# 财务系统的开发

# 财务系统的简介

财务系统是我们最核心的系统之一,掌握着对我们系统资产的增删改查,该资产包含:

数字货币,现金。其他系统要支出或输入资产都要通过财务系统进行审核结算,财务系统必须严格的保证数据的强一致性,以免造成用户资产的缺失或突出。构建财务系统的难点在于锁安全和事务保证。

# 财务系统数据库

## 2.1 用户的资产

### 表account (用户财产记录)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 名称 | 数据类型 | 长度 | 小数位 | 允许空值 | 主键 | 默认值 | 说明 |
| 1 | id | bigint | 20 | 0 | N | Y |  | 自增id |
| 2 | user\_id | bigint | 20 | 0 | N | N |  | 用户id |
| 3 | coin\_id | bigint | 20 | 0 | N | N |  | 币种id |
| 4 | status | bit | 1 | 0 | N | N |  | 账号状态：1，正常；2，冻结； |
| 5 | balance\_amount | decimal | 41 | 20 | N | N |  | 币种可用金额 |
| 6 | freeze\_amount | decimal | 41 | 20 | N | N |  | 币种冻结金额 |
| 7 | recharge\_amount | decimal | 41 | 20 | N | N |  | 累计充值金额 |
| 8 | withdrawals\_amount | decimal | 41 | 20 | N | N |  | 累计提现金额 |
| 9 | net\_value | decimal | 41 | 20 | N | N |  | 净值 |
| 10 | lock\_margin | decimal | 41 | 20 | N | N |  | 占用保证金 |
| 11 | float\_profit | decimal | 41 | 20 | N | N |  | 持仓盈亏/浮动盈亏 |
| 12 | total\_profit | decimal | 41 | 20 | N | N |  | 总盈亏 |
| 13 | rec\_addr | varchar | 100 | 0 | Y | N |  | 充值地址 |
| 14 | version | bigint | 20 | 0 | N | N |  | 版本号 |
| 15 | last\_update\_time | datetime | 19 | 0 | N | N | CURRENT\_TIMESTAMP | 更新时间 |
| 16 | created | datetime | 19 | 0 | N | N | CURRENT\_TIMESTAMP | 创建时间 |

### 表account\_detail (资金账户流水)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 名称 | 数据类型 | 长度 | 小数位 | 允许空值 | 主键 | 默认值 | 说明 |
| 1 | id | bigint | 20 | 0 | N | Y |  |  |
| 2 | user\_id | bigint | 20 | 0 | N | N |  | 用户id |
| 3 | coin\_id | bigint | 20 | 0 | N | N |  | 币种id |
| 4 | account\_id | bigint | 20 | 0 | N | N |  | 账户id |
| 5 | ref\_account\_id | bigint | 20 | 0 | N | N |  | 该笔流水资金关联方的账户id |
| 6 | order\_id | bigint | 20 | 0 | N | N | 0 | 订单ID |
| 7 | direction | bit | 1 | 0 | N | N |  | 入账为1，出账为2 |
| 8 | business\_type | varchar | 100 | 0 | N | N |  | 业务类型: 充值(recharge\_into) 提现审核通过(withdrawals\_out) 下单(order\_create) 成交(order\_turnover) 成交手续费(order\_turnover\_poundage) 撤单(order\_cancel) 注册奖励(bonus\_register) 提币冻结解冻(withdrawals) 充人民币(recharge) 提币手续费(withdrawals\_poundage) 兑换(cny\_btcx\_exchange) 奖励充值(bonus\_into) 奖励冻结(bonus\_freeze) |
| 9 | amount | decimal | 41 | 20 | N | N |  | 资产数量 |
| 10 | fee | decimal | 41 | 20 | Y | N |  | 手续费 |
| 11 | remark | varchar | 255 | 0 | N | N |  | 流水状态： 充值 提现 冻结 解冻 转出 转入 |
| 12 | created | datetime | 19 | 0 | N | N | CURRENT\_TIMESTAMP | 日期 |

## 2.2 币种相关(coin)

### 表coin (币种配置信息)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 名称 | 数据类型 | 长度 | 小数位 | 允许空值 | 主键 | 默认值 | 说明 |
| 1 | id | bigint | 20 | 0 | N | Y |  | 币种ID |
| 2 | name | varchar | 100 | 0 | N | N |  | 币种名称 |
| 3 | title | varchar | 100 | 0 | N | N |  | 币种标题 |
| 4 | img | varchar | 255 | 0 | N | N |  | 币种logo |
| 5 | type | varchar | 50 | 0 | N | N |  | xnb：人民币 default：比特币系列 ETH：以太坊 ethToken：以太坊代币 |
| 6 | wallet | varchar | 50 | 0 | N | N |  | rgb：认购币 qbb：钱包币 |
| 7 | round | tinyint | 4 | 0 | N | N |  | 小数位数 |
| 8 | base\_amount | decimal | 21 | 8 | Y | N |  | 最小提现单位 |
| 9 | min\_amount | decimal | 21 | 8 | Y | N |  | 单笔最小提现数量 |
| 10 | max\_amount | decimal | 21 | 8 | Y | N |  | 单笔最大提现数量 |
| 11 | day\_max\_amount | decimal | 21 | 8 | Y | N |  | 当日最大提现数量 |
| 12 | status | bit | 1 | 0 | N | N | 1 | status=1：启用 0：禁用 |
| 13 | auto\_out | double | 23 | 0 | Y | N | 10 | 自动转出数量 |
| 14 | rate | double | 23 | 0 | Y | N | 0 | 手续费率 |
| 15 | min\_fee\_num | decimal | 21 | 8 | Y | N |  | 最低收取手续费个数 |
| 16 | withdraw\_flag | tinyint | 4 | 0 | Y | N | 1 | 提现开关 |
| 17 | recharge\_flag | tinyint | 4 | 0 | Y | N | 1 | 充值开关 |
| 18 | last\_update\_time | datetime | 19 | 0 | N | N | CURRENT\_TIMESTAMP | 更新时间 |
| 19 | created | datetime | 19 | 0 | N | N | CURRENT\_TIMESTAMP | 创建时间 |

### 表coin\_type (币种类型)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 名称 | 数据类型 | 长度 | 小数位 | 允许空值 | 主键 | 默认值 | 说明 |
| 1 | id | bigint | 20 | 0 | N | Y |  | 主键 |
| 2 | code | varchar | 16 | 0 | N | N |  | 代码 |
| 3 | description | varchar | 32 | 0 | N | N |  | 描述 |
| 4 | status | tinyint | 4 | 0 | N | N |  | 状态：0-无效；1-有效； |
| 5 | created | datetime | 19 | 0 | N | N | CURRENT\_TIMESTAMP | 创建时间 |
| 6 | last\_update\_time | datetime | 19 | 0 | N | N | CURRENT\_TIMESTAMP | 更新时间 |

### 表coin\_config (币种配置信息)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 名称 | 数据类型 | 长度 | 小数位 | 允许空值 | 主键 | 默认值 | 说明 |
| 1 | id | bigint | 20 | 0 | N | Y |  | 币种ID(对应coin表ID) |
| 2 | name | varchar | 100 | 0 | N | N |  | 币种名称 |
| 3 | coin\_type | varchar | 50 | 0 | N | N |  | btc-比特币系列；eth-以太坊；ethToken-以太坊代币；etc-以太经典；\r\n\r\n |
| 4 | credit\_limit | decimal | 21 | 8 | Y | N |  | 钱包最低留存的币 |
| 5 | credit\_max\_limit | decimal | 21 | 8 | Y | N |  | 当触发改状态的时候,开始归集 |
| 6 | rpc\_ip | varchar | 20 | 0 | Y | N |  | rpc服务ip |
| 7 | rpc\_port | varchar | 10 | 0 | Y | N |  | rpc服务port |
| 8 | rpc\_user | varchar | 30 | 0 | Y | N |  | rpc用户 |
| 9 | rpc\_pwd | varchar | 200 | 0 | Y | N |  | rpc密码 |
| 10 | last\_block | varchar | 100 | 0 | Y | N |  | 最后一个区块 |
| 11 | wallet\_user | varchar | 50 | 0 | Y | N |  | 钱包用户名 |
| 12 | wallet\_pass | varchar | 50 | 0 | Y | N |  | 钱包密码 |
| 13 | contract\_address | varchar | 100 | 0 | Y | N |  | 代币合约地址 |
| 14 | context | varchar | 50 | 0 | Y | N |  | context |
| 15 | min\_confirm | int | 10 | 0 | Y | N | 1 | 最低确认数 |
| 16 | task | varchar | 50 | 0 | Y | N |  | 定时任务 |
| 17 | status | int | 10 | 0 | N | N | 0 | 是否可用0不可用,1可用 |
| 18 | auto\_draw\_limit | decimal | 21 | 8 | Y | N |  |  |
| 19 | auto\_draw | int | 10 | 0 | Y | N |  |  |

### 表coin\_recharge (数字货币充值记录)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 名称 | 数据类型 | 长度 | 小数位 | 允许空值 | 主键 | 默认值 | 说明 |
| 1 | id | bigint | 20 | 0 | N | Y |  | 自增id |
| 2 | user\_id | bigint | 20 | 0 | N | N |  | 用户id |
| 3 | coin\_id | bigint | 20 | 0 | N | N |  | 币种id |
| 4 | coin\_name | varchar | 255 | 0 | N | N |  | 币种名称 |
| 5 | coin\_type | varchar | 50 | 0 | N | N |  | 币种类型 |
| 6 | address | varchar | 255 | 0 | Y | N |  | 钱包地址 |
| 7 | confirm | int | 10 | 0 | N | N |  | 充值确认数 |
| 8 | status | int | 10 | 0 | Y | N | 0 | 状态：0-待入帐；1-充值失败，2到账失败，3到账成功； |
| 9 | txid | varchar | 80 | 0 | Y | N |  | 交易id |
| 10 | amount | decimal | 21 | 8 | Y | N |  |  |
| 11 | last\_update\_time | datetime | 19 | 0 | N | N | CURRENT\_TIMESTAMP | 修改时间 |
| 12 | created | datetime | 19 | 0 | N | N | CURRENT\_TIMESTAMP | 创建时间 |

### 表coin\_withdraw (数字货币提现记录)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 名称 | 数据类型 | 长度 | 小数位 | 允许空值 | 主键 | 默认值 | 说明 |
| 1 | id | bigint | 20 | 0 | N | Y |  | 自增id |
| 2 | user\_id | bigint | 20 | 0 | N | N |  | 用户id |
| 3 | coin\_id | bigint | 20 | 0 | N | N |  | 币种id |
| 4 | coin\_name | varchar | 255 | 0 | N | N |  | 币种名称 |
| 5 | coin\_type | varchar | 50 | 0 | N | N |  | 币种类型 |
| 6 | address | varchar | 255 | 0 | N | N |  | 钱包地址 |
| 7 | txid | varchar | 80 | 0 | Y | N |  | 交易id |
| 8 | num | decimal | 21 | 8 | N | N |  | 提现量 |
| 9 | fee | decimal | 21 | 8 | N | N |  | 手续费() |
| 10 | mum | decimal | 21 | 8 | N | N |  | 实际提现 |
| 11 | type | bit | 1 | 0 | Y | N | 0 | 0站内1其他2手工提币 |
| 12 | chain\_fee | decimal | 21 | 8 | Y | N |  | 链上手续费花费 |
| 13 | block\_num | int | 10 | 0 | Y | N | 0 | 区块高度 |
| 14 | remark | varchar | 255 | 0 | Y | N |  | 后台审核人员提币备注备注 |
| 15 | wallet\_mark | varchar | 255 | 0 | Y | N |  | 钱包提币备注备注 |
| 16 | step | tinyint | 4 | 0 | Y | N |  | 当前审核级数 |
| 17 | status | bit | 1 | 0 | N | N |  | 状态：0-审核中；1-成功；2-拒绝；3-撤销；4-审核通过；5-打币中； |
| 18 | audit\_time | datetime | 19 | 0 | Y | N |  | 审核时间 |
| 19 | last\_update\_time | datetime | 19 | 0 | N | N | CURRENT\_TIMESTAMP | 修改时间 |
| 20 | created | datetime | 19 | 0 | N | N | CURRENT\_TIMESTAMP | 创建时间 |

### 表coin\_withdraw\_audit\_record (提币审核记录)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 名称 | 数据类型 | 长度 | 小数位 | 允许空值 | 主键 | 默认值 | 说明 |
| 1 | id | bigint | 20 | 0 | N | Y |  | 主键 |
| 2 | order\_id | bigint | 20 | 0 | Y | N |  | 提币订单号 |
| 3 | status | tinyint | 4 | 0 | Y | N |  | 状态 |
| 4 | remark | text | 65535 | 0 | Y | N |  | 审核备注 |
| 5 | step | tinyint | 4 | 0 | Y | N |  | 当前审核级数 |
| 6 | audit\_user\_id | bigint | 20 | 0 | Y | N |  | 审核人ID |
| 7 | audit\_user\_name | varchar | 128 | 0 | Y | N |  | 审核人 |
| 8 | created | datetime | 19 | 0 | Y | N | CURRENT\_TIMESTAMP | 创建时间 |

### 表coin\_balance (币种余额)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 名称 | 数据类型 | 长度 | 小数位 | 允许空值 | 主键 | 默认值 | 说明 |
| 1 | id | bigint | 20 | 0 | N | Y |  | 主键 |
| 2 | coin\_id | bigint | 20 | 0 | Y | N |  | 币种ID |
| 3 | coin\_name | varchar | 16 | 0 | Y | N |  | 币种名称 |
| 4 | system\_balance | decimal | 21 | 8 | Y | N |  | 系统余额（根据充值提币计算） |
| 5 | coin\_type | varchar | 50 | 0 | Y | N |  | 币种类型 |
| 6 | collect\_account\_balance | decimal | 21 | 8 | Y | N |  | 归集账户余额 |
| 7 | loan\_account\_balance | decimal | 21 | 8 | Y | N |  | 钱包账户余额 |
| 8 | fee\_account\_balance | decimal | 21 | 8 | Y | N |  | 手续费账户余额(eth转账需要手续费) |
| 9 | last\_update\_time | datetime | 19 | 0 | Y | N | CURRENT\_TIMESTAMP | 更新时间 |
| 10 | created | datetime | 19 | 0 | Y | N | CURRENT\_TIMESTAMP | 创建时间 |
| 11 | recharge\_account\_balance | decimal | 21 | 8 | Y | N |  |  |

### 表coin\_server (监测当前服务器Ip状态)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 名称 | 数据类型 | 长度 | 小数位 | 允许空值 | 主键 | 默认值 | 说明 |
| 1 | id | bigint | 20 | 0 | N | Y |  | 自增id |
| 2 | rpc\_ip | varchar | 50 | 0 | N | N |  | 钱包服务器ip |
| 3 | rpc\_port | varchar | 50 | 0 | N | N |  | 钱包服务器ip |
| 4 | running | int | 10 | 0 | N | N |  | 服务是否运行 0:正常,1:停止 |
| 5 | wallet\_number | bigint | 20 | 0 | N | N |  | 钱包服务器区块高度 |
| 6 | coin\_name | varchar | 50 | 0 | Y | N |  |  |
| 7 | mark | varchar | 100 | 0 | Y | N |  | 备注信息 |
| 8 | real\_number | bigint | 20 | 0 | Y | N |  | 真实区块高度 |
| 9 | last\_update\_time | datetime | 19 | 0 | N | N | CURRENT\_TIMESTAMP | 修改时间 |
| 10 | created | datetime | 19 | 0 | N | N | CURRENT\_TIMESTAMP | 创建时间 |

## 2.3 现金相关(cash)

### 表cash\_recharge (充值表)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 名称 | 数据类型 | 长度 | 小数位 | 允许空值 | 主键 | 默认值 | 说明 |
| 1 | id | bigint | 20 | 0 | N | Y |  | 自增id |
| 2 | user\_id | bigint | 20 | 0 | N | N |  | 用户id |
| 3 | coin\_id | bigint | 20 | 0 | N | N |  | 币种id |
| 4 | coin\_name | varchar | 255 | 0 | N | N |  | 币种名：cny，人民币； |
| 5 | num | decimal | 20 | 2 | N | N |  | 数量（充值金额） |
| 6 | fee | decimal | 20 | 2 | N | N |  | 手续费 |
| 7 | feecoin | varchar | 255 | 0 | Y | N |  | 手续费币种 |
| 8 | mum | decimal | 20 | 2 | N | N |  | 成交量（到账金额） |
| 9 | type | varchar | 255 | 0 | N | N |  | 类型：alipay，支付宝；cai1pay，财易付；bank，银联； |
| 10 | tradeno | varchar | 255 | 0 | N | N |  | 充值订单号 |
| 11 | outtradeno | varchar | 255 | 0 | Y | N |  | 第三方订单号 |
| 12 | remark | varchar | 8 | 0 | Y | N |  | 充值备注备注 |
| 13 | audit\_remark | varchar | 128 | 0 | Y | N |  | 审核备注 |
| 14 | step | tinyint | 4 | 0 | Y | N |  | 当前审核级数 |
| 15 | status | tinyint | 4 | 0 | N | N |  | 状态：0-待审核；1-审核通过；2-拒绝；3-充值成功； |
| 16 | created | datetime | 19 | 0 | N | N | CURRENT\_TIMESTAMP | 创建时间 |
| 17 | last\_update\_time | datetime | 19 | 0 | N | N | CURRENT\_TIMESTAMP | 更新时间 |
| 18 | name | varchar | 100 | 0 | Y | N |  | 银行卡账户名 |
| 19 | bank\_name | varchar | 100 | 0 | Y | N |  | 开户行 |
| 20 | bank\_card | varchar | 100 | 0 | Y | N |  | 银行卡号 |
| 21 | last\_time | datetime | 19 | 0 | Y | N |  | 最后确认到账时间。 |

### 表cash\_withdrawals (提现表)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 名称 | 数据类型 | 长度 | 小数位 | 允许空值 | 主键 | 默认值 | 说明 |
| 1 | id | bigint | 20 | 0 | N | Y |  | 主键 |
| 2 | user\_id | bigint | 20 | 0 | N | N |  | 用户ID |
| 3 | coin\_id | bigint | 20 | 0 | N | N |  | 币种ID |
| 4 | account\_id | bigint | 20 | 0 | N | N |  | 资金账户ID |
| 5 | num | decimal | 21 | 2 | N | N |  | 数量（提现金额） |
| 6 | fee | decimal | 21 | 2 | N | N |  | 手续费 |
| 7 | mum | decimal | 21 | 2 | N | N |  | 到账金额 |
| 8 | truename | varchar | 255 | 0 | N | N |  | 开户人 |
| 9 | bank | varchar | 255 | 0 | N | N |  | 银行名称 |
| 10 | bank\_prov | varchar | 255 | 0 | Y | N |  | 银行所在省 |
| 11 | bank\_city | varchar | 255 | 0 | Y | N |  | 银行所在市 |
| 12 | bank\_addr | varchar | 255 | 0 | Y | N |  | 开户行 |
| 13 | bank\_card | varchar | 255 | 0 | N | N |  | 银行账号 |
| 14 | remark | varchar | 255 | 0 | Y | N |  | 备注 |
| 15 | step | tinyint | 4 | 0 | Y | N |  | 当前审核级数 |
| 16 | status | tinyint | 4 | 0 | N | N |  | 状态：0-待审核；1-审核通过；2-拒绝；3-提现成功； |
| 17 | created | datetime | 19 | 0 | N | N | CURRENT\_TIMESTAMP | 创建时间 |
| 18 | last\_update\_time | datetime | 19 | 0 | N | N | CURRENT\_TIMESTAMP | 更新时间 |
| 19 | last\_time | datetime | 19 | 0 | Y | N |  | 最后确认提现到账时间 |

