Web 基础小实训

Web 基础小实训
1. 任务1：英雄人物管理 API 接口开发
2. 任务2：axios 请求获取所有英雄人物数据并操作

1. 任务1：英雄人物管理 API 接口开发

现有两个模型类如下：BookInfo 和 HeroInfo

```
class BookInfo(models.Model):
    """图书模型类"""
    btitle = models.CharField(max_length=20, verbose_name='标题')
    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='删除标记')

    class Meta:
        db_table = 'tb_books'
        verbose_name = '图书'

    def __str__(self):
        return self.btitle


class HeroInfo(models.Model):
    """英雄模型类"""
    hname = models.CharField(max_length=20, verbose_name='名称')
    # True：表示男  False：表示女
    hgender = models.BooleanField(default=False, verbose_name='性别')
    hcomment = models.CharField(max_length=200, null=True, verbose_name='备注')
    is_delete = models.BooleanField(default=False, verbose_name='删除标记')
    # 一对多关联属性
    hbook = models.ForeignKey('BookInfo', on_delete=models.CASCADE,
related_name='heros', verbose_name='所属图书')

    class Meta:
        db_table = 'tb_heros'
        verbose_name = '英雄'

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

需求说明：

1）创建一个名为：`web_app` 的 Django 项目，并创建一个名为：`booktest` 的项目子应用，将上面的两个模型类放到 `booktest` 子应用中，同时配置使用名为 `book_db` mysql 数据库，迁移生成数据库中的表，添加测试数据如下：

```
insert into tb_books(btitle,bpub_date,bread,bcomment,is_delete) values
('射雕英雄传','1980-5-1',12,34,0),
('天龙八部','1986-7-24',36,40,0),
('笑傲江湖','1995-12-24',20,80,0),
('雪山飞狐','1987-11-11',58,24,0);

insert into tb_heros(hname,hgender,hbook_id,hcomment,is_delete) values
('郭靖',1,1,'降龙十八掌',0),
('黄蓉',0,1,'打狗棍法',0),
('黄药师',1,1,'弹指神通',0),
('欧阳锋',1,1,'蛤蟆功',0),
('梅超风',0,1,'九阴白骨爪',0),
('乔峰',1,2,'降龙十八掌',0),
('段誉',1,2,'六脉神剑',0),
('虚竹',1,2,'天山六阳掌',0),
('王语嫣',0,2,'神仙姐姐',0),
('令狐冲',1,3,'独孤九剑',0),
('任盈盈',0,3,'弹琴',0),
('岳不群',1,3,'华山剑法',0),
('东方不败',0,3,'葵花宝典',0),
('胡斐',1,4,'胡家刀法',0),
('苗若兰',0,4,'黄衣',0),
('程灵素',0,4,'医术',0),
('袁紫衣',0,4,'六合拳',0);
```

2）根据下面提供的 API 接口文档，使用类视图来实现英雄人物管理的 5 个API接口，使用 postman 进行测试

```
# 接口1：获取所有的英雄人物数据
================================================================================
API: GET /heros/
================================================================================
请求参数：
    无
================================================================================
响应数据：
    {
      "code": "响应码",
      "message": "提示信息",
      "heros": [
        {
          "id": "英雄ID",
          "hname": "英雄名称",
```

```
            "hgender": "英雄性别",
            "hcomment": "英雄备注",
            "hbook": "所属图书名称"
        },
        ...
    ]
}
```

响应举例：

```
{
    "code": 0,
    "message": "OK",
    "heros": [
        {
            "id": "1",
            "hname": "郭靖",
            "hgender": true,
            "hcomment": "降龙十八掌",
            "hbook": "射雕英雄传"
        },
        {
            "id": "2",
            "hname": "黄蓉",
            "hgender": false,
            "hcomment": "打狗棍法",
            "hbook": "射雕英雄传"
        },
        ...
    ]
}
```

==========

接口2：新增一个英雄人物数据

==========

API: POST /heros/

==========

请求参数：

注：前端通过 json 传递参数

```
{
    "hname": "英雄名称",
    "hgender": "英雄性别",
    "hcomment": "英雄备注",
    "hbook_id": "所属图书id"
}
```

参数举例：

```
{
    "hname": "周伯通",
    "hgender": true,
    "hcomment": "双手互博",
    "hbook_id": 1
```

```
    }
================================================================================
响应数据：
    {
      "code": "响应码",
      "message": "提示信息",
      "hero": {
        "id": "英雄ID",
        "hgender": "英雄性别",
        "hname": "英雄名称",
        "hcomment": "英雄备注",
        "hbook": "所属图书名称"
      }
    }
响应举例：
    {
      "code": 0,
      "message": "OK",
      "hero": {
        "id": 18,
        "hname": "周伯通",
        "hgender": true,
        "hcomment": "双手互博",
        "hbook": "射雕英雄传"
      }
    }
================================================================================


# 接口3：获取指定的英雄人物数据(根据英雄ID)
================================================================================
API：GET /heros/英雄ID/
================================================================================
请求参数：
    注：前端通过 URL 地址传递指定英雄人物的ID
    例如：GET /heros/1/
================================================================================
响应数据：
    {
      "code": "响应码",
      "message": "提示信息",
      "hero": {
        "id": "英雄ID",
        "hgender": "英雄性别",
        "hname": "英雄名称",
        "hcomment": "英雄备注",
        "hbook": "所属图书名称"
      }
    }
```

响应举例：

