Django项目实践之内容管理minicms

# 建立minicms项目

* django-admin startproject minicms
* cd minicms
* python manage.py startapp news
* 添加 news 到 settings.py 中的 INSTALLED\_APPS 中。

# Model

* 栏目：名称，网址，简介等
* 文章：标题，作者，网址，内容等
* 一篇文章只有一个作者（文章和作者是多对一的关系）
* 一篇文章可以属于多个栏目（栏目和文章是多对多的关系）
* 作者可以为空，栏目不能为空。

#models.py

from django.db import models

class Column(models.Model):

name = models.CharField('栏目名称', max\_length=256)

slug = models.CharField('栏目网址', max\_length=256, db\_index=True)

intro = models.TextField('栏目简介', default='')

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

return self.name

class Meta:

verbose\_name = '栏目'

verbose\_name\_plural = '栏目'

ordering = ['name'] # 按照哪个栏目排序

class Article(models.Model):

column = models.ManyToManyField(Column, verbose\_name='归属栏目')

title = models.CharField('标题', max\_length=256)

slug = models.CharField('网址', max\_length=256, db\_index=True)

author = models.ForeignKey('auth.User', blank=True, null=True, verbose\_name='作者', on\_delete=models.CASCADE,)

content = models.TextField('内容', default='', blank=True)

published = models.BooleanField('正式发布', default=True)

pub\_date = models.DateTimeField('发表时间', auto\_now\_add=True, editable=True)

update\_time = models.DateTimeField('更新时间', auto\_now=True, null=True)

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

return self.title

class Meta:

verbose\_name = '教程'

verbose\_name\_plural = '教程'

python manage.py makemigrations

python manage.py migrate

创建一个脚本，导入一些示例数据到数据库中

#create\_demo\_records.py

# 这个文件放在项目minicms目录下

# 执行方式： python create\_demo\_records.py

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

# @Date :

# @Author :

# @Link :

'''

create some records for demo database

'''

from minicms.wsgi import \*

from news.models import Column, Article

def main():

columns\_urls = [

('体育新闻', 'sports'),

('社会新闻', 'society'),

('科技新闻', 'tech'),

]

for column\_name, url in columns\_urls:

c = Column.objects.get\_or\_create(name=column\_name, slug=url)[0]

# 创建 10 篇新闻

for i in range(1, 11):

article = Article.objects.get\_or\_create(

title='{}\_{}'.format(column\_name, i),

slug='article\_{}'.format(i),

content='新闻详细内容： {} {}'.format(column\_name, i)

)[0]

article.column.add(c)

if \_\_name\_\_ == '\_\_main\_\_':

main()

print("Done!")

# 完善后台的功能

在后台添加，编辑，删除数据

编辑文件：news/admin.py

from django.contrib import admin

from .models import Column, Article

class ColumnAdmin(admin.ModelAdmin):

list\_display = ('name', 'slug', 'intro',)

class ArticleAdmin(admin.ModelAdmin):

list\_display = ('title', 'slug', 'author', 'pub\_date', 'update\_time')

admin.site.register(Column, ColumnAdmin)

admin.site.register(Article, ArticleAdmin)

创建一个超级管理员：python manage.py createsuperuser

# 集成 DjangoUeditor 编辑器

（1）安装 DjangoUeditor包: pip3 install DjangoUeditor

（2）这个包DjangoUeditor是python2版本下的，在python3下运行有问题。需要做修改。已经有人完成此工作，在<https://github.com/liyaopinner/DjangoUeditor3_imooc>下载后，解压缩，把解压缩的文件替换到site-package目录对应的djangouedit目录的文件即可。

（3）修改minicms/settings.py 中加入 DjangoUeditor 这个应用

INSTALLED\_APPS = (

...

'news',

'DjangoUeditor',

)

（4）minicms/urls.py 中添加：

from django.conf.urls import include, url

...

