个人领取商家红包简单设计

## 需求描述

某个商家提供1亿资金用于普通个人红包领取，普通个人领取红包后记录流水日志，一个用户只能领取一份，领完为止，个人领取金额随机。每人红包金额大小可以限制,比如最多一个人100元红包.

## 软件架构

接入处理：负责解析请求信息并映射到内部实体

鉴权认证：对用户有效性进行校验

策略控制：实现申请有效性，恶意申请，分配策略等  
 资金扣减：从缓存和物理库同时对红包进行扣减

红包产生：产生红包信息以及流水信息,并对信息进行缓存.

记录流水：将流水信息记录到物理库

红包输出：按照展现的要求输出红包信息

软件架构示意图，仅供参考：



上图中，实现说明如下：

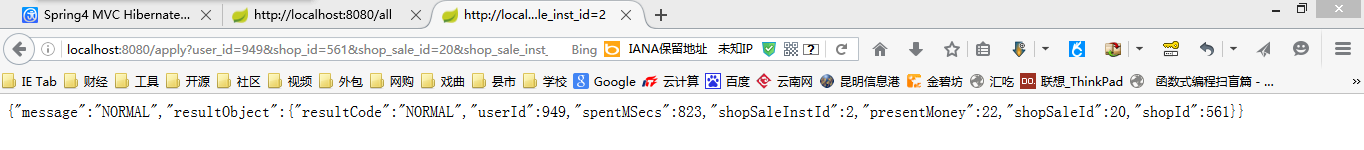
1. 缓存采用HashTable私有内存实现，且暂不考虑缓存数据周期,多线程环境需要加锁实现,可以考虑放入KV\_MDB,如redis等,可以避免锁机制,提高效率
2. 鉴权认证简单实现用户标识的校验.
3. 暂不多加log4j日志信息和配置

## 软件环境

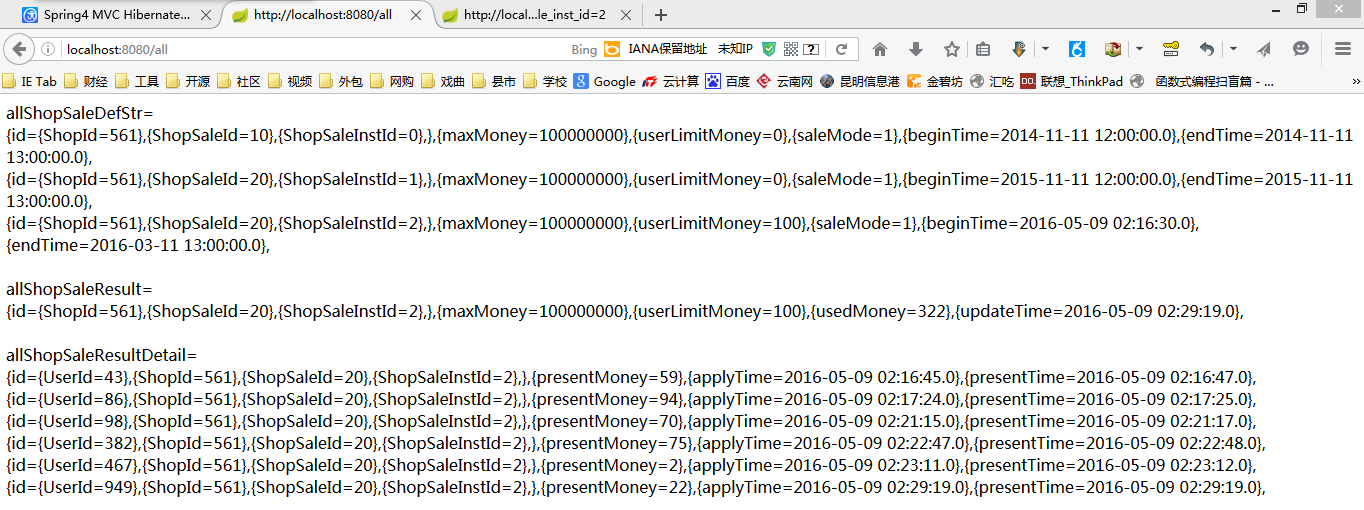
* JDK1.8
* Eclipse4.5+Maven3.7
* Spring4.2.6
* Hibernate4
* Mysql5.5
* Mysql-Connector-java-5.1.38

## 软件结果样例

* 用户红包申请和结果信息



* 商户营销活动信息



## 实体设计

### 物理库实体

* shop\_sale\_def商户活动定义:一个商户可以进行多个红包活动,预先运维配置,为了简化,不再单独定义商户和商户营销活动信息.

