# 配置目录源码

├── asgi.py

├── database\_app\_router.py

├── \_\_init\_\_.py

├── settings.py

├── urls.py

└── wsgi.py

**asgi.py**

"""

ASGI config for myblog project.

It exposes the ASGI callable as a module-level variable named ``application``.

For more information on this file, see

https://docs.djangoproject.com/en/3.0/howto/deployment/asgi/

"""

import os

from django.core.asgi import get\_asgi\_application

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'myblog.settings')

application = get\_asgi\_application()

## ****database\_app\_router.py****

from django.conf import settings

DATABASE\_MAPPING = settings.DATABASE\_APPS\_MAPPING

class DatabaseAppsRouter(object):

"""

A router to control all database operations on models for different

databases.

In case an app is not set in settings.DATABASE\_APPS\_MAPPING, the router

will fallback to the `default` database.

Settings example:

DATABASE\_APPS\_MAPPING = {'app1': 'db1', 'app2': 'db2'}

"""

def db\_for\_read(self, model, \*\*hints):

""""Point all read operations to the specific database."""

if model.\_meta.app\_label in DATABASE\_MAPPING:

return DATABASE\_MAPPING[model.\_meta.app\_label]

return None

def db\_for\_write(self, model, \*\*hints):

"""Point all write operations to the specific database."""

if model.\_meta.app\_label in DATABASE\_MAPPING:

return DATABASE\_MAPPING[model.\_meta.app\_label]

return None

def allow\_relation(self, obj1, obj2, \*\*hints):

"""Allow any relation between apps that use the same database."""

db\_obj1 = DATABASE\_MAPPING.get(obj1.\_meta.app\_label)

db\_obj2 = DATABASE\_MAPPING.get(obj2.\_meta.app\_label)

if db\_obj1 and db\_obj2:

if db\_obj1 == db\_obj2:

return True

else:

return False

return None

def allow\_syncdb(self, db, model):

"""Make sure that apps only appear in the related database."""

if db in DATABASE\_MAPPING.values():

return DATABASE\_MAPPING.get(model.\_meta.app\_label) == db

elif model.\_meta.app\_label in DATABASE\_MAPPING:

return False

return None

def allow\_migrate(self, db, app\_label, model=None, \*\*hints):

"""

Make sure the auth app only appears in the 'auth\_db'

database.

"""

if db in DATABASE\_MAPPING.values():

return DATABASE\_MAPPING.get(app\_label) == db

elif app\_label in DATABASE\_MAPPING:

return False

return None

## ****settings.py****

"""

Django settings for myblog project.

Generated by 'django-admin startproject' using Django 3.0.1.

For more information on this file, see

https://docs.djangoproject.com/en/3.0/topics/settings/

For the full list of settings and their values, see

https://docs.djangoproject.com/en/3.0/ref/settings/

"""

import os

# Build paths inside the project like this: os.path.join(BASE\_DIR, ...)

BASE\_DIR = os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_)))

# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/3.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!

SECRET\_KEY = '1dk13+$oiww#bc6#=7gd75dd=l6(%dzkzzqp#vgfpug-nx19t8'

# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = False

ALLOWED\_HOSTS = ['\*']

# Application definition

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'blog.apps.BlogConfig',

'DjangoUeditor',

]

MIDDLEWARE = [

'django.middleware.security.SecurityMiddleware',

'django.contrib.sessions.middleware.SessionMiddleware',

'django.middleware.common.CommonMiddleware',

#'django.middleware.csrf.CsrfViewMiddleware',

'django.contrib.auth.middleware.AuthenticationMiddleware',

'django.contrib.messages.middleware.MessageMiddleware',

'django.middleware.clickjacking.XFrameOptionsMiddleware',

]

ROOT\_URLCONF = 'myblog.urls'

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates',

'DIRS': [os.path.join(BASE\_DIR, 'templates')]

,

'APP\_DIRS': True,

'OPTIONS': {

'context\_processors': [

'django.template.context\_processors.debug',

'django.template.context\_processors.request',

'django.contrib.auth.context\_processors.auth',

'django.contrib.messages.context\_processors.messages',

'blog.views.global\_variable',

],

},

},

]

WSGI\_APPLICATION = 'myblog.wsgi.application'

# Database

# https://docs.djangoproject.com/en/3.0/ref/settings/#databases

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.mysql',

'NAME': 'blog',

'USER': 'root',

'PASSWORD': '\*\*\*\*\*\*',

'HOST': '127.0.0.1',

'PORT': '3306',

'OPTIONS': {

"init\_command": "SET foreign\_key\_checks = 0;",

},

},

'db2': {

'ENGINE': 'django.db.backends.mysql',

'NAME': 'rhyme',

'USER': 'root',

'PASSWORD': '\*\*\*\*\*\*',

'HOST': '127.0.0.1',

'PORT': '3306',

'OPTIONS': {

"init\_command": "SET foreign\_key\_checks = 0;",

},

}

}

DATABASE\_ROUTERS = ['myblog.database\_app\_router.DatabaseAppsRouter']

DATABASE\_APPS\_MAPPING = {

# example:

# 'app\_name':'database\_name',

'blog': 'default',

}

# Password validation

# https://docs.djangoproject.com/en/3.0/ref/settings/#auth-password-validators

AUTH\_PASSWORD\_VALIDATORS = [

{

'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',

},

]

# Internationalization

# https://docs.djangoproject.com/en/3.0/topics/i18n/

LANGUAGE\_CODE = 'zh-hans'

TIME\_ZONE = 'Asia/Shanghai'

USE\_I18N = True

USE\_L10N = True

USE\_TZ = True

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/3.0/howto/static-files/

STATIC\_URL = '/static/'

#这个是设置静态文件夹目录的路径

STATICFILES\_DIRS = (

os.path.join(BASE\_DIR, 'static'),

)

#设置文件上传路径，图片上传、文件上传都会存放在此目录里

MEDIA\_URL = '/media/'

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

## ****urls.py****

"""myblog URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:

https://docs.djangoproject.com/en/3.0/topics/http/urls/

Examples:

Function views

1. Add an import: from my\_app import views

2. Add a URL to urlpatterns: path('', views.home, name='home')

Class-based views

1. Add an import: from other\_app.views import Home

2. Add a URL to urlpatterns: path('', Home.as\_view(), name='home')

Including another URLconf

1. Import the include() function: from django.urls import include, path

2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))

"""

from django.contrib import admin

from django.urls import path

from blog import views

from django.urls import path, include, re\_path

from django.views.static import serve

#导入静态文件模块

from django.conf import settings

#导入配置文件里的文件上传配置

urlpatterns = [

path('admin/', admin.site.urls),

path('get\_ip/',views.get\_ip),

path('showlist/', views.showlist),

path('ueditor/', include('DjangoUeditor.urls')),

re\_path('^media/(?P<path>.\*)$', serve, {'document\_root': settings.MEDIA\_ROOT}),

path('bk/', views.bkindex), # 网站首页

path('', views.index, name='index'), # 网站首页

path('list-<int:lid>.html', views.listx, name='list'), # 列表页

path('show-<int:sid>.html', views.show, name='show'), # 内容页

path('tag/<tag>', views.tag, name='tags'), # 标签列表页

path('s/', views.search, name='search'), # 搜索列表页

path('about/', views.about, name='about'), # 联系我们单页

path('testdb/', views.testdb, name='testdb'),

path('wxpost/', views.wxpost, name='wxpost'),

]

## ****wsgi.py****

"""

WSGI config for myblog project.

It exposes the WSGI callable as a module-level variable named ``application``.

