

## 数据库模型的设计与建立

### models.py

- models.py是DjangoMVC设计模式中最关键的“M”层面，实现了与数据库的模型交互以及数据表的设计建立
- 上层APP\_NAME为 `kblog`，因此所创建的数据表的前缀名称为 `kblog_`
- 全部源码置于文后

对象

打开表

设计表

新建表

删除表

导入向导

导出向导

名	自动递...	修改日期	数据长度	引擎	行	注释
auth_group	1		16 KB	InnoDB	0	
auth_group_permissions	1		16 KB	InnoDB	0	
auth_permission	57	2022-11-30 13:22:29	16 KB	InnoDB	56	
auth_user	1		16 KB	InnoDB	0	
auth_user_groups	1		16 KB	InnoDB	0	
auth_user_user_permissions	1		16 KB	InnoDB	0	
django_admin_log	1		16 KB	InnoDB	0	
django_content_type	15	2022-11-30 13:22:29	16 KB	InnoDB	14	
django_migrations	22	2022-11-30 13:22:29	16 KB	InnoDB	21	
django_session	0		16 KB	InnoDB	0	
kblog_about	1		16 KB	InnoDB	0	
kblog_article	1		16 KB	InnoDB	0	
kblog_article_tag	1		16 KB	InnoDB	0	
kblog_category	1		16 KB	InnoDB	0	
kblog_notice	1		16 KB	InnoDB	0	
kblog_site	1		16 KB	InnoDB	0	
kblog_skill	1		16 KB	InnoDB	0	
kblog_tag	1		16 KB	InnoDB	0	
kblog_valine	1		16 KB	InnoDB	0	

快捷操作

文章

分类

标签

公告栏

基本信息

技能

网站设置

valine评论

用户

权限组

HappyGoing

### 文章分类模型Category

- 对应数据表-kblog\_Category
- 实现了文章分类这一功能

```
class Category(models.Model):  
    '''文章分类'''  
    name = models.CharField(max_length=20, verbose_name='分类名称')  
    index = models.IntegerField(default=1, verbose_name='分类排序')
```

```

add_menu = models.BooleanField(default=False, verbose_name='添加到导航栏')
icon = models.CharField(max_length=30, default='fas fa-home', verbose_name='导航图标')

class Meta:
    verbose_name_plural = verbose_name = '分类'

# 统计分类对应文章数,并放入后台
def get_items(self):
    return len(self.article_set.all())

get_items.short_description = '文章数' # 设置后台显示表头

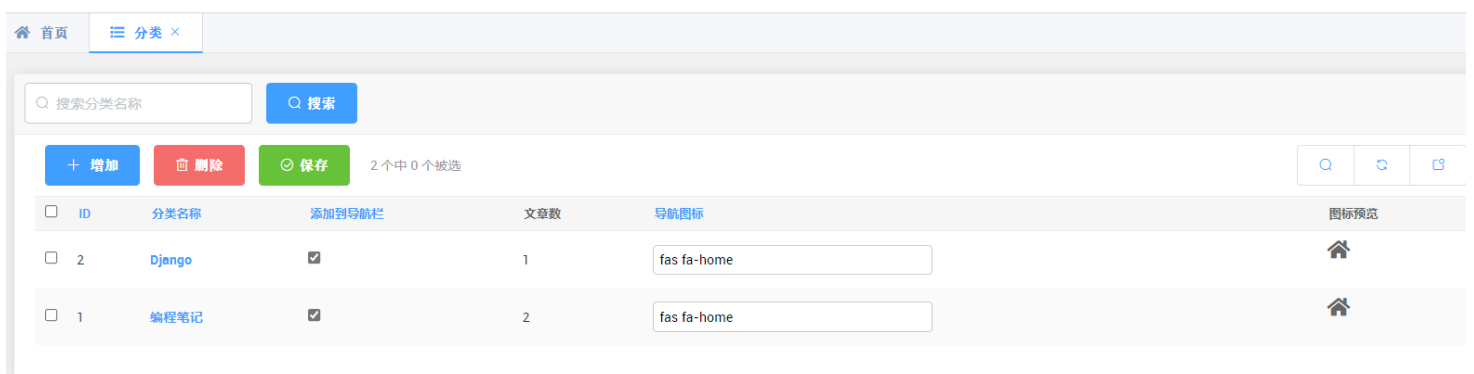
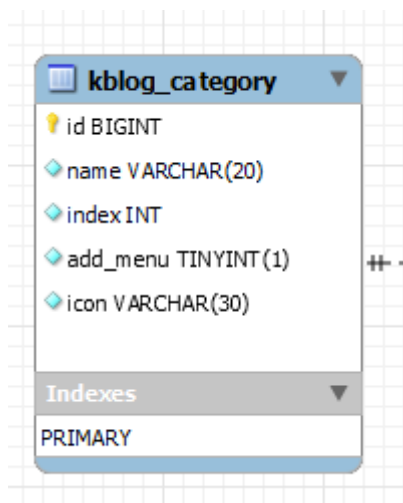
# 后台图标预览
def icon_data(self): # 引入Font Awesome Free 5.11.1
    return format_html('<h1><i class="{self.icon}"></i></h1>', self.icon) # 转化为<i class="{self.icon}">
</i>

icon_data.short_description = '图标预览'

def __str__(self):
    return self.name

