视频地址：<https://www.bilibili.com/video/BV1i7411e7CY/?spm_id_from=333.788.videocard.19>

源码地址：

# 项目1：博客项目

## 需求分析

1. 首页展示

2. 帖子分类

3. 帖子归档

4. 近期文章

5. 关于博主

6. 全文搜索

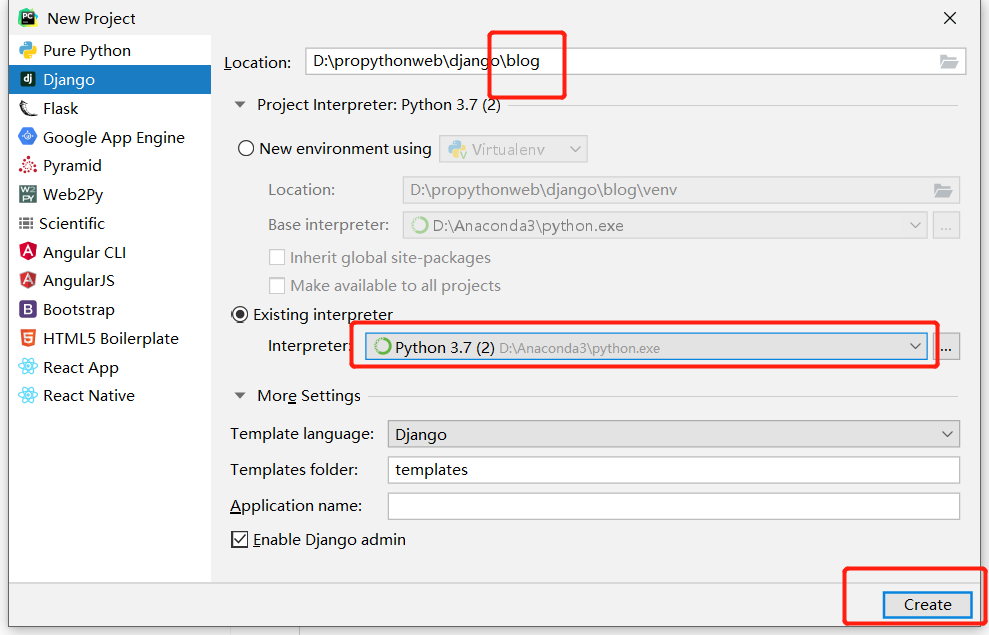
7. 发布帖子

## 项目实现

### 环境搭建

完成项目创建，挂载静态页面

* 在pycharm中file-new project-django，



* 创建post应用，在terminal中执行python manage.py startapp post
* 模版选取，打开网页，右键另存（网页全部）后，创建和网页同名文件夹，将网页放到改文件中；
* 右键项目名创建static（存放静态文件）和在post文件夹下创建tempaltes（存放网页模版）文件夹
* 将静态文件放到static中，将style.css放到staitc的css文件夹中
* 在tempaltes中创建base.html;从源码复制header.html,right.html,footer.html

关联css

<link rel**="stylesheet"** type**="text/css"** href**="/static/css/style.css"**>

* 配置settings.py中的静态文件，关联APP：INSTALLED\_APPS：中加’post’

STATIC\_URL = **'/static/'**#新添加的，指定了css文件路径  
STATICFILES\_DIRS = [  
 os.path.join(BASE\_DIR,**'static'**,**'css'**)  
]  
# global\_settings 从这个包中找到STATICFILES\_DIRS

* 修改跟路由urls

from django.conf.urls import url  
from django.contrib import admin  
from django.urls import path, include  
  
urlpatterns = [  
 path(**'admin/'**, admin.site.urls),  
 url(**r'^'**,include(**'post.urls'**))  
]

* 在post下新建子路由urls

from django.conf.urls import url  
  
from . import views  
  
urlpatterns=[  
 url(**r'^$'**,views.queryAll)  
]

