目录

[更新历史 2](#_Toc477471343)

[一、 引言 3](#_Toc477471344)

[(一) 编制目的 3](#_Toc477471345)

[(二) 词汇表 3](#_Toc477471346)

[(三) 参考资料 4](#_Toc477471347)

[二、 产品描述 5](#_Toc477471348)

[三、 逻辑视角 6](#_Toc477471349)

[四、 组合视角 8](#_Toc477471350)

[(一) 开发包图 8](#_Toc477471351)

[(二) 运行时进程 10](#_Toc477471352)

[(三) 物理部署 10](#_Toc477471353)

[五、 接口视角 11](#_Toc477471354)

[(一) 模块的职责 11](#_Toc477471355)

[1. 用户端模块视图 11](#_Toc477471356)

[2. 服务器端模块视图 11](#_Toc477471357)

[3. 用户端各层的职责 11](#_Toc477471358)

[4. 服务器端各层的职责 12](#_Toc477471359)

[5. 各层联系 12](#_Toc477471360)

[(二) 用户界面层的分解 13](#_Toc477471361)

[1. 用户界面层模块的职责 13](#_Toc477471362)

[2. 用户界面模块的接口规范 13](#_Toc477471363)

[3. 用户界面模块设计原理 16](#_Toc477471364)

[(三) 业务逻辑层的分解 17](#_Toc477471365)

[1. 业务逻辑层模块的职责 17](#_Toc477471366)

[2. 业务逻辑模块的接口模范 17](#_Toc477471367)

[(四) 数据层的分解 23](#_Toc477471368)

[1. 数据层模块的职责 23](#_Toc477471369)

[2. 数据层模块的接口规范 23](#_Toc477471370)

[六、 信息视角 28](#_Toc477471371)

[(一) 数据持久化对象 28](#_Toc477471372)

[(二) 文件格式 28](#_Toc477471373)

# 更新历史

|  |  |  |  |
| --- | --- | --- | --- |
| **修改人员** | **修改日期** | **修改原因** | **版本号** |
| 高源 | 2017/3/12 | 提交组合视角 |  |
| 高源 | 2017/3/12 | 提交界面层接口规范 |  |
| 冯俊杰 | 2017/3/12 | 提交数据层接口规范 |  |
| 冯俊杰 | 2017/3/12 | 提交信息视角 |  |
| 龚尘淼 | 2017/3/12 | 提交逻辑层接口规范 |  |
| 冯俊杰 | 2017/3/15 | 提交逻辑视角 |  |
| 冯俊杰 | 2017/3/16 | 整合所有相关文档 |  |

# 引言

## 编制目的

本报告详细完成对互联网酒店预订系统的概要设计，达到指导详细设计和开发的目的，同时实现测试人员及用户的沟通。

本报告面向开发人员、测试人员及最终用户编写，是了解系统的导航。

## 词汇表

|  |  |
| --- | --- |
| **缩写或单词** | **解释** |
| Quantourist | ASI小组开发的A股金融证券分析系统 |
| stock | 股票 |
| stockSituation | 股票市场温度计 |
| chart | 图表 |
| stockComparision | 股票比较 |
| increaseMargin | 涨跌幅 |
| logarithmicYield | 对数收益率 |
| logarithmicYieldVariance | 对数收益率方差 |
| adj close | 复权后的收盘指数 |
| candlestick | K线 |
| average | 均线 |
| user | 客户 |
| logIn | 登录 |
| service | 逻辑层 |
| dao | 数据层 |
| dataHelper | 数据助手 |

## 参考资料

1. 丁二玉，刘钦.计算与软件工程（卷二）[M]机械工业出版2012：134—182
2. IEEE std 1471-2000
3. A股金融证券分析系统需求规格说明文档 V1.1

# 产品描述

参考《A股金融证券分析系统需求规格说明文档》中对产品的概括描述。A股金融证券分析系统主要功能见用例图如下。

# 逻辑视角

互联网酒店预订系统中，选择了分层体系结构的风格，将系统分为3层（展示层、业务逻辑层、数据层）能够很好的示意整个高层抽象。展示层包括GUI页面的实现，业务逻辑层包含业务逻辑处理的实现，数据层负责数据的持久化和访问。分层体系结构的逻辑视角和逻辑设计方案如图1和图2所示。



