时，需要对路径上的每一个url赋name

其中带include的，在include里添加namespace=’’，并且还要带上app的名字，和include的url组成一个元组

path('blog/', include(('blog.urls','blog'),namespace='blog'))

不带include的，在path里添加name=’’

path('article/<int:article\_id>', views.article\_page, name='article\_page'),