* 在queryAll按alt+enter创建views

def queryAll(request):  
 return render(request,**'index.html'**)

* 在应用中加tempaltes文件夹加index.html，参考代码编写，引用base.html{% extends 'base.html' %}

### 发帖功能

创建数据表

* 在mysql数据库创建blog数据库
* 在setting中修改数据库连接为mysql，DATABASES，

**'ENGINE'**: **'django.db.backends.mysql'**,  
**'NAME'**: **'blog'**,  
**'HOST'**:**'127.0.0.1'**,  
**'PORT'**:3306,  
**'USER'**:**'root'**,  
**'PASSWORD'**:**'123456'**

* 在models创建数据表

from django.db import models  
  
# Create your models here.  
class Category(models.Model):  
 cname = models.CharField(max\_length=30,unique=True,verbose\_name=**u'类别名称'**)  
  
 class Meta:  
 db\_table = **'t\_category'** verbose\_name\_plural=**u'类别'** def \_\_str\_\_(self):  
 return **u'Category:%s'**%self.cname  
  
class Tag(models.Model):  
 tname = models.CharField(max\_length=30,unique=True,verbose\_name=**u'标签名称'**)  
  
 class Meta:  
 db\_table = **'t\_tag'** verbose\_name\_plural = **u'标签'** def \_\_str\_\_(self):  
 return **u'Tag:%s'** % self.tname  
  
class Post(models.Model):  
 title = models.CharField(max\_length=100,unique=True)  
 desc = models.CharField(max\_length=100)  
 content = models.TextField()  
 created = models.DateTimeField(auto\_now\_add=True)  
 category = models.ForeignKey(Category,on\_delete=models.CASCADE)  
 tag = models.ManyToManyField(Tag)  
  
 class Meta:  
 db\_table = **'t\_post'** verbose\_name\_plural = **u'帖子'** def \_\_str\_\_(self):  
 return **u'Post:%s'** % self.title

* 在terminal中用python manage.py makemigrations 映射models中的自建的数据表，在用python manage.py migrate创建到数据库，
* 在admin中注册models类

from .models import \*  
#注册models到后台  
admin.site.register(Category)  
admin.site.register(Tag)  
admin.site.register(Post)

* 创建管理用户：在terminal中：Python manage.py createsuperuser 用户名/密码：admin/admin
* 在setting中修改时区和语言 ：

LANGUAGE\_CODE = **'zh-Hans'**TIME\_ZONE = **'Asia/Shanghai'**

* 模型类在后台显示中文

模型名显示中文，在models中加：verbose\_name\_plural=**u'类别'**

**应用名显示中文：在post应用下，的\_\_init\_\_.py中修改：见源码**

字段名显示中文，在models中加verbose\_name=**u'类别名称'**

* 后台添加帖子

### 显示帖子内容

* 显示到前端

1. 修改views

def queryAll(request):  
 #获取所有帖子信息  
 postList = Post.objects.all().order\_by(**'-created'**)  
 return render(request,**'index.html'**,{**'postList'**:postList})

1. 修改index.html具体见源码

* 制作分页

1. 修改views

def queryAll(request,num=1):

1. 修改子路由：url(**r'^page/(\d+)$'**, views.queryAll)
2. 修改index.html具体见源码

<nav id**="page-nav"**>  
 {% if postList.has\_previous %}  
 <a class**="extend prev"** rel**="next"** href**="/page/**{{ postList.previous\_page\_number}}**"**>**Prev »**</a>  
 {% endif %}  
 {% for page in pageList %}  
 {% if currentNum == page %}  
 <span class**="page-number current"**>{{page}}</span>  
 {% else %}  
 <a class**="page-number"** href**="/page/**{{page}}**"**>{{page}}</a>  
 {% endif %}  
 {% endfor %}  
  
 {% if postList.has\_next %}  
 <a class**="extend next"** rel**="next"** href**="/page/**{{ postList.next\_page\_number}}**"**>**Next »**</a>  
 {% endif %}   
</nav>