**图1：分层结构**



**图2：逻辑设计方案**

# 组合视角

## 开发包图

与抽象的逻辑设计相比，实现物理设计要考虑更多的实现细节，这些细节有：

1. 所有的presentation层开发包都需要使用图形类型建立界面，都要依赖于图形界面类库包。
2. 在presentation层实现时，由mainUI包负责不同参与者在整个页面之间的跳转逻辑。其他各包负责各自页面自身的功能，并由各自的controller负责自身内部页面之间的跳转逻辑。
3. presentation层被置于用户端，data层与businessLogic层被置于服务器端，那么presentation层的开发包不可能依赖于businessLogic层的开发包。
4. 所有的data层开发包都需要进行数据持久化（例如读写数据库、读写文件等），所以它们会有一些重复代码，可以将重复代码独立为新的开发包。
5. 在分层风格的典型设计中，不希望高层直接依赖于低层，而是为低层建立接口包，实现依赖倒置原则
6. 在分层风格的典型设计中，presentation层与logic层之间、logic层与data层之间可能会传递复杂数据对象，那么相邻两层都需要使用数据对象声明，所以需要将数据对象声明独立为开发包——VO包和PO包。其中VO包负责在presentation层与logic层之间传递，PO包负责在logic层与data层之间传递。
7. 使用依赖倒置原则消除包的循环依赖现象，将循环依赖变为单向依赖

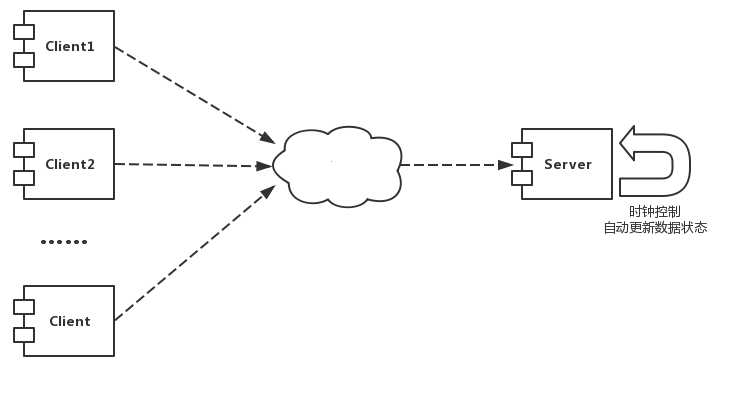
8）在logic层中，初始化和业务逻辑层上下文的工作被分配到utilities包中。

界面层依据功能需求分为登录注册界面和4个功能主界面；逻辑层主要依据逻辑分包；数据层进行整合合并，依照时间或者公司名称等实际情况进行分类。

|  |  |
| --- | --- |
| 开发（物理）包 | 依赖的其他包 |
| MainUI | 界面类库包 |
| LoginUI | loginPanelListener，界面类库包 |
| RegisterUI | registerPanelListener, 界面类库包 |
| KStringUI | StockService，jfreeChart, 界面类库包, VO |
| CompareUI | StockService, jfreeChart, 界面类库包,VO |
| ThermometerUI | StockSituationService, jfreeChart, 界面类库包,VO |
| FavoritesPanel | StockService, jfreeChart, 界面类库包,VO |
| ChartService | VO |
| StockService | VO |
| StockSituationService | VO |
| UserService | VO |
| ChartServiceImpl | StockDao, ChartService, PO，VO |
| StockServiceImpl | StockDao，StockService, PO,VO |
| StockSituationServiceImpl | StockSituationService，StockSituationDao, PO,VO |
| UserServiceImpl | StockDao, UserDao，UserService，PO,VO |
| StockDao | PO |
| StockSituationDao | PO |
| UserDao | PO |
| StockDaoImpl | StockDataHelper, PrivateStockDataHelper，StockDao, PO |
| StockSituationDaoImpl | StockSituationDataHelper, StockSituationDao, PO |
| UserDaoImpl | UserDao, UserDataHelper, LogDataHelper, PO |
| VO |  |
| PO |  |
| 界面类库包 |  |
| java.javafx |  |
| java.swing,java.awt,java.2D |  |

