链滴

Django 个人博客搭建 (3)- 定制 admin

作者：zyk

原文链接：https://ld246.com/article/1566828009902

来源网站：链滴

一. 前言

注意

在开始教程之前，为了更合理地拆分功能模块，我把之前创建的名称为front_blog的APP改为了fron，admin_blog改为了article，希望小伙伴也能改一下，抱歉了。

在上一篇博文 **Django个人博客搭建(2)-编写视图** 中，主要讲述了如何在Django中编写视图，粗略利用django自带的admin添加了一篇文章。接下来，将讲述如何定制admin以及一些使用技巧。

二. 定制admin

1. 侧边栏优化

修改模型名称

运行Django项目，进入admin。发现侧边栏的 App名称和模型名称都为英文，需要将其改为中文。

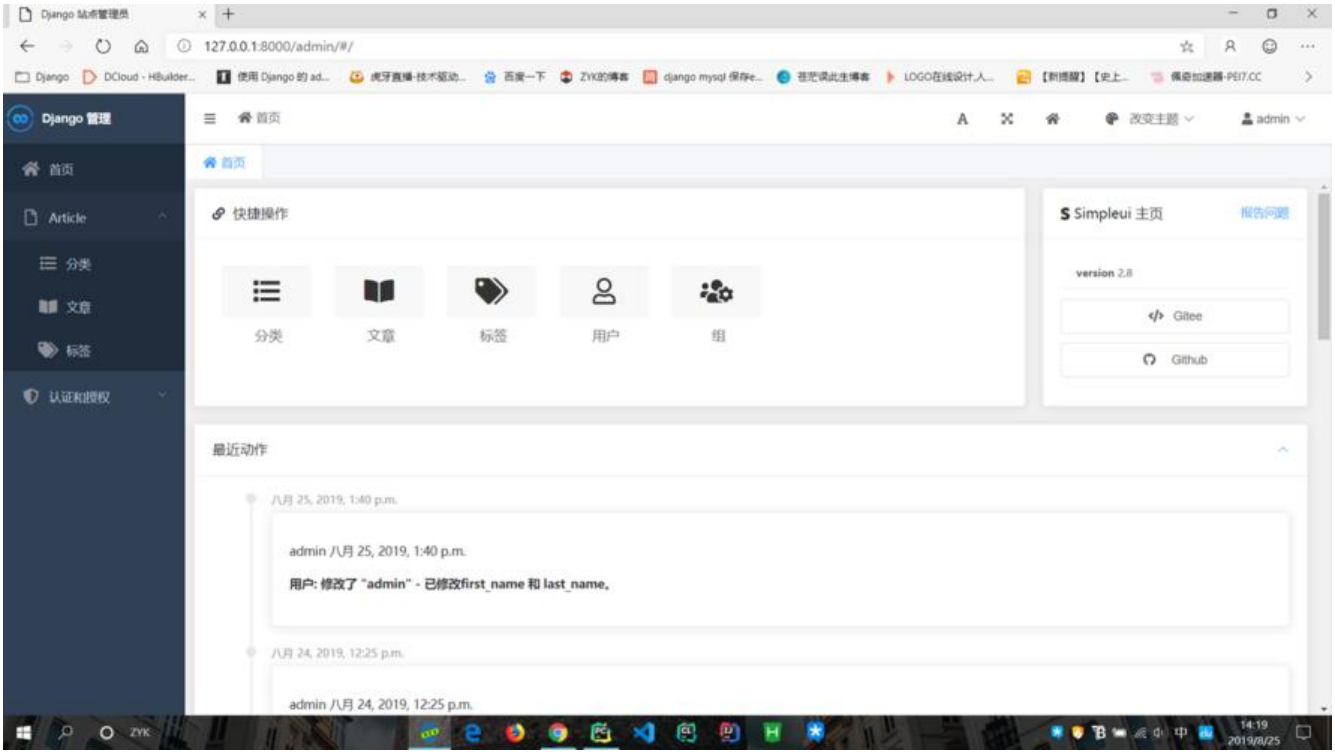
● 修改 **article/models.py**，为每一个模型类添加Meta的class属性。verbose_name表示后台界面型显示名称，verbose_name_plural表示复数形式名称，中文不分单复数，其值和verbose_name相即可。具体代码如下：

```
from django.conf import settings
from django.db import models

# 文章模型类
class Article(models.Model):
    id = models.BigAutoField(primary_key=True)  # 主键
    user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE, ver
ose_name="作者")  # 与自带的auth.user关联
    label = models.ManyToManyField('Label', verbose_name="标签")  # Label和Article为多对多
系
    title = models.CharField(max_length=100, verbose_name="标题")  # 标题
    content = models.TextField(max_length=100000, verbose_name="内容")  # 内容
    summary = models.CharField(blank=True, max_length=200, verbose_name="摘要")  # 摘要
    gmt_created = models.DateTimeField(blank=True, null=True, auto_now_add=True)  # 创建
间
    gmt_modified = models.DateTimeField(blank=True, null=True, auto_now=True)  # 修改时间

    class Meta:
        verbose_name = '文章'  # 后台模型显示名称
        verbose_name_plural = '文章'  # 后台模型复数显示名称

# 分类模型类
class Category(models.Model):
    id = models.BigAutoField(primary_key=True)
    category_name = models.CharField(max_length=32, verbose_name="分类名称")  # 分类名
    gmt_created = models.DateTimeField(blank=True, null=True, auto_now_add=True)  # auto
now_add为添加时的时间，更新对象时不会有变动。
    gmt_modified = models.DateTimeField(blank=True, null=True, auto_now=True)  # auto_n
```

