

Python + Django搭建个人博客 Dya03

在后台添加数据，方便测试

修改导航显示

1. 在views.py增加方法用来查询所有的导航

```
from django.shortcuts import render

from .models import Category #将数据模型导入
# Create your views here.

def getCategory():
    category_list = Category.objects.all() # 获取所有的Category数据
    return category_list

def index(request):
    '''显示首页'''
    template_name = "index.html" # 要显示的模板名称

    category_list = getCategory() # 获取Category数据

    return render(request, template_name, {'category_list':category_list}) #将我们定义
好的页面返回到页面
```

- 导入数据模型

```
from .models import Category #将数据模型导入
```

- 获取模型中所有的数据

```
category_list = Category.objects.all() # 获取所有的Category数据
```

- 将数据全部返回到页面

```
return render(request, template_name, {'category_list':category_list})
```

这里后面的字典就是将获取的数据，返回到前台页面。

2. 修改base.html页面显示

这里用到了Django的模板语言

```
{% for category in category_list %}
{% endfor %}
```

这里的category_list就是后台方法传过来的参数。通过for循环来获取里面所有的数据
最后用{% endfor %}结尾。

修改后的HTML页面为：

```
{% for category in category_list %}
<li><a href="about.html">{{category.name}}</a></li>
{% endfor %}
```

{{category.name}}也是Django的模板语言，用来获取对象中的属性。这里是获取category对象中的name值。
页面显示效果为：



这里暂时告一段落，后面还会再修改的。

获取文章列表

1. 修改views.py来获取文章列表

```
def getArticles():
    '''获取所有的文章'''
    article_list = Article.objects.all()
    return article_list
```

添加如上方法

并在index方法中调用

```
category_list = getCategory() # 获取Category数据
# 因为首页不需要将所有的文章都显示出来，这里只显示前6条
article_list = getArticles()[:6] # 获取Article数据
```

2. 修改页面，和导航的写法基本类似

```

70      <!--文章列表开始-->
71      <div class="contentList">
72
73          <div class="panel panel-default">...
85      </div>
86
87      <div class="panel panel-default">...
106  </div>
107
108      <div class="panel panel-default">...
119  </div>
120
121      <div class="panel panel-default">...
140  </div>
141
142      <div class="panel panel-default">...
162  </div>
163
164      <div class="panel panel-default">...
183  </div>
184
185  </div>
186  <!--文章列表结束-->