For more information on this file, see

https://docs.djangoproject.com/en/3.0/howto/deployment/wsgi/

"""

import os

from django.core.wsgi import get\_wsgi\_application

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'myblog.settings')

application = get\_wsgi\_application()

# 视图目录源码

├── admin.py

├── apps.py

├── \_\_init\_\_.py

├── migrations

│   ├── 0001\_initial.py

│   ├── 0002\_auto\_20200831\_2048.py

│   ├── 0003\_auto\_20201029\_1631.py

│   ├── 0004\_niceday.py

│   ├── \_\_init\_\_.py

├── models.py

└── views.py

## ****admin.py****

from django.contrib import admin

from .models import Banner, Category, Tag, Tui, Article, Link

# Register your models here.

admin.site.site\_header = '管理后台'

admin.site.site\_title = 'Django Blog'

@admin.register(Article)

class ArticleAdmin(admin.ModelAdmin):

list\_display = ('id', 'category', 'title', 'tui', 'user', 'views', 'created\_time')

# 文章列表里显示想要显示的字段

list\_per\_page = 50

# 满50条数据就自动分页

ordering = ('-created\_time',)

#后台数据列表排序方式

list\_display\_links = ('id', 'title')

# 设置哪些字段可以点击进入编辑界面

@admin.register(Banner)

class BannerAdmin(admin.ModelAdmin):

list\_display = ('id', 'text\_info', 'img', 'link\_url', 'is\_active')

@admin.register(Category)

class CategoryAdmin(admin.ModelAdmin):

list\_display = ('id', 'name', 'index')

@admin.register(Tag)

class TagAdmin(admin.ModelAdmin):

list\_display = ('id', 'name')

@admin.register(Tui)

class TuiAdmin(admin.ModelAdmin):

list\_display = ('id', 'name')

@admin.register(Link)

class LinkAdmin(admin.ModelAdmin):

list\_display = ('id', 'name','linkurl')

## ****apps.py****

from django.apps import AppConfig

class BlogConfig(AppConfig):

name = 'blog'

## migrations/****0001\_initial.py****

# Generated by Django 3.0.1 on 2019-12-23 03:41

from django.conf import settings

from django.db import migrations, models

import django.db.models.deletion

class Migration(migrations.Migration):

initial = True

dependencies = [

migrations.swappable\_dependency(settings.AUTH\_USER\_MODEL),

]

operations = [

migrations.CreateModel(

name='Banner',

fields=[

('id', models.AutoField(auto\_created=True, primary\_key=True, serialize=False, verbose\_name='ID')),

('text\_info', models.CharField(default='', max\_length=50, verbose\_name='标题')),

('img', models.ImageField(upload\_to='banner/', verbose\_name='轮播图')),

('link\_url', models.URLField(max\_length=100, verbose\_name='图片链接')),

('is\_active', models.BooleanField(default=False, verbose\_name='是否是active')),

],

options={

'verbose\_name': '轮播图',

'verbose\_name\_plural': '轮播图',

},

),

migrations.CreateModel(

name='Category',

fields=[

('id', models.AutoField(auto\_created=True, primary\_key=True, serialize=False, verbose\_name='ID')),

('name', models.CharField(max\_length=100, verbose\_name='博客分类')),

('index', models.IntegerField(default=999, verbose\_name='分类排序')),

],

options={

'verbose\_name': '博客分类',

'verbose\_name\_plural': '博客分类',

},

),

migrations.CreateModel(

name='Link',

fields=[

('id', models.AutoField(auto\_created=True, primary\_key=True, serialize=False, verbose\_name='ID')),

('name', models.CharField(max\_length=20, verbose\_name='链接名称')),

('linkurl', models.URLField(max\_length=100, verbose\_name='网址')),

],

options={

'verbose\_name': '友情链接',

'verbose\_name\_plural': '友情链接',

},

),

migrations.CreateModel(

name='Tag',

fields=[

('id', models.AutoField(auto\_created=True, primary\_key=True, serialize=False, verbose\_name='ID')),

('name', models.CharField(max\_length=100, verbose\_name='文章标签')),

],

options={

'verbose\_name': '文章标签',

'verbose\_name\_plural': '文章标签',

},

),

migrations.CreateModel(

name='Tui',

fields=[

('id', models.AutoField(auto\_created=True, primary\_key=True, serialize=False, verbose\_name='ID')),

('name', models.CharField(max\_length=100, verbose\_name='推荐位')),

],

options={

'verbose\_name': '推荐位',

'verbose\_name\_plural': '推荐位',

},

),

migrations.CreateModel(

name='Article',

fields=[

('id', models.AutoField(auto\_created=True, primary\_key=True, serialize=False, verbose\_name='ID')),

('title', models.CharField(max\_length=70, verbose\_name='标题')),

('excerpt', models.TextField(blank=True, max\_length=200, verbose\_name='摘要')),

('img', models.ImageField(blank=True, null=True, upload\_to='article\_img/%Y/%m/%d/', verbose\_name='文章图片')),

('body', models.TextField()),

('views', models.PositiveIntegerField(default=0, verbose\_name='阅读量')),

('created\_time', models.DateTimeField(auto\_now\_add=True, verbose\_name='发布时间')),

('modified\_time', models.DateTimeField(auto\_now=True, verbose\_name='修改时间')),

('category', models.ForeignKey(blank=True, null=True, on\_delete=django.db.models.deletion.DO\_NOTHING, to='blog.Category', verbose\_name='分类')),

('tags', models.ManyToManyField(blank=True, to='blog.Tag', verbose\_name='标签')),

('tui', models.ForeignKey(blank=True, null=True, on\_delete=django.db.models.deletion.DO\_NOTHING, to='blog.Tui', verbose\_name='推荐位')),

('user', models.ForeignKey(on\_delete=django.db.models.deletion.CASCADE, to=settings.AUTH\_USER\_MODEL, verbose\_name='作者')),

],

options={

'verbose\_name': '文章',

'verbose\_name\_plural': '文章',

},

),

]

## migrations/****0002\_auto\_20200831\_2048.py****

# Generated by Django 3.0.3 on 2020-08-31 12:48

import DjangoUeditor.models

from django.db import migrations, models

class Migration(migrations.Migration):

dependencies = [

('blog', '0001\_initial'),

]

operations = [

migrations.CreateModel(

name='Cihai',

fields=[

('id', models.AutoField(db\_column='ID', primary\_key=True, serialize=False)),

('words', models.CharField(blank=True, max\_length=255, null=True)),

('content', models.CharField(blank=True, max\_length=255, null=True)),

('yin', models.CharField(blank=True, max\_length=255, null=True)),

('yun', models.CharField(blank=True, max\_length=255, null=True)),

('key1', models.CharField(blank=True, max\_length=255, null=True)),

('key2', models.CharField(blank=True, max\_length=255, null=True)),

('key3', models.CharField(blank=True, max\_length=255, null=True)),

('key4', models.CharField(blank=True, max\_length=255, null=True)),

('key5', models.CharField(blank=True, max\_length=255, null=True)),

('times', models.IntegerField(blank=True, default=0)),

],

options={

'db\_table': 'cihai',

'managed': True,

},

),

migrations.CreateModel(

name='Words',

fields=[

('id', models.AutoField(auto\_created=True, primary\_key=True, serialize=False, verbose\_name='ID')),

('word', models.CharField(max\_length=255)),

('pron', models.CharField(blank=True, max\_length=255, null=True)),

('ch', models.CharField(blank=True, max\_length=255, null=True)),

('key1', models.CharField(blank=True, max\_length=255, null=True)),

('key2', models.CharField(blank=True, max\_length=255, null=True)),

('key3', models.CharField(blank=True, max\_length=255, null=True)),

('key4', models.CharField(blank=True, max\_length=255, null=True)),

('times', models.IntegerField(blank=True, default=0)),

],

options={

'db\_table': 'words',

'managed': True,

},

),

migrations.AlterField(

model\_name='article',

name='body',

field=DjangoUeditor.models.UEditorField(blank=True, verbose\_name='内容'),

),

]

## migrations/****0003\_auto\_20201029\_1631.py****

# Generated by Django 3.0.3 on 2020-10-29 08:31

from django.db import migrations

class Migration(migrations.Migration):

dependencies = [

('blog', '0002\_auto\_20200831\_2048'),

]

operations = [

migrations.AlterModelTable(

name='cihai',

table='Cihai',

),

migrations.AlterModelTable(

name='words',

table='Words',

),

]

## migrations/****0004\_niceday.py****

# Generated by Django 3.0.3 on 2021-05-03 13:33

from django.db import migrations, models

class Migration(migrations.Migration):

dependencies = [

('blog', '0003\_auto\_20201029\_1631'),

]

operations = [

migrations.CreateModel(

name='Niceday',

fields=[

('id', models.AutoField(db\_column='ID', primary\_key=True, serialize=False)),

('sentence', models.CharField(blank=True, max\_length=255, null=True)),

],

options={

'db\_table': 'Niceday',

'managed': True,

},

),

]

## ****models.py****

from django.db import models

from DjangoUeditor.models import UEditorField

# Create your models here.

from django.contrib.auth.models import User

#导入Django自带用户模块

# 文章分类

class Category(models.Model):

name = models.CharField('博客分类', max\_length=100)

index = models.IntegerField(default=999, verbose\_name='分类排序')

class Meta:

verbose\_name = '博客分类'

verbose\_name\_plural = verbose\_name

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

return self.name

#文章标签

class Tag(models.Model):

name = models.CharField('文章标签',max\_length=100)

class Meta:

verbose\_name = '文章标签'

verbose\_name\_plural = verbose\_name

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

return self.name

#推荐位

class Tui(models.Model):

name = models.CharField('推荐位',max\_length=100)

class Meta:

verbose\_name = '推荐位'

verbose\_name\_plural = verbose\_name

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

return self.name

#文章

class Article(models.Model):

title = models.CharField('标题', max\_length=70)

excerpt = models.TextField('摘要', max\_length=200, blank=True)

category = models.ForeignKey(Category, on\_delete=models.DO\_NOTHING, verbose\_name='分类', blank=True, null=True)