w无论是你添加还是修改对象，时间为你添加或者修改的时间。

```
    class Meta:
        verbose_name = '分类'
        verbose_name_plural = '分类'

# 标签模型类
class Label(models.Model):
    id = models.BigAutoField(primary_key=True, verbose_name="标签名称")
    label_name = models.CharField(max_length=32)  # 标签名
    category = models.ManyToManyField('Category')  # Label和Category是多对多关系
    gmt_created = models.DateTimeField(blank=True, null=True, auto_now_add=True)  # auto
now_add为添加时的时间，更新对象时不会有变动。
    gmt_modified = models.DateTimeField(blank=True, null=True, auto_now=True)  # auto_n
w无论是你添加还是修改对象，时间为你添加或者修改的时间。

    class Meta:
        verbose_name = '标签'
        verbose_name_plural = '标签'
```

- 修改完之后刷新后台界面，查看具体效果。



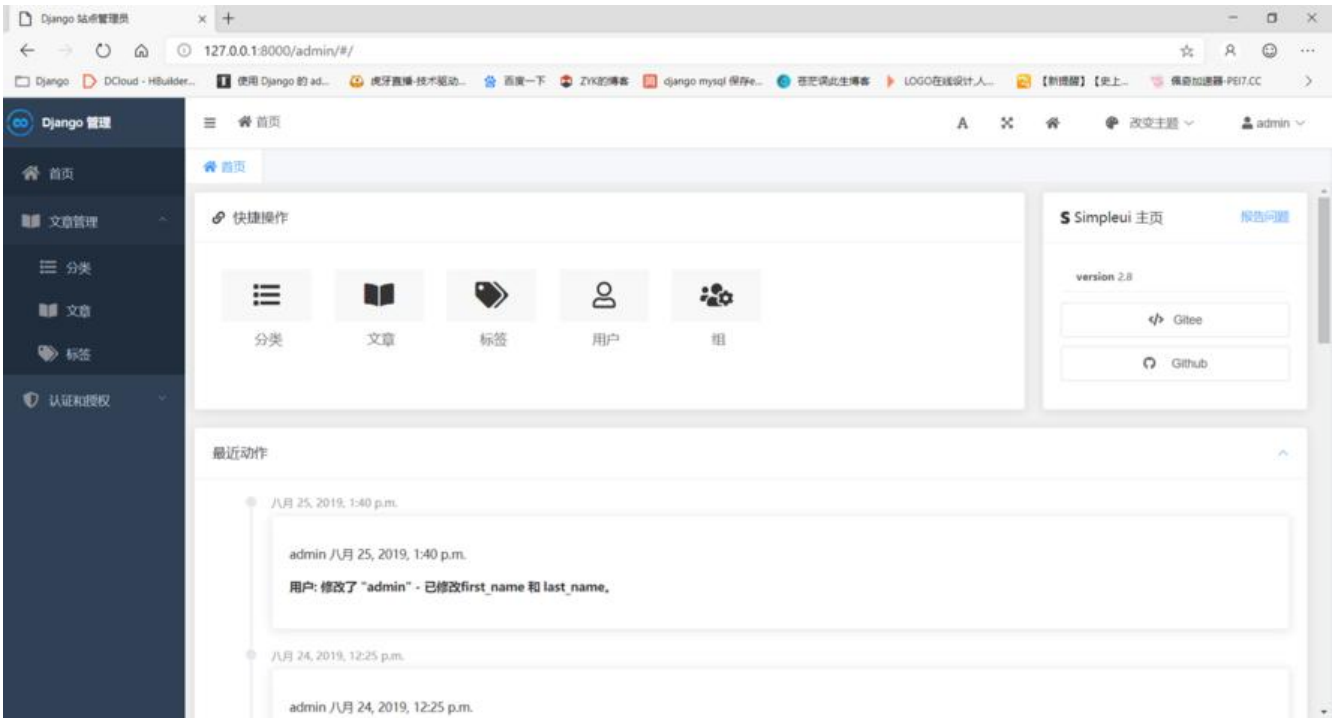
修改App名称

- 修改模型显示名称之后，我们发现app名称仍然为英文Article，需要将其改为中文。
- 进入 **article/init.py**，设置默认app_config，重新定义AppConfig，加入verbose_name属性，verbose_name的值即为后台app显示名称，如下。

```
from django.apps import AppConfig
import os
```

```
default_app_config = 'article.ArticleConfig'

# 获取apps所在文件夹名字，如果文件夹名字修改，这里可以动态调整
def get_current_app_name(_file):
    return os.path.split(os.path.dirname(_file))[-1]

class ArticleConfig(AppConfig):
    # 这里apps所在文件夹名字直接固定，如果更改则也需要调整
    # name = 'article'
    name = get_current_app_name(__file__)  # 这里的结果是：article
    verbose_name = '文章管理'
```