### 帖子详情、分享功能

1、帖子详情

* 修改index.html：

<a href**="/post/**{{ post.id }}**"** target**="\_blank"**>**阅读全文**</a>

* 修改子路由

url(**r'^post/(\d+)$'**, views.detail),

* 修改views视图

def detail(request,postid):  
 postid = int(postid)  
 #根据postid查询帖子的详情信息  
 post = Post.objects.get(id=postid)  
 return render(request,**'detail.html'**,{**'post'**:post})

* 新建detail页面，代码加源码

2、自定义过滤器实现代码换行

* 在应用下新建文件夹templatetag，在该文件夹下新建myfilter文件；

from django.template import Library  
  
register = Library()  
@register.filter()  
def md(value):  
 import markdown  
 return markdown.markdown(value)

* 修改detail网页

代码加源码 {{ post.content|md|safe }}

3、分享功能

* 修改detail
* <div class**="bdsharebuttonbox"**><a href**="#"** class**="bds\_more"** data-cmd**="more"**></a><a href**="#"** class**="bds\_qzone"** data-cmd**="qzone"**></a><a href**="#"** class**="bds\_tsina"** data-cmd**="tsina"**></a><a href**="#"** class**="bds\_tqq"** data-cmd**="tqq"**></a><a href**="#"** class**="bds\_renren"** data-cmd**="renren"**></a><a href**="#"** class**="bds\_weixin"** data-cmd**="weixin"**></a></div>  
  <script>*window*.\_bd\_share\_config={"common":{"bdSnsKey":{},"bdText":"","bdMini":"2","bdPic":"","bdStyle":"0","bdSize":"16"},"share":{},"image":{"viewList":["qzone","tsina","tqq","renren","weixin"],"viewText":"分享到：","viewSize":"16"},"selectShare":{"bdContainerClass":null,"bdSelectMiniList":["qzone","tsina","tqq","renren","weixin"]}};with(*document*)0[(getElementsByTagName('head')[0]||body).appendChild(createElement('script')).src='http://bdimg.share.baidu.com/static/api/js/share.js?v=89860593.js?cdnversion='+~(-new *Date*()/36e5)];</script>  
  </div>

### 利用富文本发帖修改context字段





* 安装django-ckeditor库，在terminal中执行：pip install django-ckeditor
* 在settings中的INSTALLED\_APPS添加‘CKeditor’，‘ckeditor\_uploader’
* 配置MEDIA映射路径：放在settings中最下端

MEDIA\_URL = '/media/'

MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media')

CKEDITOR\_UPLOAD\_PATH = 'upload/'

* 在跟路由urls中添加路径，

并读取图片信息，导入包：from blog.settings import DEBUG, MEDIA\_ROOT

url(r'ckeditor/', include('ckeditor\_uploader.urls'))

]

from django.views.static import serve

if DEBUG:

urlpatterns+=url(r'^media/(?P<path>.\*)/$', serve, {"document\_root": MEDIA\_ROOT}),

* 修改models

from ckeditor\_uploader.fields import RichTextUploadingField

content = RichTextUploadingField()

* 在terminal中执行全局迁移文件

python manage.py makemigrations

python manage.py migrate

注：要上传图片需安装Pillow包

### 分类及近期文章功能

设置全局分类功能，通过全局上下文mycontextprocessor实现

* 在post下新建mycontextprocessor文件

from django.db.models import Count  
from post.models import Post  
  
def getRightInfo(request):  
 #1、获取分类信息  
 r\_catepost = Post.objects.values(**'category\_\_cname'**,**'category'**).annotate(c=Count(**'\*'**)).order\_by(**'-c'**)  
 #2、近期文章  
 r\_recpost = Post.objects.all().order\_by(**'-created'**)[:3]  
 return {**'r\_catepost'**:r\_catepost,**'r\_recpost'**:r\_recpost}