#使用外键关联分类表与分类是一对多关系

tags = models.ManyToManyField(Tag,verbose\_name='标签', blank=True)

#使用外键关联标签表与标签是多对多关系

img = models.ImageField(upload\_to='article\_img/%Y/%m/%d/', verbose\_name='文章图片', blank=True, null=True)

body = UEditorField('内容', width=800, height=500,

toolbars="full", imagePath="upimg/", filePath="upfile/",

upload\_settings={"imageMaxSize": 1204000},

settings={}, command=None, blank=True

)

user = models.ForeignKey(User, on\_delete=models.CASCADE, verbose\_name='作者')

"""

文章作者，这里User是从django.contrib.auth.models导入的。

这里我们通过 ForeignKey 把文章和 User 关联了起来。

"""

views = models.PositiveIntegerField('阅读量', default=0)

tui = models.ForeignKey(Tui, on\_delete=models.DO\_NOTHING, verbose\_name='推荐位', blank=True, null=True)

created\_time = models.DateTimeField('发布时间', auto\_now\_add=True)

modified\_time = models.DateTimeField('修改时间', auto\_now=True)

class Meta:

verbose\_name = '文章'

verbose\_name\_plural = '文章'

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

return self.title

#Banner

class Banner(models.Model):

text\_info = models.CharField('标题', max\_length=50, default='')

img = models.ImageField('轮播图', upload\_to='banner/')

link\_url = models.URLField('图片链接', max\_length=100)

is\_active = models.BooleanField('是否是active', default=False)

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

return self.text\_info

class Meta:

verbose\_name = '轮播图'

verbose\_name\_plural = '轮播图'

#友情链接

class Link(models.Model):

name = models.CharField('链接名称', max\_length=20)

linkurl = models.URLField('网址',max\_length=100)

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

return self.name

class Meta:

verbose\_name = '友情链接'

verbose\_name\_plural = '友情链接'

class Cihai(models.Model):

id = models.AutoField(db\_column='ID', primary\_key=True) # Field name made lowercase.

words = models.CharField(max\_length=255, blank=True, null=True)

content = models.CharField(max\_length=255, blank=True, null=True)

yin = models.CharField(max\_length=255, blank=True, null=True)

yun = models.CharField(max\_length=255, blank=True, null=True)

key1 = models.CharField(max\_length=255, blank=True, null=True)

key2 = models.CharField(max\_length=255, blank=True, null=True)

key3 = models.CharField(max\_length=255, blank=True, null=True)

key4 = models.CharField(max\_length=255, blank=True, null=True)

key5 = models.CharField(max\_length=255, blank=True, null=True)

times = models.IntegerField(blank=True, default=0)

class Meta:

managed = True

db\_table = 'Cihai'

class Words(models.Model):

word = models.CharField(max\_length=255)

pron = models.CharField(max\_length=255, blank=True, null=True)

ch = models.CharField(max\_length=255, blank=True, null=True)

key1 = models.CharField(max\_length=255, blank=True, null=True)

key2 = models.CharField(max\_length=255, blank=True, null=True)

key3 = models.CharField(max\_length=255, blank=True, null=True)

key4 = models.CharField(max\_length=255, blank=True, null=True)

times = models.IntegerField(blank=True, default=0)

class Meta:

managed = True

db\_table = 'Words'

class Niceday(models.Model):

id = models.AutoField(db\_column='ID', primary\_key=True) # Field name made lowercase.

sentence = models.CharField(max\_length=255, blank=True, null=True)

class Meta:

managed = True

db\_table = 'Niceday'

## ****views.py****

from django.db.models import F

from django.shortcuts import render

from django.http import HttpResponse ,JsonResponse

from .models import Article

from .models import Category, Banner, Article, Tag, Link, Cihai, Words, Niceday

from django.core.paginator import Paginator, EmptyPage, PageNotAnInteger

import random

import requests

import re

# Create your views here.

def global\_variable(request):

allcategory = Category.objects.all()

remen = Article.objects.filter(tui\_\_id=2)[:6]

tags = Tag.objects.all()

return locals()

def getprovince(ipaddr):

headers = {"User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_12\_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36"}

url = 'http://ip.ws.126.net/ipquery?ip={}'.format(ipaddr)

r = requests.get(url,headers = headers)

#out = re.findall('localAddress={city:"荆门市", province:"湖北省"}',r.text)

out = re.findall('province:"(.\*?)"', r.text)

# print(out)

pr = {}

pr.update(province=out[0])

#return JsonResponse(pr,safe=False)

return pr

def get\_ip(request):

if 'HTTP\_X\_FORWARDED\_FOR' in request.META:

ip = request.META['HTTP\_X\_FORWARDED\_FOR']

else:

ip = request.META['REMOTE\_ADDR']

return JsonResponse(getprovince(ip),safe=False)

def testdb(request):

#ll = Niceday.objects.all().using('db2').filter(key3\_\_contains='ang')[0:10]

ll = Niceday.objects.all().using('db2').order\_by('?').first()

return HttpResponse(ll.sentence)

def wxpost(request):

if request.POST.get('language') == 'Chinese':

mapping = {'s': 'key1', 'd': 'key2', 't': 'key3', 'q': 'key4'}

json\_data = {}

data\_list = []

key = mapping[request.POST.get('mult')]

kw = request.POST.get('keyword')

#print(kw)

# items = Cihai.objects.all().using('db2').raw(

# "SELECT \* from Cihai WHERE key1=(SELECT key1 from Cihai where binary words='一心一意')")

items = Cihai.objects.all().using('db2').raw("SELECT \* from Cihai WHERE {}=(SELECT {} from Cihai where words='{}') and CHAR\_LENGTH(words)={}".format(key, key, kw, len(kw)))

#Cihai.objects.using('db2').update\_or\_create(words=kw, times=F('times') + 1, defaults={'words': kw},)

#print(len(items))

if len(items):

Cihai.objects.using('db2').filter(words=kw).update(times=F('times') + 1)

#print(len(items))

#print(type(items))

items = random.sample(list(items), 100 if len(items) > 100 else len(items))

#print(len(items))

for item in items:

#print(type(item))

data = {}

data["words"] = item.words

data["content"] = item.content

data["yin"] = item.yin

data["key"] = getattr(item, key)

#data["key"] = item.key1

data\_list.append(data)

json\_data['data'] = data\_list[:100]

return JsonResponse(json\_data)

#return HttpResponse(a.raw\_query)

else:

kw = re.sub("[A-Za-z0-9\!\%\[\]\,\。\ ]", "", kw)

Cihai.objects.using('db2').update\_or\_create(words=kw[0:4], times=1, content='暂未收录', yin='zan wei shou lu', key1='u', key2='ou,u', key3='ei,ou,u', key4='an,ei,ou,u')

json\_data['data'] = [

{

"words": kw,

"content": "暂未收录",

"yin": "zan wei shou lu",

"key": "an,ei,ou,u"

},

]

return JsonResponse(json\_data)

elif request.POST.get('language') == 'English':

mapping = {'s': 'key1', 'd': 'key2', 't': 'key3', 'q': 'key4'}

json\_data = {}

data\_list = []

key = mapping[request.POST.get('mult')]

kw = request.POST.get('keyword')

#print(kw)

# items = Cihai.objects.all().using('db2').raw(

# "SELECT \* from Cihai WHERE key1=(SELECT key1 from Cihai where binary words='一心一意')")

items = Words.objects.all().using('db2').raw(

"SELECT \* from Words WHERE {}=(SELECT {} from Words where binary word='{}') ".format(key, key, kw))

# print(len(items))

# print(type(items))

if len(items):

Words.objects.using('db2').filter(word=kw).update(times=F('times') + 1)

items = random.sample(list(items), 100 if len(items) > 100 else len(items))

#print(len(items))

for item in items:

# print(type(item))

data = {}

data["words"] = item.word

data["content"] = item.ch

data["yin"] = item.pron

data["key"] = getattr(item, key)

# data["key"] = item.key1

data\_list.append(data)

json\_data['data'] = data\_list[:100]

return JsonResponse(json\_data)