- 修改完之后刷新后台界面，操作无误的话可以看到APP名称也变成中文了。



2. 表单优化

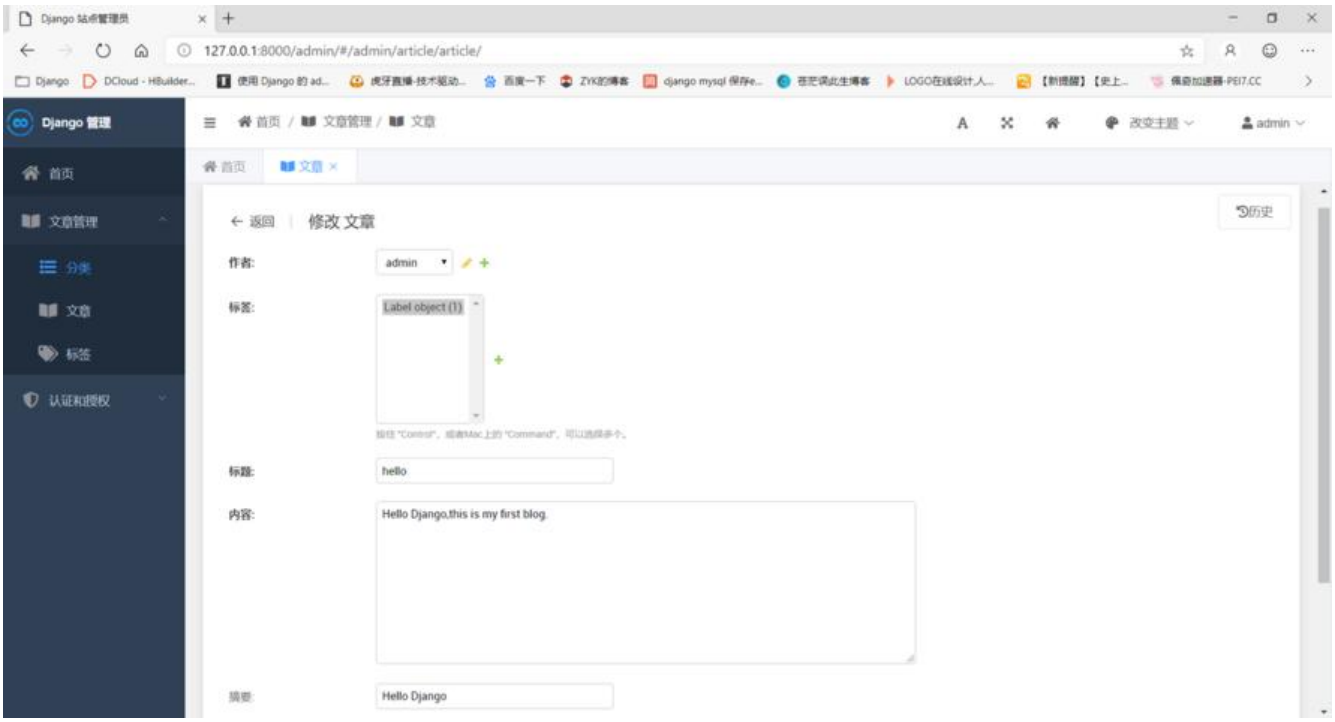
表单字段显示优化

- 点击 **文章管理/文章**，然后点击之前添加的那篇文章，发现表单名称都为英文。
- 将其修改为自定义中文字段，修改 **article/models.py**，为每一个字段添加verbose_name属性该属性的值即为后台表单中显示的字段名称。例如我们将Article模型类中的tile字段添加属性verbose name="标签"，则它将在后台表单中显示为标签。代码如下。

```
# Create your models here.
from django.conf import settings
from django.db import models


# 文章模型类
class Article(models.Model):
    id = models.BigAutoField(primary_key=True)  # 主键
```

```
    user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE, verose_name="作者")  # 与自带的auth.user关联
    label = models.ManyToManyField('Label', verbose_name="标签")  # Label和Article为多对多系
    title = models.CharField(max_length=100, verbose_name="标题")  # 标题
    content = models.TextField(max_length=100000, verbose_name="内容")  # 内容
    summary = models.CharField(blank=True, max_length=200, verbose_name="摘要")  # 摘要
    gmt_created = models.DateTimeField(blank=True, null=True, auto_now_add=True)  # 创建间
    gmt_modified = models.DateTimeField(blank=True, null=True, auto_now=True)  # 修改时间

    class Meta:
        verbose_name = '文章'
        verbose_name_plural = '文章'


# 分类模型类
class Category(models.Model):
    id = models.BigAutoField(primary_key=True)
    category_name = models.CharField(max_length=32, verbose_name="分类名称")  # 分类名
    gmt_created = models.DateTimeField(blank=True, null=True, auto_now_add=True)  # auto_now_add为添加时的时间，更新对象时不会有变动。