* 修改settings中的template模块

**'post.mycontextprocessor.getRightInfo'**,

* 修改right.html中的分类

{% for cp in r\_catepost %}

<li class="category-list-item">

<a class="category-list-link" href="/category/{{ cp.category }}">{{ cp.category\_\_cname }}</a>

<span class="category-list-count">{{ cp.c }}</span>

</li>

{% endfor %}

### 归档功能

mySql语句编写：select DATE\_FORMAT(created,'%Y-%m'),count(\*) from t\_post GROUP BY DATE\_FORMAT(created,'%Y-%m')

* 在post下修改mycontextprocessor文件

#3、获取日期归档信息  
from django.db import connection  
cursor = connection.cursor()  
cursor.execute(**"select created,count(\*) c from t\_post GROUP BY DATE\_FORMAT(created,'%Y-%m') ORDER BY c"**)  
r\_filepost = cursor.fetchall()

* 修改right.html中的归档

{% for fp in r\_filepost %}  
  
 <li class**="archive-list-item"**>  
 <a class**="archive-list-link"** href**="/archive/**{{ fp.0|date:'Y' }}**/**{{ fp.0|date:'m' }}**"**>{{ fp.0|date:'Y年m月' }}</a>  
 <span class**="archive-list-count"**>{{ fp.1 }}</span>  
 </li>  
  
{% endfor %}

### 分类、归档\_帖子列表

* 先找到right.html的分类块找到对应的连接路由：href**="/category/**{{ cp.category }}
* 在子路由中添加：

url(**r'^category/(\d+)$'**, views.queryPostByCid),  
url(**r'^archive/(\d+)/(\d+)$'**, views.queryPostByCreated),

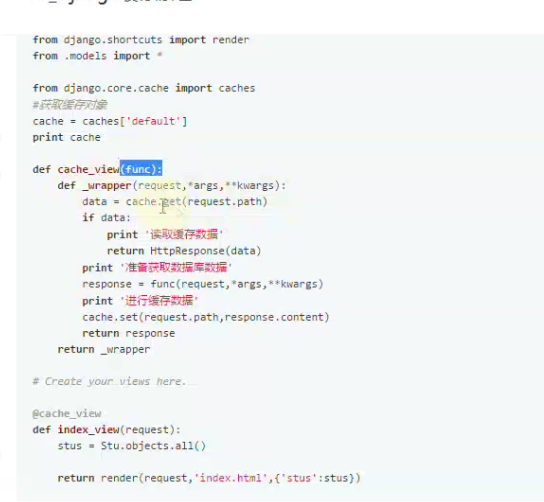
* 修改views

#根据类别id查询所有帖子  
def queryPostByCid(request,cid):  
  
 postList = Post.objects.filter(category\_id=cid)  
 # Post.objects.filter(category\_\_id=cid)  
  
 return render(request,**'article.html'**,{**'postList'**:postList})  
  
#根据发帖时间查询所有帖子  
def queryPostByCreated(request,year,month):  
  
 postList = Post.objects.filter(created\_\_year=year,created\_\_month=month)  
 return render(request,**'article.html'**,{**'postList'**:postList})

* 传到到article页面，见源码
* 注意修改settings中的时区：USE\_TZ = False

### 缓存原理

将多次查询的数据存到本地，通过装饰器实现caches



* 测试cache，新建cache应用，在terminal中执行python manage.py startapp cache
* 在settings中设置INSTALLED\_APPS，‘cache’
* 在models中创建clazz表，

class Clazz(models.Model):

cname = models.CharField(max\_length=30)

用python manage.py makemigrations cache 映射models中的自建的数据表，在用python manage.py migrate cache创建到数据库

* 跟路由设置：url(**r'^cache/'**, include(**'cache.urls'**))
* 在cache应用中添加urls子路由，并设置：

from django.conf.urls import url  
from . import views  
urlpatterns= {  
 url(**r'^$'**, views.index\_view)  
}