#return HttpResponse(ll.values())

else:

return JsonResponse({'data': [{

"words": kw,

"content": "暂未收录",

"yin": "",

"key": ""

}]}, safe=False)

else:

return HttpResponse("出错了哦~")

def showlist(request):

#添加两个变量，并给它们赋值

sitename = 'Django中文网'

url = 'www.django.cn'

#把两个变量封装到上下文里

list = [

'开发前的准备',

'项目需求分析',

'数据库设计分析',

'创建项目',

'基础配置',

'欢迎页面',

'创建数据库模型',

]

mydict={

'name': '吴秀峰',

'qq': '445813',

'wx': 'vipdjango',

'email': '445813@qq.com',

'Q群': '10218442',

}

context = {

'sitename': sitename,

'url': url,

'list': list,

'mydict': mydict,

}

#把上下文传递到模板里

return render(request, 'showlist.html', context)

def bkindex(request):

#对Article进行声明并实例化，然后生成对象allarticle

allarticle = Article.objects.all()

#把查询到的对象，封装到上下文

context = {

'allarticle': allarticle,

}

#把上传文传到模板页面index.html里

return render(request, 'bkindex.html', context)

def index(request):

allarticle = Article.objects.all().order\_by('-id')[0:10]#通过Category表查出所有分类

#把查询出来的分类封装到上下文里

banner = Banner.objects.filter(is\_active=True)[0:4] # 查询所有幻灯图数据，并进行切片

tui = Article.objects.filter(tui\_\_id=1)[:3]

#hot = Article.objects.all().order\_by('?')[:10]#随机推荐

#hot = Article.objects.filter(tui\_\_id=3)[:10] #通过推荐进行查询，以推荐ID是3为例

hot = Article.objects.all().order\_by('views')[:10]#通过浏览数进行排序

link = Link.objects.all()

return render(request, 'index.html', locals())#把上下文传到index.html页面

#列表页

def listx(request,lid):

list = Article.objects.filter(category\_id=lid).order\_by('-id')#获取通过URL传进来的lid，然后筛选出对应文章

cname = Category.objects.get(id=lid)#获取当前文章的栏目名

page = request.GET.get('page') # 在URL中获取当前页面数

paginator = Paginator(list, 5) # 对查询到的数据对象list进行分页，设置超过5条数据就分页

try:

list = paginator.page(page) # 获取当前页码的记录

except PageNotAnInteger:

list = paginator.page(1) # 如果用户输入的页码不是整数时,显示第1页的内容

except EmptyPage:

list = paginator.page(paginator.num\_pages) # 如果用户输入的页数不在系统的页码列表中时,显示最后一页的内容

return render(request, 'list.html', locals())

#内容页

def show(request,sid):

show = Article.objects.get(id=sid)#查询指定ID的文章

hot = Article.objects.all().order\_by('?')[:10]#内容下面的您可能感兴趣的文章，随机推荐

previous\_blog = Article.objects.filter(created\_time\_\_gt=show.created\_time,category=show.category.id).first()

netx\_blog = Article.objects.filter(created\_time\_\_lt=show.created\_time,category=show.category.id).last()

show.views = show.views + 1

show.save()

return render(request, 'show.html', locals())

#标签页

def tag(request, tag):

list = Article.objects.filter(tags\_\_name=tag).order\_by('-id')

tname = Tag.objects.get(name=tag)#获取当前搜索的标签名

page = request.GET.get('page')

paginator = Paginator(list, 5)

try:

list = paginator.page(page) # 获取当前页码的记录

except PageNotAnInteger:

list = paginator.page(1) # 如果用户输入的页码不是整数时,显示第1页的内容

except EmptyPage:

list = paginator.page(paginator.num\_pages) # 如果用户输入的页数不在系统的页码列表中时,显示最后一页的内容

return render(request, 'tags.html', locals())

# 搜索页

def search(request):

ss=request.GET.get('search')#获取搜索的关键词

list = Article.objects.filter(title\_\_icontains=ss)#获取到搜索关键词通过标题进行匹配

page = request.GET.get('page')

paginator = Paginator(list, 10)

try:

list = paginator.page(page) # 获取当前页码的记录

except PageNotAnInteger:

list = paginator.page(1) # 如果用户输入的页码不是整数时,显示第1页的内容

except EmptyPage:

list = paginator.page(paginator.num\_pages) # 如果用户输入的页数不在系统的页码列表中时,显示最后一页的内容

return render(request, 'search.html', locals())

# 关于我们

def about(request):

return render(request, 'page.html',locals())

# 定时任务源码

## ****wordsrank.py****

#-\*-coding:utf-8-\*-

import numpy as np

from pandas import DataFrame

import pandas as pd

from sqlalchemy import create\_engine

import pymysql

import matplotlib.pyplot as plt

import matplotlib.dates as mdates

from matplotlib.dates import DateFormatter, WeekdayLocator, DayLocator, MO, TU

from datetime import datetime

from matplotlib.ticker import MultipleLocator, FormatStrFormatter

import mysql.connector

from tqdm import tqdm

import time

from matplotlib.font\_manager import \_rebuild

\_rebuild()

pymysql.install\_as\_MySQLdb()

ce = create\_engine("mysql+mysqlconnector://root:\*\*\*\*\*\*@localhost:3306/rhyme", encoding='utf-8')

sql1 = "select words,times from `Cihai` where times>=1 order by times desc limit 10"

data1 = pd.read\_sql\_query(sql1, con=ce)

print(data1)

data1 = data1.set\_index(data1['times']).sort\_index(ascending=True)

fig1 = plt.figure(figsize=(9, 16))

ax1 = fig1.add\_subplot(1,1,1)

#data1['ratio'].plot(label='比值')

plt.rcParams['font.sans-serif'] = ['SimHei'] #显示中文标签

plt.rcParams['axes.unicode\_minus'] = False

xs = [d for d in data1['words']]

#xmajorLocator = MultipleLocator(5) # 定义横向主刻度标签的刻度差为2的倍数。就是隔几个刻度才显示一个标签文本

plt.barh(xs, data1.index, color=['c', 'g', 'b', 'y', 'm', 'r', 'grey', 'gold', 'darkviolet', 'turquoise'], label='热词TOP10')

plt.xticks([])# 不显示x轴

plt.yticks(size='medium', rotation=45, fontsize=16)

ax1.legend(loc='upper left')

ax1.set\_title('更新时间：{}'.format(time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())), fontsize=16)

plt.savefig('rank.png')

plt.show()

**shforcron.sh**

#!/bin/bash

export PATH=$PATH:/usr/local/bin

source /root/anaconda3/bin/activate py37

cd ~/myblog/static/images/

python ~/myblog/static/images/wordsrank.py

# 存储过程源码

**updatetimes**

CREATE DEFINER=`root`@`localhost` PROCEDURE `top10`()

BEGIN

#Routine body goes here...

update Cihai set times=1 where times>=1 ORDER BY times desc LIMIT 9;

END

**newwordsproccess**

CREATE DEFINER=`root`@`localhost` PROCEDURE `tt`(IN `mount` INT,out `outww` varchar(128))

BEGIN

DECLARE ww varchar(32);

DECLARE wlen INT(11);

DECLARE clen INT(11);

DECLARE y1 VARCHAR(25);

DECLARE y2 VARCHAR(25);

DECLARE y3 VARCHAR(25);

DECLARE y4 VARCHAR(25);

SELECT words into ww from Cihai where content='暂未收录' and times>=1 ORDER BY times desc LIMIT mount;

set wlen = char\_LENGTH(ww);

set clen = LENGTH(ww);

if wlen != clen then

CASE wlen

WHEN 2 THEN

select key1 into y1 from Cihai where words=convert((select substring(ww, 2, 1)) using utf8) collate utf8\_unicode\_ci;

select key1 into y2 from Cihai where words=convert((select substring(ww, 1, 1)) using utf8) collate utf8\_unicode\_ci;

update Cihai set content="",yin="",key1:=y1,key2:=CONCAT\_WS(',',y2,y1),key3="",key4="" where words = ww;

set outww = '2';

when 3 THEN

select key1 into y1 from Cihai where words=convert((select substring(ww, 3, 1)) using utf8) collate utf8\_unicode\_ci;

select key1 into y2 from Cihai where words=convert((select substring(ww, 2, 1)) using utf8) collate utf8\_unicode\_ci;

select key1 into y3 from Cihai where words=convert((select substring(ww, 1, 1)) using utf8) collate utf8\_unicode\_ci;

update Cihai set content="",yin="",key1:=y1,key2:=CONCAT\_WS(',',y2,y1),key3:=CONCAT\_WS(',',y3,y2,y1),key4="" where words = ww;

set outww = '3';

when 4 THEN

select key1 into y1 from Cihai where words=convert((select substring(ww, 4, 1)) using utf8) collate utf8\_unicode\_ci;

select key1 into y2 from Cihai where words=convert((select substring(ww, 3, 1)) using utf8) collate utf8\_unicode\_ci;

select key1 into y3 from Cihai where words=convert((select substring(ww, 2, 1)) using utf8) collate utf8\_unicode\_ci;

select key1 into y4 from Cihai where words=convert((select substring(ww, 1, 1)) using utf8) collate utf8\_unicode\_ci;

update Cihai set content="",yin="",key1:=y1,key2:=CONCAT\_WS(',',y2,y1),key3:=CONCAT\_WS(',',y3,y2,y1),key4:=CONCAT\_WS(',',y4,y3,y2,y1) where words = ww;

set outww = '4';

else

delete from Cihai where words = ww;

set outww = 'error';

end case;

else

delete from Cihai where words = ww;

set outww="not chinese";

end if;