## 运行时进程

在Quantourist系统中，会有多个用户端进程和一个服务器端进程，其进程图如下图所示。结合部署图，用户端进程是在用户端机器上运行，服务器端进程是在服务器端机器上运行。数据库的自动更新会在服务器端自动进行，作为另一个进程。



## 物理部署

Quantourist系统中用户端构件是放在用户端机上，服务器端构件是放在服务器端机器上。在系统JDK环境已经设置好的情况下，不需要独立部署。部署图如下图所示。



# 接口视角

## 模块的职责

### 用户端模块视图



### 服务器端模块视图



### 用户端各层的职责

|  |  |
| --- | --- |
| 层 | 职责 |
| 启动模块 | 负责初始化通信机制，启动用户界面 |
| 用户界面层 | 基于窗口的互联网预订系统用户端用户界面 |
| 业务逻辑层 | 对于用户界面的输入进行响应并进行业务处理逻辑 |

### 服务器端各层的职责

|  |  |
| --- | --- |
| 层 | 职责 |
| 启动模块 | 负责初始化通信机制，启动用户界面 |
| 数据层 | 负责数据的持久化及数据访问接口 |

### 各层联系

每一层只是使用下方直接接触的层。层与层之间仅仅是通过接口的调用来完成的。层之间调用的接口如下图所示。

|  |  |  |
| --- | --- | --- |
| **接口** | **服务调用方** | **服务提供方** |
| ChartService  StockService  StockSituationService  UserService | 用户端  展示层  （presentation） | 服务器端  业务逻辑层  （businessLogic） |
| StockDao  StockSituationDao  UserDao | 服务器端  业务逻辑层  （businessLogic） | 服务器端  数据层  （data） |

## 用户界面层的分解

### 用户界面层模块的职责

根据需求，系统存在6个用户界面（加粗为主界面）： **登录界面**、**注册界面**、日线K线均线显示界面，股票比较界面，市场温度计界面，股票总体界面。

用户界面层模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| MainPanel | 界面frame，负责界面的显示和界面的跳转 |

### 用户界面模块的接口规范

1. **用户界面层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| mainUI | 语法 | init(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示Frame |
| logInUI.logIn | 语法 | boolean logIn(String userName,String password) |
| 前置条件 | 得到界面的用户名密码 |
| 后置条件 | 显示是否登陆成功 |
| logInUI.register | 语法 | public boolean registerUser(UserVO userVO, String password2) |
| 前置条件 | 从界面得到用户信息 |
| 后置条件 | 返回注册的结果 |
| kStringPanel. getSingleStockRecords | 语法 | Iterator<StockVO> getSingleStockRecords (ChartShowCriteriaVO chartShowCriteriaVO) |
| 前置条件 | 填写日期和股票号码 |
| 后置条件 | 获取单支股票的一段日期内的信息 |
| kStringPanel. getSingleStockRecords | 语法 | Iterator<StockVO> getSingleStockRecords(String code) |
| 前置条件 | 填写股票号码 |
| 后置条件 | 获取单支股票的所有数据 |
| kStringPanel.getAveData | 语法 | Map<Integer, Iterator<MovingAverageVO>> getAveData(ChartShowCriteriaVO chartShowCriteriaVO, List<Integer> days) |
| 前置条件 | 填写相应的日期股票号，查看的均线类型 |
| 后置条件 | 获取单支股票一段日期内，用户所选天数的均线图的平均值. |
| ComparePanel  .getComparision | 语法 | StockComparisionVO getComparision(StockComparsionCriteriaVO stockComparsionCriteriaVO) |
| 前置条件 | 用户输入日期和股票编号 |
| 后置条件 | 获取两只股票的比较信息 |
| ThermometerPanel. getStockStituationData | 语法 | Iterator<PriceRiseOrFallVO> getStockStituationData(LocalDate date); |
| 前置条件 | 用户查找到想要的酒店 |
| 后置条件 | 显示对应酒店的信息 |
| FavoritePanel. getAllStocks | 语法 | Iterator<StockVO> getAllStocks(LocalDate date) |
| 前置条件 | 用户填写了日期 |
| 后置条件 | 显示所有股票信息的列表 |
| FavoritePanel. getPrivateStocks | 语法 | Iterator<StockVO> getPrivateStocks(String userName, LocalDate date) |
| 前置条件 | 用户尚未注册会员 |
| 后置条件 | 显示用户的自选股信息列表 |
| FavoritePanel. deletePrivateStock | 语法 | public boolean deletePrivateStock(String userName, String stockCode); |
| 前置条件 | 用户登陆系统 |
| 后置条件 | 用户删除自选股 |