    gmt_modified = models.DateTimeField(blank=True, null=True, auto_now=True)  # auto_now无论是你添加还是修改对象，时间为你添加或者修改的时间。

    class Meta:
        verbose_name = '分类'
        verbose_name_plural = '分类'


# 标签模型类
class Label(models.Model):
    id = models.BigAutoField(primary_key=True, verbose_name="标签名称")
    label_name = models.CharField(max_length=32)  # 标签名
    category = models.ManyToManyField('Category')  # Label和Category是多对多关系
    gmt_created = models.DateTimeField(blank=True, null=True, auto_now_add=True)  # auto_now_add为添加时的时间，更新对象时不会有变动。
    gmt_modified = models.DateTimeField(blank=True, null=True, auto_now=True)  # auto_now无论是你添加还是修改对象，时间为你添加或者修改的时间。

    class Meta:
        verbose_name = '标签'
        verbose_name_plural = '标签'
```

- 点击文章列表中的文章，进入修改页面，检验修改效果。



列表字段显示优化

之前我们在**artilce/admin.py**中利用了**admin.site.register**注册模型类，但这只能够简单地对数据行管理。

```
from django.contrib import admin
# 导入模型类
from article.models import Article, Category, Label
# 注册
admin.site.register(Article)
admin.site.register(Category)
admin.site.register(Label)
```

文章列表并不能显示标题和作者等信息。因此我们需要借助Django为我们提供的ModelAdmin进行度定制，ModelAdmin类是一个可以继承的基类，它负责admin页面里的数据展示。

● 修改 **artilce/admin.py**，新建 ArticleAdmin 类，在其中写入 list_display ， list_display 的值是一元祖，它表明哪些字段将在列表中显示。例如，将ArticleAdmin 中的 list_display 值设为('title', 'user', gmt_created')，如下：