END

# 数据处理源码

## ****operate.py****

#-\*-coding:utf-8-\*-

import json

import re

import mysql.connector

import sys

import time

config = {

'user': 'root',

'password': '\*\*\*\*\*\*',

'host': 'localhost',

'database': 'rhymes',

'charset': 'utf8',

"connection\_timeout": 20,

"use\_pure": True,

"auth\_plugin": 'mysql\_native\_password',

}

def ins(mycs,yunlist,n,id):

# print(mydb)

if n == 5:

sql = "update cihai set yun=(%s),key1=(%s),key2=(%s),key3=(%s),key4=(%s),key5=(%s) where id={}".format(id)

#mycs = mydb.cursor(dictionary=True)

mycs.execute(sql, yunlist)

#mydb.commit() # 数据表内容有更新，必须使用到该语句

print('已更新' + str(mycs.rowcount) + '条数据')

elif n == 4:

sql = "update cihai set yun=(%s),key1=(%s),key2=(%s),key3=(%s),key4=(%s) where id={}".format(id)

#mycs = mydb.cursor(dictionary=True)

mycs.execute(sql, yunlist)

#mydb.commit() # 数据表内容有更新，必须使用到该语句

print('已更新' + str(mycs.rowcount) + '条数据')

elif n == 3:

sql = "update cihai set yun=(%s),key1=(%s),key2=(%s),key3=(%s) where id={}".format(id)

#mycs = mydb.cursor(dictionary=True)

mycs.execute(sql, yunlist)

#mydb.commit() # 数据表内容有更新，必须使用到该语句

print('已更新' + str(mycs.rowcount) + '条数据')

elif n == 2:

sql = "update cihai set yun=(%s),key1=(%s),key2=(%s) where id={}".format(id)

#mycs = mydb.cursor(dictionary=True)

mycs.execute(sql, yunlist)

#mydb.commit() # 数据表内容有更新，必须使用到该语句

print('已更新' + str(mycs.rowcount) + '条数据')

elif n == 1:

sql = "update cihai set yun=(%s),key1=(%s) where id={}".format(id)

#mycs = mydb.cursor(dictionary=True)

mycs.execute(sql, yunlist)

print('已更新' + str(mycs.rowcount) + '条数据')

else:

print("参数不对")

def sqlproccess(n):

try:

mydb = mysql.connector.connect(\*\*config)

except Exception as e:

print("数据库连接失败：" + str(e))

log4 = (time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())) + '数据库连接失败\n' + str(e) + '\n'

with open('inject.log', 'a', encoding='utf-8') as f:

f.write(log4)

sys.exit()

sql1 = "SELECT yin FROM cihai where ID <{}".format(n)

mycs = mydb.cursor(dictionary=True)

mycs.execute(sql1)

#mydb.commit() # 数据表内容有更新，必须使用到该语句

myresult = mycs.fetchall()

#print(type(myresult))

for index, x in enumerate(myresult):

#print(x['yin'])

x['yin'] = re.sub('ā|á|à|ǎ', 'a', x['yin'])

x['yin'] = re.sub('ī|ì|í|ǐ', 'i', x['yin'])

x['yin'] = re.sub('ō|ó|ǒ|ò', 'o', x['yin'])

x['yin'] = re.sub('ū|ú|ǔ|ù', 'u', x['yin'])

x['yin'] = re.sub('ē|é|ě|è', 'e', x['yin'])

x['yin'] = re.sub('ǖ|ǘ|ǚ|ǜ', 'u', x['yin'])

#print(x['yin'])5t4r

sql2 = "UPDATE cihai SET yun='{}' WHERE ID={}".format(x['yin'], index+1)

#mycs.execute(sql2)

key = re.findall('iang|ian|iao|ia|iong|uai|uang|uan|ua|uo|ang|eng|ing|ong|ai|ei|ui|ao|ou|iu|ie|ue|er|an|en|in|un|a|o|e|i|u', x['yin'])

#print(key)

yunl = list()

yunl.append(''.join(x['yin']))

if len(key) >= 5:

key5 = key[-5:]

key4 = key[-4:]

key3 = key[-3:]

key2 = key[-2:]

key1 = key[-1:]

yunl.append(",".join(key1))

yunl.append(",".join(key2))

yunl.append(",".join(key3))

yunl.append(",".join(key4))

yunl.append(",".join(key5))

#yunl.append(''.join(str(index + 1)))

print(yunl)

ins(mycs, yunl, 5, index+1)

elif len(key) == 4:

key4 = key[-4:]

key3 = key[-3:]

key2 = key[-2:]

key1 = key[-1:]

yunl.append(",".join(key1))

yunl.append(",".join(key2))

yunl.append(",".join(key3))

yunl.append(",".join(key4))

print(yunl)

ins(mycs, yunl, 4, index+1)

elif len(key) == 3:

key3 = key[-3:]

key2 = key[-2:]

key1 = key[-1:]

yunl.append(",".join(key1))

yunl.append(",".join(key2))

yunl.append(",".join(key3))

print(yunl)

ins(mycs, yunl, 3, index+1)

elif len(key) == 2:

key2 = key[-2:]

key1 = key[-1:]

yunl.append(",".join(key1))

yunl.append(",".join(key2))

print(yunl)

ins(mycs, yunl, 2, index+1)

elif len(key) == 1:

key1 = key[-1:]

yunl.append(",".join(key1))

print(yunl)

ins(mycs, yunl, 1, index+1)

else:

pass

if not index % 5000:

mydb.commit()

mydb.commit()

time.sleep(3)

mycs.close()

mydb.close()

sqlproccess(380580)

## ****single.py****

#-\*-coding:utf-8-\*-

import json

import re

import mysql.connector

import sys

import time

import xlrd

config = {

'user': 'root',

'password': '\*\*\*\*\*\*',

'host': '127.0.0.1',

'database': 'rhymes',

'charset': 'utf8',

"connection\_timeout": 20,

"use\_pure": True,

"auth\_plugin": 'mysql\_native\_password',

}

# 打开文件

data = xlrd.open\_workbook("汉字.xls")

# 查看工作表

#print("sheets：" + str(data.sheet\_names()))

# 通过文件名获得工作表,获取工作表1

table = data.sheet\_by\_name('Sheet1')

# print("总行数：" + str(table.nrows))

# print("总列数：" + str(table.ncols))

#

# print("整行值：" + str(table.row\_values(0,0)))

# print("整列值：" + str(table.col\_values(0,0)))

mydb = mysql.connector.connect(\*\*config)

mycs = mydb.cursor(dictionary=True)

for rowNum in range(table.nrows):

if rowNum > 0:

rr = table.row\_values(rowNum)[0]

yin = table.row\_values(rowNum)[1]

key1 = re.findall('iang|ian|iao|ia|iong|uai|uang|uan|ua|uo|ang|eng|ing|ong|ai|ei|ui|ao|ou|iu|ie|ue|er|an|en|in|un|a|o|e|i|u', yin)

for c in rr:

insertlist = list()

insertlist.append("".join(c))

insertlist.append("".join(yin))

insertlist.append("".join(key1))

print(insertlist)

sql = "replace into cihai (words,yin,key1) values (%s,%s,%s)"

mycs.execute(sql, insertlist)

if not rowNum % 300:

mydb.commit()

mydb.commit()

time.sleep(8)

mycs.close()

mydb.close()