* 设置views

from django.shortcuts import render  
from cache.models import \*  
# Create your views here.  
def index\_view(request):  
 clist = Clazz.objects.all()  
 return render(request,**'indexn.html'**,{**'clist'**:clist})

* 新建termpaltes文件夹，和indexn.html文件

{% for c in clist %}  
 {{ c.cname }} <br>  
{% endfor %}

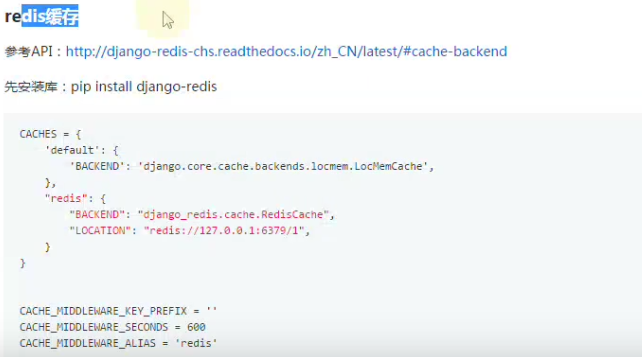
* 修改views，添加cache模块，见代码
* 在settings调整middleware及加cache，主要用全站缓存







### Redis缓存



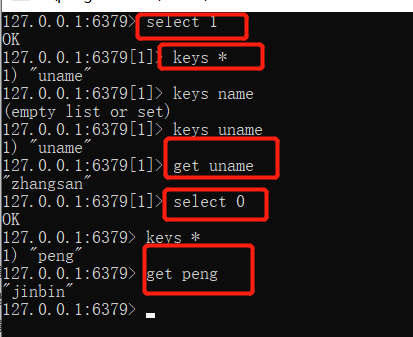
#### 下载redis数据库并安装

参考：<https://www.jianshu.com/p/e16d23e358c0>

默认有0-15个数据库，默认端口：6379

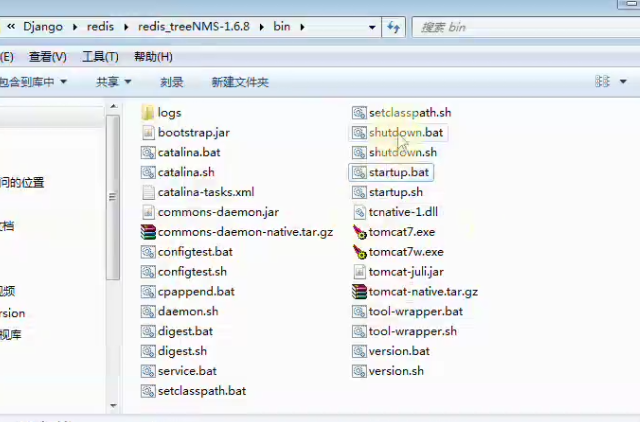
在安装目录，找到自带的客户端redis-cli，执行查询

Set uname zhangsan



#### 数据连接工具treesNMS treeNMS使用

* 工具 参考：<https://www.cnblogs.com/lxwphp/p/10007749.html>，见下载中的《windows布署说明》
* 在D:\treeDMS-2.3.12下的bin中找到startup.bat启动，默认端口：8085
* 可以连接DMS是mysql的treeDMS-2.3.12。NMS才是redis的treeNMS-1.7.5通过地址栏登录：打开浏览器，输入<http://127.0.0.1:8085/treesoft>，NMS才是redis的treeNMS-1.7.5http://127.0.0.1:8086/treenms默认用户名：admin 、treesoft，密码均为：treesoft，登录后请及时修改密码。



#### Django实现

* Python安装redis：

在terminal中执行pip install django\_redis

* 配置settings关联到redis数据库

CACHES = {

'default': {

'BACKEND': 'django.core.cache.backends.locmem.LocMemCache',

},

**'redis':{**

**"BACKEND": "django\_redis.cache.RedisCache",**

**"LOCATION": "redis://127.0.0.1:6379/1",**

**}**

}

最下边加

CACHE\_MIDDLEWARE\_KEY\_PREFIX = ''

