## 修订记录

|  |  |  |  |
| --- | --- | --- | --- |
| 修订内容 | 版本 | 修订人 | 修订时间 |
| 创建文档 | v1.0 | 叶耀雄 | 2017/8/18 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 1.项目信息

## 1.1模块划分

rst-common

底层核心实现、公共服务接口、工具类等

rst-common-web

* 基础模块，单点rbac、dict、area等系统功能
* 依赖rst-common

rst-common-remote

以Restful API（HTTP+JSON）供其他项目调用

* 基础服务。如地区数据（area）、数字字典（dict）
* 依赖rst-common

rst-mall 商城

* 依赖rst-common

rst-agent

代理商、粉丝以及任务管理

* 代理商，粉丝，提成，提现，分销任务、推广任务、广告任务。
* 依赖rst-common、

rst-wechat 微信接入。依赖rst-common、rst-common-support

## 1.2 项目规划

初步阶段：

不适用分布式服务（如dubbo等），依赖项目直接引入jar。没有依赖的项目使用HTTP Restful API相互调用（交互）。

其他阶段：可能出现的改造

* 按服务调用，引入dubbo类似的分布式治理框架
* 消息队列

## 1.3技术选型

### 1.3.1后台

基于Spring boot 1.5.6

\* 核心框架：Spring Framework 4.0

\* 安全框架：Apache Shiro 1.2

\* 视图框架：Spring MVC 4.3.0

\* 服务端验证：Hibernate Validator 5.1

\* 任务调度：Spring Task 4.0

\* 持久层框架：MyBatis 3.4.0

\* 数据库连接池：Alibaba Druid 1.0

\* 缓存框架：Ehcache 2.6、Redis

\* 日志管理：SLF4J 1.7、Logback1.1.11

\* 工具类：Apache Commons、Jackson 2.2、Xstream 1.4、Dozer 5.3、POI 3.9

### 1.3.2 前端

\* 模板框架：thymeleaf2.1.5

\* JS框架：JQuery 1.10。

\* 客户端验证：JQuery Validation Plugin 1.11。

\* 富文本：ueditor

\* 文件管理：CKFinder

\* 树结构控件：jQuery zTree

\* 日期控件： My97DatePicker

### 1.3.3数据库

使用mysql，分mall、agent，support，wechat四个库（逻辑上分库，有需求才实际分库）。

**数据库表命名规定**

* rstmall：存储商城信息，以ml开头，ml\_xxxx，如ml\_goods\_info
* rstagent:代理商、任务、粉丝信息，以ta开头，ta\_xxxx， 如ta\_fans\_info
* rstsys:存储系统及基础信息，以sys开头，sys\_xxxx。如sys\_dict
* rstwechat: 微信接入相关，以wc开头， wc\_xxxx。如wc\_pay\_inf

# 2. 项目服务

## 2.1 文件服务

## 2.2 短信服务

# 3. 项目规范

## 3.1 实现Demo

用系统字典模块作简要说明

### 3.1.2 Java

* Entity：实体类，通常继承DataEntity，如果是树形可继承TreeEntity

*/\*\*  
 \* 系统数据字典Entity  
 \* Created by yeyx on 2017/8/30.  
 \*/***public class** Dict **extends** DataEntity<Dict> {  
  
 **private static final long *serialVersionUID*** = 1L;  
 **private** String **value**; *// 数据值* **private** String **label**; *// 标签名* **private** String **type**; *// 类型* **private** String **description**;*// 描述* **private** Integer **sort**; *// 排序*

* DAO：继承CrudDao满足增删该查（分页）等需求，必须用@Mapper注解。还可以继承TreeDao以实现树形需求

*/\*\*  
 \* 字典DAO接口  
 \* Created by yeyx on 2017/8/30.  
 \*/*@Mapper  
**public interface** DictDao **extends** CrudDao<Dict> {  
 **public** List<String> findTypeList(Dict dict);

//public long getTotal();//获取总数

}

* mapper：放置在各项目的src/main/resources/mybatis/\*\*中，请认真读以下批注信息。

*<?***xml version="1.0" encoding="UTF-8"** *?>***<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">**<**mapper namespace="com.rstang.support.sys.dao.DictDao"**>  
   
 <**select id="get" resultType="Dict"**>  
 SELECT   
 \*   
 FROM sys\_dict   
 WHERE id = #{id}  
 </**select**>  
   
 <**select id="findList" resultType="Dict"**>  
 SELECT   
 \*   
 FROM sys\_dict   
 WHERE del\_flag = #{DEL\_FLAG\_NORMAL}   
 <**if test="type != null and type != ''"**>  
 AND type = #{type}  
 </**if**>  
 <**if test="description != null and description != ''"**>  
 AND description LIKE CONCAT('%', #{description}, '%')  
 </**if**>  
 ORDER BY type, sort, update\_date DESC  
 </**select**>  
   
 <**select id="findAllList" resultType="Dict"**>  
 SELECT   
 \*   
 FROM sys\_dict   
 WHERE del\_flag = #{DEL\_FLAG\_NORMAL}   
 ORDER BY type, sort, update\_date DESC  
 </**select**>  
   
 <**select id="findTypeList" resultType="string"**>  
 SELECT   
 type   
 FROM sys\_dict   
 WHERE del\_flag = #{DEL\_FLAG\_NORMAL}   
 GROUP BY type  
 ORDER BY type  
 </**select**>  
   