```

- 各有实际意义字段解释如下：
  - `name`:分类名称
  - `index`:分类排序
- 定义了统计函数 `get_items()` 和后台图标预览规格化HTML函数 `icon_data()`
- 



## 文章标签模型Tag

- 对应 `kblog_tag` 数据表
- `class Tag(models.Model):`

```

''' 标签 '''
name = models.CharField(max_length=20, verbose_name='标签名称')

class Meta:
    verbose_name = '标签'
    verbose_name_plural = verbose_name

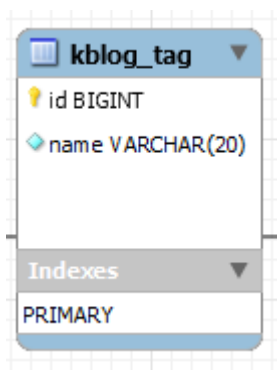
# 统计分类对应文章数,并放入后台
def get_items(self):
    return len(self.article_set.all())

get_items.short_description = '文章数' # 设置后台显示表头

def __str__(self):
    return self.name

```

- 只有一个字段 `name` -标签名称
- 实现了统计分类对应文章数函数 `get_items()`
- 



🏠 首页    ☰ 分类 ×    🏷️ 标签 ×

🔍 搜索标签名称    🔍 搜索

➕ 增加    🗑️ 删除    5个中 0个被选    🔍 ↺ 📄

<input type="checkbox"/> ID	标签名称	文章数
<input type="checkbox"/> 5	数据库	1
<input type="checkbox"/> 4	编程	1
<input type="checkbox"/> 3	Python	1
<input type="checkbox"/> 2	博客开发实录	2
<input type="checkbox"/> 1	Django	2

## 文章模型Article

- 对应 `kblog_article` 数据表

- ```

class Article(models.Model):
    '''文章'''
    title = models.CharField(max_length=50, verbose_name='文章标题')
    author = models.CharField(max_length=10, verbose_name='作者', default='李欢欢', blank=True, null=True)
    desc = models.CharField(max_length=50, verbose_name='文章描述')
    cover = models.URLField(max_length=200, default='https://happygoing.oss-cn-beijing.aliyuncs.com/img/Apartment-rain.png',
                             verbose_name='文章封面')
    content = MDTextField(verbose_name='文章内容')
    click_count = models.PositiveIntegerField(default=0, verbose_name='阅读量')
    is_recommend = models.BooleanField(default=False, verbose_name='是否推荐')
    # 文章创建时间.参数 default=datetime.now 指定其在创建数据时将默认写入当前的时间
    add_time = models.DateTimeField(default=datetime.now, verbose_name='发布时间')

