Django框架基础-day03

Admin站点

0.使用步骤

使用Django的管理模块，需要按照如下步骤操作：

1. 管理界面本地化
2. 创建管理员
3. 注册模型类
4. 自定义管理页面

1.管理界面本地化

```
# LANGUAGE_CODE = 'en-us'
# 简体中文
LANGUAGE_CODE = 'zh-hans'

# TIME_ZONE = 'UTC'
# 亚洲上海时区
TIME_ZONE = 'Asia/Shanghai'
```

2.创建管理员

```
python manage.py createsuperuser
```

3.注册模型类

```
admin.py
admin.site.register(models.BookInfo)
```

4.定义模型类站点管理类

1.方式一

```
class BookInfoAdmin(admin.ModelAdmin):
    """图书模型类管理类"""
    pass
admin.site.register(models.BookInfo, BookInfoAdmin)
```

2.方式二

使用了装饰器之后，不用再调用admin.site.register()

```
@admin.register(models.HeroInfo)
class HeroInfoAdmin(admin.ModelAdmin):
    """英雄模型类管理类"""
    pass
```

5.调整列表页展示

BookInfoAdmin

```
# 默认每页显示的数据数量
list_per_page = 2

# 操作选项
actions_on_bottom = True

# 显示的列
list_display = ['id', 'btitle', 'format_pub_date']
```

将方法作为列

1.models.BookInfo类中

```
def format_pub_date(self):
    """将日期格式化后展示"""
    return datetime.strftime(self.bpub_date, '%Y-%m-%d')
format_pub_date.short_description = '发布日期' # 设置方法字段在admin中显示的标题
format_pub_date.admin_order_field = 'bpub_date' # 指定排序依据
```

2.将方法添加到要显示的列

```
# 显示的列
list_display = ['id', 'btitle', 'format_pub_date']
```

HeroInfoAdmin

```
# 右侧过滤栏
list_filter = ['hbook', 'hgender']

# 顶部搜索框
search_fields = ['hname']
```

关联对象

1.需求　　展示英雄时，将英雄所在的书的阅读量关联出来

2.在HeroInfo模型类中指定关联

```
#定义英雄模型类HeroInfo
class HeroInfo(models.Model):
    ......
    def read(self):
        """将该英雄所属于的书的阅读量关联进来"""
        return self.hbook.bread

    read.short_description = '阅读量'
    read.admin_order_field = 'hbook__bread' # 指定排序依据,外键属性用双下划
```

3.添加到列

```
@admin.register(models.HeroInfo)
class HeroInfoAdmin(admin.ModelAdmin):
    """英雄模型类管理类"""
    ......
    list_display = ['id', 'hname', 'hbook', 'read']
```

6.调整编辑页展示

```
# 允许展示编辑的字段
fields = ['btitle', 'bpub_date']

# 允许展示编辑的字段分组
fieldsets = (
    ('基本', {'fields': ['btitle', 'bpub_date']}),
    ('高级', {
        'fields': ['bread', 'bcomment'],
        'classes': ('collapse',)  # 是否折叠显示
    })
)
```

关联

1.准备关联的类

```
class HeroInfoStackInline(admin.StackedInline):
    model = models.HeroInfo  # 要编辑的对象
    extra = 1  # 附加编辑的数量
```

2.添加关联

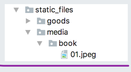
```
class BookInfoAdmin(admin.ModelAdmin):
    """图书模型类管理类"""
    ......

    # 关联
    inlines = [HeroInfoStackInline]
```

7.站点上传图片

1.安装图片处理模块

```
pip install Pillow
```

2.指定上传的文件存储位置




```
# 指定上传的文件存储路径
MEDIA_ROOT = os.path.join(BASE_DIR, 'static_files/media')
```

3.模型类准备image上传字段

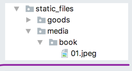
```
#定义图书模型类BookInfo
class BookInfo(models.Model):
    btitle = models.CharField(max_length=20, verbose_name='名称')
    bpub_date = models.DateField(verbose_name='发布日期')
    bread = models.IntegerField(default=0, verbose_name='阅读量')
    bcomment = models.IntegerField(default=0, verbose_name='评论量')
    is_delete = models.BooleanField(default=False, verbose_name='逻辑删除')
    image = models.ImageField(upload_to='book', verbose_name='图书图片', null=True)
```

4.迁移

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

5.image字段展示

```
class BookInfoAdmin(admin.ModelAdmin):
    """图书模型类管理类"""
    ......
    # 编辑页面
    # 允许展示编辑的字段分组
    fieldsets = (
        ('基本', {'fields': ['btitle', 'bpub_date', 'image']}),
        ('高级', {
            'fields': ['bread', 'bcomment'],
            'classes': ('collapse',)  # 是否折叠显示
        })
    )
```



6.点击上传结果

id	btitle	bpub_date	bread	bcomment	is_delete	image
1	射雕英雄传	1980-05-01	12	34	0	<null>
2	天龙八部	1986-07-24	36	40	0	<null>
3	笑傲江湖	1995-12-24	20	80	0	<null>
4	雪山飞狐	1987-11-11	58	24	0	<null>
5	西游记	1989-01-01	10	10	1	book/01.jpeg

8.调整站点全局页面信息

1.需求

- admin.site.site_header 设置网站页头
- admin.site.site_title 设置页面标题
- admin.site.index_title 设置首页标语

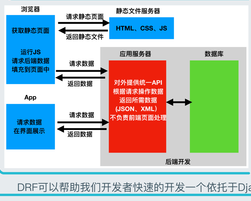
2.实现

```
# 设置admin网页信息
admin.site.site_header = '传智书城'
admin.site.site_title = '传智书城MIS'
admin.site.index_title = '欢迎使用传智书城MIS'
```

前后端分离介绍

1.前后端不分离图解

2.前后端分离图解



3.为什么要学习DRF　　DRF可以帮助我们开发者快速的开发一个依托于Django的前后端分离的项目

RESTfull 介绍

1.认识 RESTful

1.在前后端分离的应用模式里，后端API接口如何定义？

需要一种规范和风格来约束后端程序员对接口的定义

RESTful 就是用来约束后端程序员对接口的定义的一种风格

2.描述

REST，即**Representational State Transfer**的缩写。维基百科称其为"**具象状态传输**"，国内大部分人理解为"**表现层状态转化**"。

RESTful是一种开发理念。维基百科说：**REST是设计风格而不是标准**

RESTful架构就是：

- **每一个URL代表一种资源；**
- **客户端和服务器之间，传递这种资源的某种表现层；**
- **客户端通过四个HTTP动词，对服务器端资源进行操作，实现"表现层状态转化"。**

2.RESTful 设计方法

- 域名
- 路径
- 版本
- 过滤信息
- 状态码
- HTTP动词
- 错误处理
- 返回结果
- 超媒体
- 数据传输格式

3.Django和DRF对比

1.Django可以实现前后端分离

Django开发前后端分离的周期长

Django如果要遵守RESTful设计风格需要自己写对应风格的路由

2.DRF专门实现前后端分离

DRF开发前后端分离的周期短

默认遵守的是RESTful设计风格