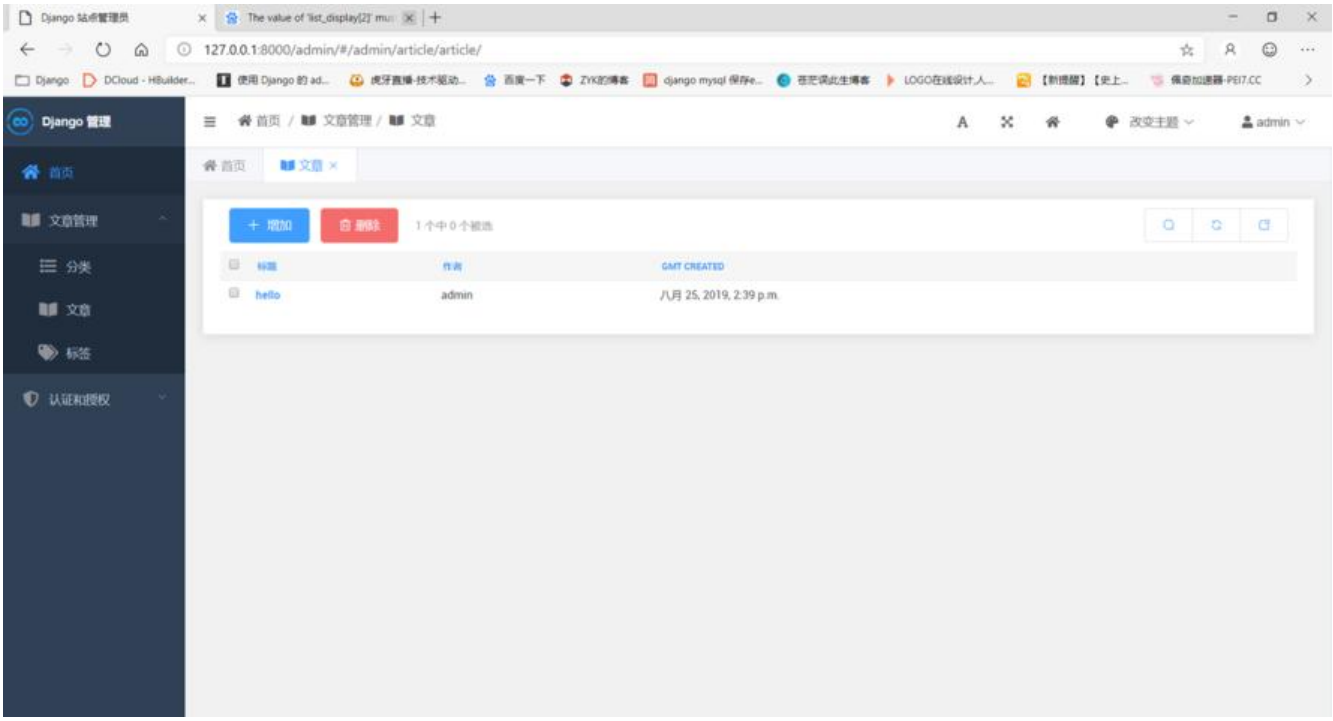
注意：不要忘记把 Article 类和 ArticleAdmin 类通过admin.site.register(Article, ArticleAdmin)注到admin站点中

```
from django.contrib import admin
# 导入模型类
from article.models import Article, Category, Label

class ArticleAdmin(admin.ModelAdmin):
    list_display = ('title', 'user',  'gmt_created')

# 注册
admin.site.register(Article, ArticleAdmin)
```

原文链接：Django 个人博客搭建 (3)- 定制 admin

```
admin.site.register(Category)
admin.site.register(Label)
```

- 刷新后台界面，查看文章列表，观察效果。



列表多对多属性显示

那如果我们想在列表中显示多对多字段属性该怎么办，例如，在**文章**列表中，想要显示**标签**属性（**文章**和**签**是ManyToMany多对多关系）

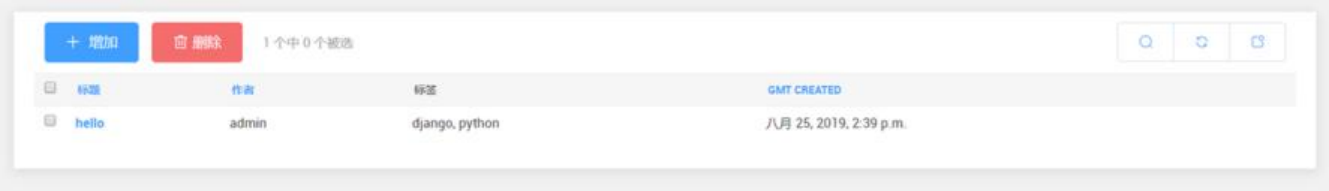
- 显示多对多关系字段，需要重新定义显示函数，利用数据库ORM方法返回关联的对象中所需要显的字段。
- 修改 **artilce/admin.py**，在list_display中加入自定义字段标签，定义并实现标签函数，如下。

注意：自定义字段名称与函数名称要**一致**。

```
from django.contrib import admin

# 导入模型类
from article.models import Article, Category, Label


class ArticleAdmin(admin.ModelAdmin):
    # 显示的字段
    list_display = ('title', 'user', '标签', 'gmt_created')

    # 定义标签显示
    def 标签(self, obj):
        return [l.label_name for l in obj.label.all()]  # 返回与Article关联的Label的label_name字段性的值

    filter_horizontal = ('label',)  # 表示对label属性进行过滤
```