```

```

# 文章更新时间。参数 auto_now=True 指定每次数据更新时自动写入当前时间
update_time = models.DateTimeField(auto_now=True, verbose_name='更新时间')
category = models.ForeignKey(Category, blank=True, null=True, verbose_name='文章分类',
on_delete=models.CASCADE) # 此处设置为一个文章只能属于一个类
tag = models.ManyToManyField(Tag, blank=True, verbose_name='文章标签')

class Meta:
    verbose_name = '文章'
    verbose_name_plural = verbose_name
    ordering = ('-add_time',) # 以创建时间倒序排列

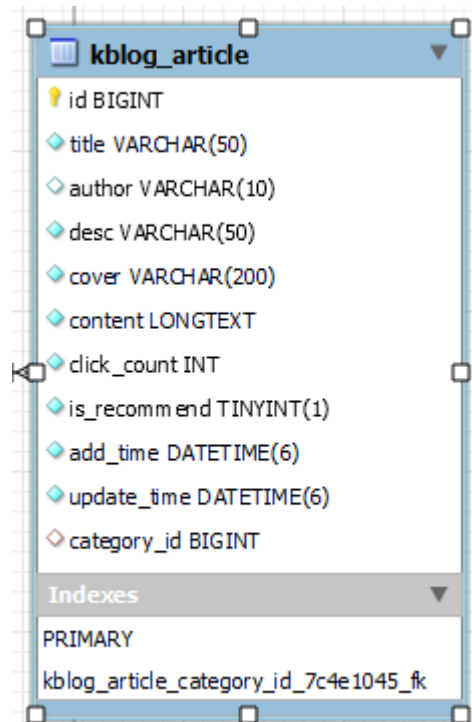
def cover_preview(self):
    return format_html('', self.cover, )

cover_preview.short_description = '文章封面预览'

def __str__(self):
    return self.title # 将文章标题返回

```

- `title`: 文章标题 CharField
- `author`: 文章作者 默认值为李欢欢 CharField
- `desc`: 文章描述 CharField
- `cover`: 文章封面 URLField预览
- `content`: 文章详细内容 MDTextField 实现Markdown编写模式
- `click_count`: 文章阅读量 活跃整型字段
- `is_recommend` 是否为推荐文章 BooleanField
- `add_time`:文章创建时间 DateTimeField 默认取值为datetime.now
- `update_time`: 文章最后更新时间
- `category`: 文章对应分类名称 ForeignKey 对应 `kblog_category`
- `tag`:文章对应标签名称 ManyToManyField多对多字段 对应 `kblog_tag`
- 