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 中文名 | 类型 | 注释 |
| shop\_id | 商户标识 | number(10) | 关键字 |
| shop\_sale\_id | 商户营销活动标识 | number(10) | 一个商户可以有多个活动,关键字 |
| shop\_sale\_inst\_id | 活动实例 | number(10) | 一个活动可以举行多次,关键字 |
| max\_money | 最大金额 | number(10) | 即1亿 |
| user\_limit\_money | 单个用户最大限额 | number(10) | 可以指定个人最大获得的红包金额 |
| sale\_mode | 营销活动模式 | number(1) | 1=随机(默认);0=平均 |
| begin\_time | 活动开始时间 | char(14) | YYYYMMDDHHMISS |
| end\_time | 活动结束时间 | char(14) | YYYYMMDDHHMISS |

* shop\_sale\_result商户活动结果信息:每个商户红包的结果信息.

实际红包分配在该表中进行和记录,也用于查询

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 中文名 | 类型 | 注释 |
| shop\_id | 商户标识 | number(10) | 关键字 |
| shop\_sale\_id | 商户营销活动标识 | number(10) | 一个商户可以有多个活动, 关键字 |
| shop\_sale\_inst\_id | 活动实例 | number(10) | 一个活动可以举行多次,关键字 |
| max\_money | 最大金额 | number(10) | 即1亿 |
| used\_money | 已分配金额 | number(10) | >=0 |
| update\_time | 更新时间 | char(14) | YYYYMMDDHHMISS |

* shop\_sale\_result\_detail商户活动明细信息(即流水日志,可用于稽核),不记录红包为0的信息

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 中文名 | 类型 | 注释 |
| user\_id | 用户标识 | number(10) | 关键字 |
| shop\_id | 商户标识 | number(10) | 关键字 |
| shop\_sale\_id | 商户营销活动标识 | number(10) | 关键字 |
| shop\_sale\_inst\_id | 活动实例 | number(10) | 一个活动实例只能参加1次,关键字 |
| present\_money | 分配的金额 | number(10) |  |
| present\_time | 分配的时间 | char(14) | YYYYMMDDHHMISS |
| apply\_time | 申请时间 | char(14) | YYYYMMDDHHMISS,可用于核查分配时延 |

### HTTP接口

* 红包申请接口

[http://localhost:8080/apply?user\_id=](http://localhost:8080/apply?user_id=392039)43678&shop\_id=561&shop\_sale\_id=20&shop\_sale\_inst\_id=2

上述的userid为用户标识，用于标示唯一的个人

* 商家营销活动信息打印接口

<http://localhost:8080/all>

### 请求缓存

RequestCache,全局单例,主要用于用户重复发起请求的过滤,恶意请求的鉴别,快速返回上次返回的红包信息,申请成功(present\_money>0)就更新缓存

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 中文名 | 类型 | 注释 |
| user\_id | 用户标识 | number(10) | 关键字 |
| shop\_id | 商户标识 | number(10) | 关键字 |
| shop\_sale\_id | 商户营销活动标识 | number(10) | 关键字 |
| shop\_sale\_inst\_id | 活动实例 | number(10) | 一个活动只能参加一次,关键字 |
| present\_money | 分配的金额 | number(10) |  |
| present\_time | 分配的时间 | char(14) | YYYYMMDDHHMISS |
| apply\_count | 申请次数 | number(10) | 可用于计算点击频度,进而进行策略控制 |

### 资金缓存

ShopMoneyCache全局单例,主要用于:

1. 提升资金查询性能
2. 第一次获取并留存此次活动红包最大金额.如果max\_money<=used\_money,即可以立即返回’红包已经抢完’
3. 已分配金额可以通过一定策略进行同步控制(比如每20秒钟就从物理库刷取最新值一次),在服务中单独配置20秒
4. 如果是仅仅查询剩余红包金额,可以粗略返回max\_money-used\_money的差值,大幅减少数据库压力
5. 申请成功(present\_money>0)就更新缓存

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 中文名 | 类型 | 注释 |
| shop\_id | 商户标识 | number(10) | 关键字 |
| shop\_sale\_id | 商户营销活动标识 | number(10) | 关键字 |
| shop\_sale\_inst\_id | 活动实例 | number(10) | 关键字 |
| used\_money | 已分配金额 | number(10) | 此字段不一定精确,仅用于控制 |
| max\_money | 最大金额 | number(10) | 即1亿 |
| update\_time | 更新时间 | char(14) | YYYYMMDDHHMISS |