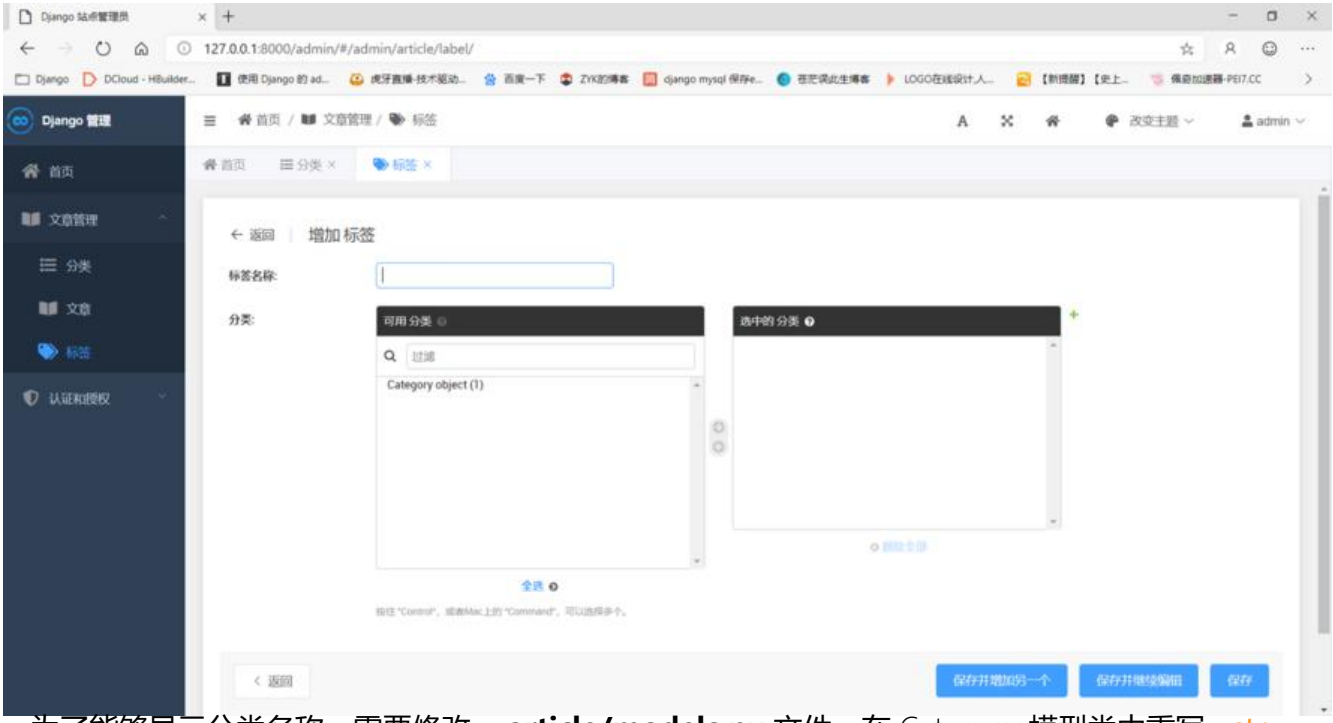
```
# 注册
admin.site.register(Article, ArticleAdmin)
admin.site.register(Category)
admin.site.register(Label)
```