### 表cash\_recharge\_audit\_record (充值审核记录)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 名称 | 数据类型 | 长度 | 小数位 | 允许空值 | 主键 | 默认值 | 说明 |
| 1 | id | bigint | 20 | 0 | N | Y |  | 主键 |
| 2 | order\_id | bigint | 20 | 0 | Y | N |  | 充值订单号 |
| 3 | status | tinyint | 4 | 0 | Y | N |  | 状态 |
| 4 | remark | text | 65535 | 0 | Y | N |  | 审核备注 |
| 5 | step | tinyint | 4 | 0 | Y | N |  | 当前审核级数 |
| 6 | audit\_user\_id | bigint | 20 | 0 | Y | N |  | 审核人ID |
| 7 | audit\_user\_name | varchar | 128 | 0 | Y | N |  | 审核人 |
| 8 | created | datetime | 19 | 0 | Y | N | CURRENT\_TIMESTAMP | 创建时间 |

### 表cash\_withdraw\_audit\_record (提现审核记录)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 名称 | 数据类型 | 长度 | 小数位 | 允许空值 | 主键 | 默认值 | 说明 |
| 1 | id | bigint | 20 | 0 | N | Y |  | 主键 |
| 2 | order\_id | bigint | 20 | 0 | Y | N |  | 提币订单号 |
| 3 | status | tinyint | 4 | 0 | Y | N |  | 状态 |
| 4 | remark | text | 65535 | 0 | Y | N |  | 审核备注 |
| 5 | step | tinyint | 4 | 0 | Y | N |  | 当前审核级数 |
| 6 | audit\_user\_id | bigint | 20 | 0 | Y | N |  | 审核人ID |
| 7 | audit\_user\_name | varchar | 128 | 0 | Y | N |  | 审核人 |
| 8 | created | datetime | 19 | 0 | Y | N | CURRENT\_TIMESTAMP | 创建时间 |

## 2.4 创新交易相关

### 表forex\_account (创新交易持仓信息)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 名称 | 数据类型 | 长度 | 小数位 | 允许空值 | 主键 | 默认值 | 说明 |
| 1 | id | bigint | 20 | 0 | N | Y |  | 主键 |
| 2 | user\_id | bigint | 20 | 0 | Y | N |  | 用户ID |
| 3 | market\_id | bigint | 20 | 0 | Y | N |  | 交易对ID |
| 4 | market\_name | varchar | 255 | 0 | Y | N |  | 交易对 |
| 5 | type | tinyint | 4 | 0 | Y | N |  | 持仓方向：1-买；2-卖 |
| 6 | amount | decimal | 21 | 8 | Y | N | 0.00000000 | 持仓量 |
| 7 | lock\_amount | decimal | 21 | 8 | Y | N | 0.00000000 | 冻结持仓量 |
| 8 | status | tinyint | 4 | 0 | Y | N |  | 状态：1-有效；2-锁定； |
| 9 | last\_update\_time | datetime | 19 | 0 | N | N | CURRENT\_TIMESTAMP | 修改时间 |
| 10 | created | datetime | 19 | 0 | N | N | CURRENT\_TIMESTAMP | 创建时间 |

### 表forex\_account\_detail (持仓账户流水)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 名称 | 数据类型 | 长度 | 小数位 | 允许空值 | 主键 | 默认值 | 说明 |
| 1 | id | bigint | 20 | 0 | N | Y |  | 主键 |
| 2 | account\_id | bigint | 20 | 0 | Y | N |  | 持仓账户ID |
| 3 | type | tinyint | 4 | 0 | Y | N |  | 收支类型：开仓；2-平仓； |
| 4 | amount | decimal | 21 | 8 | Y | N |  | 持仓量 |
| 5 | remark | text | 65535 | 0 | Y | N |  | 备注 |
| 6 | created | datetime | 19 | 0 | Y | N | CURRENT\_TIMESTAMP | 创建时间 |

### 表forex\_coin (创新交易币种表)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 名称 | 数据类型 | 长度 | 小数位 | 允许空值 | 主键 | 默认值 | 说明 |
| 1 | id | bigint | 20 | 0 | N | Y |  |  |
| 2 | name | varchar | 100 | 0 | N | N |  | 币种名称 |
| 3 | title | varchar | 100 | 0 | N | N |  | 币种标题 |
| 4 | sort | tinyint | 4 | 0 | N | N | 0 | 排序 |
| 5 | status | bit | 1 | 0 | N | N | 1 | 状态: 0禁用 1启用 |
| 6 | last\_update\_time | datetime | 19 | 0 | N | N | CURRENT\_TIMESTAMP | 修改时间 |
| 7 | created | datetime | 19 | 0 | N | N | CURRENT\_TIMESTAMP | 创建时间 |

### 表forex\_open\_position\_order (开仓订单信息)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 名称 | 数据类型 | 长度 | 小数位 | 允许空值 | 主键 | 默认值 | 说明 |
| 1 | id | bigint | 20 | 0 | N | Y |  | 主键 |
| 2 | user\_id | bigint | 20 | 0 | Y | N |  | 用户ID |
| 3 | market\_id | bigint | 20 | 0 | Y | N |  | 交易对ID |
| 4 | market\_name | varchar | 255 | 0 | Y | N |  | 交易对名称 |
| 5 | coin\_id | bigint | 20 | 0 | Y | N |  | 结算币种 |
| 6 | type | tinyint | 4 | 0 | Y | N |  | 持仓方向：1-买；2-卖 |
| 7 | account\_id | bigint | 20 | 0 | Y | N |  | 资金账户ID |
| 8 | entrust\_order\_id | bigint | 20 | 0 | Y | N |  | 委托订单 |
| 9 | order\_id | bigint | 20 | 0 | Y | N |  | 成交订单号 |
| 10 | price | decimal | 21 | 8 | Y | N |  | 成交价格 |
| 11 | num | decimal | 21 | 8 | Y | N | 0.00000000 | 成交数量 |
| 12 | lock\_margin | decimal | 21 | 8 | Y | N | 0.00000000 | 扣除保证金 |
| 13 | close\_num | decimal | 21 | 8 | Y | N | 0.00000000 | 平仓量 |
| 14 | status | tinyint | 4 | 0 | Y | N |  | 状态：1：未平仓；2-已平仓 |
| 15 | last\_update\_time | datetime | 19 | 0 | Y | N | CURRENT\_TIMESTAMP | 修改时间 |
| 16 | created | datetime | 19 | 0 | Y | N | CURRENT\_TIMESTAMP | 创建时间 |

### 表forex\_close\_position\_order (平仓详情)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 名称 | 数据类型 | 长度 | 小数位 | 允许空值 | 主键 | 默认值 | 说明 |
| 1 | id | bigint | 20 | 0 | N | Y |  | 主键 |
| 2 | user\_id | bigint | 20 | 0 | Y | N |  | 用户ID |
| 3 | market\_id | bigint | 20 | 0 | Y | N |  | 交易对ID |
| 4 | market\_name | varchar | 255 | 0 | Y | N |  | 交易对名称 |
| 5 | type | tinyint | 4 | 0 | Y | N |  | 持仓方向：1-买；2-卖 |
| 6 | account\_id | bigint | 20 | 0 | Y | N |  | 资金账户ID |
| 7 | entrust\_order\_id | bigint | 20 | 0 | Y | N |  | 委托订单号 |
| 8 | order\_id | bigint | 20 | 0 | Y | N |  | 成交订单号 |
| 9 | price | decimal | 21 | 8 | Y | N |  | 成交价 |
| 10 | num | decimal | 21 | 8 | Y | N |  | 成交数量 |
| 11 | open\_id | bigint | 20 | 0 | Y | N |  | 关联开仓订单号 |
| 12 | profit | decimal | 21 | 8 | Y | N |  | 平仓盈亏 |
| 13 | unlock\_margin | decimal | 21 | 8 | Y | N |  | 返回还保证金 |
| 14 | last\_update\_time | datetime | 19 | 0 | N | N | CURRENT\_TIMESTAMP | 修改时间 |
| 15 | created | datetime | 19 | 0 | N | N | CURRENT\_TIMESTAMP | 创建时间 |

## 2.5 账号资产冻结相关

### **表 user\_account\_freeze(冻结用户金额)**

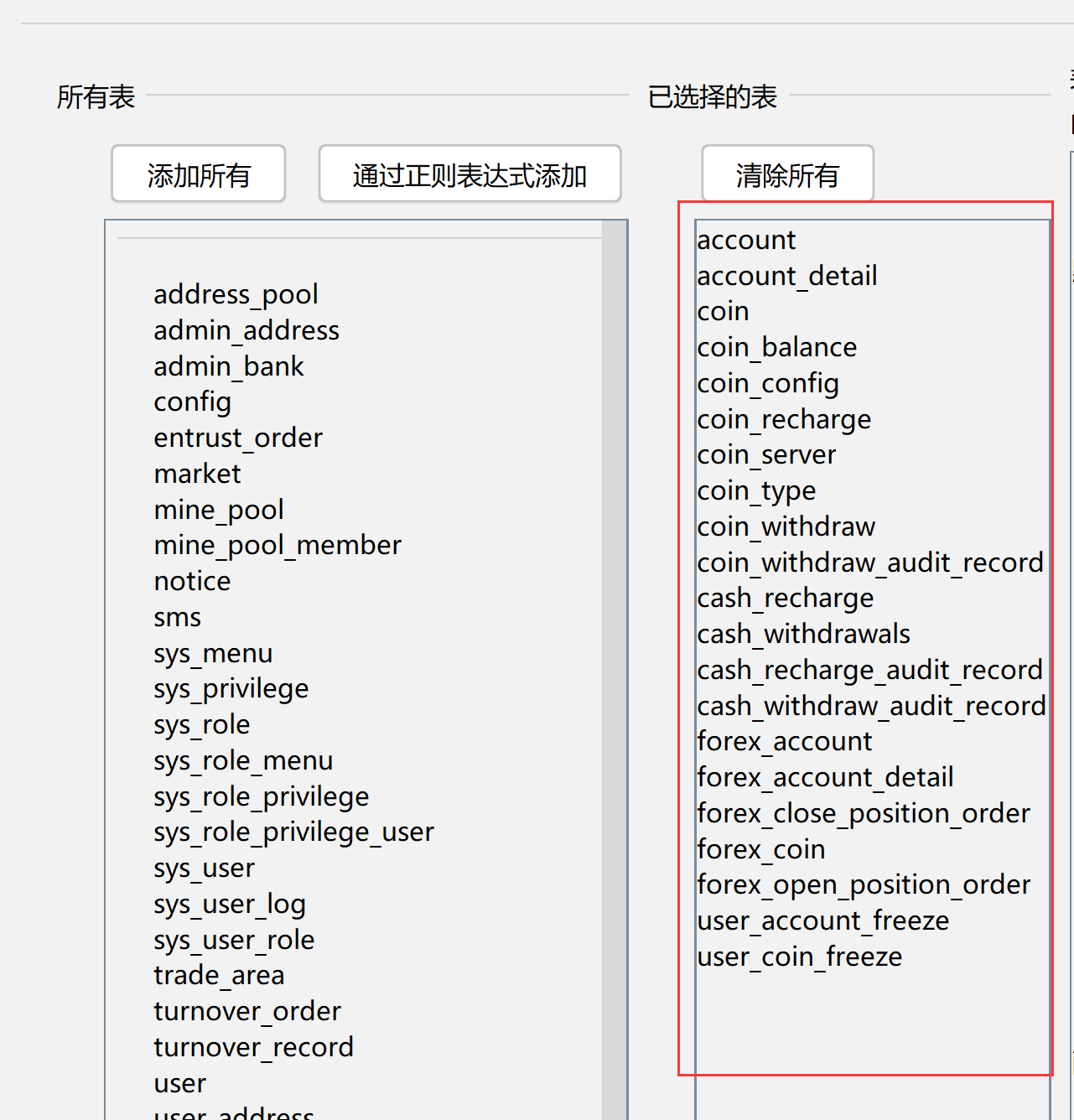
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 名称 | 数据类型 | 长度 | 小数位 | 允许空值 | 主键 | 默认值 | 说明 |
| 1 | user\_id | bigint | 20 | 0 | N | Y |  |  |
| 2 | freeze | decimal | 41 | 20 | Y | N |  |  |

### **表user\_coin\_freeze(冻结用户币种金额)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 名称 | 数据类型 | 长度 | 小数位 | 允许空值 | 主键 | 默认值 | 说明 |
| 1 | user\_id | bigint | 20 | 0 | N | Y |  |  |
| 2 | coin\_id | bigint | 20 | 0 | Y | N |  |  |
| 3 | freeze | decimal | 11 | 0 | Y | N |  |  |

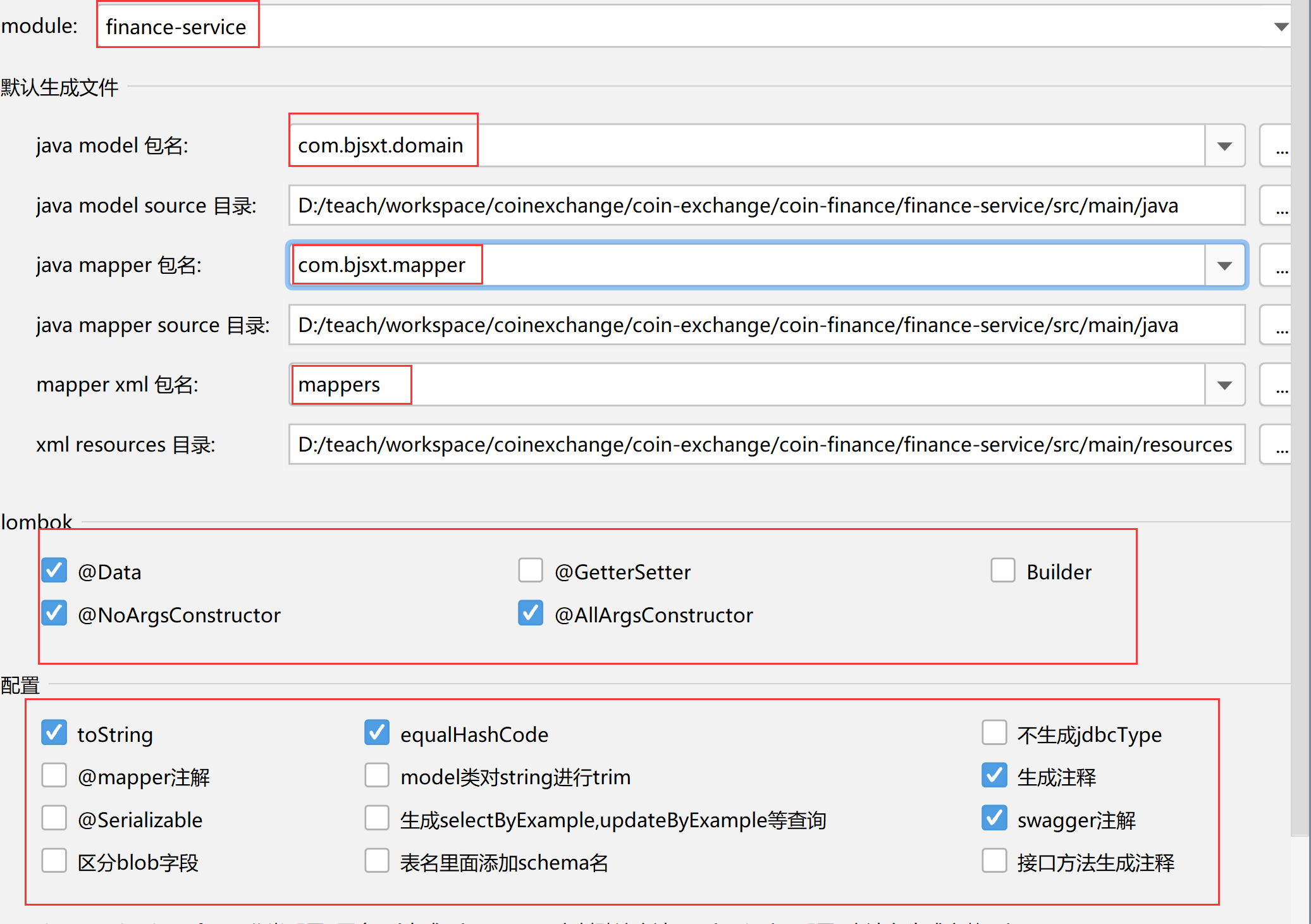
# 财务系统代码的生成

## 3.1 选择要生成的表

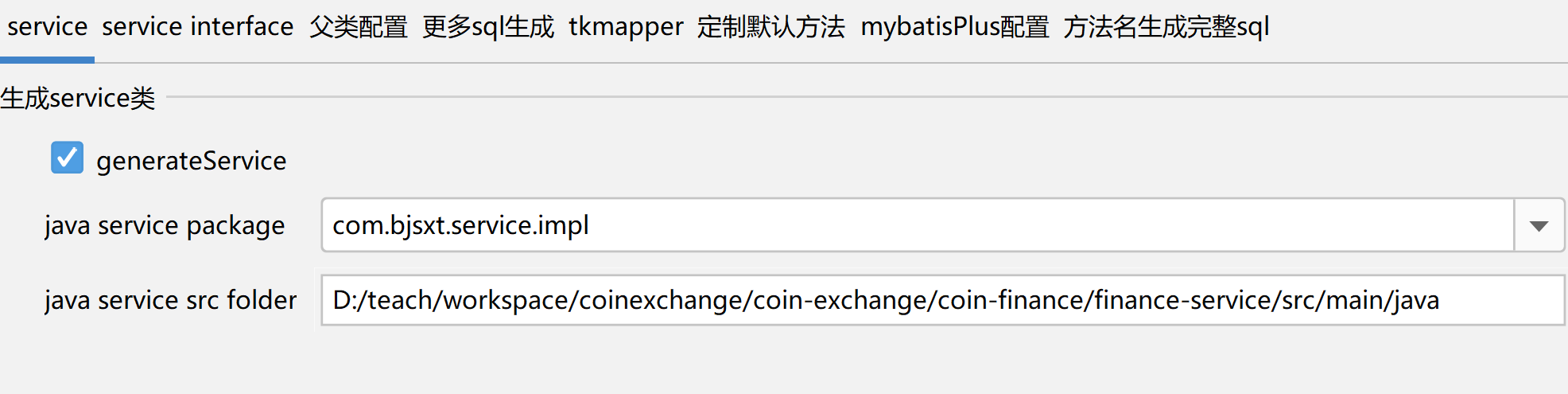


## 3.2 代码生成

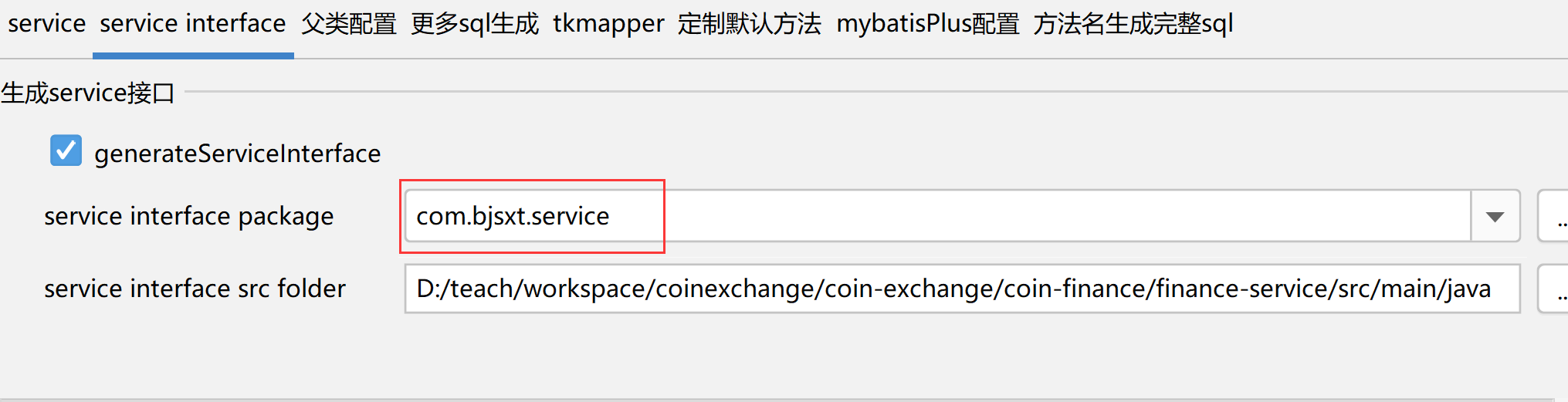
### 配置信息一:



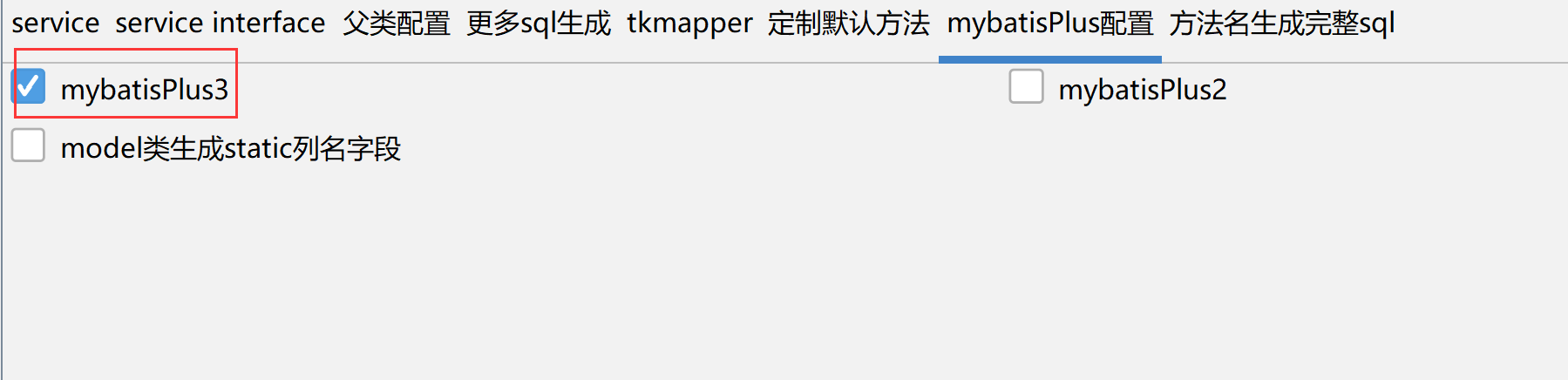
### 配置信息二:



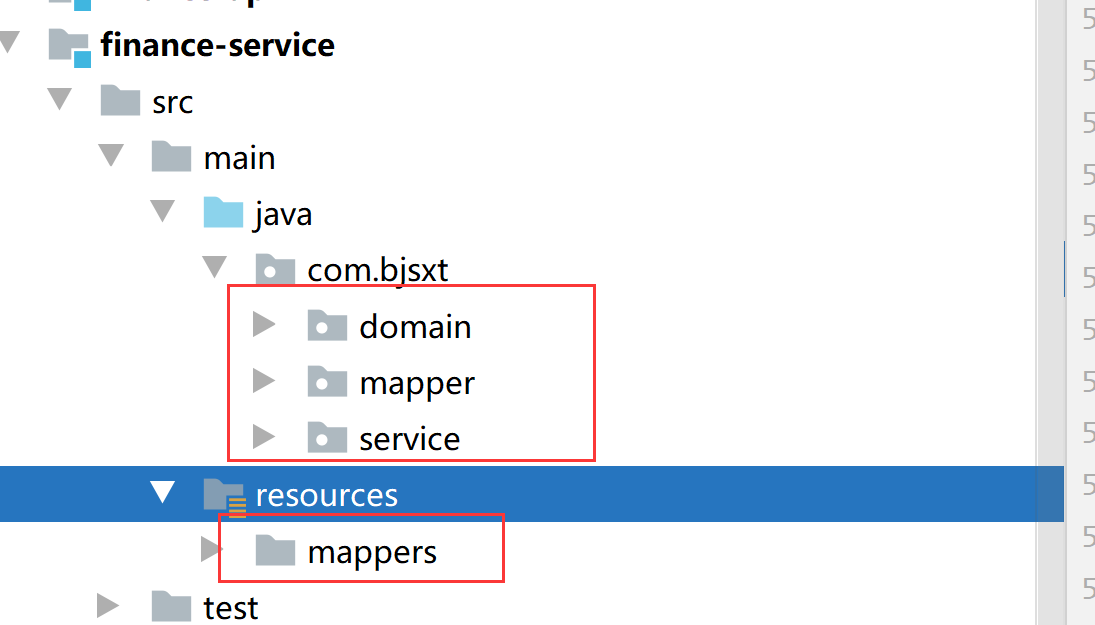
### 配置信息三:



### 配置信息四:



确认后,点击代码生成,即可产生代码.



# 财务系统测试运行

## 添加配置文件

### 本地bootstrap.yml

|  |
| --- |
| spring:  application:  name: finance-service  cloud: *# nacos地址* nacos:  server-addr: nacos-server:8848  config:  file-extension: yaml  profiles:  active: dev *# 拉取的配置文件的dataID = finance-service-dev.yaml* |

### Nacos上finance-service-dev.yaml



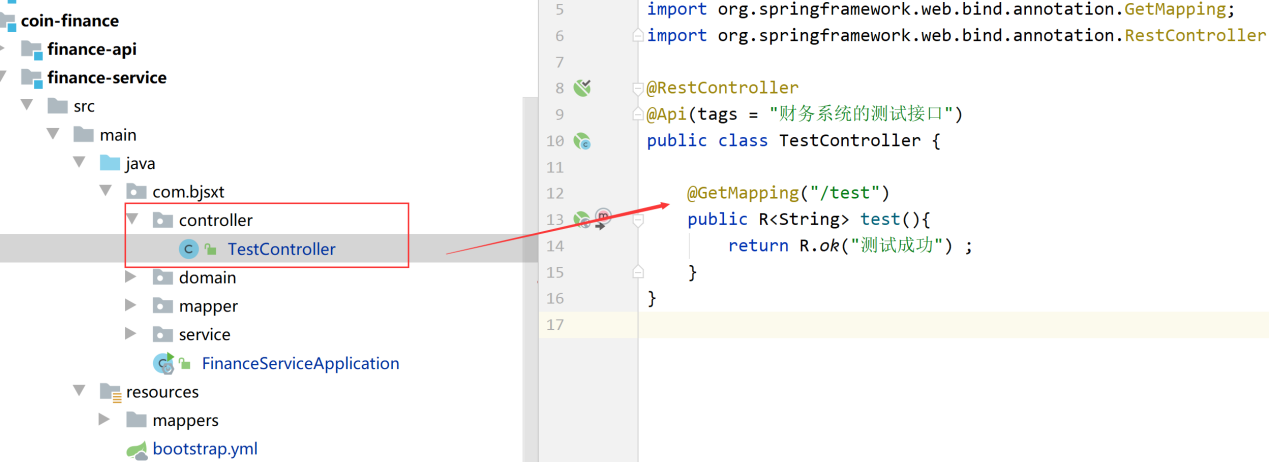
|  |
| --- |
| server:  port: 8070  spring:  datasource:  driver-class-name: com.mysql.cj.jdbc.Driver  url: jdbc:mysql://mysql-server:3307/coin-exchange?serverTimezone=GMT%2B8  username: root  password: Ltd3411??  redis:  host: redis-server  port: 6380  password: Ltd3411??  cloud:  sentinel:  transport:  dashboard: sentinel-server:8858  alicloud:  access-key: LTAI4GEXnLoaQg6UraLEJL1v  secret-key: txr1DH29usTcQo5MUSsDGoVBusQQHX  oss:  endpoint: oss-cn-beijing.aliyuncs.com  mybatis-plus:  configuration:  log-impl: org.apache.ibatis.logging.stdout.StdOutImpl  mapper-locations: classpath:/mappers/\*Mapper.xml  jetcache:  statIntervalMinutes: 15  areaInCacheName: false  local:  default:  type: linkedhashmap  keyConvertor: fastjson  remote:  default:  type: redis  keyConvertor: fastjson  valueEncoder: kryo  valueDecoder: kryo  poolConfig:  minIdle: 5  maxIdle: 20  maxTotal: 50  host: ${spring.redis.host}  port: ${spring.redis.port}  password: ${spring.redis.password}  swagger2:  basePackage: com.bjsxt.controller  name: liangtiandong  url: www.liangtiandong.com  email: liangtiandong@live.com  title: 财务系统API接口  description: 财务系统API接口演示  version: 1.0  termsOfServiceUrl: www.bjsxt.com  geetest:  geetest-id: 3a01ffc01c1d63b37c3dbe8ee9555290  geetest-key: 27c7b4a18124d5d649b9c58ca1830871  identify:  url: https://idcert.market.alicloudapi.com/idcard?idCard=%s&name=%s  appKey: 203866940  appSecret: KF6LRDKejSejx2vfyLqiLP7bVvWOZYCp  appCode: 39e89acacec14a9d80782d8aa2893295 |

## 添加启动类

|  |
| --- |
| */\*\*  \* 财务系统的启动类  \*/* @SpringBootApplication @EnableDiscoveryClient @EnableFeignClients public class FinanceServiceApplication {   public static void main(String[] args) {  SpringApplication.*run*(FinanceServiceApplication.class ,args) ;  } } |

## 添加测试接口

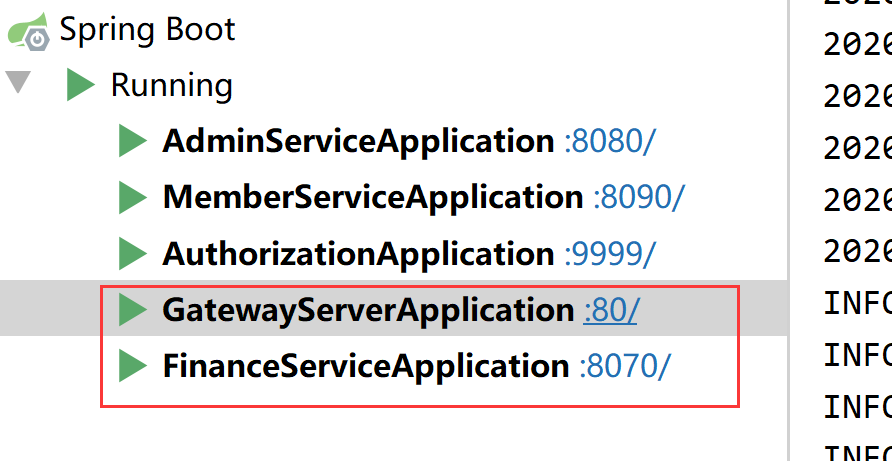
|  |
| --- |
| @RestController @Api(tags = "财务系统的测试接口") public class TestController {   @GetMapping("/test")  public R<String> test(){  return R.*ok*("测试成功") ;  } } |



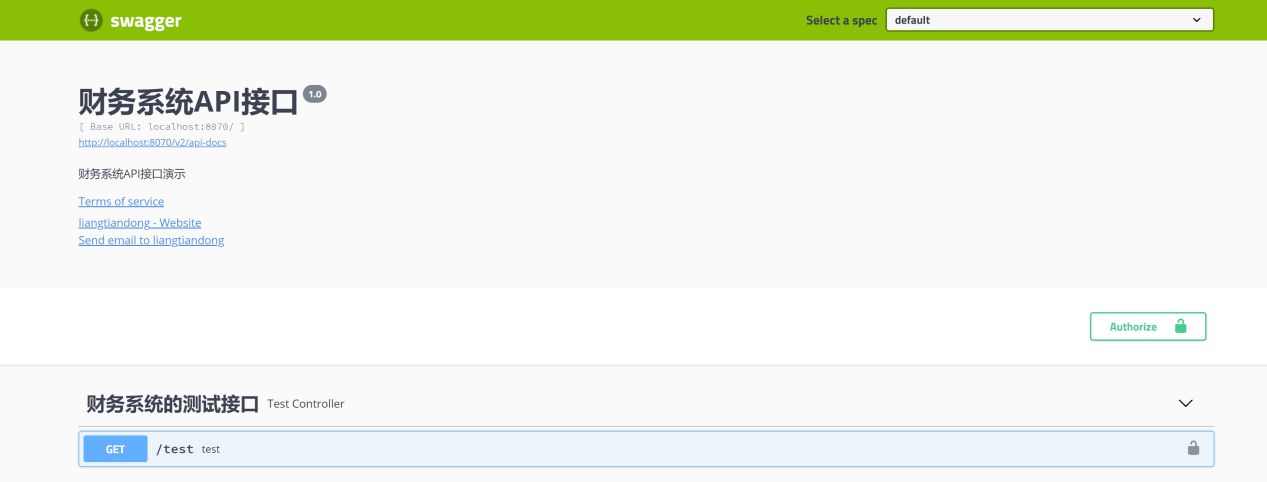
## 在网关里面添加路由

|  |
| --- |
| - id: finance-service\_router  uri: lb://finance-service *# 转发到那个目的地* predicates:  - Path=/finance/\*\*  filters:  - StripPrefix=1 *# 当前端访问*/finance/xx*去掉*finance |

## 启动finance-service/重启gateway-server



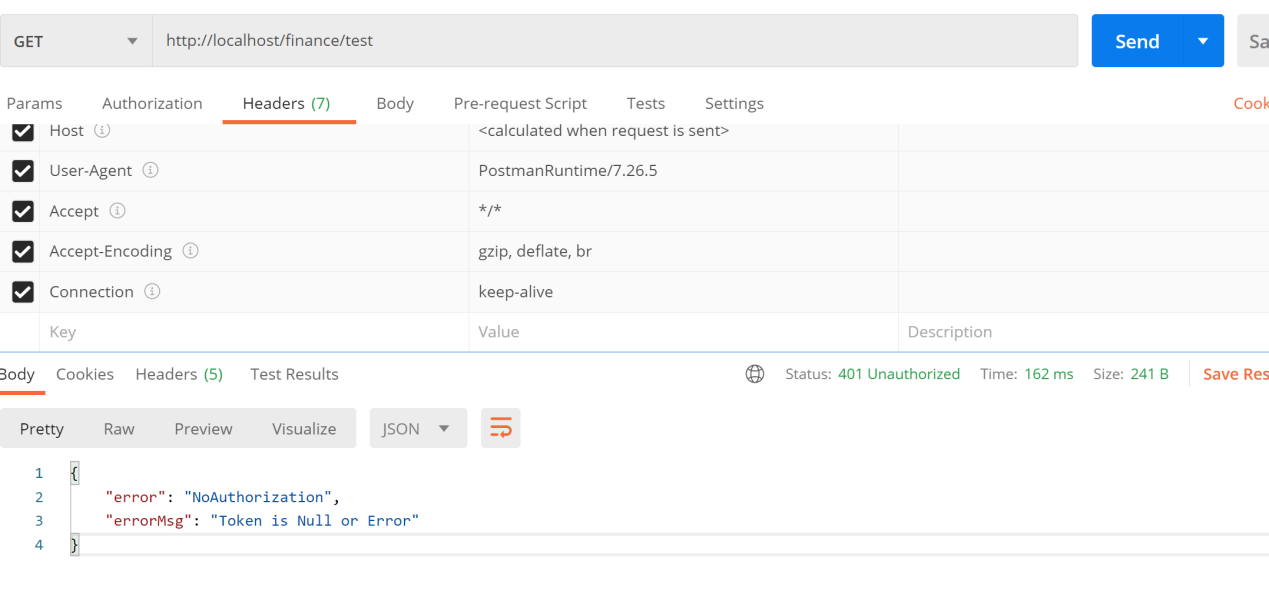
### swagger-ui测试



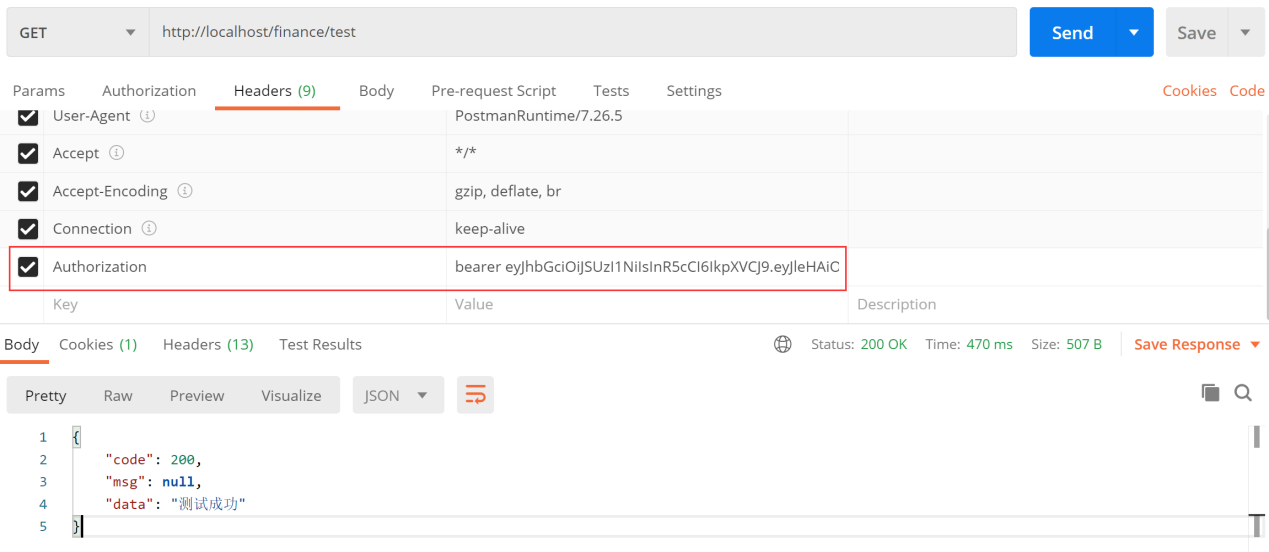
### 访问接口测试

使用网关访问:

无token:



有token:



# 后台系统-币币交易参数配置



## ConType币种类型配置

### API接口

### CoinTypeController

|  |
| --- |
| @RestController @RequestMapping("/coinTypes") @Api(value = "币种类型的控制器") public class CoinTypeController {    @Autowired  private CoinTypeService coinTypeService ;   @GetMapping  @ApiOperation(value = "条件分页查询我们的币种类型")  @ApiImplicitParams({  @ApiImplicitParam(name = "current",value = "当前页") ,  @ApiImplicitParam(name = "size",value = "每页显示的条数") ,  @ApiImplicitParam(name = "code",value = "币种类型") ,  })  @PreAuthorize("hasAuthority('trade\_coin\_type\_query')")  public R<Page<CoinType>> findByPage(@ApiIgnore Page<CoinType> page,String code){  page.addOrder(OrderItem.*desc*("last\_update\_time")) ;  Page<CoinType> coinTypePage = coinTypeService.findByPage(page,code) ;  return R.*ok*(coinTypePage) ;  }    @PostMapping  @ApiOperation(value = "新增货币类型")  @ApiImplicitParams({  @ApiImplicitParam(name = "coinType" ,value = "coinType 的json")  })  @PreAuthorize("hasAuthority('trade\_coin\_type\_create')")  public R add(@RequestBody @Validated CoinType coinType){  boolean save = coinTypeService.save(coinType);  if(save){  return R.*ok*() ;  }  return R.*fail*("新增失败") ;  }    @PatchMapping  @ApiOperation(value = "修改货币类型")  @ApiImplicitParams({  @ApiImplicitParam(name = "coinType" ,value = "coinType 的json")  })  @PreAuthorize("hasAuthority('trade\_coin\_type\_update')")  public R update(@RequestBody @Validated CoinType coinType){  boolean update = coinTypeService.updateById(coinType);  if(update){  return R.*ok*() ;  }  return R.*fail*("修改失败") ;  }    @PostMapping("/setStatus")  @ApiOperation(value = "修改货币状态")  @ApiImplicitParams({  @ApiImplicitParam(name = "id" ,value = "coinType的Id") ,  @ApiImplicitParam(name = "status" ,value = "修改成的状态")  })  @PreAuthorize("hasAuthority('trade\_coin\_type\_update')")  public R setStatus(@RequestBody CoinType coinType){  boolean update = coinTypeService.updateById(coinType);  if(update){  return R.*ok*() ;  }  return R.*fail*("修改失败") ;  }    @GetMapping("/all")  @ApiOperation(value = "查询所有的币种类型")  @ApiImplicitParams({  @ApiImplicitParam(name = "status",value = "币种的状态")  })  @PreAuthorize("hasAuthority('trade\_coin\_type\_query')")  public R<List<CoinType>> findAllCoinTypeByStatus(Byte status){  List<CoinType> coinTypes = coinTypeService.listByStatus(status) ;  return R.*ok*(coinTypes) ;  } } |

### CoinTypeService

|  |
| --- |
| public interface CoinTypeService extends IService<CoinType> {    */\*\*  \* 条件分页查询货币类型  \* @param page  \* 分页参数  \* @param code  \* 类型的Code  \* @return  \* 分页货币类型  \*/* Page<CoinType> findByPage(Page<CoinType> page, String code);   */\*\*  \* 使用币种类型的状态查询所有的币种类型值  \* @param status  \* @return  \*/* List<CoinType> listByStatus(Byte status); } |

### CoinTypeServiceImpl

|  |
| --- |
| @Service public class CoinTypeServiceImpl extends ServiceImpl<CoinTypeMapper, CoinType> implements CoinTypeService {   */\*\*  \* 条件分页查询货币类型  \*  \* @param page 分页参数  \* @param code 类型的Code  \* @return 分页货币类型  \*/* @Override  public Page<CoinType> findByPage(Page<CoinType> page, String code) {  return page(page,new LambdaQueryWrapper<CoinType>()  .like(!StringUtils.*isEmpty*(code),CoinType::getCode,code));  }   */\*\*  \* 使用币种类型的状态查询所有的币种类型值  \*  \* @param status  \* @return  \*/* @Override  public List<CoinType> listByStatus(Byte status) {  return list(new LambdaQueryWrapper<CoinType>().eq(status!=null ,CoinType::getStatus,status));  } } |

### 5.1.5CoinType

|  |
| --- |
| @ApiModel(value = "com-bjsxt-domain-CoinType") @Data @AllArgsConstructor @NoArgsConstructor @TableName(value = "coin\_type") public class CoinType {  */\*\*  \* 主键  \*/* @TableId(value = "id", type = IdType.*AUTO*)  @ApiModelProperty(value = "主键")  private Long id;   */\*\*  \* 代码  \*/* @TableField(value = "code")  @ApiModelProperty(value = "代码")  @NotBlank  private String code;   */\*\*  \* 描述  \*/* @TableField(value = "description")  @ApiModelProperty(value = "描述")  private String description;   */\*\*  \* 状态：0-无效；1-有效；  \*/* @TableField(value = "status")  @ApiModelProperty(value = "状态：0-无效；1-有效；")  @NotNull  private Byte status;   */\*\*  \* 创建时间  \*/* @TableField(value = "created",fill = FieldFill.*INSERT*)  @ApiModelProperty(value = "创建时间")  private Date created;   */\*\*  \* 更新时间  \*/* @TableField(value = "last\_update\_time",fill = FieldFill.*INSERT\_UPDATE*)  @ApiModelProperty(value = "更新时间")  private Date lastUpdateTime; } |

## Coin币种接口(分页查询/状态修改/详情)

### API接口

### CoinController

|  |
| --- |
| @RestController @RequestMapping("/coins") @Api(tags = "数字货币的数据接口") public class CoinController {   @Autowired  private CoinService coinService ;   */\*\*  \* http://localhost:9527/finance/coins?name=xxx&type=usdt&status=1&title=xxx&wallet\_type=rgb&current=1&size=15  \* @return  \*/* @GetMapping  @ApiOperation(value = "分页条件查询数字货币")  @ApiImplicitParams({  @ApiImplicitParam(name = "name" ,value = "数字货币的名称") ,  @ApiImplicitParam(name = "type" ,value = "数字货币类型的名称") ,  @ApiImplicitParam(name = "status" ,value = "数字货币类型的状态") ,  @ApiImplicitParam(name = "status" ,value = "数字货币类型的标题") ,  @ApiImplicitParam(name = "wallet\_type" ,value = "数字货币钱包类型") ,  @ApiImplicitParam(name = "current" ,value = "当前页") ,  @ApiImplicitParam(name = "size" ,value = "每页显示的条数") ,  })  public R<Page<Coin>> findByPage(  String name , String type , Byte status ,  String title ,@RequestParam(name = "wallet\_type",required = false) String walletType ,  @ApiIgnore Page<Coin> page   ){  Page<Coin> coinPage = coinService.findByPage(name,type,status,title,walletType,page) ;  return R.*ok*(coinPage) ;  }    */\*\*  \* 禁用或启用  \*/* @PostMapping("/setStatus")  @ApiOperation(value = "禁用或启用币种")  @ApiImplicitParams({  @ApiImplicitParam(name = "coin" ,value = "coin的json数据")  })  public R setStatus(@RequestBody Coin coin){  boolean updateById = coinService.updateById(coin);  if(updateById){  return R.*ok*() ;  }  return R.*fail*("设置状态失败") ;   }    @GetMapping("/info/{id}")  @ApiOperation(value = "查询币种的详细信息")  @ApiImplicitParams({  @ApiImplicitParam(name = "id",value = "币种的id")  })  public R<Coin> info(@PathVariable("id") Long id){  Coin coin = coinService.getById(id);  return R.*ok*(coin) ;  }   @GetMapping("/all")  @ApiImplicitParams({  @ApiImplicitParam(name = "status",value = "币种当前的状态")  })  @ApiOperation(value = "通过状态查询所有的币种信息")  public R<List<Coin>> getCoinAll(Byte status){  List<Coin> coins = coinService.getCoinsByStatus(status) ;  return R.*ok*(coins) ;  }    @PatchMapping  @ApiOperation(value = "修改我们的币种的信息")  @ApiImplicitParams({  @ApiImplicitParam(name = "coin" ,value = "coin的json数据")  })  public R update(@RequestBody @Validated Coin coin){  boolean updateById = coinService.updateById(coin);  if(updateById){  return R.*ok*() ;  }  return R.*fail*("修改失败") ;  } |

### CoinService

|  |
| --- |
| public interface CoinService extends IService<Coin> {    */\*\*  \* 数字货币的条件分页查询  \* @param name  \* 数字货币的名称  \* @param type  \* 数字货币类型的名称  \* @param status  \* 数字货币的状态  \* @param title  \* 字货币的标题  \* @param walletType  \* 树字货币的钱包类型名称  \* @param page  \* 分页参数  \* @return  \* 数据货币的分页数据  \*/* Page<Coin> findByPage(String name, String type, Byte status, String title, String walletType, Page<Coin> page);   */\*\*  \* 使用币种的状态查询所有的币种信息  \* @param status  \* @return  \*/* List<Coin> getCoinsByStatus(Byte status); } |

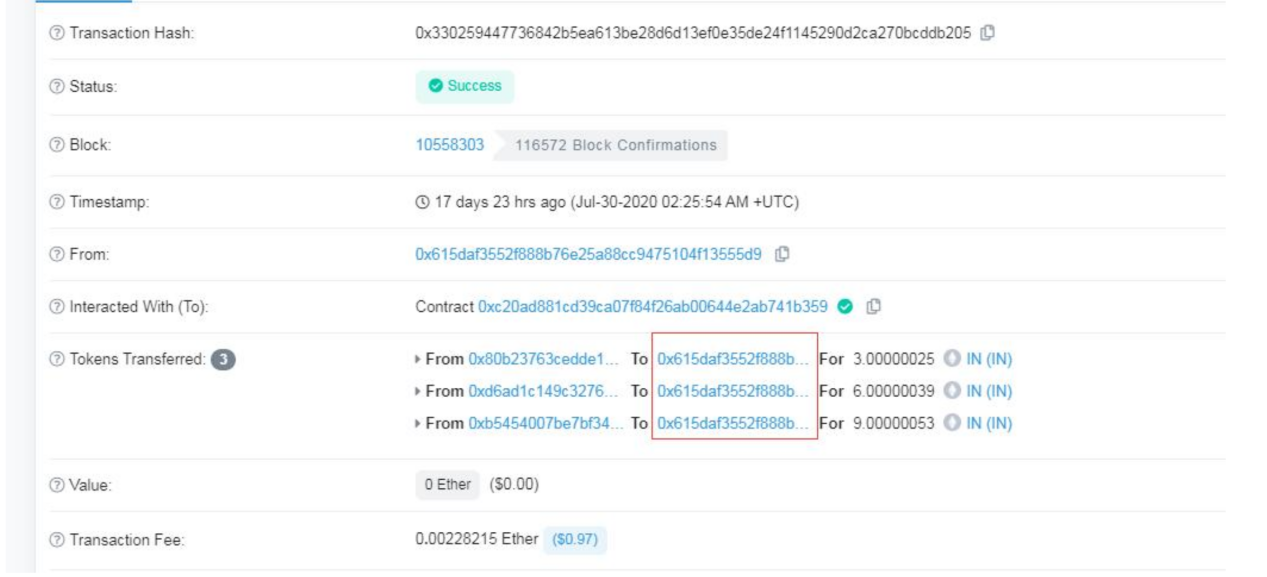
### CoinServiceImpl

|  |
| --- |
| @Service public class CoinServiceImpl extends ServiceImpl<CoinMapper, Coin> implements CoinService {   */\*\*  \* 数字货币的条件分页查询  \*  \* @param name 数字货币的名称  \* @param type 数字货币类型的名称  \* @param status 数字货币的状态  \* @param title 字货币的标题  \* @param walletType 树字货币的钱包类型名称  \* @param page 分页参数  \* @return 数据货币的分页数据  \*/* @Override  public Page<Coin> findByPage(String name, String type, Byte status, String title, String walletType, Page<Coin> page) {  return page(page,  new LambdaQueryWrapper<Coin>()  .like(!StringUtils.*isEmpty*(name), Coin::getName, name) *// 名称的查询* .like(!StringUtils.*isEmpty*(title), Coin::getTitle, title) *// 标题的查询* .eq(status != null, Coin::getStatus, status) *// 状态的查询* .eq(!StringUtils.*isEmpty*(type), Coin::getType, type) *// 货币类型名称的查询* .eq(!StringUtils.*isEmpty*(walletType), Coin::getWallet, walletType) *// 货币钱包类型的查询* );  }   */\*\*  \* 使用币种的状态查询所有的币种信息  \*  \* @param status  \* @return  \*/* @Override  public List<Coin> getCoinsByStatus(Byte status) {  return list(new LambdaQueryWrapper<Coin>().eq(Coin::getStatus,status));  } } |

## 钱包的归集提币地址

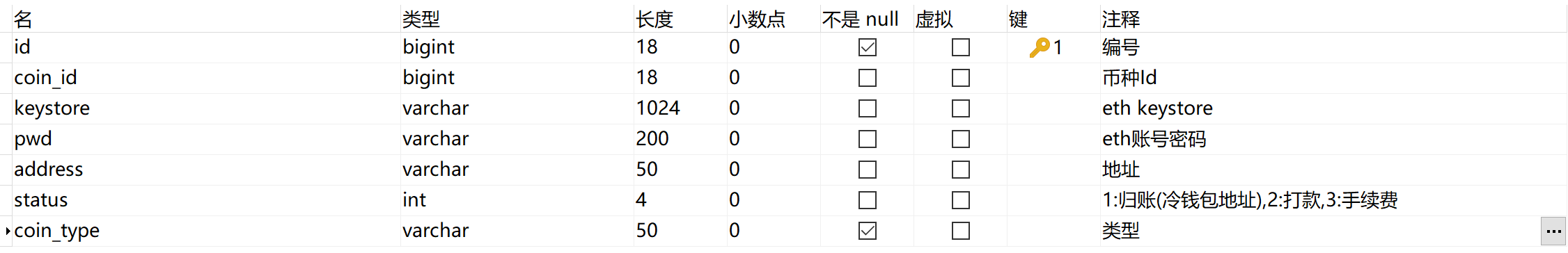
在交易系统里面在给大家详细的解释:

比如：有三个地址上面的代币或者usdt需要归集到一个地址，无需再往这三个地址里面转入eth矿工费。直接发起归集，这个三个地址的币。就直接归集到指定的地址；

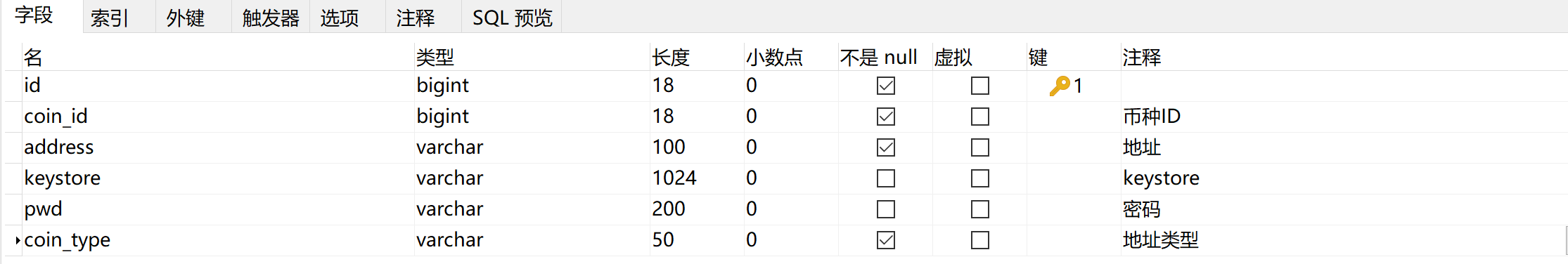


### 表的设计

#### admin\_address:

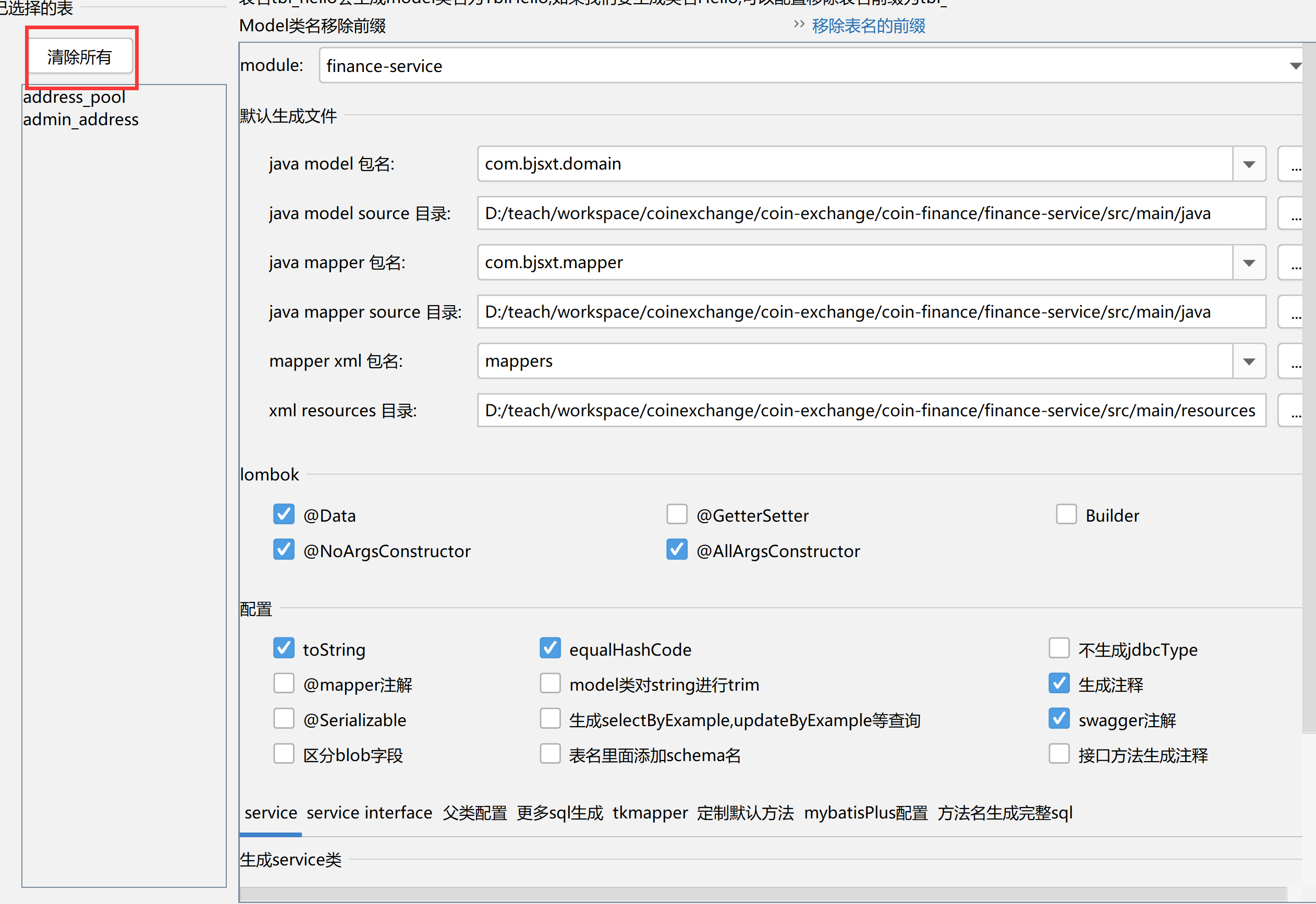


#### address\_pool:



### 代码的生成

配置和之前代码生成没有任何区别



### 5.3.3 AdminAddressController

|  |
| --- |
| @RestController @RequestMapping("/adminAddress") @Api(tags = "归集地址的控制器") public class AdminAddressController {    @Autowired  private AdminAddressService adminAddressService;   @GetMapping  @ApiOperation(value = "查询归集地址")  @ApiImplicitParams({  @ApiImplicitParam(name = "coinId", value = "币种的id"),  @ApiImplicitParam(name = "current", value = "当前页"),  @ApiImplicitParam(name = "size", value = "每页显示的条数"),  })  public R<Page<AdminAddress>> findByPage(@ApiIgnore Page<AdminAddress> page, Long coinId) {  Page<AdminAddress> adminAddressPage = adminAddressService.findByPage(page, coinId);  return R.*ok*(adminAddressPage);  }    @PostMapping  @ApiOperation(value = "归集地址的新增")  @ApiImplicitParams({  @ApiImplicitParam(name = "adminAddress" ,value = "adminAddress json")  })  public R save(@RequestBody @Validated AdminAddress adminAddress) {  boolean save = adminAddressService.save(adminAddress);  if (save) {  return R.*ok*();  }  return R.*fail*("新增失败");  }    @PatchMapping  @ApiOperation(value = "归集地址的修改")  @ApiImplicitParams({  @ApiImplicitParam(name = "adminAddress" ,value = "adminAddress json")  })  public R update(@RequestBody @Validated AdminAddress adminAddress) {  boolean update = adminAddressService.updateById(adminAddress);  if (update) {  return R.*ok*();  }  return R.*fail*("修改失败");  } } |

### 5.3.4AdminAddressService

|  |
| --- |
| public interface AdminAddressService extends IService<AdminAddress>{    */\*\*  \* 条件分页查询归集地址  \* @param page  \* 分页参数  \* @param coinId  \* 币种的Id  \* @return  \*/* Page<AdminAddress> findByPage(Page<AdminAddress> page, Long coinId); } |

### 5.3.5AdminAddressServiceImpl

|  |
| --- |
| @Service  public class AdminAddressServiceImpl extends ServiceImpl<AdminAddressMapper, AdminAddress> implements AdminAddressService{  @Autowired  private CoinService coinService ;  /\*\*  \* 条件分页查询归集地址  \*  \* @param page 分页参数  \* @param coinId 币种的Id  \* @return  \*/  @Override  public Page<AdminAddress> findByPage(Page<AdminAddress> page, Long coinId) {  return page(page,new LambdaQueryWrapper<AdminAddress>().eq(coinId!=null ,AdminAddress::getCoinId,coinId));  }  /\*\*  \* 重新save ,为了让我们的归集地址里面包含coinType  \* @param entity  \* @return  \*/  @Override  public boolean save(AdminAddress entity) {  Long coinId = entity.getCoinId();  Coin coin = coinService.getById(coinId);  if(coin==null){  throw new IllegalArgumentException("输入的币种id错误") ;  }  String type = coin.getType();  entity.setCoinType(type);  return super.save(entity) ;  }  } |

### 5.3.6AdminAddress

|  |
| --- |
| */\*\*  \* 平台归账手续费等账户  \*/* @ApiModel(value="com-bjsxt-domain-AdminAddress") @Data @AllArgsConstructor @NoArgsConstructor @TableName(value = "admin\_address") public class AdminAddress {  */\*\*  \* 编号  \*/* @TableId(value = "id", type = IdType.*AUTO*)  @ApiModelProperty(value="编号")  private Long id;   */\*\*  \* 币种Id  \*/* @TableField(value = "coin\_id")  @ApiModelProperty(value="币种Id")  @NotNull  private Long coinId;   */\*\*  \* eth keystore  \*/* @TableField(value = "keystore")  @ApiModelProperty(value="eth keystore")  private String keystore;   */\*\*  \* eth账号密码  \*/* @TableField(value = "pwd")  @ApiModelProperty(value="eth账号密码")  private String pwd;   */\*\*  \* 地址  \*/* @TableField(value = "address")  @ApiModelProperty(value="地址")  private String address;   */\*\*  \* 1:归账(冷钱包地址),2:打款,3:手续费  \*/* @TableField(value = "status")  @ApiModelProperty(value="1:归账(冷钱包地址),2:打款,3:手续费")  private Integer status;   */\*\*  \* 类型  \*/* @TableField(value = "coin\_type")  @ApiModelProperty(value="类型") *// @NotBlank* private String coinType; } |

## 新增币种信息

新增币种: 包含: 1新增币种,2新增币种的配置信息,3新增币种的归集地址

### CoinController

|  |
| --- |
| @PatchMapping @ApiOperation(value = "修改我们的币种的信息") @ApiImplicitParams({  @ApiImplicitParam(name = "coin" ,value = "coin的json数据") }) public R update(@RequestBody @Validated Coin coin){  boolean updateById = coinService.updateById(coin);  if(updateById){  return R.*ok*() ;  }  return R.*fail*("修改失败") ; }   @PostMapping @ApiOperation(value = "新增我们的币种的信息") @ApiImplicitParams({  @ApiImplicitParam(name = "coin" ,value = "coin的json数据") }) public R<Coin> save(@RequestBody @Validated Coin coin){  coin.setStatus((byte)1);  boolean save = coinService.save(coin);  *// coin新增成功后,会有Id ,这是mybatis-plus在新增成功后,  // 会自动的进行一个sql语句的查询,查询的结果就是id,之后把id设置给coin* if(save){  return R.*ok*(coin) ;  }  return R.*fail*("新增失败") ; } |

### CoinConfigController

|  |
| --- |
| @RestController @RequestMapping("/coinConfigs") @Api(tags = "币种配置的控制器") public class CoinConfigController {   @Autowired  private CoinConfigService coinConfigService ;    @GetMapping("/info/{coinId}")  @ApiOperation(value = "查询币种的配置信息")  @ApiImplicitParams({  @ApiImplicitParam(name = "coinId" ,value = "币种的id值")  })  public R<CoinConfig> getCoinConfig(@PathVariable("coinId") Long coinId){  CoinConfig coinConfig = coinConfigService.findByCoinId(coinId) ;  return R.*ok*(coinConfig) ;  }    @PatchMapping  @ApiOperation(value = "币种配置的修改操作")  @ApiImplicitParams({  @ApiImplicitParam(name = "coinConfig" ,value ="coinConfig的json数据" )  })  public R update(@RequestBody @Validated CoinConfig coinConfig){  boolean saveOrUpdate = coinConfigService.updateOrSave(coinConfig) ;  if(saveOrUpdate){  return R.*ok*() ;  }  return R.*fail*("修改失败") ;  } } |

### CoinConfigService

|  |
| --- |
| public interface CoinConfigService extends IService<CoinConfig> {    */\*\*  \* 通过币种的id 查询币种的配置信息  \* @param coinId  \* 币种的id  \* @return  \* 币种的配置信息  \*/* CoinConfig findByCoinId(Long coinId);   */\*\*  \* 新增或修改币种配置  \* @param coinConfig  \* @return  \*/* boolean updateOrSave(CoinConfig coinConfig); } |

### CoinConfigServiceImpl

|  |
| --- |
| @Service public class CoinConfigServiceImpl extends ServiceImpl<CoinConfigMapper, CoinConfig> implements CoinConfigService {   @Autowired  private CoinService coinService;   */\*\*  \* 通过币种的id 查询币种的配置信息  \*  \* @param coinId 币种的id  \* @return 币种的配置信息  \*/* @Override  public CoinConfig findByCoinId(Long coinId) {  *// coinConfig的id 和Coin的id 值是相同的* return getOne(new LambdaQueryWrapper<CoinConfig>().eq(CoinConfig::getId, coinId));  }   */\*\*  \* 新增或修改币种配置  \*  \* @param coinConfig  \* @return  \*/* @Override  public boolean updateOrSave(CoinConfig coinConfig) {  *//* Coin coin = coinService.getById(coinConfig.getId());  if(coin==null){  throw new IllegalArgumentException("coin-Id不存在") ;  }  coinConfig.setCoinType(coin.getType());  coinConfig.setName(coin.getName());  *// 如何是新增/修改呢?* CoinConfig config = getById(coinConfig.getId());  if (config == null) { *// 新增操作* return save(coinConfig);  } else { *// 修改操作* return updateById(coinConfig);  }  } } |

### CoinConfig

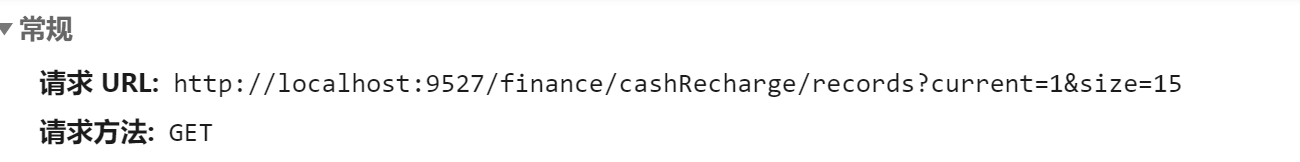
|  |
| --- |
| @ApiModel(value = "com-bjsxt-domain-CoinConfig") @Data @AllArgsConstructor @NoArgsConstructor @TableName(value = "coin\_config") public class CoinConfig {  */\*\*  \* 币种ID(对应coin表ID)  \*/* @TableId(value = "id", type = IdType.*INPUT*)  @ApiModelProperty(value = "币种ID(对应coin表ID)")  @NotNull  private Long id;   */\*\*  \* 币种名称  \*/* @TableField(value = "name")  @ApiModelProperty(value = "币种名称") *// @NotBlank* private String name;   */\*\*  \* btc-比特币系列；eth-以太坊；ethToken-以太坊代币；etc-以太经典；\r\n\r\n  \*/* @TableField(value = "coin\_type")  @ApiModelProperty(value = "btc-比特币系列；eth-以太坊；ethToken-以太坊代币；etc-以太经典；\r\n\r\n") *// @NotBlank* private String coinType;   */\*\*  \* 钱包最低留存的币  \*/* @TableField(value = "credit\_limit")  @ApiModelProperty(value = "钱包最低留存的币")  @NotNull  private BigDecimal creditLimit;   */\*\*  \* 当触发改状态的时候,开始归集  \*/* @TableField(value = "credit\_max\_limit")  @ApiModelProperty(value = "当触发改状态的时候,开始归集")  private BigDecimal creditMaxLimit;   */\*\*  \* rpc服务ip  \*/* @TableField(value = "rpc\_ip")  @ApiModelProperty(value = "rpc服务ip")  @NotBlank  private String rpcIp;   */\*\*  \* rpc服务port  \*/* @TableField(value = "rpc\_port")  @ApiModelProperty(value = "rpc服务port")  @NotBlank  private String rpcPort;   */\*\*  \* rpc用户  \*/* @TableField(value = "rpc\_user")  @ApiModelProperty(value = "rpc用户")  private String rpcUser;   */\*\*  \* rpc密码  \*/* @TableField(value = "rpc\_pwd")  @ApiModelProperty(value = "rpc密码")  private String rpcPwd;   */\*\*  \* 最后一个区块  \*/* @TableField(value = "last\_block")  @ApiModelProperty(value = "最后一个区块")  private String lastBlock;   */\*\*  \* 钱包用户名  \*/* @TableField(value = "wallet\_user")  @ApiModelProperty(value = "钱包用户名")  private String walletUser;   */\*\*  \* 钱包密码  \*/* @TableField(value = "wallet\_pass")  @ApiModelProperty(value = "钱包密码")  private String walletPass;   */\*\*  \* 代币合约地址  \*/* @TableField(value = "contract\_address")  @ApiModelProperty(value = "代币合约地址")  private String contractAddress;   */\*\*  \* context  \*/* @TableField(value = "context")  @ApiModelProperty(value = "context")  private String context;   */\*\*  \* 最低确认数  \*/* @TableField(value = "min\_confirm")  @ApiModelProperty(value = "最低确认数")  private Integer minConfirm;   */\*\*  \* 定时任务  \*/* @TableField(value = "task")  @ApiModelProperty(value = "定时任务")  private String task;   */\*\*  \* 是否可用0不可用,1可用  \*/* @TableField(value = "status")  @ApiModelProperty(value = "是否可用0不可用,1可用")  private Integer status;   @TableField(value = "auto\_draw\_limit")  @ApiModelProperty(value = "")  private BigDecimal autoDrawLimit;   @TableField(value = "auto\_draw")  @ApiModelProperty(value = "")  private Integer autoDraw; |

# 后台系统-资金明细



## 6.1 GCN充值记录查询

### 6.1.1 请求的接口





### 6.1.2 CashRechargeController

|  |
| --- |
| @RestController @RequestMapping("/cashRecharges") @Api(tags = "GCN充值控制器") public class CashRechargeController {    @Autowired  private CashRechargeService cashRechargeService;   @GetMapping("/records")  @ApiOperation(value = "条件分页查询")  @ApiImplicitParams({  @ApiImplicitParam(name = "current", value = "当前页"),  @ApiImplicitParam(name = "size", value = "每页显示的条数"),  @ApiImplicitParam(name = "coinId", value = "币种的Id"),  @ApiImplicitParam(name = "userId", value = "用户的Id"),  @ApiImplicitParam(name = "userName", value = "用户的名称"),  @ApiImplicitParam(name = "mobile", value = "用户的手机号"),  @ApiImplicitParam(name = "status", value = "充值审核状态"),  @ApiImplicitParam(name = "numMin", value = "充值最小金额"),  @ApiImplicitParam(name = "numMax", value = "充值最大金额"),  @ApiImplicitParam(name = "startTime", value = "充值开始时间"),  @ApiImplicitParam(name = "endTime", value = "充值结束时间"),   })  public R<Page<CashRecharge>> findByPage(  @ApiIgnore Page<CashRecharge> page,  Long coinId, Long userId, String userName,  String mobile, Byte status, String numMin, String numMax,  String startTime, String endTime  ) {  Page<CashRecharge> pageData = cashRechargeService.findByPage(page, coinId, userId, userName, mobile, status, numMin, numMax, startTime, endTime);  return R.*ok*(pageData);  } } |

### 6.1.3 CashRechargeService

|  |
| --- |
| public interface CashRechargeService extends IService<CashRecharge> {    */\*\*  \* 条件分页查询  \* @param page  \* 分页参数  \* @param coinId  \* 币种的ID  \* @param userId  \* 用户的Id  \* @param userName  \* 用户的名称  \* @param mobile  \* 用户的手机号  \* @param status  \* 审核的状态  \* @param numMin  \* 充值数量的最小值  \* @param numMax  \* 充值数量的最大值  \* @param startTime  \* 充值的开始时间  \* @param endTime  \* 充值数量的结束时间  \* @return  \*/* Page<CashRecharge> findByPage(Page<CashRecharge> page, Long coinId, Long userId, String userName, String mobile,  Byte status, String numMin, String numMax, String startTime, String endTime); } |