```

通过分析页面可以看出，所有的的文章都是一个独立的DIV，所以我们这里只要保留一个，其他的通过for循环遍历出来就可以了。修改后的效果：

wfyvv.com

UUID

java

个人网站正在建设中。。。

admin 阅读:666 评论:18 2017-02-07

wfyvv.com

UUID

java

个人网站正在建设中。。。

admin 阅读:666 评论:18 2017-02-07

wfyvv.com

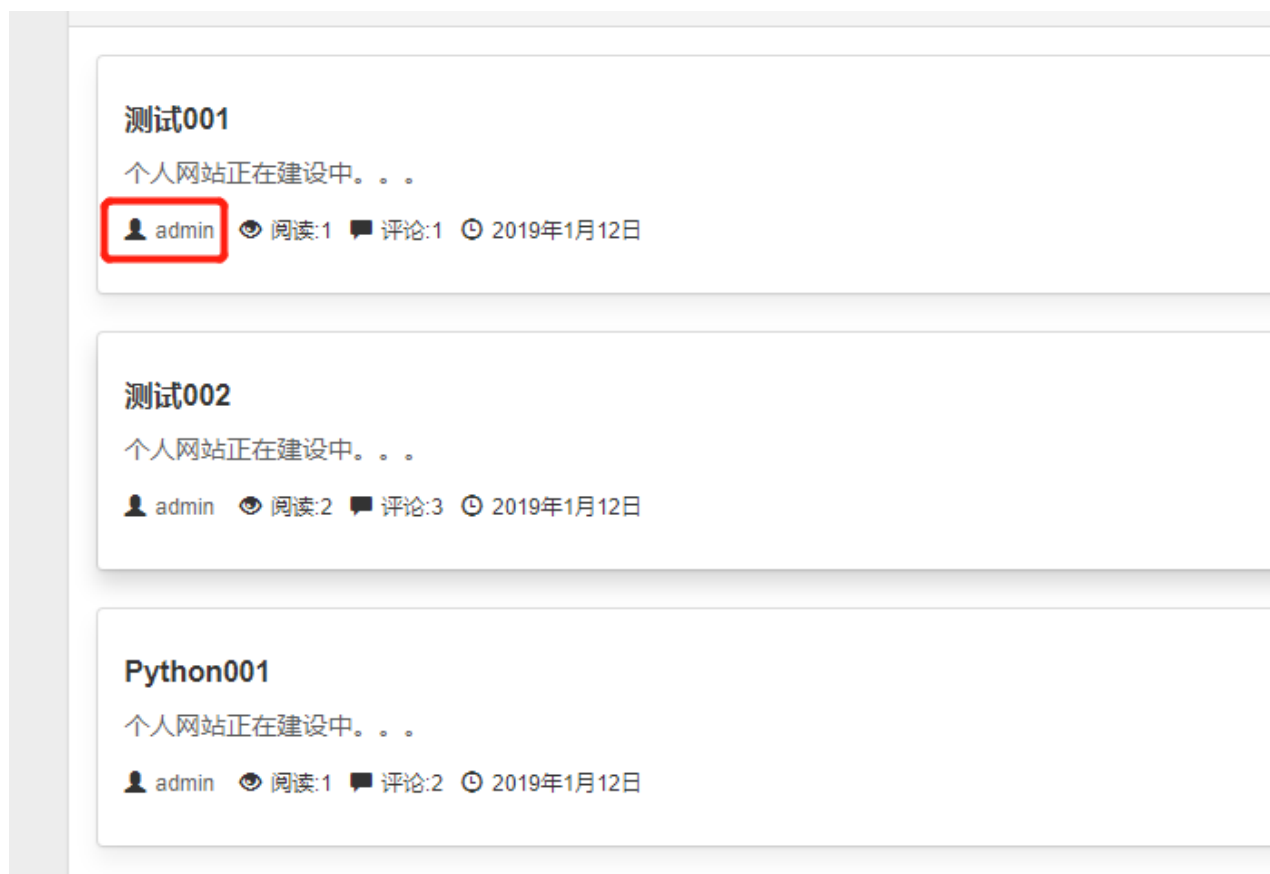
UUID

java

个人网站正在建设中。。。

admin 阅读:666 评论:18 2017-02-07

接下来就是将数据换成我们自己的数据，通过{{}} 这个标签就可以。



这里有两个地方还没有修改，一个是用户名，还有一个是标签名。

接下来我们来修改一下：

- 获取用户信息

再views.py中添加

```
from django.contrib.auth import *  
...  
user_name = request.user.username
```




修改前台代码：

```
"#">{{user_name}}</a></spa
```

看到效果：

测试001

个人网站正在建设中。。。

 Tango  阅读:1  评论:1  2019年1月12日

测试002

个人网站正在建设中。。。

 Tango  阅读:2  评论:3  2019年1月12日

Python001

个人网站正在建设中。。。

 Tango  阅读:1  评论:2  2019年1月12日

■ 显示标签名

```
{% for tagInfo in article.tags.all %}
<a class="label label-default">{{tagInfo.name}}</a>
{% endfor %}
```

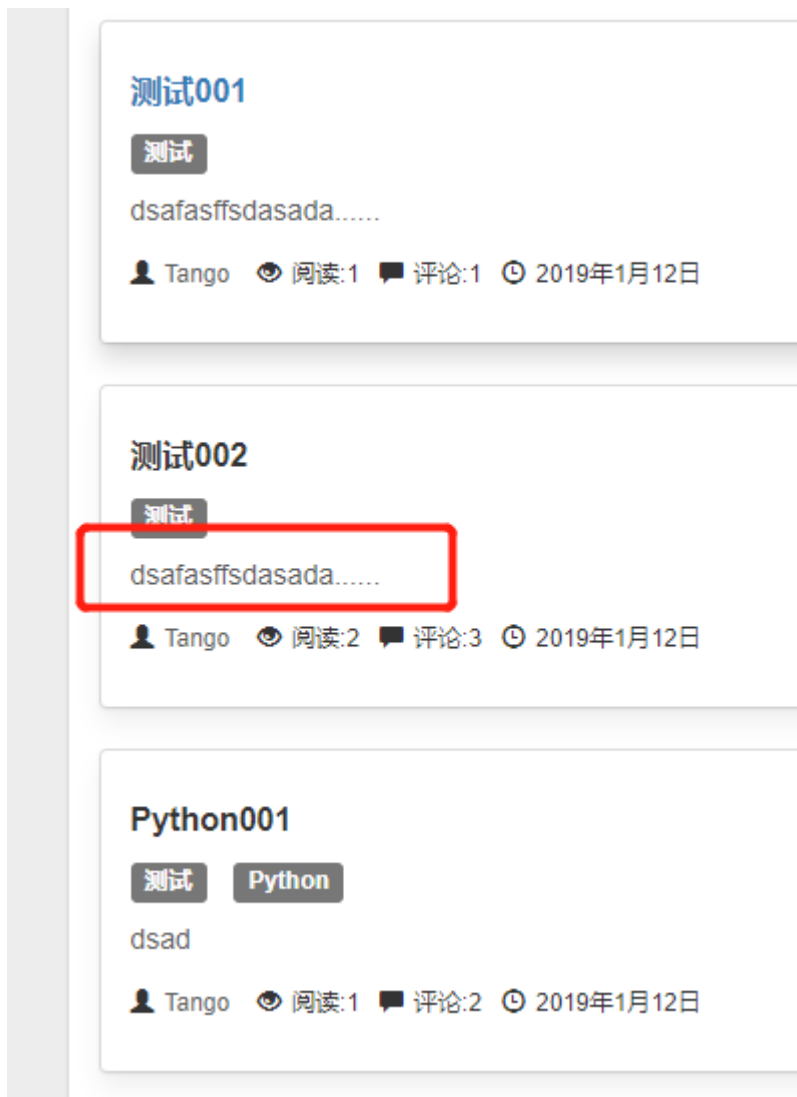
这里的`article.tags.all`是因为`tags`是`article`的一个外联表，所以要通过这样的方式来获取数据。