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

（5）为了让上传的图片，文件可以在本地调试的时候可以正常显示，下载。

在 minicms/settings.py 设置 static 和 media

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/1.8/howto/static-files/

STATIC\_URL = '/static/'

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

# 公共的 static 文件，比如 jquery.js 可以放这里，这里面的文件夹不能包含 STATIC\_ROOT

STATICFILES\_DIRS = (

os.path.join(BASE\_DIR, "common\_static"),

)

# upload folder

MEDIA\_URL = '/media/'

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

（6）在 minicms/urls.py 最后加入以下代码：

# use Django server /media/ files

from django.conf import settings

#原有的urlpatterns，下面的代码一定要放在原有的urlpatterns后面

if settings.DEBUG:

from django.conf.urls.static import static

urlpatterns += static(

settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

（7）修改 models.py

# 引入这个 Field，其它的不要变

from DjangoUeditor.models import UEditorField

class Article(models.Model):

...省略

#仅修改 content 字段

content = UEditorField(verbose\_name='内容',width=800, height=300, toolbars="full", imagePath="uploads/images/", filePath="uploads/files/", upload\_settings={"imageMaxSize":12040000},default='')

...

（8）重新启动django server，测试富文本编辑器，可能出现以下错误提示：

* DjangoUeditor ImportError: No module named ‘widgets’的解决办法

由于ueditor为百度开发的一款富文本编辑框，现已停止维护。python2，则直接pip install djangoueditor，python3以上的版本需要下载压缩包解压安装。前面的（1）和（2）就是针对这个错误的处理方法。

* 报错93行：boundfield.py in as\_widget, line 93

这个文件在\site-packages\django\forms\ boundfield.py，网上提供的解决方案是注释这一行即可。

#renderer=self.form.renderer,

# 前台内容显示

首页显示一些栏目，每个栏目显示五篇相关的文章，栏目可以点击进去，文章也可以点击进去。

## 网址与内容关联

/column/column\_slug/ 访问栏目的内容

/news/article\_slug/ 查看文章详情

修改 minicms/urls.py 添加上面的规则：

|  |
| --- |
|  |

url(r'^$', 'news.views.index', name='index'),

url(r'^column/(?P<column\_slug>[^/]+)/$', 'news.views.column\_detail', name='column'),

url(r'^news/(?P<article\_slug>[^/]+)/$', 'news.views.article\_detail', name='article'),

url(r'^admin/', include(admin.site.urls)),

以下不用通用视图，因为通用视图都帮大家做好了，学不到什么东西

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

from django.shortcuts import render

from django.http import HttpResponse

def index(request):

return HttpResponse(u'欢迎学习Django')

def column\_detail(request, column\_slug):

return HttpResponse('column slug: ' + column\_slug)

def article\_detail(request, article\_slug):

return HttpResponse('article slug: ' + article\_slug)

访问比如：<http://127.0.0.1:8000/column/tech/>

## 用代码来生成相关的网址

运行 python manage.py shell 进入有项目环境的终端，相当于下面两句：

python

from minicms.wsgi import \*

运行

>>> from django.core.urlresolvers import reverse

这个有错，应该是 from django.urls import reverse

>>>

>>> # column

>>> reverse('column', args=('tech',))

u'/column/tech/'

>>> reverse('column', args=('sports',))

u'/column/sports/'

>>>

>>> # article

>>> reverse('article', args=('article\_slug',))

u'/news/article\_slug/'

>>> reverse('article', args=('windows\_10',))

u'/news/windows\_10/'

>>>

修改 models.py ，将获取网址的功能写成一个函数 get\_absolute\_url ，然后在模板或其它脚本中调用。后台默认也会调用这个函数，可以理解成一个约定俗成的名称。

from django. urls import reverse

...

class Column(models.Model):

...

slug = models.CharField('栏目网址', max\_length=256, db\_index=True)

...

def get\_absolute\_url(self):

return reverse('column', args=(self.slug,))

...

class Article(models.Model):

...

slug = models.CharField('网址', max\_length=256, db\_index=True)

...

def get\_absolute\_url(self):

return reverse('article', args=(self.slug,))

...