## ****tract.py****

#-\*-coding:utf-8-\*-

import json

import re

import mysql.connector

import sys

import time

def mysq(orglist,n):

config = {

'user': 'root',

'password': '\*\*\*\*\*\*',

'host': '127.0.0.1',

'database': 'rhymes',

'charset': 'utf8',

"connection\_timeout": 20,

"use\_pure": True

}

try:

mydb = mysql.connector.connect(\*\*config)

except Exception as e:

print("数据库连接失败：" + str(e))

log4 = (time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())) + '数据库连接失败\n' + str(e) + '\n'

with open('inject.log', 'a', encoding='utf-8') as f:

f.write(log4)

sys.exit()

# print(mydb)

if n == 4:

sql = "replace INTO words (word,pron,ch,key1,key2,key3,key4) VALUES (%s,%s,%s,%s,%s,%s,%s)"

mycs = mydb.cursor(dictionary=True)

try:

mycs.executemany(sql, orglist)

mydb.commit() # 数据表内容有更新，必须使用到该语句

print('已更新' + str(mycs.rowcount) + '条数据')

except Exception as e:

mydb.rollback()

print("导入失败：" + str(e))

log5 = (time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())) + '导入失败\n' + str(e) + '\n'

with open('inject.log', 'a', encoding='utf-8') as f:

f.write(log5)

finally:

mycs.close()

mydb.close()

elif n == 3:

sql = "replace INTO words (word,pron,ch,key1,key2,key3) VALUES (%s,%s,%s,%s,%s,%s)"

mycs = mydb.cursor(dictionary=True)

try:

mycs.executemany(sql, orglist)

mydb.commit() # 数据表内容有更新，必须使用到该语句

print('已更新' + str(mycs.rowcount) + '条数据')

except Exception as e:

mydb.rollback()

print("导入失败：" + str(e))

log5 = (time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())) + '导入失败\n' + str(e) + '\n'

with open('inject.log', 'a', encoding='utf-8') as f:

f.write(log5)

finally:

mycs.close()

mydb.close()

elif n == 2:

sql = "replace INTO words (word,pron,ch,key1,key2) VALUES (%s,%s,%s,%s,%s)"

mycs = mydb.cursor(dictionary=True)

try:

mycs.executemany(sql, orglist)

mydb.commit() # 数据表内容有更新，必须使用到该语句

print('已更新' + str(mycs.rowcount) + '条数据')

except Exception as e:

mydb.rollback()

print("导入失败：" + str(e))

log5 = (time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())) + '导入失败\n' + str(e) + '\n'

with open('inject.log', 'a', encoding='utf-8') as f:

f.write(log5)

finally:

mycs.close()

mydb.close()

elif n == 1:

sql = "replace INTO words (word,pron,ch,key1) VALUES (%s,%s,%s,%s)"

mycs = mydb.cursor(dictionary=True)

try:

mycs.executemany(sql, orglist)

mydb.commit() # 数据表内容有更新，必须使用到该语句

print('已更新' + str(mycs.rowcount) + '条数据')

except Exception as e:

mydb.rollback()

print("导入失败：" + str(e))

log5 = (time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())) + '导入失败\n' + str(e) + '\n'

with open('inject.log', 'a', encoding='utf-8') as f:

f.write(log5)

finally:

mycs.close()

mydb.close()

else:

print("参数不对")

file\_content = open('yh.txt', encoding='utf-8', errors='ignore')

clist = file\_content.readlines()

words = {}

for i, val in enumerate(clist):

#print(i, val)

pattern = u'(.\*?) <.\*?<pron>(.\*?)</pron>.\*?<span class="zh">(.\*?)</span>'

res = re.findall(pattern, val)

if res:

#print(res[0][1])

key = re.findall('iː|ɜː|ɑː|ɔː|uː|eɪ|aɪ|ɔɪ|əʊ|aʊ|ɪə|eə|ʊə|ɪ|e|æ|ʌ|a|ʊ|ə|ɒ', res[0][1])

if len(key) >= 4:

key4 = key[-4:]

key3 = key[-3:]

key2 = key[-2:]

key1 = key[-1:]

res[0] = list(res[0])

res[0].append(",".join(key1))

res[0].append(",".join(key2))

res[0].append(",".join(key3))

res[0].append(",".join(key4))

print(res)

mysq(res, 4)

elif len(key) == 3:

key3 = key[-3:]

key2 = key[-2:]

key1 = key[-1:]

res[0] = list(res[0])

res[0].append(",".join(key1))

res[0].append(",".join(key2))

res[0].append(",".join(key3))

print(res)

mysq(res, 3)

elif len(key) == 2:

key2 = key[-2:]

key1 = key[-1:]

res[0] = list(res[0])

res[0].append(",".join(key1))

res[0].append(",".join(key2))

print(res)

mysq(res, 2)

elif len(key) == 1:

key1 = key[-1:]

res[0] = list(res[0])

res[0].append(",".join(key1))

print(res)

mysq(res, 1)

else:

pass

file\_content.close()

# 小程序前端源码

│ .gitignore

│ app.js

│ app.json

│ app.wxss

│ project.config.json

│ sitemap.json

│

├─images

│ code.jpg

│ navi1.jpg

│ navi2.jpg

│ ques.png

│ star.png

│ totop.png

│ zs.png

│

├─pages

│ ├─index

│ │ help.js

│ │ help.json

│ │ help.wxml

│ │ help.wxss

│ │ index.js

│ │ index.json

│ │ index.wxml

│ │ index.wxss

│ │ yun,json

│ │ yun.js

│ │ yun.wxml

│ │ yun.wxss

│ │

│ └─logs

│ logs.js

│ logs.json

│ logs.wxml

│ logs.wxss

│

└─utils

util.js

**.gitignore**

# Windows

[Dd]esktop.ini

Thumbs.db

$RECYCLE.BIN/

# macOS

.DS\_Store

.fseventsd

.Spotlight-V100

.TemporaryItems

.Trashes

# Node.js

node\_modules/

**app.js**

//app.js

App({

onLaunch: function () {

const updateManager = wx.getUpdateManager()

updateManager.onCheckForUpdate(function (res) {

// 请求完新版本信息的回调

console.log(res.hasUpdate)

})

updateManager.onUpdateReady(function () {

wx.showModal({

title: '更新提示',

content: '新版本已经准备好，是否重启应用？',

success: function (res) {

if (res.confirm) {

// 新的版本已经下载好，调用 applyUpdate 应用新版本并重启

updateManager.applyUpdate()

}

}

})

})

updateManager.onUpdateFailed(function () {

// 新版本下载失败

})

try {

const res = wx.getSystemInfoSync()

this.globalData.windowHeight = res.windowHeight;

this.globalData.windowWidth = res.windowWidth;

//console.log(this.globalData)

} catch (e){}

// 展示本地存储能力

var logs = wx.getStorageSync('logs') || []

logs.unshift(Date.now())

wx.setStorageSync('logs', logs)

// 登录

wx.login({

success: res => {

// 发送 res.code 到后台换取 openId, sessionKey, unionId

}

})

// 获取用户信息

wx.getSetting({

success: res => {

if (res.authSetting['scope.userInfo']) {

// 已经授权，可以直接调用 getUserInfo 获取头像昵称，不会弹框

wx.getUserInfo({

success: res => {

// 可以将 res 发送给后台解码出 unionId

this.globalData.userInfo = res.userInfo

// 由于 getUserInfo 是网络请求，可能会在 Page.onLoad 之后才返回

// 所以此处加入 callback 以防止这种情况

if (this.userInfoReadyCallback) {

this.userInfoReadyCallback(res)

}

}

})

}

}

})

},

onShow(opt){

//console.log(opt);

},

globalData: {

userInfo: null

}

})

**app.json**

{

"pages":[

"pages/index/index",

"pages/index/yun",

"pages/index/help",

"pages/logs/logs"

],

"permission": {

"scope.userLocation": {

"desc": "你的位置信息将用于小程序位置接口的效果展示"

}

},

"window":{

"backgroundTextStyle":"dark",

"navigationBarBackgroundColor": "#ddd",

"navigationBarTitleText": "ZE's 押韵助手",

"navigationBarTextStyle":"black",

"enablePullDownRefresh": true

},

"tabBar": {

"selectedColor": "#2288ff",

"borderStyle": "black",

"position":"bottom",

"list": [{

"pagePath": "pages/index/index",

"text": "首页",

"iconPath": "images/navi1.jpg",

"selectedIconPath": "images/navi2.jpg"

}, {

"pagePath": "pages/logs/logs",

"text": "热词",

"iconPath": "images/navi1.jpg",

"selectedIconPath": "images/navi2.jpg"

}]

},

"networkTimeout": {

"request": 10000,

"connectSocket": 20000,

"uploadFile": 20000,

"downloadFile": 20000

},

"debug": true,

"style": "v2",

"sitemapLocation": "sitemap.json"