- 刷新后台界面，检查修改结果。



表单多对多字段显示

- 进入添加标签页面，我们发现分类显示并未显示分类名称。

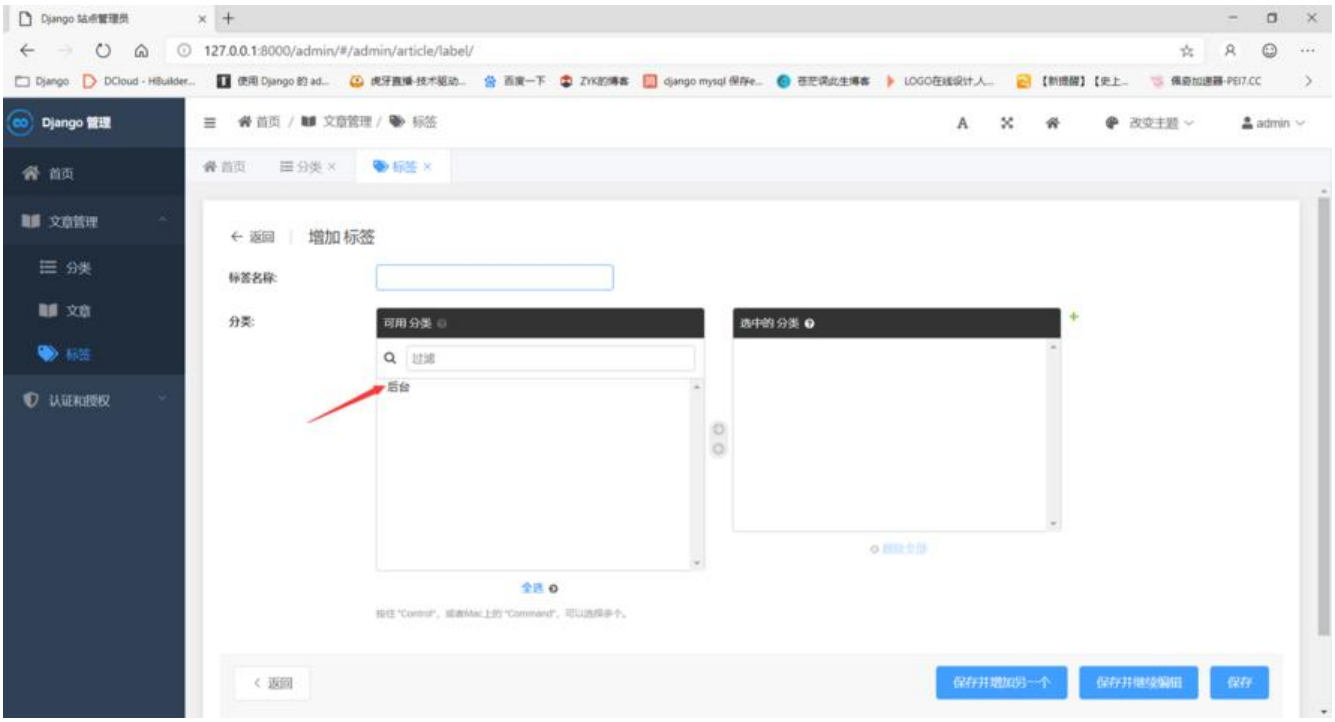


- 为了能够显示分类名称，需要修改 **article/models.py** 文件，在 Category 模型类中重写__str__法，返回分类名称 category_name 属性。同时为了能在添加文章时显示标签名称，也在Lable模型类重写__str__方法。具体代码如下：

```
from django.conf import settings
from django.db import models

# 文章模型类
class Article(models.Model):
    id = models.BigAutoField(primary_key=True)  # 主键
    user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE, ver
ose_name="作者")  # 与自带的auth.user关联
    label = models.ManyToManyField('Label', verbose_name="标签")  # Label和Article为多对多
系
```

```
    title = models.CharField(max_length=100, verbose_name="标题")  # 标题
    content = models.TextField(max_length=100000, verbose_name="内容")  # 内容
    summary = models.CharField(blank=True, max_length=200, verbose_name="摘要")  # 摘要
    gmt_created = models.DateTimeField(blank=True, null=True, auto_now_add=True)  # 创建
间
    gmt_modified = models.DateTimeField(blank=True, null=True, auto_now=True)  # 修改时间

    class Meta:
        verbose_name = '文章'
        verbose_name_plural = '文章'

    def __str__(self):
        return self.title

# 分类模型类
class Category(models.Model):
    id = models.BigAutoField(primary_key=True)
    category_name = models.CharField(max_length=32, verbose_name="分类名称")  # 分类名
    gmt_created = models.DateTimeField(blank=True, null=True, auto_now_add=True)  # auto
now_add为添加时的时间，更新对象时不会有变动。
    gmt_modified = models.DateTimeField(blank=True, null=True, auto_now=True)  # auto_n
w无论是你添加还是修改对象，时间为你添加或者修改的时间。

    class Meta:
        verbose_name = '分类'
        verbose_name_plural = '分类'

    def __str__(self):
        return self.category_name  # 在后台表单中显示分类名

# 标签模型类
class Label(models.Model):
    id = models.BigAutoField(primary_key=True)
    label_name = models.CharField(max_length=32, verbose_name="标签名称")  # 标签名
    category = models.ManyToManyField('Category', verbose_name="分类")  # Label和Categor
是多对多关系
    gmt_created = models.DateTimeField(blank=True, null=True, auto_now_add=True)  # auto
now_add为添加时的时间，更新对象时不会有变动。
    gmt_modified = models.DateTimeField(blank=True, null=True, auto_now=True)  # auto_n
w无论是你添加还是修改对象，时间为你添加或者修改的时间。

    class Meta:
        verbose_name = '标签'
        verbose_name_plural = '标签'

    def __str__(self):
        return self.label_name
```

- 刷新后台页面，进入添加标签页面，不出意外就能显示出分类名称了。



3. 常用功能集成

添加搜索栏和分页

后台中我们有时候需要查询参数，默认是没有开启搜索功能的，我们需要在 **article/admin.py** 中加入earch_fields = ('title', 'label__label_name')来开启搜索功能。

