一 介绍

作为项目基础夹包各业务服务工程需引入 项目依赖lombok插件，需安装lombok

|— vap-utils  
 |—src  
 |—main  
 |—java  
 |—com.vrv.vap  
 |— base ---mybatis，hibernate，jpa数据层封装

|—BaseMapper mybatis Mapper

|—BaseRepository jpa公用Repository

|—BaseService mybatis service

|—common ---工具，异常处理，验证

|—config ---配置

|—bean bean的初始化如{RestTemplate}

|—datasource 数据源初始化

|—feign feign 初始化重写

|—orm 初始化jpa,hibernate

|—swagger 在线文档初始化配置

|—-ExceptionControllerAdvice 初始化异常拦截

|—MapperScanConfig 初始化扫描mybatiS mapper

|—RedisConfig 初始化 Redis

|—TransactionConfig 初始化事物

----Jpa

public <T> Page<T> queryPageBySQL(StringBuffer sql,Object clazz, Pageable pageable, Sort sort,Class<T> resultClass );  
public <T> Page<T> queryPageBySQL(StringBuffer sql, Pageable pageable, Map<String, Object> params, Class<T> resultClass);  
public Page<Map<String,Object>> queryPageBySQL(StringBuffer sql ,Map<String, Object> params, Pageable pageable);  
public int queryDataCountBySQL(StringBuffer sql,Object params);  
public <T> T findByJpql(String jpql,Object... objects);

findAll(criteria)

|  |
| --- |
| Criteria<User>criteria= new Criteria<>(); |
| criteria.add(QueryCondition.eq("uname", "2"));  多条件  QueryCondition{  isNotEmpty,isEmpty,isNull,isNotNull,eq.ne,like,gt,lt,lte,gte,or  ,between,in,notIn  }  //criteria.add(QueryCondition.eq("age", "2")); |  |
| List<User> userList = userRepository.findAll(criteria); |  |

二 开发建议

请遵循项目规范  
 - 映射数据库对象（DO) 放在业务服务项目包目录下 比如com.vrv.vap.业务服务名.model  
 Model内成员变量建议与表字段数量对应，  
 - 表名，建议使用小写，多个单词使用下划线拼接  
 -数据传输层对象（DTO,VO,BO）如需扩展成员变量（比如连表查询）建议创建DTO，  
 -如果数据层使用mybatis可以使用代码生成器（vap-generator）使用方式vap-generator项目下的md文件已详细说明  
 - mybatis使用Mapper插件，详情见通用Mapper插件文档说明  
 - 大量工具都在eglsc-helper中  
 对象拷贝工具BeanByRefMapper  
 -如果数据层使用jpa 请遵循jpa规范，使用spring-data-jpa，映射数据库对象（DO)model 在实体类上增加@Entity  
 @Table,业务数据层包规范om.vrv.vap.业务服务名.repository.dao  
 dao层接口需继承BaseRepository，  
 - 如果数据层使用hibernate 请遵循规范   
 建议使用注解对应数据库映射  
 如果数据库映射使用XML，请把生成的\*.hbm.xml 放在业务服务resources下的mapping下  
-hibernate,jpa，尽量不要建外键关联，建议不要用关联关系映射，会增加复杂度  
- 增加easyjdbc组件 ，支持部分常用的JPA注解，使得经过注解的实体可以像Hibernate,jpa一样进行增、删、改和获取。SQL构造工具、链式API等让查询操作更为灵活，请遵循规范组件规范 详情见通用easyjdbc插件文档说明（https://github.com/xphsc/easyjdbc-spring-boot-starter）

- 增加多数据源组件dynamic-datasource-spring-boot-starter  
 通过注解方式的转换数据源@DynamicDataSource("slave")  
 建议注解在service层 方法级别大于类级别

- 建议在公司内部使用SpringFox-Swagger2开源项目来编写、管理API文档  
 目前已集成Swagger

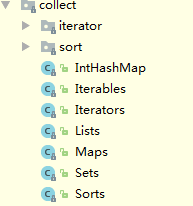
三 xtool 工具



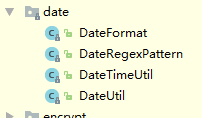
对象拷贝



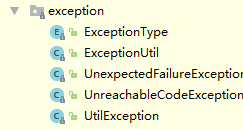
集合操作



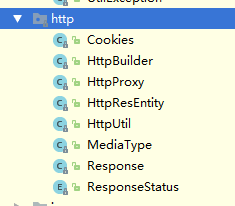
日期操作

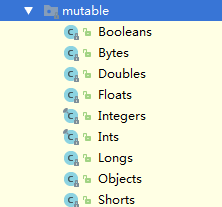


异常操作工具

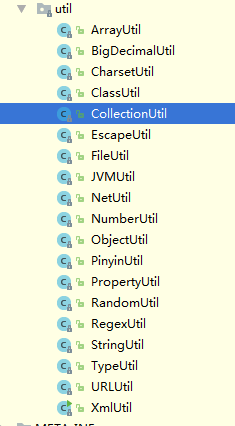


HTTP





常用工具



***多数据源配置****~~~  
spring:  
 datasource:  
 datasource-name: default  
 url: jdbc:mysql://localhost:3306/target\_datasource?useUnicode=true&characterEncoding=utf-8&useSSL=false  
 username: root  
 password: root  
 datasources:  
 - datasource-name: slave  
 url: jdbc:mysql://localhost:3306/target\_datasource1?useUnicode=true&characterEncoding=utf-8&useSSL=false  
 username: root  
 password: root  
 - datasource-name: master  
 url: jdbc:mysql://localhost:3306/target\_datasource2?useUnicode=true&characterEncoding=utf-8&useSSL=false  
 username: root  
 password: root  
~~~~~*