 <**insert id="insert"**>  
 INSERT INTO sys\_dict(  
 id,   
 value,   
 label,   
 type,   
 description,   
 sort,   
 create\_by,   
 create\_date,   
 update\_by,   
 update\_date,   
 remarks,   
 del\_flag  
 ) VALUES (  
 #{id},   
 #{value},   
 #{label},   
 #{type},   
 #{description},   
 #{sort},   
 #{createBy.id},   
 #{createDate},   
 #{updateBy.id},   
 #{updateDate},   
 #{remarks},   
 #{delFlag}  
 )  
 </**insert**>  
   
 <**update id="update"**>  
 UPDATE sys\_dict SET   
 value = #{value},   
 label = #{label},   
 type = #{type},   
 description = #{description},   
 sort = #{sort},   
 update\_by = #{updateBy.id},   
 update\_date = #{updateDate},   
 remarks = #{remarks}  
 WHERE id = #{id}  
 </**update**>   
 <**update id="delete"**>  
 UPDATE sys\_dict SET   
 del\_flag = #{DEL\_FLAG\_DELETE}  
 WHERE id = #{id}  
 </**update**>   
</**mapper**>

* service：继承CrudService，TreeService。如果涉及多个表操作的事务需引入其它表的DAO在service层作事务处理。

*/\*\*  
 \* 数据字典服务类  
 \* Created by yeyx on 2017/8/30.  
 \*/*@Service  
@Transactional(readOnly = **true**)  
**public class** DictService **extends** CrudService<DictDao, Dict> {  
   
  
 @Transactional(readOnly = **false**)  
 **public void** save(Dict dict) {  
 **super**.save(dict);  
  
 }  
  
 @Transactional(readOnly = **false**)  
 **public void** delete(Dict dict) {  
 **super**.delete(dict);  
  
 }  
}

### 3.1.2 数据库

每一张表必需使用id字段作为主键，作为外键时需要加上标识明确标识外键的来源。

如：sys\_user 系统用户表sys\_user.id为主键，作为外键时标识为user\_id

主键生成规范请参考[3.2 主键规范](#_3.2_主键规范)

需补充表格

### 3.1.3 注意事项

* 通过设置entity bean的isNewRecord属性来手动传入ID。

将setIsNewRecord(true)，需手动手动传入ID后新增记录。

将setIsNewRecord(false)，默认为false；没有ID框架自动生成后新增记录，有ID则更新记录。

Dict dict = **new** Dict();  
dict.setId(IDCreator.*getInstance*().getID(**"sys.dict"**));  
dict.setLabel(**"归属公司"**);  
dict.setIsNewRecord(**true**);*//将setIsNewRecord(true)，手动传入ID：IDCreator.getInstance().getID("sys.dict")***dictService**.save(dict);

* 更新时无需设置

Dict dict = **new** Dict();  
dict.setId(**"S0217090716315601"**);  
*//dict.setIsNewRecord(true); 更新时默认即可，无需设置isNewRecord***dictService**.save(dict);//此为更新操作

## 3.2 主键规范

使用以下规范生成主键，目的：

1. 保证多个数据唯一
2. 看到ID就知道属于哪个数据库、哪个功能点的业务数据。

规则及说明如下：

*##id生成器，需要结合id-rules.properties文件使用  
##用法：  
##1.在id-rules.properties中配置:  
## OPEN\_LC=LC{0}[DATE2-4][NUM5]  
##2.调用函数：  
## String id = IDCreator.getInstance().getID(String key, String value);<每次递增>  
## 注：key 取 OPEN\_LC；  
## value对应{\*}参数，多个参数value以","分隔,如有value="a,b",则{0}=a,{1}=b；  
## userUID 用户内部ID号  
##其中：  
## A、[DATE2-4] 表示DATE为： yyyyMMddHHmmss格式取2到4位的字符串，即yy;  
## B、[NUM5]：表示5位数字；  
## 注意：key 的字符长度最大【10】位;  
  
#商城规则：1.以大写M开头，模块以编码标示；2.订单特殊：8开头自营、6第三方商品; 3. 其它自定义，唯一即可  
#模块规则：商品（80、60） 类别：01***mall.goods.self**=**M80[DATE02-14]  
mall.goods.3rd**=**M60[DATE02-14]  
mall.type**=**M01[DATE02-14]  
mall.order.self**=**8[yyDDDHHmmss]  
mall.order.3rd**=**6[yyDDDHHmmss]***#任务规则：1.以大写T开头，模块以编码标示 2.自定义，唯一即可  
#模块规则：广告任务00、推广任务01、商品分销任务02***task.ad**=**T00  
task.expand**=**T01  
task.goods**=**TO2***#代理商规则：1.以大写A开头，模块以编码标示 2.自定义，唯一即可  
#模块规则：代理商00、粉丝02***agent.mgr**=**A00  
agent.fans**=  
**agent.grow**=  
  
*#系统模块规则：1. 以大写S开头，模块以编码标示 2.自定义，唯一即可  
#模块规则：用户管理00、菜单管理01、数据字典02***sys.user**=  
**sys.menu**=  
**sys.dict**=**S02[DATE2-14][NUM2]  
sys.role**=  
**sys.user.role**=  
**sys.app**=  
  
*#微信规则：1. 以大写W开头，模块以编码标示 2.自定义，唯一即可  
#模块规则：微信菜单00、微信支付01***wechat.menu**=  
**wechat.pay**=  
  
*#共用规则：1.按类型获取，img图片、txt文本、doc办公文档***file.img.name**=**img  
file.txt.name**=**txt  
file.doc.name**=**doc**