仿抖音后端项目
By wangguo

- **技术选型**
  - Redis
    - 在计数/缓存/过期策略等场景的使用
  - Mybatis
    - 完成数据库相关操作-mapper层
      - 包括通过tk-mapper反向代理生成的，以及自定义的映射代码（xml文件中参数类型parameterType是map类型，返回类型是VO实体类（查出来的内容与实体类对应））
  - Lombok
  - Spring Boot
  - RabbitMQ
    - 消息中间件，用于组件之间得解耦
      - 对非重要信息的存储与重要信息的存储异步解耦防止非重要信息的存储失败对重要信息的保存操作进行回滚。
  - Nacos
  - Minio
    - 云存储：用于保存用户的头像，信息页背景。而把图片存放的地址保存到数据库中
  - Maven
    - 软件项目管理以及自动构建工具：完成项目管理以及对项目依赖进行管理
  - knife4j
    - 前后端分离接口管理工具
      - GET
      - POST
      - DELETE
  - org.n3r.idworker
    - 分布式全局主键id生成

- PageHelper
  - 分页工具
- MySQL

项目中用到的BO

- RegistLoginBO
  - 注册登录BO（主要包含手机号、验证码）
- VlogBO
  - 前端传送的视频发布BO参数信息，用于发布视频，包含用户id，视频url，视频封面，视频标题，视频此尺寸信息
- CommentBO
  - 用于包装前端发送评论的时候包装的对象（业务对象），包括：被评论的视频的博主id/这条评论有没有父评论/vlog的id/发表评论的用户id/评论内容。
- UpdatedUserBO
  - 更新用户信息的时候包装用户更新信息的业务对象

项目中用到的VO

- IndexVlogVO
  - 主要用于返回给前端的展示每一个视频页面的信息
- CommentVO
  - 区别于POJO/Comment，返回给前端的时候还需要显示评论者的头像/昵称等这些在Comment表中没有定义的信息。
- FansVO
  - 用于展示粉丝列表的视图对象：包括粉丝id/粉丝昵称/粉丝头像/是否和粉丝是朋友关系
- UsersVO
  - 用于展示登录用户的信息：包括用户主键id/用户手机号/用户昵称/用户idouyin号/用户头像/用户性别/用户生日/国家/省份/城市/地区/个人描述/背景图片/Idouyin号能否修改/用户注册时间/用户信息更新时间/用户Token信息/关注的数量/粉丝数量/所有点赞我的视频的总和。
- VlogerVO
  - 用于展示我已经关注的人的视图对象：包括：视频主id/视频主昵称/视频主头像/我是否关注他。

Redis在项目中的应用

- 缓存验证码（设置15分钟的过期策略）
- 缓存用户Token会话信息（
- 计数用户的关注数
- 计数用户的粉丝数
- 计数用户获赞总数
- 在把用户喜欢一个视频的信息保存到数据库的时候，缓存用户是否点赞一个视频（REDIS_USER_LIKE_VLOG)，1表示攒点了，0表示没有点赞
- 缓存计数视频被点赞的个数REDIS_VLOG_BE_LIKED_COUNTS，当点赞视频数量达到一个动态阈值的时候，再把最新的点赞个数保存到数据库中。能够减轻数据库的压力。
- 缓存某个用户与另外一个用户（视频发布者）的关系（朋友/非朋友）

控制层

- passport
  - 主要实现对登录/注册相关业务的拦截处理的控制层。包括登录/注册/获取验证码等。
- userInfo
  - 主要是用户信息相关的模块的接口，包括查询用户信息、修改用户信息、修改头像/背景等。
- vlog
  - 短视频相关业务功能的控制层，主要包括：发布视频、视频列表展示、获取视频详情、私密/公开设置、展示公开列表、展示私密列表、点赞视频、取消点赞视频、获取我的点赞列表、获取我的关注列表、获取朋友列表等。
- fans
  - 粉丝相关业务的控制层，主要包括：关注用户/取消关注/查询我关注的博主列表/查询我是否关注某个博主/查询我的粉丝等。
- comment
  - 评论相关业务的控制层，主要包括：创建评论/返回评论的总数/分页查询评论的列表/点赞评论/取消点赞评论/删除评论等。
- msg
  - 系统消息业务的控制层，主要是展示出所有的系统消息（包括点赞消息/关注消息/评论消息/回复消息）

