股票模拟生成器及web展示设计文档

许仙明

目录

[1 必要的说明 2](#_Toc355316363)

[2 效果预览 2](#_Toc355316364)

[3 股票模拟生成器 2](#_Toc355316365)

[3.1 股票行情数据结构 2](#_Toc355316366)

[3.2 股票行情信息生成规则 4](#_Toc355316367)

[3.2.1 盘口数据规则 4](#_Toc355316368)

[3.2.2 成交价格确定规则 5](#_Toc355316369)

[4 技术实现细节 5](#_Toc355316370)

[4.1 系统结构 5](#_Toc355316371)

[4.1.1 股票行情生成器 5](#_Toc355316372)

[4.1.2 Thrift 5](#_Toc355316373)

[4.1.3 Socket.io 5](#_Toc355316374)

[4.2 设计思路 6](#_Toc355316375)

[4.3 后续改进 6](#_Toc355316376)

# 必要的说明

1. 本程序（文档）中关于股票的相关知识，大部分属于本人在笔试期间学习而来，因此可能不够完善，其中对专业词汇及规则的理解可能与实际有出入；
2. 由于行情模拟生成器设计了一个买卖委托生产规则及交易撮合逻辑，是根据本人对实际交易规则的知识及理解，对实际交易规则的简化；

# 效果预览

Chrome浏览器打开<http://localhost:8080/> 页面后，可以看到如下所示结果；

服务器在收到每个股票的行情数据后，通过websocket推动数据到浏览器，浏览器更新数据；



# 股票模拟生成器

## 股票行情数据结构

对于股票行情数据（不考虑指数、ETF等），我们以Level-1快照行情为例（不考虑Level-2快照、逐笔委托、逐笔委托数据），数据字段如下表所示，可简单划分为几个部分：

**(1)标识字段**

股票代码、时间戳；

**(2)价格统计**

上一日收盘价（开盘前已存在，盘中固定不变）、今日开盘价（集合竞价结束后固定不变）、今日最高价（开盘时为开盘价，盘中如最新成交价突破，则更新）、今日最低价（开盘时为开盘价，盘中如最新成交价突破，则更新）、今日最新价（根据最新成交价实时变动）；