CACHE\_MIDDLEWARE\_SECONDS = 600

CACHE\_MIDDLEWARE\_ALIAS = 'redis'

* 在python console中测试

from django.core.cache import caches

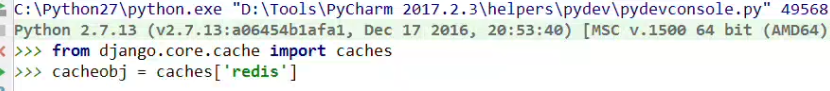
cacheobj = caches['redis']

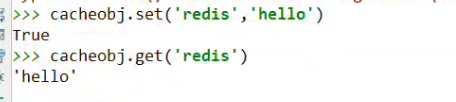
cacheobj.set('redis','hello')

Out[4]: True

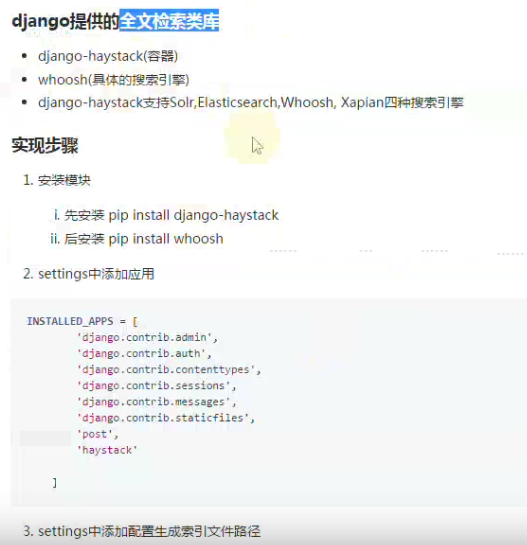
cacheobj.get('redis')

Out[5]: 'hello'

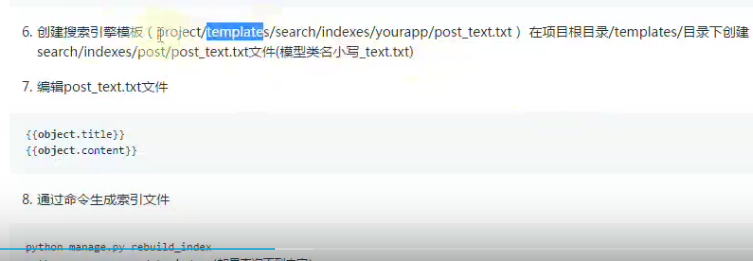


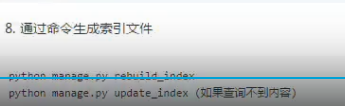


### 全文检索功能









* 安装django-haystack 和 whoosh包，在pycharm中file-setting安装，或terminal中pip install 。。
* 在settings添加INSTALLED\_APPS，‘haystack’
* 在settings中添加配置生成的索引文件，最下端

# 指定生成的索引路径

HAYSTACK\_CONNECTIONS = {

'default': {

'ENGINE': 'haystack.backends.whoosh\_backend.WhooshEngine',

'PATH': os.path.join(BASE\_DIR, 'whoosh\_index'),

},

}

# 实时生成索引文件

HAYSTACK\_SIGNAL\_PROCESSOR = 'haystack.signals.RealtimeSignalProcessor'

* 在postAPP中创建search\_indexes.py

from haystack import indexes  
from post.models import \*  
  
#注意格式(模型类名+Index)  
class PostIndex(indexes.SearchIndex,indexes.Indexable):  
 text = indexes.CharField(document=True, use\_template=True)  
  
 #给title,content设置索引  
 title = indexes.NgramField(model\_attr=**'title'**)  
 content = indexes.NgramField(model\_attr=**'content'**)  
  
 def get\_model(self):  
 return Post  
  
 def index\_queryset(self, using=None):  
 return self.get\_model().objects.order\_by(**'-created'**)

* 创建搜索引擎模版（project/templates/search/indexes/yourapp/post\_text.txt）模型类名post
* 在terminal中 执行：python manage.py rebuild\_index

(base) D:\propythonweb\django\blog>python manage.py rebuild\_index

WARNING: This will irreparably remove EVERYTHING from your search index in connection 'default'.