■ 文章概要，我们这里只显示前15个字，其他的省略。

```
{% if article.content|length >= 15 %}
    {{article.content|slice:'15'}}.....
{% else %}
    {{article.content}}
{% endif %}
```

`{% if article.content|length >= 15 %}`是Django模板语言中的判断语句。以`{% endif %}`结尾。

这里还用到了Django的选择标签“`|length`”是获取文字的长度，“`| slice:'15'`”这个类似于切片操作。效果如下：



暂时修改到这里，后面还要修改。

标签展示

1. 修改views.py

```
def getTags():  
    '''获取所有的标签'''  
    tags = Tag.objects.all()  
    return tags  
  
....  
def index(request):  
    ....  
    tags= getTags() # 获取所有的标签
```

2. 修改base.html页面

```

class="panel panel-default">


观察页面可以看出一个标签就是一个a标签，这里我们只保留一个，其他的用循环来获取。



```

<div class="labelList">
 {% for tag in tags %}
 {{tag.name}}
 {% endfor %}
</div>

```



效果如下：



## 最新发布文章列表



### 1. 修改views.py



```

def index(request):

 article_list_new = Article.objects.all().order_by('add_date')[:10] # 根据发布时间来
 排序,展现前10条

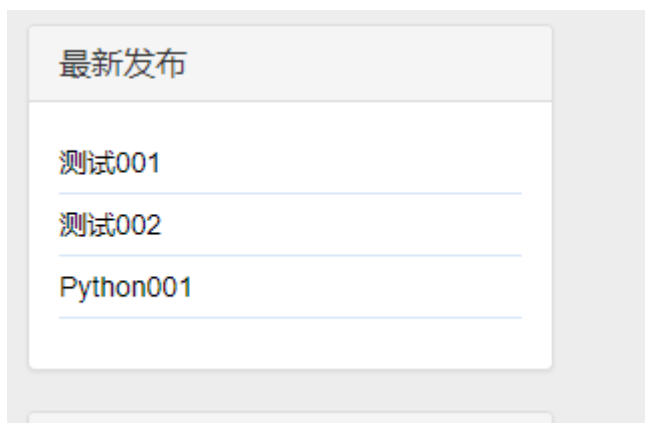
```


```


2. 修改base.html

```
{% for article in article_list_new %}
<li>
    <a href="/post/04928311">{{article.title}}</a>
</li>
{% endfor %}
```

效果如下：



但是感觉顺序不太对，因该是倒序。

所以需要修改一下代码：

```
article_list_new = Article.objects.all().order_by('-add_date')[:10] # 根据发布时间来排序,展现前10条
```

友情链接

和之前的做法一下，这里略。

文章详情页面

1. 修改views.py

```
def readArticle(request, id):
    template_name = "article_detail.html" # 要显示的模板名称
    print(type(id))
    article = Article.objects.filter(id = id )
    print(article)
    return render(request, template_name, {'article':article})
```

这里：

```
article = Article.objects.filter(id = id )
```

是根据传入的id来获取指定的博客信息

为了传入博客id需要修改一下index.html，修改如下

```
<div class="contentlist">
  {% for article in article_list%}
  <div class="panel panel-default">
    <div class="panel-body">

      <h4><a class="title" href="{% url 'readArticle' article.id %}">{{article.title}}<
      <p>
        {% for tagInfo in article.tags.all %}
        <a class="label label-default">{{tagInfo.name}}</a>
        {% endfor %}
      </p>
    </div>
  </div>
</div>
```

