Week 4

MyBatis-Plus是Springboot基础上的一个快速开发库。借助这个库我们能够快速完成一些互联网基础服务的开发。

实际上，新闻门户的基本功能难度不大，多数情况下只是对资源请求的响应而已。等到用户量级增大之后则需要考虑鉴权、高并发、分布式等问题。

在这里，需要用户学习MyBatis-Plus的使用。

https://baomidou.com/guide/

为了完成新闻门户的构建，需要理解MyBatis-Plus官方样例中的以下几条：

https://github.com/baomidou/mybatis-plus-samples

- mybatis-plus-sample-quickstart: 快速开始示例
- mybatis-plus-sample-quickstart-springmvc: 快速开始示例（Spring MVC版本）
- mybatis-plus-sample-reduce-springmvc: 简化掉默认mapper类示例（Spring MVC版本）
- mybatis-plus-sample-generator: 代码生成器示例
- mybatis-plus-sample-crud: 完整 CRUD 示例
- mybatis-plus-sample-wrapper: 条件构造器示例
- mybatis-plus-sample-pagination: 分页功能示例
- mybatis-plus-sample-active-record: ActiveRecord示例
- mybatis-plus-sample-sequence: Sequence示例
- mybatis-plus-sample-execution-analysis: Sql执行分析示例

任务

我们的任务是纯后端，所以在调试的时候感觉会不那么直观、可视化，强烈推荐Postman工具。

借助Springboot和MyBatis-Plus搭建一个**资讯门户后端**，要求使用RESTful规范进行通信，使用Postman测试。

包含以下模块对应的CRUD操作：

- 新闻文章管理：

  新闻列表的获取、单条新闻内容的获取、新闻的上传和修改

- 新闻评论管理：

  获取评论列表、发表评论、删除评论

- 用户管理（基础版）：

  用户注册、用户个人信息修改

- 广告管理：

  广告上传、特定类型的广告获取、广告点击反馈。注意，评论内广告和新闻内嵌广告这两种类型的广告，在上述"新闻文章"接口的返回内容中直接包含。

> 以下是一个参考，一个音乐门户的接口设计方案：

```
<GET>       /album                  →获取数据库中的所有album列表
<GET>       /album/<pid>            →获取单个album的信息
<PUT>       /album/<pid>            →将专辑添加到用户收藏中
<DELETE>    /album                  →删除用户收藏的专辑

<GET>       /playlist               →获取数据库的所有playlist列表
<GET>       /playlist/<pid>         →获取单个playlist的信息
<POST>      /playlist               →创建播放列表
<PUT>       /playlist/<pid>         →将播放列表添加到用户收藏中
<DELETE>    /playlist/<pid>         →删除用户收藏的播放列表

<GET>       /track                  →获取数据库的所有歌曲列表
<GET>       /track/<track_id>       →获取单条歌曲信息
<PUT>       /track/<track_id>       →将歌曲添加到用户收藏中
<DELETE>    /track/<track_id>       →删除用户收藏的歌曲

// User用户信息
<GET>       /user/login             →登陆，获取用户的jwt token（这一部分先跳过）
<POST>      /user/login             →注册
<DELETE>    /user/login             →登出
<GET>       /user/<uid>             →获取用户统计信息
<POST>      /user/<uid>             →更新用户的统计信息

// image图片信息
<GET>       /image/<album_id>       →获取专辑封面图片，返回文件
```

`<GET>  /album`

`<GET>  /playlist`

`<GET>  /track`

后端参考的返回格式

成功

获取列表

```
{
    "data": [],
    "total": 0 // 条目数量
}
```

获取单个条目

```
{
    "data":{

    }
}
```

失败

```
{
    "err": 201,
    "msg": "Resource doesn't exist"
}
```

在获取列表时，data中包含的数据视显示内容需求而定。最基本的，需要有id、封面图、标题和description。

REST API的一种请求payload形式如图所示：

> JSON Server
>
> This Data Provider fits REST APIs powered by JSON Server , such as JSONPlaceholder .
>
> | Method | API calls |
> | --- | --- |
> | `getList` | GET `/posts?_sort=title&_order=ASC&_start=0&_end=24&title=bar&_embed=comments&_expand=user` |
> | `getOne` | GET `/posts/123` |
> | `getMany` | GET `/posts?id=123&id=456&id=789` |
> | `create` | POST `/posts/123` |
> | `update` | PUT `/posts/123` |
> | `updateMany` | Multiple calls to PUT `/posts/{id}` |
> | `delete` | DELETE `/posts/123` |
> | `deleteMany` | Multiple calls to DELETE `/posts/{id}` |
>
> 在后续的API开发中，必然会遇到筛选和排序问题，我们将会按照类似JSONPlaceholder这种数据请求格式去完成前后端query的资源对接。

本任务要求最后实现的后端系统，能够完成获取新闻列表、按类型索引获取新闻列表以供首页渲染，获取新闻内容、获取新闻评论内容、发布评论、上传新闻；获取开屏广告、首页弹框广告这几个核心功能对应的接口。