⋮ 首页

A⌵🏠🌐 改变主题admin

🏠 首页

☰ 分类

🏷 标签

📄 文章

🔍 搜索文章标题,文章描述,文

文章标题

🕒 发布时间 - 发布时间

🔍 搜索

📅 2022 11月27日

+ 增加

🗑 删除

💾 保存

📶 导入

📶 导出

3 个中 0 个被选

🔍

🔄

📄

☐ 发布时间 1

📄 文章标题

🖼 文章封面预览

📄 文章分类

🏷 标签


✅ 是否推荐 2

📖 阅读量

🕒 更新时间

☐ 2022年11月27日 09:24

HuanBlog开发注解-Part3



Django

🔧 +

Django


✅

1

2022年11月27日 09:26

☐ 2022年11月27日 05:48

HuanBlog开发注解-Part2



编程笔记

🔧 +

博客开发实录


✅

0

2022年11月27日 05:50

☐ 2022年11月27日 01:16

HuanBlog开发注解-Part1



编程笔记

🔧 +

Django, 博客开发实录, Python, 编程, 数据库

✅

9

2022年11月27日 01:19

## 公告栏目Notice

- 对应 kblog\_notice 数据表

```
class Notice(models.Model):
    '''公告栏'''
    title = models.CharField(max_length=30, verbose_name='公告栏标题')
    content = models.TextField(max_length=500, verbose_name='公告内容')
    icon = models.CharField(default='far fa-lightbulb', max_length=50, verbose_name='公告图标')

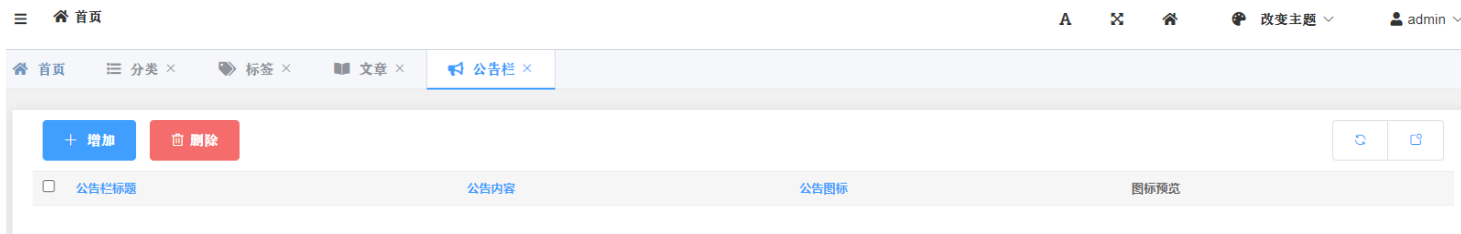
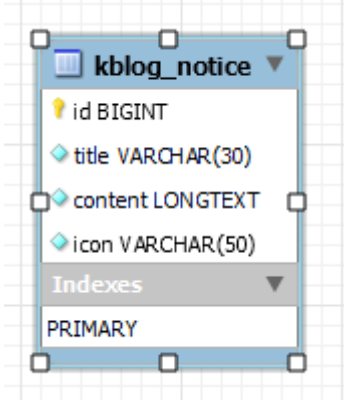
    class Meta:
        verbose_name = verbose_name_plural = '公告栏'

    def icon_data(self):
        return format_html('<h1><i class="{self.icon}"></i></h1>', self.icon) # 转化为<i class="{self.icon}"></i>

    icon_data.short_description = '图标预览'

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

- title:公告栏标题 CharField
- content: 公告内容 TextField
- icon: 公告图标 CharField
-



## 文章评论模型Valine

- 对应 kblog\_valine 数据表
- 文章评论的功能通过使用 Valine 富文本评论库实现



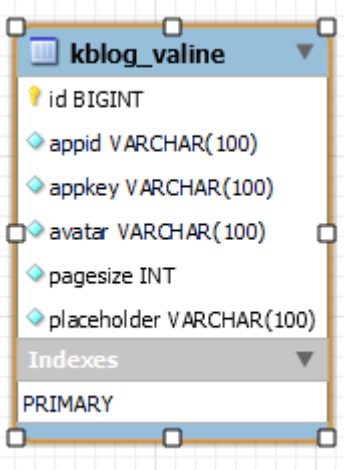
### 介绍

Valine - 一款快速、简洁且高效的无后端评论系统。

[在Github上编辑此页](#)