Your choices after this are to restore from backups or rebuild via the `rebuild\_index` command.

Are you sure you wish to continue? [y/N] y

Removing all documents from your index because you said so.

All documents removed.

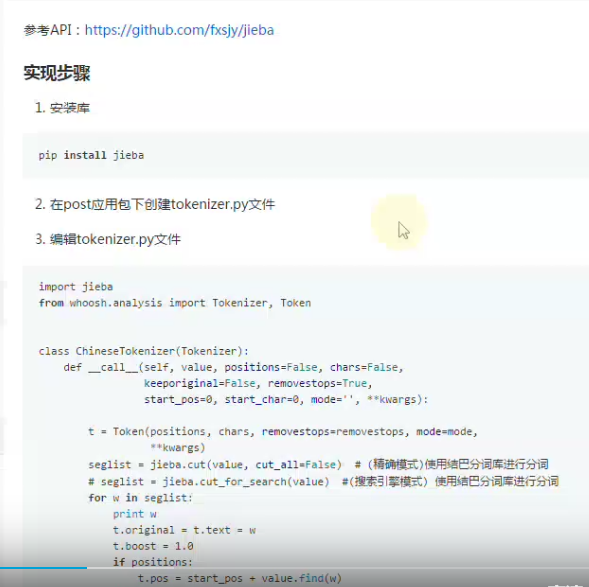
Indexing 5 帖子

* 编辑搜索框请求，header.html

<form method**="get"** action**="/search/"**>  
 <input type**="text"** name**="q"** placeholder**="搜索"**>  
</form>

* 配置跟路由，url(**r'^search/'**, include(**'haystack.urls'**)),
* 在项目根目录的templates—search中，创建search.html