净值字段用于基金，此处不予考虑；

**(3)成交统计**

今日累计成交笔数、今日累计成交数量、今日累计成交金额；

**(4)盘口数据**

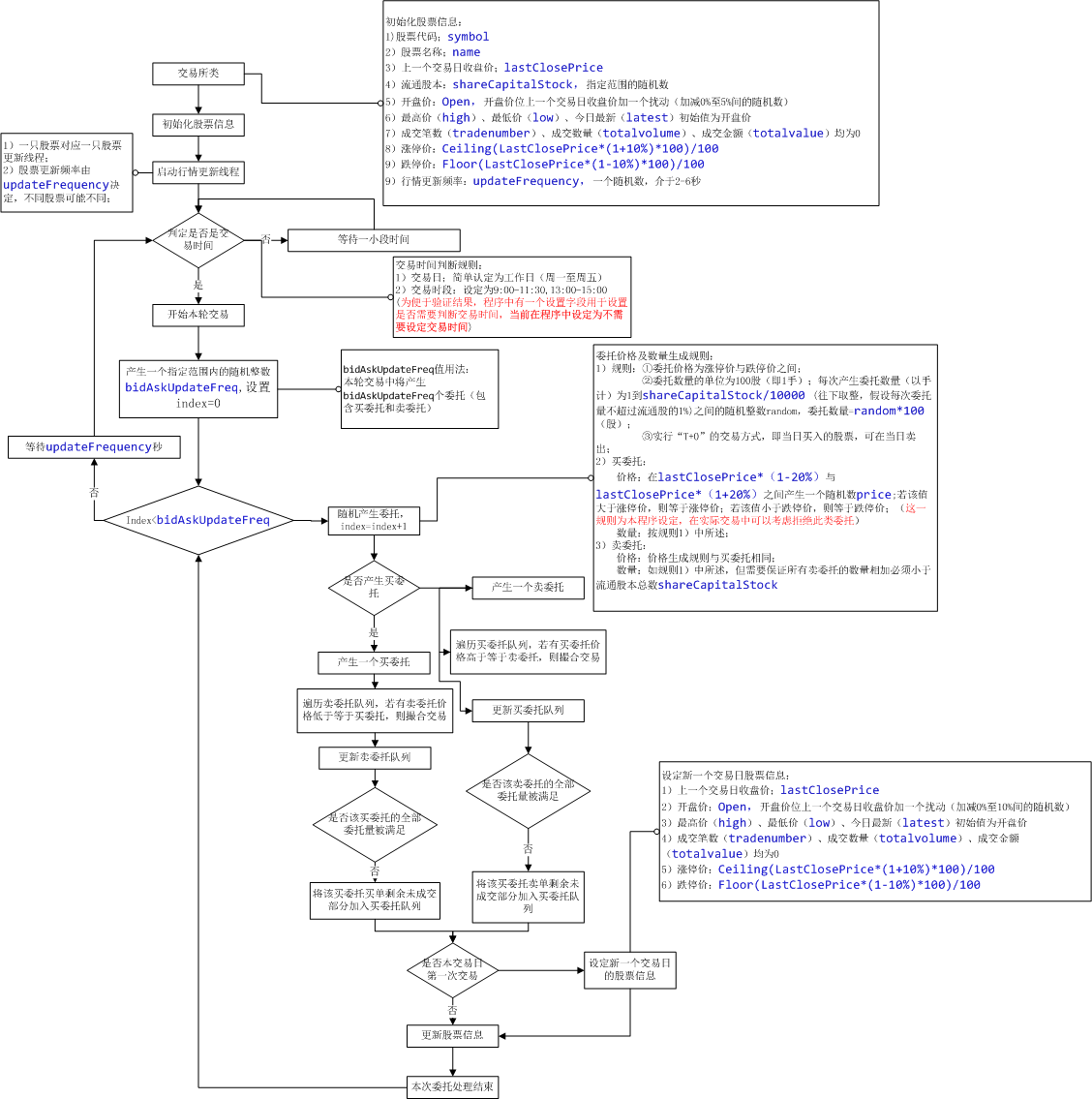
买\卖五档报价及对应的委托量

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 表名 | **L1\_Snapshot** | | | |
| 描述 | 股票、ETF等Level-1级别快照数据 | | | |
| 编号 | 字段名 | 字段描述 | 类型 | 备注 |
| 1 | Symbol | 证券代码 | 长度6位字符串 |  |
| 2 | Name | 股票名称 | String |  |
| 3 | Time | 时间 | TIMESTAMP | Messagetime |
| 4 | LastClose | 上一日收盘价 | Double |  |
| 5 | Open | 今日开盘 | DOUBLE |  |
| 6 | High | 今日最高 | DOUBLE |  |
| 7 | Low | 今日最低 | DOUBLE |  |
| 8 | Latest | 今日最新 | DOUBLE |  |
| 9 | Tradenumber | 成交笔数 | int |  |
| 10 | Totalvolume | 成交数量 | int |  |
| 11 | Totalvalue | 成交金额 | DOUBLE |  |
|  | IOPV | 净值（用于基金） | DOUBLE | 本程序不予考虑 |
| 12 | BidPrices1 | 买一价格 | DOUBLE |  |
| 13 | BidVolumes1 | 买一数量 | int |  |
| 14 | BidPrices2 | 买二价格 | DOUBLE |  |
| 15 | BidVolumes2 | 买二数量 | int |  |
| 16 | BidPrices3 | 买三价格 | DOUBLE |  |
| 17 | BidVolumes3 | 买三数量 | int |  |
| 18 | BidPrices4 | 买四价格 | DOUBLE |  |
| 19 | BidVolumes4 | 买四数量 | int |  |
| 20 | BidPrices5 | 买五价格 | DOUBLE |  |
| 21 | BidVolumes5 | 买五数量 | int |  |
| 22 | AskPrices1 | 卖一价格 | DOUBLE |  |
| 23 | AskVolumes1 | 卖一数量 | int |  |
| 24 | AskPrices2 | 卖二价格 | DOUBLE |  |
| 25 | AskVolumes2 | 卖二数量 | int |  |
| 26 | AskPrices3 | 卖三价格 | DOUBLE |  |
| 27 | AskVolumes3 | 卖三数量 | int |  |
| 28 | AskPrices4 | 卖四价格 | DOUBLE |  |
| 29 | AskVolumes4 | 卖四数量 | int |  |
| 30 | AskPrices5 | 卖五价格 | DOUBLE |  |
| 31 | AskVolumes5 | 卖五数量 | int |  |