项目结构

- API-控制层

- 也可以叫做Controller层，主要是一些和前端交互的请求和响应控制，接受前端的请求，调用service层，完成相应业务并接收service层返回的数据，并回复给前端进行渲染

- Service-业务逻辑层
  - 主要是完成业务功能设计，比如：点赞业务/评论业务/视频业务等，通过调用Mapper层接口，接收Mapper层返回的数据
- Model-数据模型层/实体层
  - 也可以叫做Entity层/POJO层
    - 是对数据库表中实体的抽象，同时也可以分成VO, MO等。VO是返回给前端的视图对象，MO是接收前端传送的信息的一种对象。
- Mapper-数据映射层/持久层
  - 也可以叫做Dao层
    - 主要是完成访问数据库，向数据库发送SQL语句，完成数据的交互操作
- Common-公共层
  - 主要完成一些基本的处理
    - 枚举类型的定义：性别枚举/消息类型枚举/用户信息修改类型等
    - 异常处理类
    - 对返回结果的封装类等
      - 统一对结果进行封装，在需要用的时候就不需要重复写代码。包括：结果状态/提示信息/成功与否/以及Object类型的数据封装。
    - 工具类Utils，不用重复造轮子
      - 微信短信验证码工具
      - JSON与String类型转换工具
      - 获取用户IP地址的类
      - 日期处理工具类
      - 脱敏工具类
      - Redis操作包装类
      - 分页结果类
      - MInio工具类

- **业务逻辑Service层**
  - UserService

- 处理与用户相关的业务逻辑

- FansService
  - 处理与粉丝相关的业务逻辑

数据库表的设计

- 视频Vlog表
  - 主键id，视频id，发布视频者id，视频的url，视频封面地址，视频标题，视频的尺寸信息，视频的点赞数，视频的评论个数，视频的公开/私密标签，视频创建视频，视频更新信息等。
- 用户喜欢视频表my_liked_vlog
  - 主键id，用户id，用户喜欢的视频id
- 用户users表
  - 主键id，电话，昵称，i抖音号，头像，性别，生日，国家，省份，城市，地区，个性签名，背景图片地址，i抖音号是否修改过，创建时间，用户信息更新时间等。
- 评论commnet表
  - 主键id，被评论视频的发布者id，该评论的父评论（默认为0表示是一级评论，否则为回复评论），视频id，发表评论的用户id，评论内容，评论的点赞个数，评论发布时间等。
- 粉丝fans表
  - 主键id，被关注的人id，关注人id，是否是朋友（互关）

配置类

- InterceptorConfig配置类
- Knife4jConfig配置类
  - 主要是对接口文档Knife4j进行配置的类，需要用@EnableSwagger2WebMvc注解，并指定需要拦截的Controller扫描包的路径。
    .apis(RequestHandlerSelectors.basePackage("com.wangguo.controller"))
- MinIOConfig配置类
  - 配置endpoint,filehost,bucketName,账号，密码，图片，文件尺寸等
- RabbitMQConfig配置类
  - 主要完成定义交换机/定义队列/创建交换机/创建队列/队列和交换机的绑定

工具类

- MinIOUtils工具类

- BaseInfoProperties
  - 设置redis的操作器以及在redis中会用到的存放字段的定义以及一些返回会用到的其他字段。以及分页相关的一些操作。

项目中用到的MO

- MessageMO消息实体
  - MO表示Mongodb Object的简写
  - 主要包括系统消息的一系列属性：消息主键id/消息发送方id/消息发送方昵称/消息发送方头像/消息接收方id/消息类型/消息内容/消息创建时间。

主要的优化总结

- 借助RabbitMQ对重要信息/非重要信息进行解耦。提高系统的可用性和效率。
- 借助Redis，缓存一些不必须马上存放入数据库的信息/频繁访问的数据，同时借助Redis的过期策略，处理有时效性的数据。同时借助Redis对需要频繁增/减的数据进行保存，在一定阈值的时候再持久化到数据库中。通过动态调整保存阈值进而减少对数据库的频繁访问。提升用户的体验。
- 借助nacos分布式配置中心，对一些需要配置的参数提取出来，便于动态修改。