Django调用url用的是{% url 'url名' %}url名是我们定义再urls.py中的。后面接的article.id就是要传入的参数，对应的是url的<int:id>

```
panel-default contenttop">...

panel-default">
panel-heading">
"panel-title">最新发布</h3>

panel-body">

<!-- 开始 -->
<div class="contentlist">
  {% for article in article_list%}
  <div class="panel panel-default">
    <div class="panel-body">

      <h4><a class="title" href="{% url 'readArticle' article.id %}">{{article.title}}<
      <p>
        {% for tagInfo in article.tags.all %}
        <a class="label label-default">{{tagInfo.name}}</a>
        {% endfor %}
      </p>
    </div>
  </div>
</div>
```

```
9 Class-based views
10 1. Add an import: from other_app.views import Home
11 2. Add a URL to urlpatterns: path('', Home.as_view()),
12 Including another URLconf
13 1. Import the include() function: from django.urls import include
14 2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
15
16 from django.contrib import admin
17 from django.urls import path
18
19 from main.views import index, readArticle # 将views中定义个
20
21 urlpatterns = [
22     path('admin/', admin.site.urls),
23     path('', index, name='index'), # 定义首页的路由规则
24     path('read/<int:id>', readArticle, name='readArticle')
25 ]
26
27
```

2. 修改urls.py

```
path('read/<int:id>', readArticle, name='readArticle'),
```

定义路由规则

3. 修改html

因为我们需要用到markdown的内容，所以需要修改一下

这里用的是django-mdeditor

地址: <https://github.com/pylixm/django-mdeditor>

- o 安装插件

```
pip install django-mdeditor
```

```
(MyblogPro) PS F:\Blog\Blog> pip install django-mdeditor
Collecting django-mdeditor
Installing collected packages: django-mdeditor
Successfully installed django-mdeditor-0.1.9
(MyblogPro) PS F:\Blog\Blog>
```

- o models.py

```
from mdeditor.fields import MDTextField
....
class Article(models.Model):
    ....
    content = MDTextField() # 使用markdown格式
```

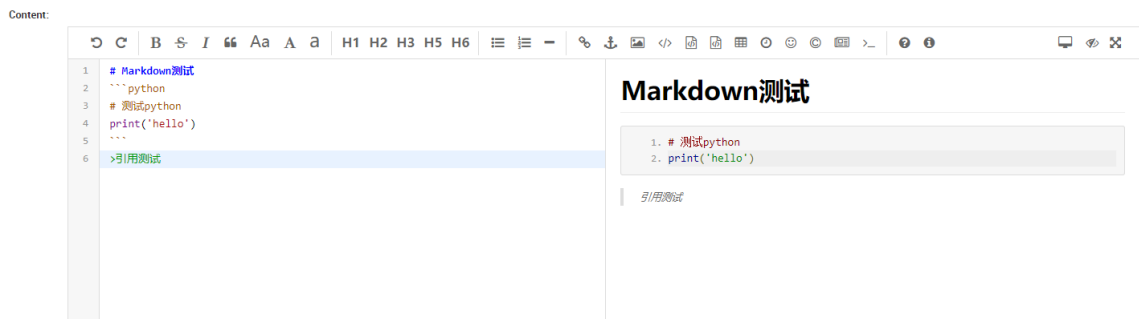
- 修改settings.py urls.py安装官方文档进行就可以。

创建一个uploads文件夹

并重新同步一下数据库

```
python .\manage.py makemigrations
python .\manage.py migrate
```

- 运行一下程序看看



后台已经有了makdown功能了，接下来再前台添加。

- 需要再base.html中添加一些样式和js可以参照例子

地址：https://github.com/pylixm/django-mdeditor/blob/master/mdeditor_demo_app/templates/show.html

运行效果如下：



同时也发现了导航和右侧的内容没有了。

修改一下我们的views.py

添加category信息

1. 为文章添加一个外键，将博客细分到各个分类中。

```
class Article(models.Model):
    ...
    category = models.ForeignKey("Category", verbose_name='分类',
on_delete=models.CASCADE, blank=True, null=True)
```

2. 根据不同的分类显示不同的文章列表



- 修改views.py和urls.py

```
def getArticleByCategory(request, cid):
    '''根据分类显示文章'''
    '''显示首页'''
    template_name = "index.html" # 要显示的模板名称

    category_list = getCategory() # 获取Category数据
    # 因为首页不需要将所有的文章都显示出来, 这里只显示前6条
    article_list = Article.objects.filter(category_id = cid) # 获取Article数据

    tags= getTags() # 获取所有的标签

    user_name = request.user.username # 获取用户名
    article_list_new = Article.objects.all().order_by('-add_date')[:10] # 根据发布时间来排序, 展现前10条

    links = getLinks() # 获取友情链接

    return render(request, template_name, {'category_list':category_list,
'article_list':article_list, 'tags':tags,
'article_list_new':article_list_new, 'links':links, 'user_name':user_name,
'isActive':cid}) #将我们定义好的页面返回到页面
```

今天的内容就是这些，下回的内容

1. 细节修改
2. 更换数据库
3. 上传服务器，正式发布