1. **用户界面层模块需要的服务接口**

|  |  |
| --- | --- |
| 服务名 | 服务 |
| ChartService | 有关统计图的业务逻辑接口 |
| StockService | 获取股票的相关的接口 |
| StockSituationService | 市场温度计的相关的接口 |
| UserService | 用户登陆注册相关的接口 |

### 用户界面模块设计原理

用户用户端界面拟用java的swing库来实现，服务器端用java的Swing、AWT实现。

## 业务逻辑层的分解

业务逻辑层包括多个针对界面及业务逻辑处理对象。例如对界面显示图表数据的获取和计算操作，对股票信息的获取操作，对市场温度计情况信息的获取操作和对用户信息的操作。

### 业务逻辑层模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| chartService | 负责处理界面显示图表信息的获取和计算 |
| stockService | 负责获取股票的具体信息 |
| stockSituationService | 负责获取和计算市场温度计的情况 |
| userService | 负责提供所有用户的注册和登录的服务 |

### 业务逻辑模块的接口模范

1. **chart模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| chartService.getSingleStockRecords | 语法 | Iterator<StockVO> getSingleStockRecords(String code) |
| 前置条件 | 无 |
| 后置条件 | 系统返回指定股票的所有交易信息 |
| chartService. getSingleStockRecords | 语法 | Iterator<StockVO> getSingleStockRecords(ChartShowCriteriaVO chartShowCriteriaVO) |
| 前置条件 | 无 |
| 后置条件 | 系统返回指定股票的所有交易信息 |
| chartService.getAveData | 语法 | Map<Integer,Iterator<MovingAverageVO>> getAveData(ChartShowCriteriaVO chartShowCriteriaVO, List<Integer> days) |
| 前置条件 | 无 |
| 后置条件 | 系统返回用户所选天数的均线图的移动平均值 |
| chartService.getAveData | 语法 | Map<Integer,Iterator<MovingAverageVO>> getAveData(String code, List<Integer> days) |
| 前置条件 | 无 |
| 后置条件 | 系统返回用户所选天数的均线图的移动平均值 |
| chartService. getComparison | 语法 | StockComparisionVO getComparision(StockComparsionCriteriaVO stockComparsionCriteriaVO) |
| 前置条件 | 无 |
| 后置条件 | 系统返回界面所需的两只股票的比较信息 |
| 需要的接口（需接口） | | |
| 服务名 | **服务内容** | |
| stockDao.getStockData(String stockCode) | 返回指定代码股票的所有数据 | |
| stockDao.getStockData(String stockCode, LocalDate start, LocalDate end  ) | 返回指定时间段内的指定股票所有数据 | |

1. **stock模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| stockService.getAllStocks | 语法 | Iterator<StockVO> getAllStocks(LocalDate date) |
| 前置条件 | 无 |
| 后置条件 | 系统返回所有股票信息列表 |
| stockService.getPrivateStocks | 语法 | Iterator<StockVO> getPrivateStocks(String userName, LocalDate date) |
| 前置条件 | 用户已经登录 |
| 后置条件 | 返回用户的自选股信息列表 |
| stockService.addPrivateStock | 语法 | public boolean addPrivateStock(String userName, String stockCode) |
| 前置条件 | 用户已经登录 |
| 后置条件 | 添加一条自选股 |
| stockService.deletePrivateStock | 语法 | public boolean deletePrivateStock(String userName, String stockCode) |
| 前置条件 | 用户已经登录 |
| 后置条件 | 删除一条自选股 |
| stockService.searchStock | 语法 | public List<StockSearchVO> searchStock(String searchString) |
| 前置条件 | 无 |
| 后置条件 | 返回符合查询条件的股票信息列表 |
| 需要的接口（需接口） | | |
| 服务名 | **服务内容** | |
| stockDao.getStockData(LocalDate date) | 返回指定日期保存的所有股票信息 | |
| stockDao.getPrivateStockData(String username,LocalDate date) | 根据用户指定的日期，返回用户自选股的数据 | |
| stockDao.addPrivateStock(String username,String stockCode) | 根据股票代码，添加一条自选股 | |
| stockDao.deletePrivateStock(String username,String stockCode) | 根据股票代码，删除一条自选股 | |
| stockDao.getAllStocksCode | 返回所有股票的代码-名称键值对 | |
| stockDao.getAllStocksFirstLetters | 返回所有股票的首字母-名称键值对 | |