}

**app.wxss**

/\*\*app.wxss\*\*/

.container {

height: 100%;

display: flex;

flex-direction: column;

align-items: center;

justify-content: space-between;

padding: 200rpx 0;

box-sizing: border-box;

background-color:black;

}

**project.config.json**

{

"description": "项目配置文件",

"packOptions": {

"ignore": []

},

"setting": {

"urlCheck": false,

"es6": true,

"enhance": false,

"postcss": true,

"preloadBackgroundData": false,

"minified": true,

"newFeature": false,

"coverView": true,

"nodeModules": false,

"autoAudits": false,

"showShadowRootInWxmlPanel": true,

"scopeDataCheck": false,

"uglifyFileName": false,

"checkInvalidKey": true,

"checkSiteMap": true,

"uploadWithSourceMap": true,

"compileHotReLoad": false,

"babelSetting": {

"ignore": [],

"disablePlugins": [],

"outputPath": ""

},

"useIsolateContext": true,

"useCompilerModule": false,

"userConfirmedUseCompilerModuleSwitch": false

},

"compileType": "miniprogram",

"libVersion": "2.12.1",

"appid": "wx2c64a9defd9de51e",

"projectname": "miniprogram-1",

"debugOptions": {

"hidedInDevtools": []

},

"scripts": {},

"isGameTourist": false,

"simulatorType": "wechat",

"simulatorPluginLibVersion": {},

"condition": {

"search": {

"current": -1,

"list": []

},

"conversation": {

"current": -1,

"list": []

},

"plugin": {

"current": -1,

"list": []

},

"game": {

"current": -1,

"list": []

},

"gamePlugin": {

"current": -1,

"list": []

},

"miniprogram": {

"current": -1,

"list": [

{

"id": -1,

"name": "pages/index/index",

"pathName": "pages/index/index",

"scene": null

}

]

}

}

}

**sitemap.json**

{

"desc": "关于本文件的更多信息，请参考文档 https://developers.weixin.qq.com/miniprogram/dev/framework/sitemap.html",

"rules": [{

"action": "allow",

"page": "\*"

}]

}

**pages\index\help.js**

//help.js

const util = require('../../utils/util.js')

Page({

data: {

},

onLoad: function () {

},

onShareAppMessage:function(){

return {

title: 'ZE\'s 押韵助手',

path: '/page/index/index',

imageUrl: '/images/navi1.jpg'

}

},

})

**pages\index\help.json**

{

"navigationBarTitleText": "帮助页",

"usingComponents": {}

}

**pages\index\help.wxml**

<view class='container'>

<text class="text">

1.制作小程序的初衷是支援武汉本土rapper的发展，给你们打call。

2.当然也欢迎所有热爱音乐的创作人使用。

3.已增加自动收录新词功能，查询显示暂未收录以后过段时间(建议15分钟以上)再查询一次吧。

4.已更新热词Top10功能。

5.发现bug可以发邮件到418011010@qq.com，感谢支持。

6.2021.5.3更新人生佳句分享与赞赏。

</text>

</view>

**pages\index\help.wxss**

.text {

height: 500rpx;

width: 90%;

color: rgb(255, 255, 255);

padding: 10px;

}

.container{

background-color: rgba(19, 196, 196, 0.26);

width: 90%;

margin: 25px;

line-height:25px;

}

page{

background-image: url("https://www.juso.top/static/images/valley.jpg");

background-size: 100% 100%;

background-repeat: no-repeat;

}

**pages\index\index.js**

//index.js

//获取应用实例

const app = getApp()

Page({

data: {

motto: '正在加载...',

arrs: ['location'],

userInfo: {},

hasUserInfo: false,

canIUse: wx.canIUse('button.open-type.getUserInfo')

},

//事件处理函数

bindViewTap: function() {

wx.navigateTo({

url: '../logs/logs'

})

},

goyun:function(param){

wx.navigateTo({

url: '/pages/index/yun',

})

},

postnum(){

var that = this

wx.request({

url: 'https://www.juso.top/postest', // 已废弃接口地址

data: {

a: '2020',

b: 'RapChina'

},

method:'post',

header: {

'content-type': 'application/x-www-form-urlencoded' // 默认值

},

success(res) {

console.log(res.data);

that.setData({

motto:res.data.number

})

}

})

},

get\_ip(){

var that = this

wx.request({

url: 'https://www.juso.top/get\_ip', // 接口地址

data: {

},

method:'get',

header: {

'content-type': 'application/x-www-form-urlencoded' // 默认值

},

success(res) {

console.log(res.data);

that.setData({

arrs:res.data

})

}

})

},

onLoad: function () {

//this.postnum();

this.get\_ip();

if (app.globalData.userInfo) {

this.setData({

userInfo: app.globalData.userInfo,

hasUserInfo: true

})

} else if (this.data.canIUse){

// 由于 getUserInfo 是网络请求，可能会在 Page.onLoad 之后才返回

// 所以此处加入 callback 以防止这种情况

app.userInfoReadyCallback = res => {

this.setData({

userInfo: res.userInfo,

hasUserInfo: true

})

}

} else {

// 在没有 open-type=getUserInfo 版本的兼容处理

wx.getUserInfo({

success: res => {

app.globalData.userInfo = res.userInfo

this.setData({

userInfo: res.userInfo,

hasUserInfo: true

})

}

})

}

},

onPullDownRefresh(){

console.log("用户下拉");

},

onShareAppMessage:function(){

return {

title: 'ZE\'s 押韵助手',

path: '/page/index/index',

imageUrl: '/images/navi1.jpg'

}

},

getUserInfo: function(e) {

console.log(e)

app.globalData.userInfo = e.detail.userInfo

this.setData({

userInfo: e.detail.userInfo,

hasUserInfo: true

})

}

})

**pages\index\index.json**

{

"usingComponents": {}

}

**pages\index\index.wxml**

<!--index.wxml-->

<view class="container">

<view class='toptitle'>押韵助手</view>

<view class='topword'>会自动收录新词的神器</view>

<view class="userinfo">

<button wx:if="{{!hasUserInfo && canIUse}}" open-type="getUserInfo" bindgetuserinfo="getUserInfo"> 获取头像昵称 </button>

<block wx:else>

<image bindtap="bindViewTap" class="userinfo-avatar" src="{{userInfo.avatarUrl}}" mode="cover"></image>

<text class="userinfo-nickname">{{userInfo.nickName}}</text>

</block>

</view>

<view class="item" wx:for='{{arrs}}'>

{{item}}

</view>

<button class='btn1' bindtap="goyun">

<image class='btnImg' src='../../images/star.png'></image>

<view>立即进入</view>

</button>

</view>

**pages\index\index.wxss**

/\*\*index.wxss\*\*/

.container{

background-color:rgb(20, 20, 20);

}

.toptitle {

position: absolute;

top: 30px;

color: rgba(255, 0, 0, 0.849);

font-size: 40rpx;

}

.topword {

color: white;

}

.userinfo {

display: flex;

flex-direction: column;

align-items: center;

}

.userinfo-avatar {

width: 128rpx;

height: 128rpx;

margin: 20rpx;

border-radius: 50%;

}

.userinfo-nickname {

color: #aaa;

}

.usermotto {

margin-top: 50rpx;

color: white;

}

.item {

margin-top: 20rpx;

color: rgb(204, 36, 36);

}

.btn1{

width: 200rpx;

height: 200rpx;

margin-top: 20rpx;

background-color:rgb(20, 20, 20);

color: white;

border-radius: 0rpx;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

font-size: 30rpx;

}

.btnImg {

width: 50rpx;

height: 50rpx;

}

.btn1::after {

border: 0;

}

page {height: 100%;}

**pages\index\yun,json**

{

"usingComponents": {},

"navigationBarBackgroundColor": "purple",

"navigationStyle": "custom",

}

**pages\index\yun.js**

// pages/index/talkPage.js

var app = getApp();

Page({

/\*\*

\* 页面的初始数据

\*/

data: {

lot:"",

lat:"",

postres:[{words: "loading..."},],

poststory:[],

kwreport:"",

boolres:true,

boolstory:false,

modalHidden: true,

},

staticData:{

type: "Chinese",

mult: "s",

},

/\*\*

\* 事件处理函数

\*/

getpos(){

wx.getLocation({

type: 'gjc02',

success: (res) => {

var latitude = res.latitude // 纬度

var longitude = res.longitude // 经度

//console.log(res);

this.setData({

lat: latitude,

lot: longitude

})

}

})