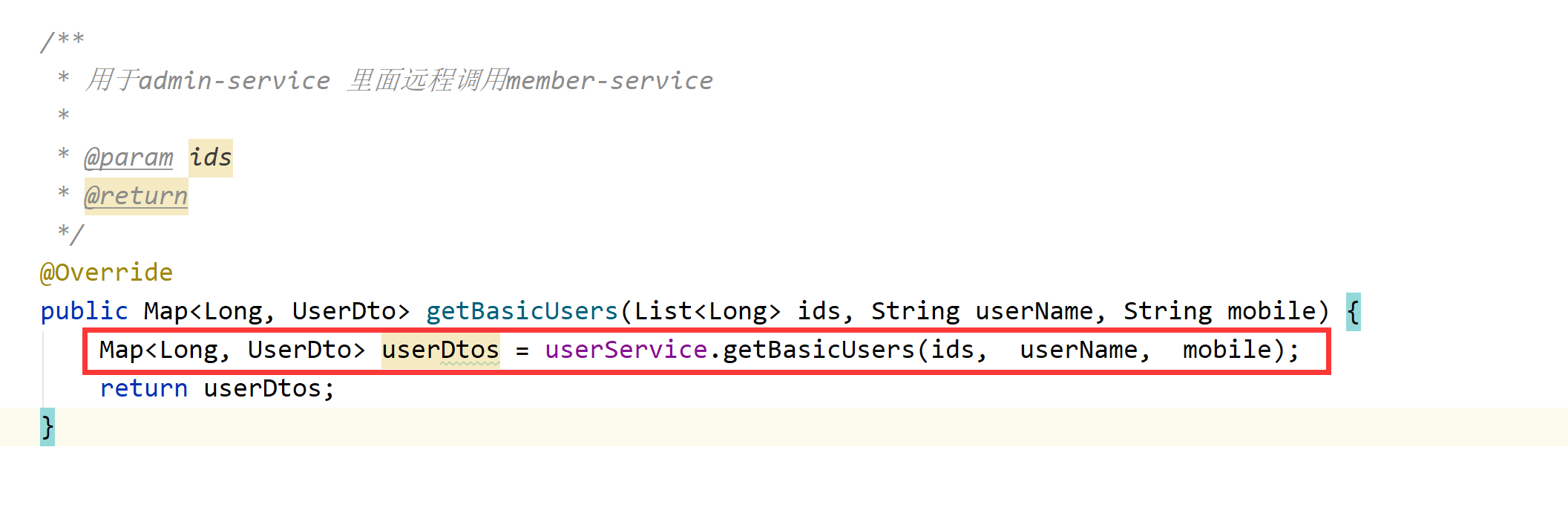
### 6.1.4 CashRechargeServiceImpl

|  |
| --- |
| @Service public class CashRechargeServiceImpl extends ServiceImpl<CashRechargeMapper, CashRecharge> implements CashRechargeService {   @Autowired  private UserServiceFeign userServiceFeign;   */\*\*  \* 条件分页查询  \*  \* @param page 分页参数  \* @param coinId 币种的ID  \* @param userId 用户的Id  \* @param userName 用户的名称  \* @param mobile 用户的手机号  \* @param status 审核的状态  \* @param numMin 充值数量的最小值  \* @param numMax 充值数量的最大值  \* @param startTime 充值的开始时间  \* @param endTime 充值数量的结束时间  \* @return  \*/* @Override  public Page<CashRecharge> findByPage(Page<CashRecharge> page, Long coinId, Long userId, String userName,  String mobile, Byte status, String numMin, String numMax, String startTime,  String endTime) {  LambdaQueryWrapper<CashRecharge> cashRechargeLambdaQueryWrapper = new LambdaQueryWrapper<>();  *// 1 使用用户相关的字段进行查询* Map<Long, UserDto> basicUsers = null;  if (userId != null || !StringUtils.*isEmpty*(userName) || !StringUtils.*isEmpty*(mobile)) {  basicUsers = userServiceFeign.getBasicUsers(userId == null ? null : Arrays.*asList*(userId), userName, mobile);  if (CollectionUtils.*isEmpty*(basicUsers)) { *// 没有用户* return page;  }  cashRechargeLambdaQueryWrapper.in(CashRecharge::getUserId, basicUsers.keySet()); *// 使用用户的信息做条件* }  *// 添加其他的条件* cashRechargeLambdaQueryWrapper.eq(coinId != null, CashRecharge::getCoinId, coinId)  .eq(status != null, CashRecharge::getStatus, status)  .between(  !(StringUtils.*isEmpty*(numMin) || StringUtils.*isEmpty*(numMax)),  CashRecharge::getNum,  new BigDecimal(numMin==null? "0" :numMin), new BigDecimal(numMax==null? "0" :numMax)  )  .between(  !(StringUtils.*isEmpty*(startTime) || StringUtils.*isEmpty*(endTime)),  CashRecharge::getCreated,  startTime, endTime + "23:23:59"  );  *// 查询* Page<CashRecharge> pageData = page(page, cashRechargeLambdaQueryWrapper);  *// 获取查询的数据* List<CashRecharge> records = pageData.getRecords();  if(!CollectionUtils.*isEmpty*(records)){  if(basicUsers==null){ *// 说明前面没有使用用户的信息查询用户* List<Long> userIds = records.stream().map(cashRecharge -> cashRecharge.getUserId()).collect(Collectors.*toList*());  basicUsers = userServiceFeign.getBasicUsers(userIds,null,null) ;  }  Map<Long, UserDto> finalBasicUsers = basicUsers;  records.forEach(record->{  UserDto userDto = finalBasicUsers.get(record.getUserId());  if(userDto!=null){  record.setUsername(userDto.getUsername());  record.setRealName(userDto.getRealName());  }  });  }  return pageData ;  } } |

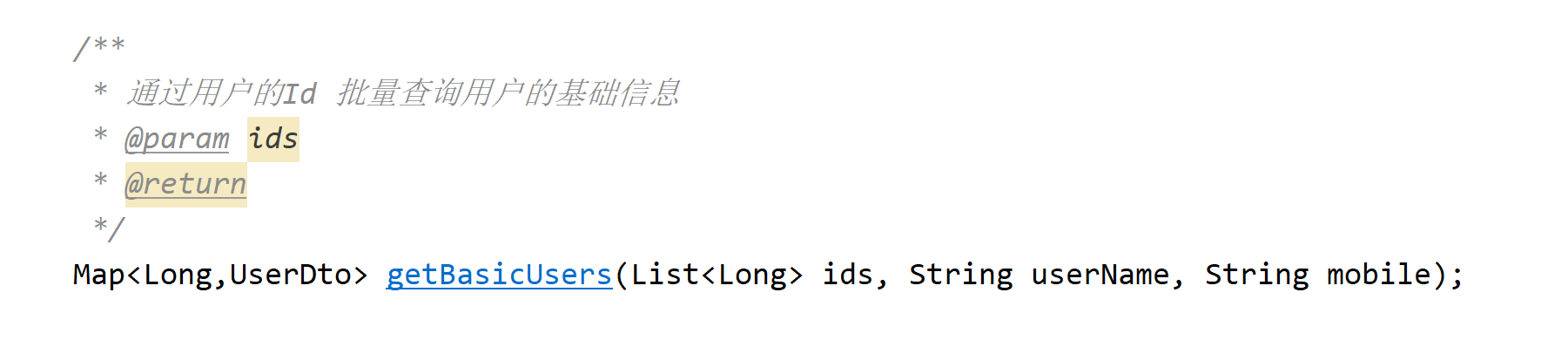
### 6.1.5 对UserServiceFeign的改造

|  |
| --- |
| @FeignClient(name = "member-service",configuration = OAuth2FeignConfig.class ,path = "/users") public interface UserServiceFeign {   */\*\*  \* 用于admin-service 里面远程调用member-service  \* @param ids  \* @return  \*/* @GetMapping("/basic/users")  Map<Long,UserDto> getBasicUsers(@RequestParam(value = "ids",required = false) List<Long> ids,  @RequestParam(value = "username",required = false)String username,  @RequestParam(value = "mobile",required = false)String mobile) ; } |

### 6.1.6 对UserController的改造



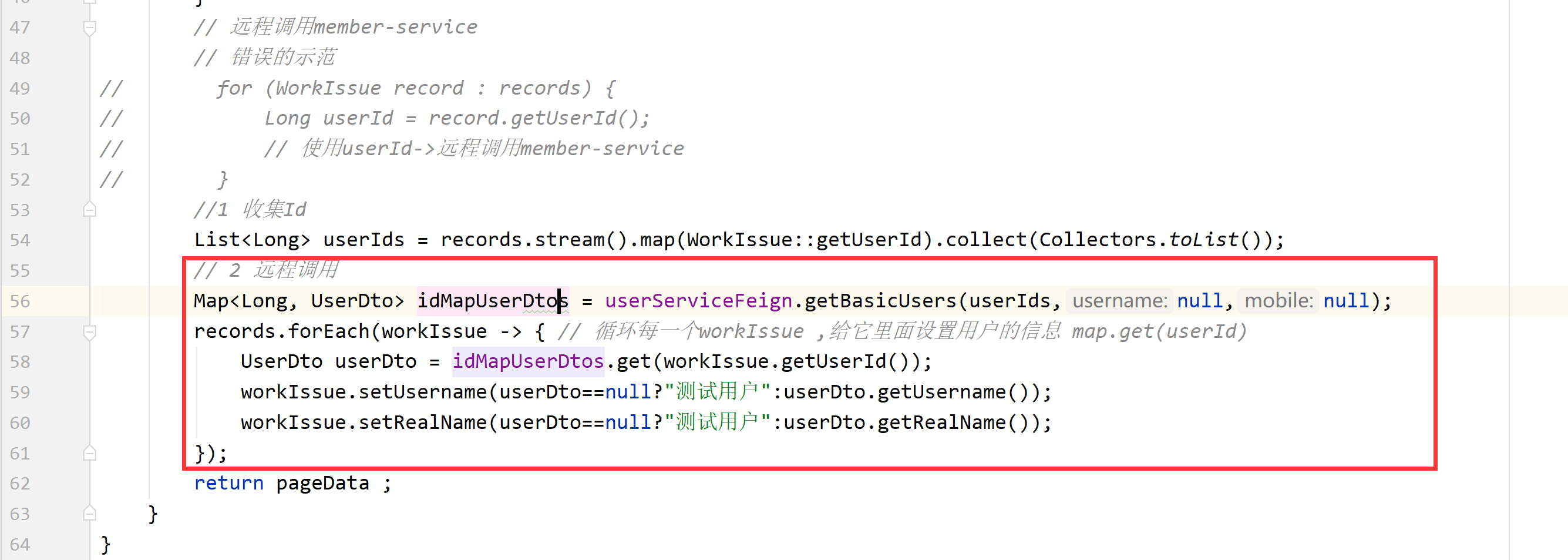
### 6.1.7 对UserService的改造



### 6.1.8 对UserServiceImpl的改造

|  |
| --- |
| */\*\*  \* 通过用户的Id 批量查询用户的基础信息  \*  \* @param ids  \* @return  \*/* @Override public Map<Long, UserDto> getBasicUsers(List<Long> ids, String userName, String mobile) {  if (CollectionUtils.*isEmpty*(ids) && StringUtils.*isEmpty*(userName) && StringUtils.*isEmpty*(mobile)) {  return Collections.*emptyMap*();  }  List<User> list = list(new LambdaQueryWrapper<User>()  .in(!CollectionUtils.*isEmpty*(ids), User::getId, ids)  .like(!StringUtils.*isEmpty*(userName), User::getUsername, userName)  .like(!StringUtils.*isEmpty*(mobile), User::getMobile, mobile));  if (CollectionUtils.*isEmpty*(list)) {  return Collections.*emptyMap*();  }  *// 将user->userDto* List<UserDto> userDtos = UserDtoMapper.*INSTANCE*.convert2Dto(list);  Map<Long, UserDto> userDtoMaps = userDtos.stream().collect(Collectors.*toMap*(UserDto::getId, userDto -> userDto));  return userDtoMaps; } |

### 6.1.9 对WorkIssueServiceImpl的影响



## 6.2 GCN提现记录查询

### 6.2.1 API请求的接口

## 6.3 充币记录查询

## 6.4 提币记录查询

## 6.5 流水记录查询

# 后台系统-财务管理



## 7.1场外交易充值审核

7.1.1 API 接口展示

7.1.2 CashRechargeController

7.1.2 CashRechargeService

7.1.2 CashRechargeService

7.1.2 CashRecharge

## 7.2场外交易提现审核

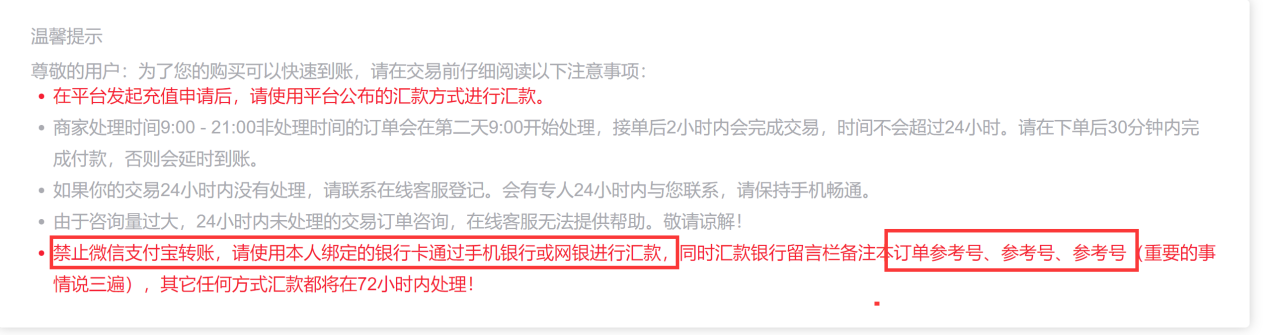
## 7.3数字货币提现审核

# GCN数字货币操作

从现在开始,我们聊聊数字货币的交易问题:在做币币交易时,绝对不允许进行人民币和数字货币的兑换操作,因此,我们需要一个别的思路:就是先将人民币兑换成我们系统的一种虚拟货币(可以认为是某种积分).然后使用虚拟币和其他币种进行兑换操作就可以了.

但是我们必须注意到:这种人民币兑换积分的方式很容易受到支付宝或微信的风控监测,因此,我们需要直接使用银行卡手动转账的方式来进行操作.

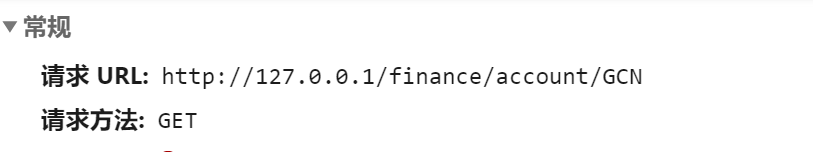






## 个人GCN资产的查询操作

### API



### AccountController

|  |
| --- |
| @RestController @RequestMapping("/account") @Api(tags = "资产服务的控制器") public class AccountController {    @Autowired  private AccountService accountService ;   @GetMapping("/{coinName}")  @ApiOperation(value = "获取当前用户的货币的资产情况")  @ApiImplicitParams({  @ApiImplicitParam(name = "coinName" ,value = "货币的名称")  })  public R<Account> getUserAccount(@PathVariable("coinName")String coinName){  Long userId = Long.*valueOf*(SecurityContextHolder.*getContext*().getAuthentication().getPrincipal().toString()) ;  Account account = accountService.findByUserAndCoin(userId,coinName) ;  return R.*ok*(account) ;  } } |

### AccountService

|  |
| --- |
| public interface AccountService extends IService<Account> {   */\*\*  \* 查询某个用户的货币资产  \* @param userId  \* 用户的id  \* @param coinName  \* 货币的名称  \* @return  \*/* Account findByUserAndCoin(Long userId, String coinName); } |

### AccountServiceImpl

|  |
| --- |
| @Service public class AccountServiceImpl extends ServiceImpl<AccountMapper, Account> implements AccountService {   @Autowired  private CoinService coinService ;   @Autowired  private ConfigService configService ;  */\*\*  \* 查询某个用户的货币资产  \*  \* @param userId 用户的id  \* @param coinName 货币的名称  \* @return  \*/* @Override  public Account findByUserAndCoin(Long userId, String coinName) {  Coin coin = coinService.getCoinByCoinName(coinName) ;  if(coin==null){  throw new IllegalArgumentException("货币不存在") ;  }  Account account = getOne(new LambdaQueryWrapper<Account>()  .eq(Account::getUserId, userId)  .eq(Account::getCoinId, coin.getId())  );  if(account==null){  throw new IllegalArgumentException("该资产不存在") ;  }   Config sellRateConfig = configService.getConfigByCode("USDT2CNY");  account.setSellRate(new BigDecimal(sellRateConfig.getValue())); *// 出售的费率* Config setBuyRateConfig = configService.getConfigByCode("CNY2USDT");  account.setBuyRate(new BigDecimal(setBuyRateConfig.getValue())); *// 买进来的费率* return account ;  } } |

### CoinService

|  |
| --- |
| */\*\*  \* 使用货币的名称来查询货币  \* @param coinName  \* @return  \*/* Coin getCoinByCoinName(String coinName); |

### CoinServiceImpl

|  |
| --- |
| */\*\*  \* 使用货币的名称来查询货币  \* 货币的名称是唯一  \* @param coinName  \*  \* @return  \*/* @Override public Coin getCoinByCoinName(String coinName) {  return getOne(new LambdaQueryWrapper<Coin>().eq(Coin::getName,coinName)); } |

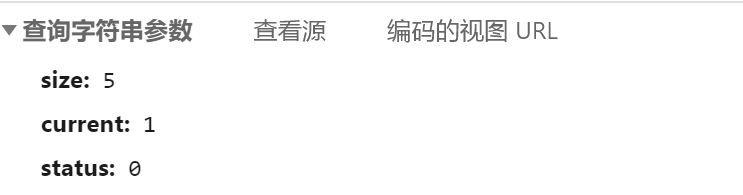
## 买入记录和卖出记录的查询

### API接口

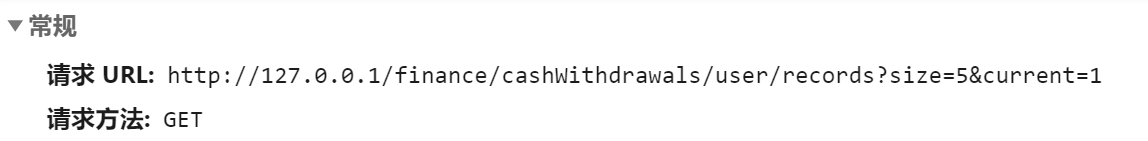


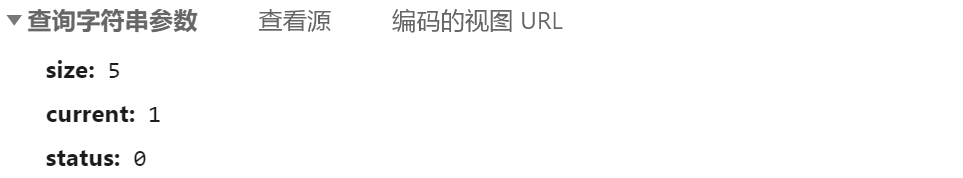
买入:





卖出:





### Controller

#### CashRechargeController:

|  |
| --- |
| @GetMapping("/user/records")  @ApiOperation(value = "查询当前用户的充值记录")  @ApiImplicitParams({  @ApiImplicitParam(name = "current",value = "当前页") ,  @ApiImplicitParam(name = "size",value = "每页显示的大小") ,  @ApiImplicitParam(name = "status",value = "充值的状态") ,  })  public R<Page<CashRecharge>> findUserCashRecharge(@ApiIgnore Page<CashRecharge> page ,Byte status){  Long userId = Long.*valueOf*(SecurityContextHolder.*getContext*().getAuthentication().getPrincipal().toString()) ;  Page<CashRecharge> cashRechargePage = cashRechargeService.findUserCashRecharge(page ,userId,status) ;  return R.*ok*(cashRechargePage) ;  } |

#### CashWithdrawalsController:

|  |
| --- |
| @GetMapping("/user/records")  @ApiOperation(value = "查询当前用户的充值记录")  @ApiImplicitParams({  @ApiImplicitParam(name = "current",value = "当前页") ,  @ApiImplicitParam(name = "size",value = "每页显示的大小") ,  @ApiImplicitParam(name = "status",value = "充值的状态") ,  })  public R<Page<CashWithdrawals>> findUserCashRecharge(@ApiIgnore Page<CashWithdrawals> page , Byte status){  Long userId = Long.*valueOf*(SecurityContextHolder.*getContext*().getAuthentication().getPrincipal().toString()) ;  Page<CashWithdrawals> cashWithdrawalsPage = cashWithdrawalsService.findCashWithdrawals(page ,userId,status) ;  return R.*ok*(cashWithdrawalsPage) ;  } |

### Service

#### CashRechargeService:

|  |
| --- |
| */\*\*  \* 查询当前用户的充值的数据  \* @param page  \* 分页对象  \* @param userId  \* 用户的Id  \* @param status  \* 订单的状态  \* @return  \*/* Page<CashRecharge> findUserCashRecharge(Page<CashRecharge> page, Long userId, Byte status); |