1. **stockSituation模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| stockSituationService.getStockSituationData | 语法 | Iterator<PriceRiseOrFallVO> getStockStituationData(LocalDate date) |
| 前置条件 | 无 |
| 后置条件 | 返回指定日期股票涨跌情况的列表 |
| 需要的接口（需接口） | | |
| 服务名 | **服务内容** | |
| stockSituationDao.getStockSituation(LocalDate date) | 返回指定日期市场温度计数据 | |

1. **user模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| userService.registerUser | 语法 | public boolean registerUser(UserVO userVO, String password2) |
| 前置条件 | 无 |
| 后置条件 | 返回用户是否注册成功 |
| userSerivce.modifyUser | 语法 | public boolean modifyUser(UserVO userVO) |
| 前置条件 | 用户已登录 |
| 后置条件 | 系统更新用户信息 |
| userService.checkUserInfo | 语法 | public UserVO checkUserInfo(String userName) |
| 前置条件 | 用户已登录 |
| 后置条件 | 返回用户信息 |
| userService.logIn | 语法 | boolean logIn(String userName,String password) |
| 前置条件 | 无 |
| 后置条件 | 返回用户是否登录成功 |
| userService.logOut | 语法 | public boolean logOut(String userName) |
| 前置条件 | 用户已登录 |
| 后置条件 | 返回用户是否注销成功 |
| 需要的接口（需接口） | | |
| 服务名 | **服务内容** | |
| userDao.getAllUseNames() | 获取已成功注册的所有用户的名称 | |
| userDao.add(UserPO userPO) | 添加一条用户信息 | |
| userDao.modify(UserPO userPO) | 修改一条用户信息 | |
| userDao.get(String username) | 得到一条用户信息 | |
| userDao.getLoginUserNames() | 获取所有已登录用户的名称 | |
| userDao.logIn(String username) | 添加一条已登录的用户信息 | |
| userDao.logOut(String username) | 删除一条已登录的用户信息 | |

## 数据层的分解

数据层主要给业务逻辑层提供数据访问服务，包括对于持久化数据的查找。例如，AccountBusinessService的服务主要由AccountDataService提供。由于持久化数据的保存可能存在多种形式：Txt文件、序列化文件、数据库等，所示抽象了数据服务。以AccountDataService为例，数据层模块的具体描述下图所示：

### 数据层模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| stockData | 负责保存股票信息，进行查找的操作 |
| stockSituationData | 负责保存股票市场温度计信息，进行查找的操作 |
| userData | 负责保存用户信息，进行增、删、改、查的操作 |

### 数据层模块的接口规范

1. **stockData模块的接口规范**

|  |  |  |
| --- | --- | --- |
| stockDao.getStockData | 语法 | public StockPO getStockData(String stockCode, LocalDate date) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| stockDao.getStockData | **语法** | public List<StockPO> getStockData(String stockCode, LocalDate start, LocalDate end) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| stockDao.getStockData | **语法** | public List<StockPO> getStockData(String stockCode) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| stockDao.getStockData | **语法** | public List<StockPO> getStockData(LocalDate date) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| stockDao.getPrivateStockData | **语法** | public List<StockPO> getPrivateStockData(String userName, LocalDate date) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| stockDao.getPrivateStocks | **语法** | public PrivateStockPO getPrivateStocks(String userName) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| stockDao.addPrivateStock | **语法** | public boolean addPrivateStock(String userName, String stockCode) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| stockDao.deletePrivateStock | **语法** | public boolean deletePrivateStock(String userName, String stockCode) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| stockDao.getFirstDay | **语法** | public LocalDate getFirstDay(String stockCode) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| stockDao.getAllStocksCode | **语法** | public Map<String, String> getAllStocksCode() |
| 前置条件 | 无 |
| 后置条件 | 无 |
| stockDao.getAllStocksFirstLetters | **语法** | public Map<String, String> getAllStocksFirstLetters () |
| 前置条件 | 无 |
| 后置条件 | 无 |