- ```
class Valine(models.Model):  
    '''valine评论'''  
    appid = models.CharField(max_length=100, verbose_name='appId')  
    appkey = models.CharField(max_length=100, verbose_name='appkey')  
    avatar = models.CharField(default='', blank=True, max_length=100, verbose_name='avatar')  
    pagesize = models.IntegerField(default='10', verbose_name='pageSize')  
    placeholder = models.CharField(max_length=100, verbose_name='placeholder')  
  
    class Meta:  
        verbose_name = 'valine评论'  
        verbose_name_plural = verbose_name
```

- 分别对应配置申请好的 appid 和 appkey



三

首页

分类 ×

标签 ×

文章 ×

公告栏 ×

valine评论 ×

增加

删除

1 个中 0 个被选

APPID	APPKEY	AVATAR	PAGESIZE	PLACEHOLDER
eriHD4UOtWONOGKdf3c5JeVJ-gzGzoHsz	PKkunDFMcOs8VnNJE9XmzPFV		10	[昵称和邮箱必填，网址选填。填上邮箱，会给你发送评论提醒] </> 请畅所欲言~

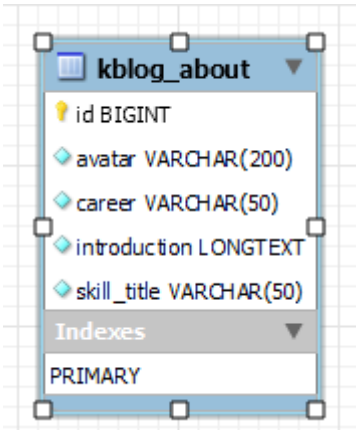
## 个人介绍模型About

```
class About(models.Model):
    '''关于'''
    avatar = models.URLField(verbose_name='头像')
    career = models.CharField(max_length=50, verbose_name='事业')
    introduction = models.TextField(verbose_name='介绍')
    skill_title = models.CharField(default='技能', max_length=50, verbose_name='技能标题')

    class Meta:
        verbose_name_plural = verbose_name = '关于'

    def avatar_admin(self):
        return format_html('', self.avatar, )

    avatar_admin.short_description = '头像预览'
```



三

首页

分类 ×

标签 ×

文章 ×

公告栏 ×

valine评论 ×

基本信息 ×

增加

删除

1 个中 0 个被选

头像预览	事业	介绍	技能标题
	学生	Kylin	Django Python

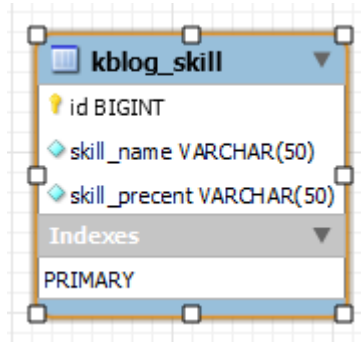
## 技能模型Skill

```
class skill(models.Model):
    """关于页技能"""
    skill_name = models.CharField(max_length=50, verbose_name='方向名')
    skill_precent = models.CharField(default='%', max_length=50, verbose_name='百分比')

    class Meta:
        verbose_name_plural = verbose_name = '技能'

# 后台图标预览
def icon_data(self):
    return format_html('<h1><i class="{}"></i></h1>', self.social_icon)

icon_data.short_description = '图标预览'
```



+ 增加
删除
2 个中 0 个被选

<input type="checkbox"/>	方向名	百分比
<input type="checkbox"/>	C++	%70
<input type="checkbox"/>	Python	%70

## Django Python

Python

%70

C++

%70

## 站点页面配置Site

```
class Site(models.Model):
    """站点配置"""
    site_name = models.CharField(default='kylinBlog', max_length=30, verbose_name='网站名字')
    keywords = models.CharField(default='keyword', max_length=50, verbose_name='网站关键词')
    logo = models.URLField(default='https://jwt1399.top/favicon.png', max_length=100,
        verbose_name='网站logo')
    desc = models.CharField(max_length=50, verbose_name='网站描述')
    slogan = models.CharField(max_length=50, verbose_name='网站标语')
    dynamic_slogan = models.CharField(max_length=50, verbose_name='动态标语')
    bg_cover = models.URLField(default='http://119.23.243.154/image/Covteam-hack.jpg',
        max_length=100,
        verbose_name='背景图片')

    class Meta:
        verbose_name = '网站设置'
        verbose_name_plural = verbose_name