注意 args 参数为元组，写 args=(self.slug) 这样是错的，注意后面有一个逗号 args=(self.slug,)

重新运行终端命令： python manage.py shell

>>> from news.models import Column, Article

>>> c = Column.objects.all()[0]

>>> c.get\_absolute\_url()

u'/column/sports/'

>>> a = Article.objects.all()[0]

>>> a.get\_absolute\_url()

u'/news/article\_1/'

这样已经可以获取到文章或者栏目的网址了，下面我们在模板中显示它们。

## 相关模板文件

先写一个 base.html 所有的其它模板都继承它，由于 base.html 不属于某一个 app，它是整个项目共用的，我们建立一个专门的 模块文件夹来放它样的文件。

baidutongji.html ，把百度统计的代码放进去来统计访问情况。

修改 settings.py，添加一个模板目录 项目下 templates 文件夹。

'DIRS': [os.path.join(BASE\_DIR, 'templates')], #修改了这一行

项目目录下 建一个 templates 文件夹，里面写一个 base.html

templates/base.html 模板文件

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>{% block title %}欢迎光临{% endblock title %} - 自强学堂</title>

{% block css %}

{% endblock css %}

{% block js %}

{% endblock js %}

</head>

<body>

{% block content %}

<h1>自强 新闻网</h1>

{% endblock content %}

{% include "baidutongji.html" %}

</body>

</html>

喜欢 BootStrap 的同学可以用这个 base.html

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge,Chrome=1">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>{% block title %}欢迎光临{% endblock title %} - 自强学堂</title>

<!-- Bootstrap -->

<link href="http://apps.bdimg.com/libs/bootstrap/3.3.0/css/bootstrap.min.css" rel="stylesheet">

<!-- 引入下面两个库让 IE8 支持 HTML5 元素 -->

<!-- 警告: Respond.js 通过 file:// 浏览的时候不能正常工作！-->

<!--[if lt IE 9]>

<script src="http://apps.bdimg.com/libs/html5shiv/3.7/html5shiv.min.js"></script>

<script src="http://apps.bdimg.com/libs/respond.js/1.4.2/respond.min.js"></script>

<![endif]-->

{% block css %}

{% endblock css %}

{% block js %}

{% endblock js %}

</head>

<body>

<div>

<div>

{% block content %}

<h1>自强 新闻网</h1>

{% endblock content %}

</div>

</div>

<script src="http://apps.bdimg.com/libs/jquery/1.11.1/jquery.min.js"></script>

<script src="http://apps.bdimg.com/libs/bootstrap/3.3.0/js/bootstrap.min.js"></script>

{% include "baidutongji.html" %}

</body>

</html>

templates/baidutongji.html

<!-- 这里放统计的代码 -->

<!-- 百度统计 begin-->

<div style="display:none" hidden>

<script type="text/javascript">

var \_bdhmProtocol = (("https:" == document.location.protocol) ? " https://" : " http://");

document.write(unescape("%3Cscript src='" + \_bdhmProtocol + "hm.baidu.com/h.js%3Fab8b04b0bf640ac196520db1cd029664' type='text/javascript'%3E%3C/script%3E"));

</script>

</div>

<!-- 百度统计 end-->

## 视图中检索结果，渲染到模板中显示出来

改写视图函数，查询数据库，得到相关的内容：

def column\_detail(request, column\_slug):

column = Column.objects.get(slug=column\_slug)

return render(request, 'news/column.html', {'column': column})

def article\_detail(request, article\_slug):

article = Article.objects.get(slug=article\_slug)

return render(request, 'news/article.html', {'article': article})

在 news 下新建 templates 文件夹，再在 templates 下新建 news 文件夹，我们来写两个模板文件：

栏目模板，视图传递过来了一个 column object，news/templates/news/column.html （或者 项目下的 templates/news/column.html 路径）Django 都能找到它们

{% extends "base.html" %}

{% block title %}

{{ column.title }}

{% endblock title %}

{% block content %}

栏目简介：{{ column.intro }}

栏目文章列表：

还需要完善

{% endblock content %}

文章模板，视图传递过来了一个 article object。news/templates/news/article.html （或者 项目下的 templates/news/article.html）

{% extends "base.html" %}

{% block title %}

{{ article.title }}

{% endblock title %}

{% block content %}

<h1>文章标题： {{ article.title }}</h1>

<div id="main">

{{ article.content }}

</div>

{% endblock content %}

news/views.py

def index(request):

columns = Column.objects.all()

return render(request, 'index.html', {'columns': columns})

项目下 templates/index.html

{% extends "base.html" %}

{% block title %} 首页 {% endblock title %}

{% block content %}

<ul>

{% for column in columns %}

<li>

<a href="{{ column.get\_absolute\_url }}"> {{ column.name }} </a>

</li>

{% endfor %}

</ul>

{% endblock content %}

## 完善首页的显示功能