},

help(e){

wx.navigateTo({

url: '/pages/index/help',

})

},

handleChooseLanguage(e) {

this.staticData.type = e.detail.value;

//console.log(this.staticData);

},

handleChooseMult(e){

this.staticData.mult = e.detail.value;

//console.log(this.staticData);

},

handleKeyword(e) {

this.staticData.keyword = e.detail.value;

//console.log(this.staticData)

},

buttonTap: function() {

this.setData({

modalHidden: false

})

},

/\*\*

\* 点击取消

\*/

modalCandel: function() {

// do something

this.setData({

modalHidden: true

})

},

/\*\*

\* 点击确认

\*/

modalConfirm: function() {

// do something

wx.previewImage({

current: 'https://www.juso.top/static/images/code.jpg', // 当前显示图片的http链接

urls: ['https://www.juso.top/static/images/code.jpg']

})

this.setData({

modalHidden: true

})

},

handleSubmit(){

if (!this.staticData.keyword){

wx.showToast({

title: '请填写搜索词',

icon: 'loading',

duration: 1000

})

return;

}

this.setData({

boolres:false,

boolstory:true,

})

wx.request({

url: 'https://www.juso.top/wxpost/',

//url: 'http://127.0.0.1/wxpost/',

data: {

language: this.staticData.type,

mult: this.staticData.mult,

keyword: this.staticData.keyword,

},

method:'POST',

header: {

'content-type': 'application/x-www-form-urlencoded' // 默认值

},

success:(res) =>{

this.setData({

postres:res.data.data,

})

//this.data.postres = res.data.data;

this.data.kwreport=this.staticData.keyword

//console.log(this.data.kwreport);

wx.pageScrollTo({

scrollTop: 220,

duration: 300

})

}

})

},

totop() {

wx.pageScrollTo({

scrollTop: 0,

duration: 300

})

},

/\*\*

\* 生命周期函数--监听页面加载

\*/

onLoad: function () {

wx.request({

url: 'https://www.juso.top/testdb/',

method:'GET',

success:(res) =>{

this.setData({

poststory:res.data,

})

}

})

},

/\*\*

\* 生命周期函数--监听页面初次渲染完成

\*/

onReady: function () {

},

/\*\*

\* 生命周期函数--监听页面显示

\*/

onShow: function () {

this.getpos();

},

/\*\*

\* 生命周期函数--监听页面隐藏

\*/

onHide: function () {

},

/\*\*

\* 生命周期函数--监听页面卸载

\*/

onUnload: function () {

},

/\*\*

\* 页面相关事件处理函数--监听用户下拉动作

\*/

onPullDownRefresh: function () {

this.setData({

boolres:true,

boolstory:false,

})

wx.request({

url: 'https://www.juso.top/testdb/',

method:'GET',

success:(res) =>{

this.setData({

poststory:res.data,

})

//console.log(this.data.poststory);

wx.stopPullDownRefresh()

}

})

},

/\*\*

\* 页面上拉触底事件的处理函数

\*/

onReachBottom: function () {

},

/\*\*

\* 用户点击右上角分享

\*/

onShareAppMessage:function(){

return {

title: 'ZE\'s 押韵助手',

path: '/page/index/index',

imageUrl: '/images/navi1.jpg'

}

},

})

**pages\index\yun.wxml**

<!--pages/getinto.wxml-->

<view class='container'>

<button class='qbtn' bindtap="help">

<image class='btnImg' src='../../images/ques.png'></image>

<view>帮助</view>

</button>

<view class="row">

<label class="title">语言</label>

<view class="info">

<radio-group bindchange="handleChooseLanguage">

<label>

<radio value="Chinese" checked="True"/>中文

<radio value="English" />英文

</label>

</radio-group>

</view>

<view class="info">

<radio-group bindchange="handleChooseMult">

<label>

<radio value="s" checked="True"/>单押

<radio value="d" />双押

<radio value="t" />三押

<radio value="q" />四押

</label>

</radio-group>

</view>

<label class="title">韵脚</label>

<view class="info">

<input bindinput="handleKeyword" placeholder="填入要搜索的关键词" auto-focus />

</view>

<label class="title">结果</label>

<view class="info" wx:for="{{postres}}" wx:key="words" hidden="{{boolres==true}}">

<text>{{item.words}}：</text>

<text>解释：{{item.content}}, </text>

<text>读音：{{item.yin}}, </text>

<text>韵母：{{item.key}} </text>

</view>

<view class="info" hidden="{{boolstory==true}}">

<text>{{poststory}}</text>

</view>

</view>

<image class='btnZ' bindtap="modalConfirm" src='../../images/zs.png'></image>

<modal title="求赞" hidden="{{modalHidden}}" bindconfirm="modalConfirm" bindcancel="modalCandel">

<view>

<image class="image" src="../../images/code.jpg" mode='aspectFill'></image>

</view>

<view>如果觉得好用，欢迎投币支持哦</view>

</modal>

<image class='btnI' bindtap="totop" src='../../images/totop.png'></image>

<view class="button" bindtap="handleSubmit">查询</view>

</view>

**pages\index\yun.wxss**

.container{

background-color:rgb(43, 50, 51);

}

.qbtn {

position: absolute;

top: 0rpx;

width: 50%;

margin-top: 10rpx;

background-color: rgba(87, 86, 83, 0.678);

color: white;

border-radius: 98rpx;

display: flex;

flex-direction: row;

align-items: center;

justify-content: center;

z-index:1;

}

.btnImg {

margin-right: 8rpx;

width: 46rpx;

height: 46rpx;

}

.btnZ {

position: fixed;

margin-top: 10rpx;

top: 0;

right: 12px;

width: 8%;

height: 30rpx;

}

.btnI {

position: fixed;

bottom: 75px;

right: 12px;

width: 5%;

height: 24rpx;

}

.qbtn::after {

border-radius: 98rpx;

border: 0;

}

.button{

position: fixed;

bottom: 0px;

width: 100%;

margin: 0px 0px 0px 0px;

background: #ff7300c9;

line-height: 50px;

border-radius: 8px;

text-align: center;

color: #fff;

font-size: 35rpx;

}

.row{

overflow: hidden;

position: absolute;

top: 5px;

left: 0px;

border-radius: 5px;

background-color: #f2f2f2;

padding: 20px;

margin-bottom: 50px;

}

.title {

margin-top: 10px;

padding-left: 10px;

width: 60px;

float: left;

}

.info {

width: 100%;

padding: 12px 16px;

margin: 10px 0 15px 0;

display: inline-block;

border: 1px solid #ccc;

border-radius: 4px;

box-sizing: border-box;

}

page{

height: 100%;

}

**pages\logs\logs.js**

//logs.js

const util = require('../../utils/util.js')

Page({

data: {

logs: [],

imageUrl: "https://www.juso.top/static/images/rankword.png",

},

onLoad: function () {

this.setData({

logs: (wx.getStorageSync('logs') || []).map(log => {

return util.formatTime(new Date(log))

})

})

},

onShareAppMessage:function(){

return {

title: 'ZE\'s 押韵助手',

path: '/page/index/index',

imageUrl: '/images/navi1.jpg'

}

},

reloadimage:function(){

var that=this;

that.setData({

imageUrl: that.data.imageUrl+' ',

})

},

onPullDownRefresh: function () {

wx.clearStorage()

var that=this;

that.setData({

imageUrl: that.data.imageUrl+' ',

})

//console.log(this.data.imageUrl);

wx.stopPullDownRefresh()

},

})

**pages\logs\logs.json**

{

"navigationBarTitleText": "查看热词Top10",

"usingComponents": {}

}

**pages\logs\logs.wxml**

<!--logs.wxml-->

<view class="container log-list">

<image class='hotpng' src="{{imageUrl}}" binderror='reloadimage' mode="aspectFill">

</image>

</view>

**pages\logs\logs.wxss**

.container{

background-color:#ffffff;

}

.log-list {

display: flex;

flex-direction: column;

padding: 40rpx;

color: azure;

}

.log-item {

margin: 10rpx;

}

.hotpng {

margin: 0px 0px 0px 0px;

width:750rpx;

height:1400rpx;

}

**utils\util.js**

const formatTime = date => {

const year = date.getFullYear()

const month = date.getMonth() + 1

const day = date.getDate()

const hour = date.getHours()

const minute = date.getMinutes()

const second = date.getSeconds()

return [year, month, day].map(formatNumber).join('/') + ' ' + [hour, minute, second].map(formatNumber).join(':')

}

const formatNumber = n => {

n = n.toString()

return n[1] ? n : '0' + n

}

module.exports = {

formatTime: formatTime

}