## 股票行情信息生成规则

关于股票行情的生成规则，请看图片“股票生成器说明.bmp”；

若在word内看不清，可以选择打开外部图片，或者打开“股票生成器说明.vsd”文件（使用ms visio软件打开）查看；



### 盘口数据规则

程序在运行过程中，记录所有未被撮合交易的买委托及卖委托；但程序在传送数据给服务器时只传送买一至买五、卖一至卖五数据；web前端显示时只显示买一和卖一数据；

委托买入价格：BidPrice1> BidPrice2> BidPrice3> BidPrice4> BidPrice5 精确度为0.01

委托卖出价格：AskPrice1< AskPrice2<AskPrice3< AskPrice4< AskPrice5 精确度为0.01

**并且**

委托买入价格<最新成交价格<委托卖出价格

注：最新成交价格可以等于最优买卖价格BidPrice1\AskPrice1中的一个。

对于委托量，存在 BidVol\AskVol >0，数量均为100的整数倍；

### 成交价格确定规则

1）买入申报价格bidprice高于即时揭示得最低卖出申报价格，以即时揭示得最低卖出申报价格为成交价格  
if BidPrice>BestAskPrice  
 tradeprice=BestAskPrice  
2）卖出申报价格低于即时揭示得最高买入价格，以即时揭示得最高买入价格为成交价；

if AskPrice>BestBidPrice

tradeprice=BestBidPrice

# 技术实现细节

## 系统结构



### 股票行情生成器

股票行情数据生成器是一个java程序；

### Thrift

股票行情生成器与node.js之间的数据传输通过thrift完成；thrift是facebook提出的一种跨平台远程通信框架，效率比较高。它结合了功能强大的软件堆栈和代码生成引擎，以构建在 C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, JavaScript, Node.js, Smalltalk, and OCaml 这些编程语言间无缝结合的、高效的服务。thrift最初由facebook开发，07年四月开放源码，08年5月进入apache孵化器。

### Socket.io

socket.io设计的目标是支持任何的浏览器，任何Mobile设备。目前支持主流的PC浏览器 (IE,Safari,Chrome,Firefox,Opera等)，Mobile浏览器(iphone Safari/ipad Safari/android WebKit/WebOS WebKit等)。socket.io基于node.js并简化了WebSocket API，统一了通信的API。它支持：WebSocket, Flash Socket, AJAX long-polling, AJAX multipart streaming, Forever IFrame, JSONP polling。

socket.io解决了实时的通信问题，并统一了服务端与客户端的编程方式。启动了socket以后，就像建立了一条客户端与服务端的管道，两边可以互通有无。

## 设计思路

（1）股票行情数据生成器：

该部分采用多线程方式，每个线程单独处理一只股票的行情数据；java对多线程编程支持比较充分，同时本人对java比较熟悉，因此该部分使用java编程；

（2）java程序与node.js的数据通信：

这一部分主要考虑java与node.js是两个不同的平台，因此需要使用一种可以跨平台进行通信的框架；在研究过程中，发现thrift是一个比较适合的方案；

当然由于对thrift还处于初步学习阶段，目前仍存在两个问题需要进一步研究：1）大数量并发处理时，是否可以支持及如何支持；2）thrift在运作过程中会出现哪些类型的异常，及其相关处理方式；

（3）服务器端与浏览器：

服务器端与浏览器的数据交互，比较直接的方案是使用ajax，也即是浏览器定期访问服务器获取更新数据；

但是在本系统中，服务器端股票行情数据是不规律的，因此若采用服务器端在收到数据后，通知浏览器端更新会比较有效率；

在研究过程中，个人认为socket.io是一个比较适合的方案；但目前由于对socket.io认识不足，仍存在如下问题需要进一步研究：

1. 网络不稳定时，socket.io是否有机制实现数据准确传输；
2. 多用户连接时，如何使用socke.io解决多并发等问题；

## 后续改进

如果有更多的时间，比如10天，本人会尝试从如下角度对本程序进行改进：

（1）web前端展示

模仿大智慧等软件，美化股票行情数据展示页面，同时加入更多的图表内容；

（2）系统的灵活性

目前股票行情生成器，所有的配置信息均放入一个只有常量属性的类中，后续可以将这些配置项移入配置文件；用户可以直接修改配置文件，而不需要改代码；

（3）多并发处理

当前程序未在多并发方面做考虑，因此程序可能会在多用户接入时出错；因此需要在大数量、多并发处理方面做进一步优化；

（4）容错处理

当前程序未在容错方面做考虑，需要进一步优化，提高程序的稳定性；

（5）历史数据

1）历史数据保存：当前程序未保存历史数据，后续可以考虑引入数据库存放历史数据；

2）历史数据展示：

* 方案一：用户打开web页面时，将一段时间内的历史数据传输到浏览器后再显示；对于传送数据的时间间隔，在浏览器端展示一个动画信息，提示用户正在准备数据；
* 方案二：用户打开web页后，先展示当前数据；对于需要历史数据的功能全部无法使用；同时在浏览器向服务器请求（或者服务器向浏览器推动）历史数据，待历史数据接收完全后，更新页面，解锁需要历史数据的功能；