def logo_preview(self): # logo预览
    return format_html('', self.logo)
```

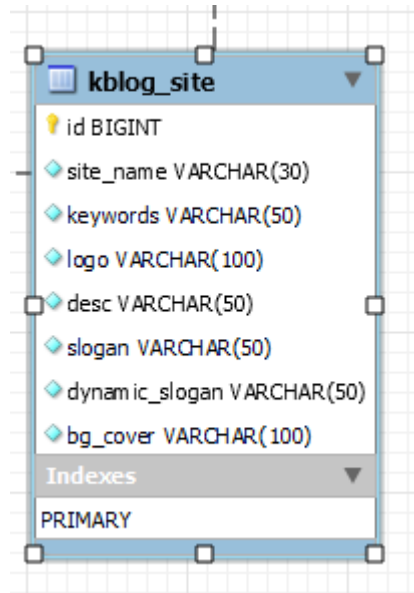


```
logo_preview.short_description = 'logo预览'
```

```
def bgcover_preview(self): # 背景图片预览
    return format_html('',
self.bg_cover)
```

```
bgcover_preview.short_description = '首页背景预览'
```

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



# ER图



```

class Category(models.Model):
    '''文章分类'''
    name = models.CharField(max_length=20, verbose_name='分类名称')
    index = models.IntegerField(default=1, verbose_name='分类排序')
    add_menu = models.BooleanField(default=False, verbose_name='添加到导航栏')
    icon = models.CharField(max_length=30, default='fas fa-home', verbose_name='导航图标')

    class Meta:
        verbose_name_plural = verbose_name = '分类'

    # 统计分类对应文章数,并放入后台
    def get_items(self):
        return len(self.article_set.all())

    get_items.short_description = '文章数' # 设置后台显示表头

    # 后台图标预览
    def icon_data(self): # 引入Font Awesome Free 5.11.1
        return format_html('<h1><i class="{}"></i></h1>', self.icon) # 转化为<i class="{self.icon}">
</i>

    icon_data.short_description = '图标预览'

    def __str__(self):
        return self.name

'''
文章标签模型
'''

class Tag(models.Model):
    '''标签'''
    name = models.CharField(max_length=20, verbose_name='标签名称')

    class Meta:
        verbose_name = '标签'
        verbose_name_plural = verbose_name

    # 统计分类对应文章数,并放入后台
    def get_items(self):
        return len(self.article_set.all())

    get_items.short_description = '文章数' # 设置后台显示表头

    def __str__(self):
        return self.name

'''
文章标签模型
'''

class Article(models.Model):
    '''文章'''
    title = models.CharField(max_length=50, verbose_name='文章标题')
    author = models.CharField(max_length=10, verbose_name='作者', default='李欢欢', blank=True,
null=True)

```

```

desc = models.CharField(max_length=50, verbose_name='文章描述')
cover = models.URLField(max_length=200, default='https://happygoing.oss-cn-beijing.aliyuncs.com/img/Apartment-rain.png',
                        verbose_name='文章封面')
content = MDTextField(verbose_name='文章内容')
click_count = models.PositiveIntegerField(default=0, verbose_name='阅读量')
is_recommnd = models.BooleanField(default=False, verbose_name='是否推荐')
# 文章创建时间。参数 default=datetime.now 指定其在创建数据时将默认写入当前的时间
add_time = models.DateTimeField(default=datetime.now, verbose_name='发布时间')
# 文章更新时间。参数 auto_now=True 指定每次数据更新时自动写入当前时间
update_time = models.DateTimeField(auto_now=True, verbose_name='更新时间')
category = models.ForeignKey(Category, blank=True, null=True, verbose_name='文章分类',
on_delete=models.CASCADE) # 此处设置为一个文章只能属于一个类
tag = models.ManyToManyField(Tag, blank=True, verbose_name='文章标签')

```