#### CashWithdrawalsService:

|  |
| --- |
| */\*\*  \* 查询用户的提现记录  \* @param page  \* @param userId  \* @param status  \* @return  \*/* Page<CashWithdrawals> findCashWithdrawals(Page<CashWithdrawals> page, Long userId, Byte status); |

### ServiceImpl

#### CashRechargeServiceImpl:

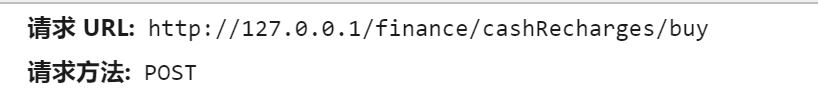
|  |
| --- |
| */\*\*  \* 查询当前用户的充值的数据  \*  \* @param page 分页对象  \* @param userId 用户的Id  \* @param status 订单的状态  \* @return  \*/* @Override public Page<CashRecharge> findUserCashRecharge(Page<CashRecharge> page, Long userId, Byte status) {  return page(page ,new LambdaQueryWrapper<CashRecharge>()  .eq(CashRecharge::getUserId,userId)  .eq(status!=null ,CashRecharge::getStatus ,status)  ); } |

#### CashWithdrawalsServiceImpl:

|  |
| --- |
| */\*\*  \* 查询用户的提现记录  \*  \* @param page  \* @param userId  \* @param status  \* @return  \*/* @Override public Page<CashWithdrawals> findCashWithdrawals(Page<CashWithdrawals> page, Long userId, Byte status) {  return page(page,new LambdaQueryWrapper<CashWithdrawals>()  .eq(CashWithdrawals::getUserId,userId)  .eq(status!=null,CashWithdrawals::getStatus,status)); } |

## GCN买入操作

### API接口





### CashParam

|  |
| --- |
| @Data @ApiModel(value = "现金充值参数") public class CashParam {   @ApiModelProperty(value = "币种的ID")  @NotNull  private Long coinId ;   @ApiModelProperty(value = "购买的数量")  @NotNull  private BigDecimal num ;   @ApiModelProperty(value = "实际金额")  @NotNull  private BigDecimal mum ; } |

### CashTradeVo



|  |
| --- |
| */\*\*  \* 收款方户名吴志锋  \* 收款方开户行北京银行  \* 收款方账号6214680131522508  \* 转账金额￥100000  \* 参考号515685(汇款务必备注)  \* 状态待审核  \*/* @Data @ApiModel(value = "现金交易的结果") public class CashTradeVo {   @ApiModelProperty(value = "收款方户名")  private String name ;   @ApiModelProperty(value = "收款方开户行")  private String bankName ;   @ApiModelProperty(value = "收款方账号")  private String bankCard ;   @ApiModelProperty(value = "转账金额")  private BigDecimal amount ;   @ApiModelProperty(value = "参考号")  private String remark ;   @ApiModelProperty(value = "状态")  private Byte status ; } |

### CashRechargeController

|  |
| --- |
| @PostMapping("/buy") @ApiOperation(value = "GCN的充值操作") @ApiImplicitParams({  @ApiImplicitParam( name = "cashParam",value = "现金交易的参数") }) public R<CashTradeVo> buy(@RequestBody @Validated CashParam cashParam){  Long userId = Long.*valueOf*(SecurityContextHolder.*getContext*().getAuthentication().getPrincipal().toString()) ;  CashTradeVo cashTradeVo = cashRechargeService.buy(userId,cashParam) ;  return R.*ok*(cashTradeVo) ; } |

### CashRechargeService

|  |
| --- |
| */\*\*  \* 进行一个GCN/充值/购买  \* @param userId  \* 用户的id  \* @param cashParam  \* 现金参数  \* @return  \*/* CashTradeVo buy(Long userId, CashParam cashParam); |

### CashRechargeServiceImpl

|  |
| --- |
| */\*\*  \* 进行一个GCN/充值/购买  \*  \* @param userId 用户的id  \* @param cashParam 现金参数  \* @return  \*/* @Override public CashTradeVo buy(Long userId, CashParam cashParam) {  *//1 校验现金参数* checkCashParam(cashParam);  *// 2 查询我们公司的银行卡* List<AdminBankDto> allAdminBanks = adminBankServiceFeign.getAllAdminBanks();  *// 仅仅需要一张银行卡* AdminBankDto adminBankDto = loadbalancer(allAdminBanks);  *//3 生成订单号\参考号* String orderNo = String.*valueOf*(snowflake.nextId());  String remark = RandomUtil.*randomNumbers*(6);   Coin coin = coinService.getById(cashParam.getCoinId());   if (coin == null) {  throw new IllegalArgumentException("coinId不存在");  }  *//cashParam.getMum()这是前端给我们的金额,前端可能因为浏览器的缓存导致价格不准确,因此,我们需要在后台进行计算* Config buyGCNRate = configService.getConfigByCode("CNY2USDT");  BigDecimal realMum = cashParam.getMum().multiply(new BigDecimal(buyGCNRate.getValue())).setScale(2, RoundingMode.*HALF\_UP*);  *// 4 在数据库里面插入一条充值的记录* CashRecharge cashRecharge = new CashRecharge();  cashRecharge.setUserId(userId);  *// 银行卡的信息* cashRecharge.setName(adminBankDto.getName());  cashRecharge.setBankName(adminBankDto.getBankName());  cashRecharge.setBankCard(adminBankDto.getBankCard());   cashRecharge.setTradeno(orderNo);  cashRecharge.setCoinId(cashParam.getCoinId());  cashRecharge.setCoinName(coin.getName());  cashRecharge.setNum(cashParam.getNum());  cashRecharge.setMum(realMum); *// 实际的交易金额* cashRecharge.setRemark(remark);  cashRecharge.setFee(BigDecimal.*ZERO*);  cashRecharge.setType("linepay"); *// 在线支付* cashRecharge.setStatus((byte) 0); *// 待审核* cashRecharge.setStep((byte) 1);*// 第一步* boolean save = save(cashRecharge);  if (save) {  *// 5 返回我们的成功对象* CashTradeVo cashTradeVo = new CashTradeVo();  *// 我们收户的银行卡信息* cashTradeVo.setAmount(realMum);  cashTradeVo.setStatus((byte)0);  cashTradeVo.setName(adminBankDto.getName());  cashTradeVo.setBankName(adminBankDto.getBankName());  cashTradeVo.setBankCard(adminBankDto.getBankCard());  cashTradeVo.setRemark(remark);  return cashTradeVo;  }  return null; }  */\*\*  \* 从一个list 里面随机选一个出来  \* @param allAdminBanks  \* @return  \*/* private AdminBankDto loadbalancer(List<AdminBankDto> allAdminBanks) {  if (CollectionUtils.*isEmpty*(allAdminBanks)) {  throw new RuntimeException("没有发现可用的银行卡");  }  int size = allAdminBanks.size();  if (size == 1) {  return allAdminBanks.get(0);  }  Random random = new Random();  return allAdminBanks.get(random.nextInt(size)); }  private void checkCashParam(CashParam cashParam) {  @NotNull BigDecimal num = cashParam.getNum(); *// 现金充值的数量* Config withDrowConfig = configService.getConfigByCode("WITH\_DROW");  @NotBlank String value = withDrowConfig.getValue();  BigDecimal minRecharge = new BigDecimal(value);  if (num.compareTo(minRecharge) < 0) {  throw new IllegalArgumentException("充值数量太小");  } } |

### 8.3.7IdConfig

|  |
| --- |
| @Configuration public class IdConfig {   @Value("${snow.app.id:1}")  private Integer appId ;   @Value("${snow.data.id:1}")  private Integer dataId ;  */\*\*  \* 雪花算法  \* @return  \*/* @Bean  public Snowflake snowflake(){  Snowflake snowflake = new Snowflake(appId,dataId);  return snowflake ;  } } |

### 8.3.8 AdminBankServiceFeign(admin-api)

|  |
| --- |
| @FeignClient(name = "admin-service",path = "/adminBanks",configuration = OAuth2FeignConfig.class) public interface AdminBankServiceFeign {   @GetMapping("/list")  List<AdminBankDto> getAllAdminBanks() ; } |

### 8.3.9 AdminBankDto(admin-api)

|  |
| --- |
| @Data @ApiModel(value = "银行卡的参数") public class AdminBankDto {   @ApiModelProperty(value = "开户行的人名称")  private String name ;   @ApiModelProperty(value = "开户行的银行名称")  private String bankName ;   @ApiModelProperty(value = "开户行的银行卡号")  private String bankCard ; } |

### 8.3.10 AdminBankController(admin-service)



|  |
| --- |
| @Override public List<AdminBankDto> getAllAdminBanks() {  List<AdminBankDto> adminBankDtos = adminBankService.getAllAdminBanks() ;  return adminBankDtos; } |

### 8.3.11 AdminBankService(admin-service)

|  |
| --- |
| */\*\*  \* 查询所有的银行开启信息  \* @return  \*/* List<AdminBankDto> getAllAdminBanks(); |

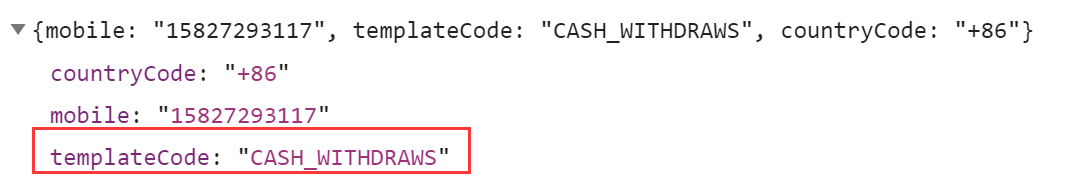
### 8.3.12 AdminBankServiceImpl(admin-service)

|  |
| --- |
| */\*\*  \* 查询所有的银行开启信息  \*  \* @return  \*/* @Override public List<AdminBankDto> getAllAdminBanks() {  List<AdminBank> adminBanks = list(new LambdaQueryWrapper<AdminBank>().eq(AdminBank::getStatus, 1));  if (CollectionUtils.*isEmpty*(adminBanks)){  return Collections.*emptyList*() ;  }  List<AdminBankDto> adminBankDtos = AdminBankDtoMappers.*INSTANCE*.toConvertDto(adminBanks);  return adminBankDtos; } |

## GCN 提现操作



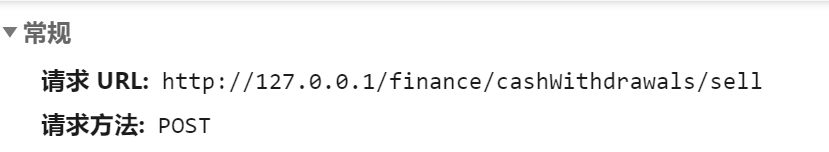
### 短信模板的创建





确定后,即可发送短信.

### 提现的API接口





### CashSellParam

|  |
| --- |
| @Data @ApiModel(value = "GCN卖出的参数") public class CashSellParam {   @ApiModelProperty(value = "要卖出的币的ID")  @NotNull  private Long coinId ;   @ApiModelProperty(value = "要卖出的币的金额")  @NotNull  private BigDecimal mum ;   @ApiModelProperty(value = "要卖出的币的数量")  @NotNull  private BigDecimal num ;   @ApiModelProperty(value = "支付密码")  @NotBlank  private String payPassword ;   @ApiModelProperty(value = "手机验证码")  @NotBlank  private String validateCode ; } |

### 8.4.4 CashWithdrawalsController

|  |
| --- |
| @PostMapping("/sell") @ApiOperation(value = "GCN的卖出操作") @ApiImplicitParams({  @ApiImplicitParam(name = "", value = "") }) public R<Object> sell(@RequestBody @Validated CashSellParam cashSellParam) {  Long userId = Long.*valueOf*(SecurityContextHolder.*getContext*().getAuthentication().getPrincipal().toString());  boolean isOk = cashWithdrawalsService.sell(userId, cashSellParam);  if (isOk) {  return R.*ok*("提交申请成功");  }  return R.*fail*("提交申请失败"); } |

### 8.4.5 CashWithdrawalsService

|  |
| --- |
| */\*\*  \* GCN的卖出操作  \* @param userId  \* @param cashSellParam  \* @return  \*/* boolean sell(Long userId, CashSellParam cashSellParam); |

### 8.4.6 CashWithdrawalsServiceImpl

|  |
| --- |
| */\*\*  \* GCN的卖出操作  \*  \* @param userId  \* @param cashSellParam  \* @return  \*/* @Override public boolean sell(Long userId, CashSellParam cashSellParam) {  *//1 参数校验* checkCashSellParam(cashSellParam);  Map<Long, UserDto> basicUsers = userServiceFeign.getBasicUsers(Arrays.*asList*(userId), null, null);  if (CollectionUtils.*isEmpty*(basicUsers)) {  throw new IllegalArgumentException("用户的id错误");  }  UserDto userDto = basicUsers.get(userId);  *// 2 手机验证码* validatePhoneCode(userDto.getMobile(),cashSellParam.getValidateCode()) ;  *// 3 支付密码* checkUserPayPassword(userDto.getPaypassword(), cashSellParam.getPayPassword());  *// 4 远程调用查询用户的银行卡* UserBankDto userBankInfo = userBankServiceFeign.getUserBankInfo(userId);  if (userBankInfo == null) {  throw new IllegalArgumentException("该用户暂未绑定银行卡信息");  }  String remark = RandomUtil.*randomNumbers*(6);  *// 5 通过数量得到本次交易的金额* BigDecimal amount = getCashWithdrawalsAmount(cashSellParam.getNum());  *// 6 计算本次的手续费* BigDecimal fee = getCashWithdrawalsFee(amount);  *// 7 查询用户的账号ID* Account account = accountService.findByUserAndCoin(userId, "GCN");  *// 7 订单的创建* CashWithdrawals cashWithdrawals = new CashWithdrawals();  cashWithdrawals.setUserId(userId);  cashWithdrawals.setAccountId(account.getId());  cashWithdrawals.setCoinId(cashSellParam.getCoinId());  cashWithdrawals.setStatus((byte) 0);  cashWithdrawals.setStep((byte) 1);  cashWithdrawals.setNum(cashSellParam.getNum());  cashWithdrawals.setMum(amount.subtract(fee)); *// 实际金额 = amount-fee* cashWithdrawals.setFee(fee);  cashWithdrawals.setBank(userBankInfo.getBank());  cashWithdrawals.setBankCard(userBankInfo.getBankCard());  cashWithdrawals.setBankAddr(userBankInfo.getBankAddr());  cashWithdrawals.setBankProv(userBankInfo.getBankProv());  cashWithdrawals.setBankCity(userBankInfo.getBankCity());  cashWithdrawals.setTruename(userBankInfo.getRealName());  cashWithdrawals.setRemark(remark);  boolean save = save(cashWithdrawals);  if (save) { *//  // 扣减总资产--account-->accountDetail* accountService.lockUserAmount(userId, cashWithdrawals.getCoinId(), cashWithdrawals.getMum(), "withdrawals\_out", cashWithdrawals.getId(), cashWithdrawals.getFee());  }  return save; }  */\*\*  \* 计算本次的手续费  \*  \* @param amount  \* @return  \*/* private BigDecimal getCashWithdrawalsFee(BigDecimal amount) {  *// 1 通过总金额\* 费率 = 手续费  // 2 若金额较小---->最小的提现的手续费   // 最小的提现费用* Config withdrawMinPoundage = configService.getConfigByCode("WITHDRAW\_MIN\_POUNDAGE");  BigDecimal withdrawMinPoundageFee = new BigDecimal(withdrawMinPoundage.getValue());   *// 提现的费率* Config withdrawPoundageRate = configService.getConfigByCode("WITHDRAW\_POUNDAGE\_RATE");    *// 通过费率计算的手续费* BigDecimal poundageFee = amount.multiply(new BigDecimal(withdrawPoundageRate.getValue())).setScale(2, RoundingMode.*HALF\_UP*);   *//min 取2 个的最小值* return poundageFee.min(withdrawMinPoundageFee).equals(poundageFee) ? withdrawMinPoundageFee : poundageFee; }  */\*\*  \* 通过数量计算金额  \*  \* @param num  \* @return  \*/* private BigDecimal getCashWithdrawalsAmount(BigDecimal num) {  *//* Config rateConfig = configService.getConfigByCode("USDT2CNY");  return num.multiply(new BigDecimal(rateConfig.getValue())).setScale(2, RoundingMode.*HALF\_UP*); }  */\*\*  \* 支付密码的校验  \*  \* @param payDBPassword  \* @param payPassword  \*/* private void checkUserPayPassword(String payDBPassword, String payPassword) {  BCryptPasswordEncoder bCryptPasswordEncoder = new BCryptPasswordEncoder();  boolean matches = bCryptPasswordEncoder.matches(payPassword, payDBPassword);  if (!matches) {  throw new IllegalArgumentException("支付密码错误");  } }  */\*\*  \* 校验用户的手机验证码  \*  \* @param mobile  \* @param validateCode  \*/* private void validatePhoneCode(String mobile, String validateCode) {   *// 验证:SMS:CASH\_WITHDRAWS:mobile* String code = redisTemplate.opsForValue().get("SMS:CASH\_WITHDRAWS:" + mobile);  if (!validateCode.equals(code)) {  throw new IllegalArgumentException("验证码错误");  }  }  */\*\*  \* 1 手机验证码  \* 2 支付密码  \* 3 提现相关的验证  \*  \* @param cashSellParam  \*/* private void checkCashSellParam(CashSellParam cashSellParam) {  *// 1 提现状态* Config cashWithdrawalsStatus = configService.getConfigByCode("WITHDRAW\_STATUS");  if (Integer.*valueOf*(cashWithdrawalsStatus.getValue()) != 1) {  throw new IllegalArgumentException("提现暂未开启");  }  *// 2 提现的金额* @NotNull BigDecimal cashSellParamNum = cashSellParam.getNum();  *// 2.1 最小的提现额度100* Config cashWithdrawalsConfigMin = configService.getConfigByCode("WITHDRAW\_MIN\_AMOUNT");  *//101* if (cashSellParamNum.compareTo(new BigDecimal(cashWithdrawalsConfigMin.getValue())) < 0) {  throw new IllegalArgumentException("检查提现的金额");  }  *// 2.1 最小的提现额度200  // 201* Config cashWithdrawalsConfigMax = configService.getConfigByCode("WITHDRAW\_MAX\_AMOUNT");  if (cashSellParamNum.compareTo(new BigDecimal(cashWithdrawalsConfigMax.getValue())) >= 0) {  throw new IllegalArgumentException("检查提现的金额");  } } |

### 8.4.7 添加UserBankDto

|  |
| --- |
| @Data @ApiModel(value = "用户银行卡的传输对象") public class UserBankDto {  */\*\*  \* 开户人  \*/* @ApiModelProperty(value="开户人")  private String realName;   */\*\*  \* 开户行  \*/* @ApiModelProperty(value="开户行")  private String bank;   */\*\*  \* 开户省  \*/* @ApiModelProperty(value="开户省")  private String bankProv;   */\*\*  \* 开户市  \*/* @ApiModelProperty(value="开户市")  private String bankCity;   */\*\*  \* 开户地址  \*/* @ApiModelProperty(value="开户地址")  private String bankAddr;   */\*\*  \* 开户账号  \*/* @ApiModelProperty(value="开户账号")  private String bankCard;  } |