search_fields中的参数是模型类中的字段名，该字段名可以用来进行关键字查询。跨表查询类似于django的数据库ORM关系，在这里先不细说。

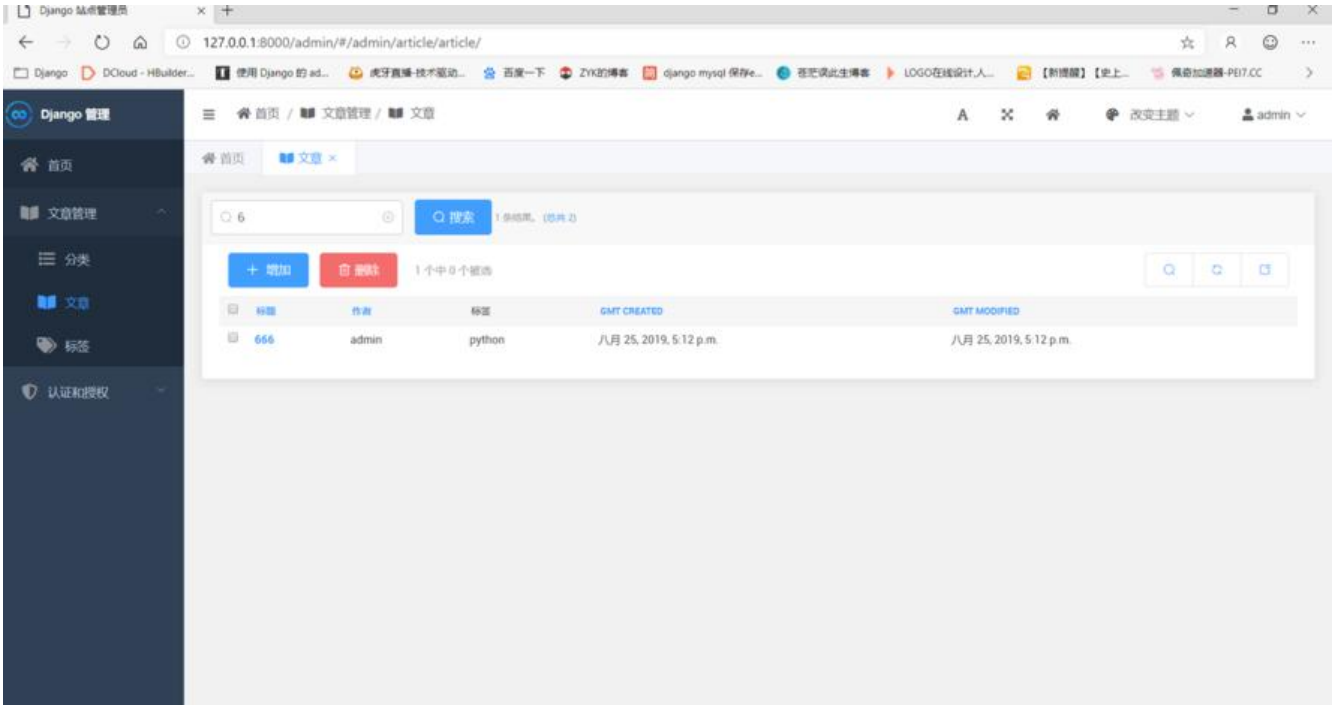
想要了解Django自带的ORM数据库API的可以查看我的另外一篇博客Django数据库常用ORM方法

同时加入分页参数list_per_page，即每页显示条数。具体代码如下:

```
from django.contrib import admin
# 导入模型类
from article.models import Article, Category, Label

class ArticleAdmin(admin.ModelAdmin):
    # 显示的字段
    list_display = ('title', 'user', 'show_labels', 'gmt_created', 'gmt_modified')

    # 定义标签显示
    def show_labels(self, obj):
        return [l.label_name for l in obj.label.all()]  # 返回与Article关联的Label的label_name字段性的值

    show_labels.short_description = "标签"
    filter_horizontal = ('label',)  # 表示对label属性进行过滤
    search_fields = ('title', 'label__label_name')  # 可搜索属性
    list_per_page = 10 # 每页条数
```

```
class LabelAdmin(admin.ModelAdmin):
    list_display = ('label_name', 'show_categories', 'gmt_created', 'gmt_modified')  # 显示的字段

    def show_categories(self, obj):
        return [c.category_name for c in obj.category.all()]

    show_categories.short_description = "分类"
    filter_horizontal = ('category',)  # 表示对label属性进行过滤
    search_fields = ('label_name', 'category__category_name')  # 可搜索属性
    list_per_page = 10


class CategoryAdmin(admin.ModelAdmin):
    # 显示的字段
    list_display = ('category_name', 'gmt_created', 'gmt_modified')
    list_per_page = 10

# 注册
admin.site.register(Article, ArticleAdmin)
admin.site.register(Category, CategoryAdmin)
admin.site.register(Label, LabelAdmin)
```

- 修改完之后，刷新后台界面，测试搜索功能。



Django自带的admin还有很多强大的功能，在这里不再一一叙述了，小伙伴们可以自行钻研一波。
一篇将讲述如何编辑博客主页视图和页面显示。