### 中文分词--jieba分词





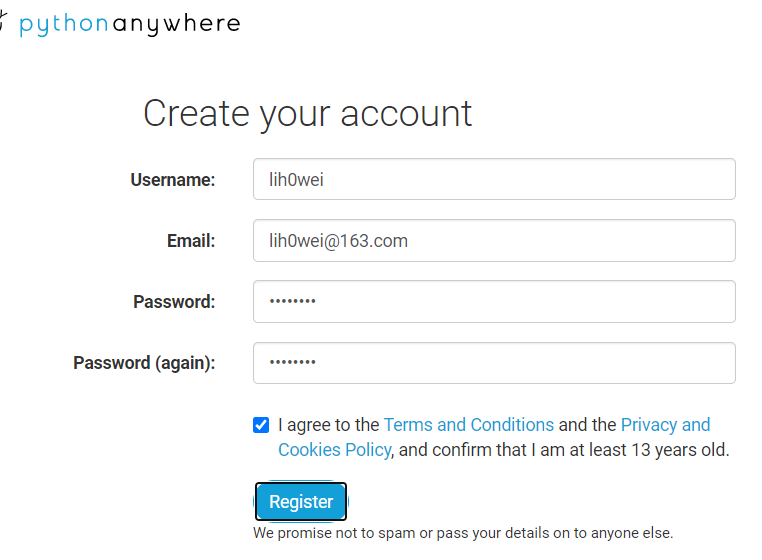
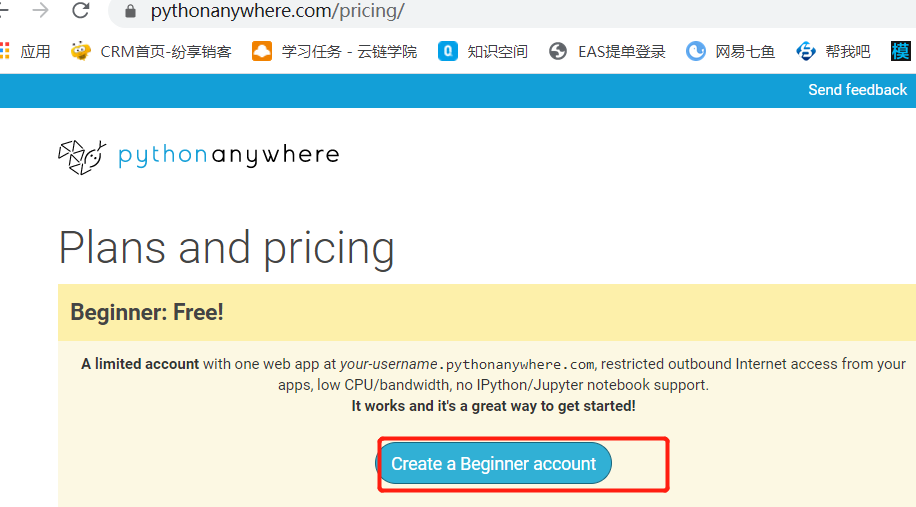
* 引入jieba包，在pycharm中file-setting安装，或terminal中pip install 。。
* 复制tokenizer.py文件到postAPP应用中；
* 拷贝D:\Anaconda3\Lib\site-packages\haystack\backends的whoosh\_backend.py文件到post目录下，改名whoosh\_cn\_backend.py
* 编辑post/ whoosh\_cn\_backend.py,ctrl+f搜索build\_schema
* 修改blog/settings中，修改

HAYSTACK\_CONNECTIONS = {  
 **'default'**: {  
 **'ENGINE'**: **'post.whoosh\_cn\_backend.WhooshEngine'**, 改此行

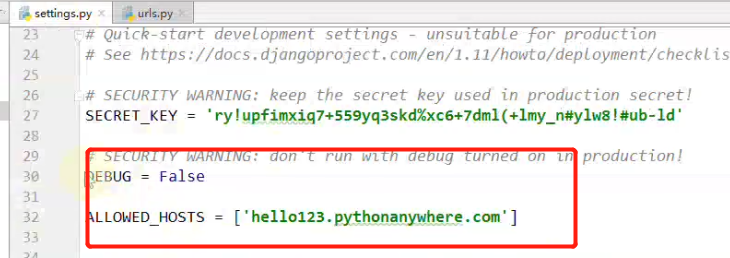
* 在terminal中 执行：python manage.py rebuild\_index，实现分词
* 在header的搜索实现搜索

## 部署项目

* 在[www.pythonanywhere.com](http://www.pythonanywhere.com)中注册



* 修改settings



**pythonanywhere.com**

**#### 虚拟环境**

**virtualenv blogenv --python=python2.7 创建虚拟环境**

**source activate 激活虚拟环境**

**deactivate 退出虚拟环境**

**workon blogenv 加入虚拟环境**

**#### 安装各种库文件**

**#### 实现步骤**

**1. 修改settings.py**

**```DEBUG = False**

**ALLOWED\_HOSTS = ['hello123.pythonanywhere.com']**

**```**

**2. 拷贝项目到桌面并且添加zip压缩包**

**3. 将blog.zip文件上传至pythonanywhere.com[files]目录中**

**4. 打开bash窗口 解压blog.zip文件**

**5. 创建虚拟环境blogenv**

**6. 激活虚拟环境**

**7. 在虚拟环境安装各种模块**

**8. 在pythonanywhere.com[web]目录中配置路径**

**9. reload服务器**

**具体可参考：**

**https://www.jianshu.com/p/c6cd366b65fe**