### 8.4.8 添加UserBankServiceFeign

|  |
| --- |
| */\*\*  \* 若FeignClient 里面的name 相同时,spring 创建对象就会报错:它认为它们2 个对象是一样的  \*/* @FeignClient(name = "member-service",contextId = "userBankServiceFeign" ,configuration = OAuth2FeignConfig.class ,path = "/userBanks") public interface UserBankServiceFeign {   @GetMapping("/{userId}/info")  UserBankDto getUserBankInfo(@PathVariable Long userId) ; } |

### 8.4.9 UserBankController

|  |
| --- |
| @Override public UserBankDto getUserBankInfo(Long userId) {  UserBank currentUserBank = userBankService.getCurrentUserBank(userId);  UserBankDto userBankDto = UserBankDtoMapper.*INSTANCE*.toConvertDto(currentUserBank);  return userBankDto ; } |

### 8.4.9 UserBankDtoMapper

|  |
| --- |
| @Mapper(componentModel = "spring") public interface UserBankDtoMapper {  UserBankDtoMapper *INSTANCE* = Mappers.*getMapper*(UserBankDtoMapper.class);   UserBank toConvertEntity(UserBankDto source);   List<UserBank> toConvertEntity(List<UserBankDto> source);    UserBankDto toConvertDto(UserBank source);   List<UserBankDto> toConvertDto(List<UserBank> source); } |

## 余额的扣减

### AccountService

|  |
| --- |
| */\*\*  \* 暂时锁定用户的资产  \* @param userId  \* 用户的id  \* @param coinId  \* 币种的id  \* @param mum  \* 锁定的金额  \* @param type  \* 资金流水的类型  \* @param orderId  \* 订单的Id  \* @param fee  \* 本次操作的手续费  \*/* void lockUserAmount(Long userId, Long coinId, BigDecimal mum, String type, Long orderId, BigDecimal fee); |

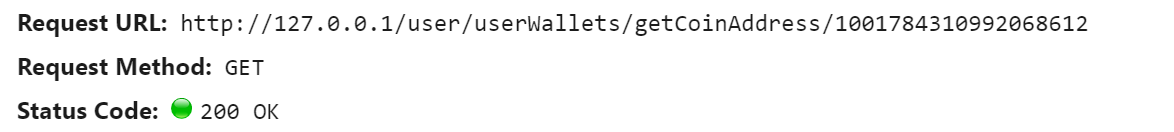
### AccountServiceImpl

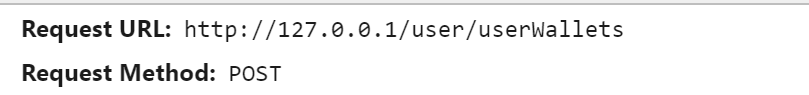
|  |
| --- |
| */\*\*  \* 暂时锁定用户的资产  \*  \* @param userId 用户的id  \* @param coinId 币种的id  \* @param mum 锁定的金额  \* @param type 资金流水的类型  \* @param orderId 订单的Id  \* @param fee  \*/* @Override public void lockUserAmount(Long userId, Long coinId, BigDecimal mum, String type, Long orderId, BigDecimal fee) {  Account account = getOne(new LambdaQueryWrapper<Account>().eq(Account::getUserId, userId)  .eq(Account::getCoinId, coinId)  );  if (account == null) {  throw new IllegalArgumentException("您输入的资产类型不存在");  }  BigDecimal balanceAmount = account.getBalanceAmount();  if (balanceAmount.compareTo(mum) < 0) { *// 库存的操作* throw new IllegalArgumentException("账号的资金不足");  }  account.setBalanceAmount(balanceAmount.subtract(mum));  account.setFreezeAmount(account.getFreezeAmount().add(mum));  boolean updateById = updateById(account);  if (updateById) { *// 增加流水记录* AccountDetail accountDetail = new AccountDetail(  null,  userId,  coinId,  account.getId(),  account.getId(), *// 如果该订单时邀请奖励,有我们的ref的account ,否则,值和account 是一样的* orderId,  (byte) 2,  type,  mum,  fee,  "用户提现",  null,  null,  null  );  accountDetailService.save(accountDetail);  } |

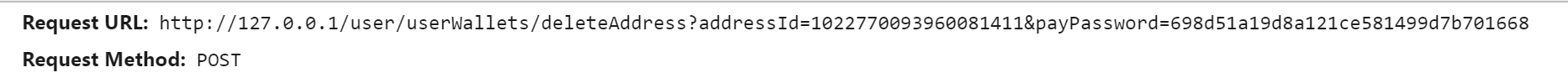
# 资产管理

## 9.1 提现地址的操作

### 9.1.1 API







### 9.1.2 UserWalletController

|  |
| --- |
| @RestController @Api(tags = "用户的提币地址") @RequestMapping("/userWallets") public class UserWalletController {    @Autowired  private UserWalletService userWalletService;   @GetMapping  @ApiOperation(value = "分页查询用户的提币地址")  @ApiImplicitParams({  @ApiImplicitParam(name = "userId", value = "用户的id"),  @ApiImplicitParam(name = "current", value = "当前页"),  @ApiImplicitParam(name = "size", value = "每页显示的条数")  })  @PreAuthorize("hasAuthority('user\_wallet\_query')")  public R<Page<UserWallet>> findByPage(@ApiIgnore Page<UserWallet> page, Long userId) {  page.addOrder(OrderItem.*desc*("last\_update\_time"));  Page<UserWallet> userWalletPage = userWalletService.findByPage(page, userId);  return R.*ok*(userWalletPage);  }   @GetMapping("/getCoinAddress/{coinId}")  @ApiOperation(value = "查询用户某种币的提现地址")  @ApiImplicitParams({  @ApiImplicitParam(name = "coinId" ,value = "币种的Id")  })  public R<List<UserWallet>> getCoinAddress(@PathVariable("coinId") Long coinId) {  Long userId = Long.*valueOf*(SecurityContextHolder.*getContext*().getAuthentication().getPrincipal().toString());  List<UserWallet> userWallets = userWalletService.findUserWallets(userId, coinId);  return R.*ok*(userWallets);  }    @PostMapping  @ApiOperation(value = "新增一个提现地址")  @ApiImplicitParams({  @ApiImplicitParam(name = "userWallet" ,value = "userWallet json数据")  })  public R save(@RequestBody @Validated UserWallet userWallet){  Long userId = Long.*valueOf*(SecurityContextHolder.*getContext*().getAuthentication().getPrincipal().toString());  userWallet.setUserId(userId);  boolean save = userWalletService.save(userWallet); *// 交易密码的交易* if (save){  return R.*ok*() ;  }  return R.*fail*("新增提现地址失败") ;  }    @PostMapping("/deleteAddress")  @ApiOperation(value = "删除某个用户的提现地址")  @ApiImplicitParams({  @ApiImplicitParam(name = "addressId" ,value = "提现地址的ID" ) ,  @ApiImplicitParam(name = "payPassword" ,value = "交易密码")  })  public R delete(@RequestParam(required = true) Long addressId ,@RequestParam(required = true) String payPassword){  boolean isOk = userWalletService.deleteUserWallet(addressId,payPassword) ;  if(isOk){  return R.*ok*("删除成功") ;  }  return R.*fail*("删除失败") ;  } } |

### 9.1.3 UserWalletService

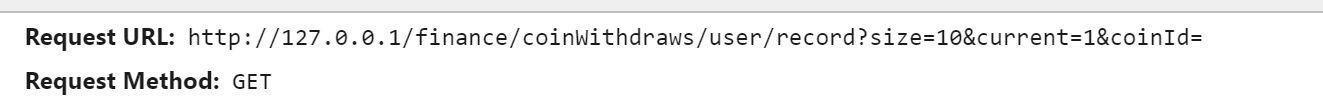
|  |
| --- |
| public interface UserWalletService extends IService<UserWallet>{    */\*\*  \* 分页查询用户的提币地址  \* @param page  \* 分页参数  \* @param userId  \* 用户的ID  \* @return  \*/* Page<UserWallet> findByPage(Page<UserWallet> page, Long userId);   */\*\*  \* 查询用户的提币的地址  \* @param userId  \* 用户的Id  \* @param coinId  \* 币种的Id  \* @return  \*/* List<UserWallet> findUserWallets(Long userId, Long coinId);   */\*\*  \* 删除用户的提现地址  \* @param addressId  \* 提现地址的Id  \* @param payPassword  \* 交易密码  \* @return  \*/* boolean deleteUserWallet(Long addressId, String payPassword); } |

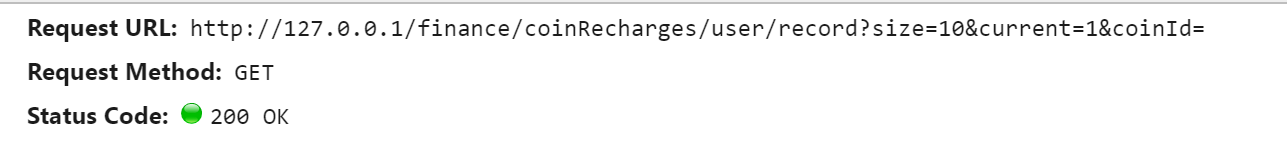
### 9.1.4 UserWalletServiceImpl

|  |
| --- |
| @Service public class UserWalletServiceImpl extends ServiceImpl<UserWalletMapper, UserWallet> implements UserWalletService{    @Autowired  private UserService userService ;  */\*\*  \* 分页查询用户的提币地址  \*  \* @param page 分页参数  \* @param userId 用户的ID  \* @return  \*/* @Override  public Page<UserWallet> findByPage(Page<UserWallet> page, Long userId) {  return page(page, new LambdaQueryWrapper<UserWallet>().eq(UserWallet::getUserId ,userId));  }   */\*\*  \* 查询用户的提币的地址  \*  \* @param userId 用户的Id  \* @param coinId 币种的Id  \* @return  \*/* @Override  public List<UserWallet> findUserWallets(Long userId, Long coinId) {  return list(new LambdaQueryWrapper<UserWallet>()  .eq(UserWallet::getUserId,userId)  .eq(UserWallet::getCoinId,coinId)  ) ;  }   @Override  public boolean save(UserWallet entity) {  Long userId = entity.getUserId();  User user = userService.getById(userId);  if(user==null){  throw new IllegalArgumentException("该用户不存在") ;  }  String paypassword = user.getPaypassword();  if(StringUtils.*isEmpty*(paypassword) || !(new BCryptPasswordEncoder().matches(entity.getPayPassword(),paypassword))){  throw new IllegalArgumentException("交易密码错误") ;  }  return super.save(entity);  }    */\*\*  \* 删除用户的提现地址  \*  \* @param addressId 提现地址的Id  \* @param payPassword 交易密码  \* @return  \*/* @Override  public boolean deleteUserWallet(Long addressId, String payPassword) {  UserWallet userWallet = getById(addressId);  if(userWallet==null){  throw new IllegalArgumentException("提现地址错误") ;  }  Long userId = userWallet.getUserId();  User user = userService.getById(userId);  if(user==null){  throw new IllegalArgumentException("用户不存在") ;  }  String paypassword = user.getPaypassword();  if(StringUtils.*isEmpty*(paypassword) || !(new BCryptPasswordEncoder().matches(payPassword,paypassword))){  throw new IllegalArgumentException("交易密码错误") ;  }  return super.removeById(addressId);  } } |

## 9.2 用户充值记录的查询

### 9.2.1 API 接口





### 9.2.2 CoinRechargeController

|  |
| --- |
| @RestController @RequestMapping("/coinRecharges") @Api(tags = "充币记录") public class CoinRechargeController {   @Autowired  private CoinRechargeService coinRechargeService;   @GetMapping("/records")  @ApiImplicitParams({  @ApiImplicitParam(name = "current", value = "当前页"),  @ApiImplicitParam(name = "size", value = "每页显示的条数"),  @ApiImplicitParam(name = "coinId", value = "当前页"),  @ApiImplicitParam(name = "userId", value = "用户的ID"),  @ApiImplicitParam(name = "userName", value = "用户的名称"),  @ApiImplicitParam(name = "mobile", value = "用户的手机号"),  @ApiImplicitParam(name = "status", value = "充值的状态"),  @ApiImplicitParam(name = "numMin", value = "充值金额的最小值"),  @ApiImplicitParam(name = "numMax", value = "充值金额的最小值"),  @ApiImplicitParam(name = "startTime", value = "充值开始时间"),  @ApiImplicitParam(name = "endTime", value = "充值结束时间"),  })  public R<Page<CoinRecharge>> findByPage(  @ApiIgnore Page<CoinRecharge> page, Long coinId,  Long userId, String userName, String mobile,  Byte status, String numMin, String numMax,  String startTime, String endTime  ) {  Page<CoinRecharge> pageData = coinRechargeService.findByPage(page, coinId, userId, userName,  mobile, status, numMin, numMax, startTime, endTime);  return R.*ok*(pageData);  }    @GetMapping("/user/record")  @ApiImplicitParams({  @ApiImplicitParam(name = "current" ,value = "当前页") ,  @ApiImplicitParam(name = "size" ,value = "显示的条数") ,  @ApiImplicitParam(name = "coinId" ,value = "币种的Id") ,   })  @ApiOperation(value = "查询用户某种币的Id")  public R<Page<CoinRecharge>> findUserCoinRecharge(@ApiIgnore Page<CoinRecharge> page ,Long coinId){  Long userId = Long.*valueOf*(SecurityContextHolder.*getContext*().getAuthentication().getPrincipal().toString()) ;  Page<CoinRecharge> pageData = coinRechargeService.findUserCoinRecharge(page ,coinId, userId) ;  return R.*ok*(pageData) ;  } } |

### 9.2.3 CoinRechargeService

|  |
| --- |
| public interface CoinRechargeService extends IService<CoinRecharge> {   */\*\*  \* 分页条件查询充值记录  \* @param page  \* 分页参数  \* @param coinId  \* 币种的Id  \* @param userId  \* 用户的Id  \* @param userName  \* 用户的名称  \* @param mobile  \* 用户的手机号  \* @param status  \* 充值的状态  \* @param numMin  \* 充值的最小金额  \* @param numMax  \* 充值的最大金额  \* @param startTime  \* 充值的开始时间  \* @param endTime  \* 充值的截至时间  \* @return  \*/* Page<CoinRecharge> findByPage(Page<CoinRecharge> page, Long coinId, Long userId, String userName, String mobile, Byte status, String numMin, String numMax, String startTime, String endTime);   */\*\*  \* 查询用户充币记录  \* @param page  \* 分页参数  \* @param coinId  \* 币种的Id  \* @param userId  \* 用户的ID  \*  \* @return  \*/* Page<CoinRecharge> findUserCoinRecharge(Page<CoinRecharge> page, Long coinId, Long userId); } |

### 9.2.4 CoinRechargeServiceImpl

|  |
| --- |
| @Service public class CoinRechargeServiceImpl extends ServiceImpl<CoinRechargeMapper, CoinRecharge> implements CoinRechargeService {   @Autowired  private UserServiceFeign userServiceFeign ;  */\*\*  \* 分页条件查询充值记录  \*  \* @param page 分页参数  \* @param coinId 币种的Id  \* @param userId 用户的Id  \* @param userName 用户的名称  \* @param mobile 用户的手机号  \* @param status 充值的状态  \* @param numMin 充值的最小金额  \* @param numMax 充值的最大金额  \* @param startTime 充值的开始时间  \* @param endTime 充值的截至时间  \* @return  \*/* @Override  public Page<CoinRecharge> findByPage(Page<CoinRecharge> page, Long coinId, Long userId, String userName, String mobile, Byte status, String numMin, String numMax, String startTime, String endTime) {  LambdaQueryWrapper<CoinRecharge> coinRechargeLambdaQueryWrapper = new LambdaQueryWrapper<>();  *// 1 若用户本次的查询中,带了用户的信息userId, userName,mobile ----> 本质就是要把用户的Id 放在我们的查询条件里面* Map<Long, UserDto> basicUsers = null;  if (userId != null || !StringUtils.*isEmpty*(userName) | !StringUtils.*isEmpty*(mobile)) { *// 使用用户的信息查询  // 需要远程调用查询用户的信息* basicUsers = userServiceFeign.getBasicUsers(userId == null ? null : Arrays.*asList*(userId), userName, mobile);  if (CollectionUtils.*isEmpty*(basicUsers)) { *// 找不到这样的用户->* return page;  }  Set<Long> userIds = basicUsers.keySet();  coinRechargeLambdaQueryWrapper.in(!CollectionUtils.*isEmpty*(userIds), CoinRecharge::getUserId, userIds);  }  *// 2 若用户本次的查询中,没有带了用户的信息* coinRechargeLambdaQueryWrapper.eq(coinId != null, CoinRecharge::getCoinId, coinId)  .eq(status != null, CoinRecharge::getStatus, status)  .between(  !(StringUtils.*isEmpty*(numMin) || StringUtils.*isEmpty*(numMax)),  CoinRecharge::getAmount,  new BigDecimal(StringUtils.*isEmpty*(numMin) ? "0" : numMin),  new BigDecimal(StringUtils.*isEmpty*(numMax) ? "0" : numMax)  )  .between(  !(StringUtils.*isEmpty*(startTime) || StringUtils.*isEmpty*(endTime)),  CoinRecharge::getCreated,  startTime, endTime + " 23:59:59"  );  Page<CoinRecharge> coinRechargePage = page(page, coinRechargeLambdaQueryWrapper);  List<CoinRecharge> records = coinRechargePage.getRecords();  if (!CollectionUtils.*isEmpty*(records)) {  List<Long> userIds = records.stream().map(CoinRecharge::getUserId).collect(Collectors.*toList*());  if (CollectionUtils.*isEmpty*(basicUsers)) {  basicUsers = userServiceFeign.getBasicUsers(userIds, null, null);  }  Map<Long, UserDto> finalBasicUsers = basicUsers;  records.forEach(coinRecharge -> { *// 设置用户相关的数据* UserDto userDto = finalBasicUsers.get(coinRecharge.getUserId());  if (userDto != null) {  coinRecharge.setUsername(userDto.getUsername()); *// 远程调用查询用户的信息* coinRecharge.setRealName(userDto.getRealName());  }  });  }   return coinRechargePage;  }   */\*\*  \* 查询用户充币记录  \*  \* @param page 分页参数  \* @param coinId 币种的Id  \* @param userId 用户的ID  \* @return  \*/* @Override  public Page<CoinRecharge> findUserCoinRecharge(Page<CoinRecharge> page, Long coinId, Long userId) {  return page(page,new LambdaQueryWrapper<CoinRecharge>()  .eq(coinId!=null,CoinRecharge::getCoinId,coinId)  .eq(CoinRecharge::getUserId ,userId)  );  } } |