1. **stockSituationData模块的接口规范**

|  |  |  |
| --- | --- | --- |
| stockSituationDao.getStockSituation | 语法 | public StockSituationPO getStockSituation(LocalDate date) |
| 前置条件 | 无 |
| 后置条件 | 无 |

1. **userData模块的接口规范**

|  |  |  |
| --- | --- | --- |
| UserDao.add | 语法 | public boolean add(UserPO userPO) |
| 前置条件 | 无 |
| 后置条件 | 系统持久化增加该用户的数据 |
| UserDao.get | 语法 | public UserPO get(String username) |
| 前置条件 | 无 |
| 后置条件 | 无 |
| UserDao.modify | 语法 | public boolean modify(UserPO userPO) |
| 前置条件 | 无 |
| 后置条件 | 系统修改并持久化增加该用户的数据 |
| UserDao.logIn | 语法 | public boolean logIn(String userName) |
| 前置条件 | 无 |
| 后置条件 | 系统保存该用户的登录数据 |
| UserDao.logOut | 语法 | public boolean logOut(String userName) |
| 前置条件 | 无 |
| 后置条件 | 系统保存该用户的登录数据并将该用户退出 |
| UserDao.getLoginUserNames | 语法 | public Set<Object> getAllUserNames() |
| 前置条件 | 无 |
| 后置条件 | 无 |
| UserDao.getLoginUserNames | 语法 | public Set<Object> getLoginUserNames() |
| 前置条件 | 无 |
| 后置条件 | 无 |

# 信息视角

## 数据持久化对象

系统的PO类就是对应的相关的实体类，如下所示。

|  |  |
| --- | --- |
| 类名 | 包含的属性 |
| StockPO | 记录编号，日期(月/日/年)，开盘指数，最高指数，最低指数，收盘指数，成交量，复权后的收盘指数，股票代码，股票名称，市场名称，昨日收盘指数，昨日复权后的收盘指数 |
| StockSituationPO | 当日总交易量，涨停股票数，跌停股票数，涨幅超过5%的股票数，跌幅超过5%的股票数，开盘‐收盘大于5%\*上一个交易日收盘价的股票个数、开盘‐收盘小于‐5%\*上一个交易日收盘价的股票个数 |
| PrivateStockPO | 用户名，用户的自选股列表 |
| AveragePO | 收盘指数，股票代码，股票名称 |
|  |  |
| UserPO | 用户姓名，用户密码 |

## 文件格式

本系统采用txt保存文件。

|  |  |  |
| --- | --- | --- |
| 所属包与表名 | 内容 | 字段设计 |
| stock\_records\_by\_code | 按股票代码分类的股票信息 | 分别为记录编号，日期(月/日/年)，开盘指数，最高指数，最低指数，收盘指数，成交量，复权后的收盘指数，股票代码，股票名称，市场名称，昨日收盘指数，昨日复权后的收盘指数。各变量之间用tab分隔。 |
| stock\_records\_by\_date | 按股票日期分类的股票信息 | 分别为记录编号，日期(月/日/年)，开盘指数，最高指数，最低指数，收盘指数，成交量，复权后的收盘指数，股票代码，股票名称，市场名称，昨日收盘指数，昨日复权后的收盘指数。各变量之间用tab分隔。 |
| stock\_situation | 股票市场温度计 | 分别为当日总交易量，涨停股票数，跌停股票数，涨幅超过5%的股票数，跌幅超过5%的股票数，开盘‐收盘大于5%\*上一个交易日收盘价的股票个数、开盘‐收盘小于‐5%\*上一个交易日收盘价的股票个数。各变量之间用tab分隔。 |
| privateStock | 用户自选股 | 分别为用户名称、用户自选股。各变量之间用tab分隔。 |
| shortPinyin | 股票名称的拼音缩写 | 分别为股票名称、拼音缩写。各变量之间用tab分隔。 |
| stockName-code | 股票名字与代码的对应 | 分别为股票名称、股票代码。各变量之间用tab分隔。 |
| userinfo | 用户信息 | 分别为用户名称、用户密码。各变量之间用tab分隔。 |
| Login\_user | 已登录的用户 | 用户名称 |