```

class Meta:
    verbose_name = '文章'
    verbose_name_plural = verbose_name
    ordering = ('-add_time',) # 以创建时间倒序排列

```

```

def cover_preview(self):
    return format_html('', self.cover, )

```

```

cover_preview.short_description = '文章封面预览'

```

```

def __str__(self):
    return self.title # 将文章标题返回

```

```

'''
公告栏目
'''

```

```

class Notice(models.Model):
    '''公告栏'''
    title = models.CharField(max_length=30, verbose_name='公告栏标题')
    content = models.TextField(max_length=500, verbose_name='公告内容')
    icon = models.CharField(default='far fa-lightbulb', max_length=50, verbose_name='公告图标')

    class Meta:
        verbose_name = verbose_name_plural = '公告栏'

    def icon_data(self):
        return format_html('<h1><i class="{}"></i></h1>', self.icon) # 转化为<i class="{self.icon}">
</i>

```

```

icon_data.short_description = '图标预览'

```

```

def __str__(self):
    return self.title

```

```

'''
评论模型 使用valine系统框架
'''

```

```

# 评论模型的实现 使用valine系统调度完成
# 在后台配置Valine的appkey和appid

```

```

class Valine(models.Model):
    '''valine评论'''
    appid = models.CharField(max_length=100, verbose_name='appId')
    appkey = models.CharField(max_length=100, verbose_name='appKey')
    avatar = models.CharField(default='', blank=True, max_length=100, verbose_name='avatar')
    pagesize = models.IntegerField(default='10', verbose_name='pageSize')
    placeholder = models.CharField(max_length=100, verbose_name='placeholder')

    class Meta:
        verbose_name = 'valine评论'
        verbose_name_plural = verbose_name

'''
作者介绍
'''

class About(models.Model):
    '''关于'''
    avatar = models.URLField(verbose_name='头像')
    career = models.CharField(max_length=50, verbose_name='事业')
    introduction = models.TextField(verbose_name='介绍')
    skill_title = models.CharField(default='技能', max_length=50, verbose_name='技能标题')

    class Meta:
        verbose_name_plural = verbose_name = '关于'

    def avatar_admin(self):
        return format_html('', self.avatar, )

    avatar_admin.short_description = '头像预览'

'''
技能模型
'''

class Skill(models.Model):
    """关于页技能"""
    skill_name = models.CharField(max_length=50, verbose_name='方向名')
    skill_precent = models.CharField(default='%', max_length=50, verbose_name='百分比')

    class Meta:
        verbose_name_plural = verbose_name = '技能'

    # 后台图标预览
    def icon_data(self):
        return format_html('<h1><i class="{}"></i></h1>', self.social_icon)

    icon_data.short_description = '图标预览'

'''
站点页面配置数据表模型
'''

```

```
class Site(models.Model):
    """站点配置"""
    site_name = models.CharField(default='kylinBlog', max_length=30, verbose_name='网站名字')
    keywords = models.CharField(default='keyword', max_length=50, verbose_name='网站关键词')
    logo = models.URLField(default='https://jwt1399.top/favicon.png', max_length=100,
verbose_name='网站logo')
    desc = models.CharField(max_length=50, verbose_name='网站描述')
    slogan = models.CharField(max_length=50, verbose_name='网站标语')
    dynamic_slogan = models.CharField(max_length=50, verbose_name='动态标语')
    bg_cover = models.URLField(default='http://119.23.243.154/image/Covteam-hack.jpg',
max_length=100,
                                verbose_name='背景图片')

    class Meta:
        verbose_name = '网站设置'
        verbose_name_plural = verbose_name

    def logo_preview(self): # logo预览
        return format_html('', self.logo)

    logo_preview.short_description = 'logo预览'

    def bgcover_preview(self): # 背景图片预览
        return format_html('',
self.bg_cover)

    bgcover_preview.short_description = '首页背景预览'

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