```
    {
      "code": 0,
      "message": "OK",
      "hero": {
        "id": 1,
        "hname": "郭靖",
        "hgender": true,
        "hcomment": "降龙十八掌",
        "hbook": "射雕英雄传"
      }
    }
```

==========

接口4：修改指定的英雄人物数据(根据英雄ID)

==========

API：PUT /heros/英雄ID/

==========

请求参数：

注1：前端通过 URL 地址传递指定英雄人物的ID

例如：PUT /heros/18/

注2：前端同时通过 json 传递参数

```
    {
      "hname": "修改后英雄名称",
      "hgender": "修改后英雄性别",
      "hcomment": "修改后的英雄备注",
      "hbook_id": "修改后所属图书id"
    }
```

参数举例：

```
    {
      "hname": "周伯通106",
      "hgender": true,
      "hcomment": "七十二路空明拳",
      "hbook_id": 1
    }
```

==========

响应数据：

```
    {
      "code": "响应码",
      "message": "提示信息",
      "hero": {
        "id": "英雄ID",
        "hname": "英雄名称",
        "hgender": "英雄性别",
        "hcomment": "英雄备注",
        "hbook": "所属图书名称"
      }
    }
```

```
响应举例:
    {
      "code": 0,
      "message": "OK",
      "hero": {
        "id": 18,
        "hname": "周伯通106",
        "hgender": true,
        "hcomment": "七十二路空明拳",
        "hbook": "射雕英雄传"
      }
    }
================================================================================


# 接口5：删除指定的英雄人物数据(根据英雄ID)
================================================================================
API: DELETE /heros/英雄ID/
================================================================================
请求参数:
    注：前端通过 URL 地址传递指定英雄人物的ID
    例如：DELETE /heros/18/
================================================================================
响应数据:
    {
      "code": "响应码",
      "message": "提示信息"
    }
响应举例:
    {
      "code": 0,
      "message": "OK"
    }
================================================================================
```

2. 任务2：axios 请求获取所有英雄人物数据并操作

任务需求：

1）完善提供的 `01-获取所有图书数据.html`，在页面加载时使用 axios 请求获取所有英雄人物数据，并在 html 页面上进行展示。

2）点击 `删除` 链接则删除指定英雄人物的数据

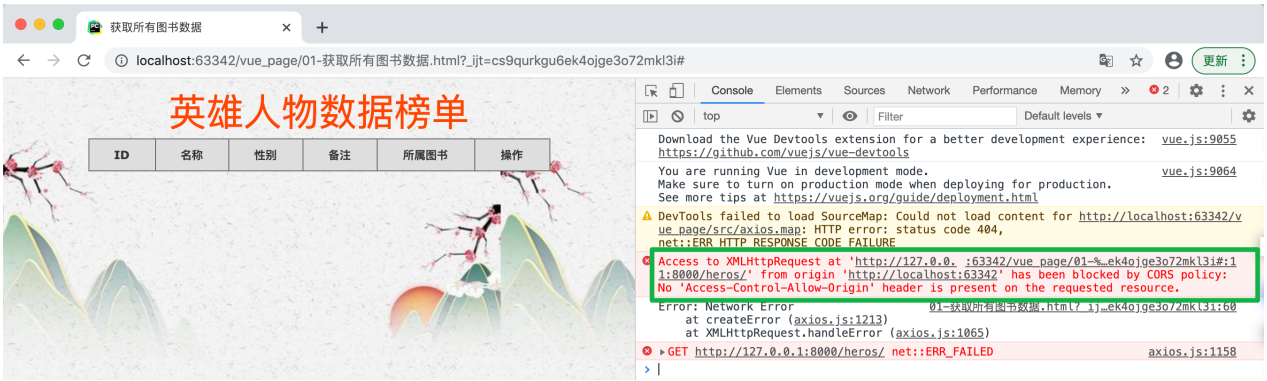
案例效果：

关键提示：

1）页面加载时，会自动调用 Vue 对象中的 mounted 钩子函数

```
<script>
    // 创建 Vue 对象
    var vm = new Vue({
        // 指定当前 Vue 对象控制的页面区域
        el: "#app",
        // mounted 钩子函数
        mounted: function() {
            // 页面加载时，会自动执行 mounted 中的代码
            ...
        },
        ...
    })
</script>
```

2）访问 html 页面时，会出现如下错误提示，解决方式如下(后续项目会讲原理)：



```
# ① 在自己的虚拟环境中安装 django-cors-headers 扩展包
pip install django-cors-headers

# ② 在 Django 项目的 INSTALLED_APPS 配置项中注册 corsheaders
INSTALLED_APPS = [
    ...
    'corsheaders',
]

# ③ 在 Django 项目的 MIDDLEWARE 配置项中添加中间件
MIDDLEWARE = [
    # 注：此中间件放在最上面
    'corsheaders.middleware.CorsMiddleware',
    ...
]

# ④ 在 Django 项目的 settings.py 中添加如下配置项：
CORS_ORIGIN_ALLOW_ALL = True
CORS_ALLOW_CREDENTIALS = True
```