AnyQuant股票分析系统详细设计文档

Ultraviolet工作室

2016.3.10

文档更新记录：

|  |  |  |
| --- | --- | --- |
| 版本 | 作者 | 日期 |
| V1.0 | 全组人员 | 2016.3.10 |
| V2.0 | 苏琰梓 | 2016.5.2 |

# 1.引言

## 1.1编制目的

本报告给出AnyQuant股票分析系统的详细设计描述和模型图，在软件架构的基础上进行了进一步分析，为系统的维护、扩展及后续开发提供参考依据。

## 1.2词汇表

|  |  |
| --- | --- |
| display market index | 显示大盘指数 |
| display stock list | 显示股票列表 |
| filter stocks | 筛选股票 |
| search stock | 搜索股票 |
| inquire market index | 查看大盘指数 |
| inquire detail info | 查看详细信息 |
| Manage self select stock | 自选股管理 |
| inquire follow history | 查看关注历史 |
| filter stock info | 筛选股票信息 |

## 1.3参考资料

1）IEEE标准。

2）AnyQuant股票分析系统软件需求规格说明文档。

# 2.产品概述

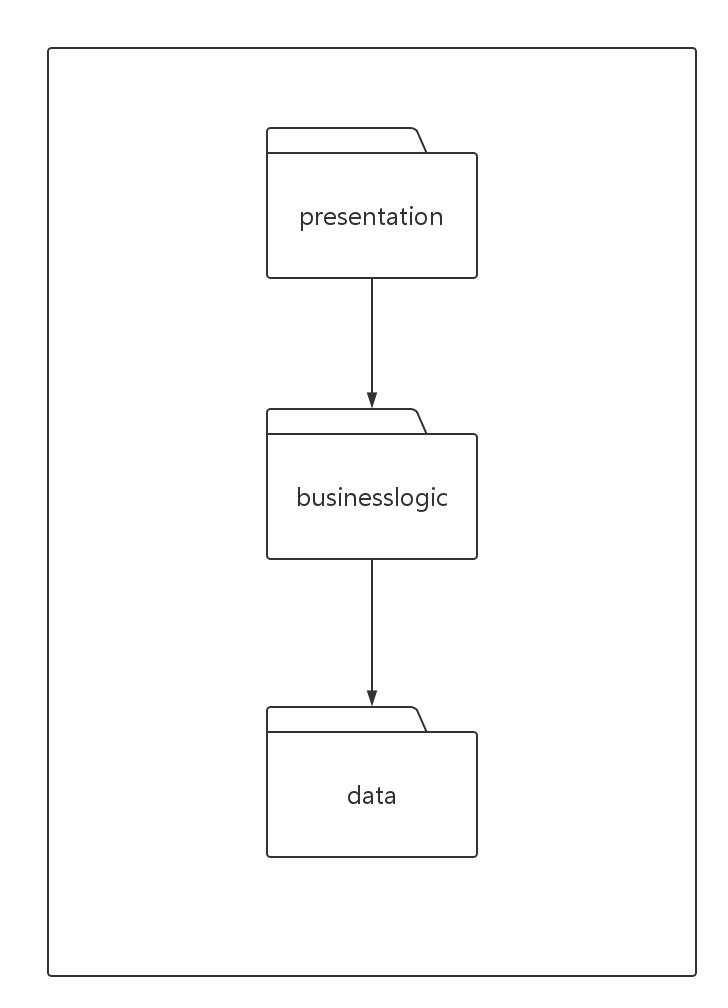
AnyQuant股票分析系统是为广大股民开发的股票分析系统，开发的目标是帮助股民及时了解股票实时动态并进行有效的股票评估以及买进卖出业务。

通过AnyQuant股票分析系统的应用，期望为股民提高对股票的操作效率、降低时间成本、提供更多的信息。

# 3.体系结构模型

## 3.1整体架构描述

为了更好地实现并行开发、提高开发速率，系统采用分层模型进行开发。传统分层模型包括表现层、业务层以及数据层3层。图1为传统分层模型图示。

图1 传统分层模型风格图

为了增强程序的灵活性和用户自由度，系统允许用户修改系统配置，为了更好地管理系统配置，特增加配置层以满足需求。图2为系统的分层架构。系统划分为以下4个层次

1. 表示层：用于前台界面展示和配置的层次。
2. 业务层：包含业务控制和逻辑的层次。
3. 数据层：定义和存储系统中相关数据的层次。
4. 配置层：管理界面配置及系统配置的层次。

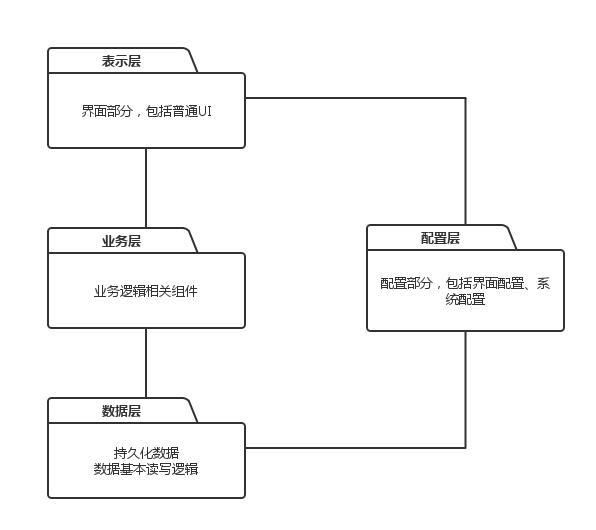


图2 系统分层架构

系统业务层分为以下五个模块，图3是业务层的分解示意图。

1. 显示模块：包括显示大盘指数、显示股票列表和显示详细信息，负责实现与表示层的直接交互。
2. 读取模块：包括获取股票列表、股票详细信息以及大盘指数，负责实现与数据层的直接交互。
3. 筛选模块：包括筛选股票和搜索股票。
4. 查询模块：包括查看大盘指数和查看股票详细信息。
5. 关注模块：包括关注股票和取消关注。
6. 历史模块：包括关注历史，后续迭代考虑添加策略历史、购买历史等。

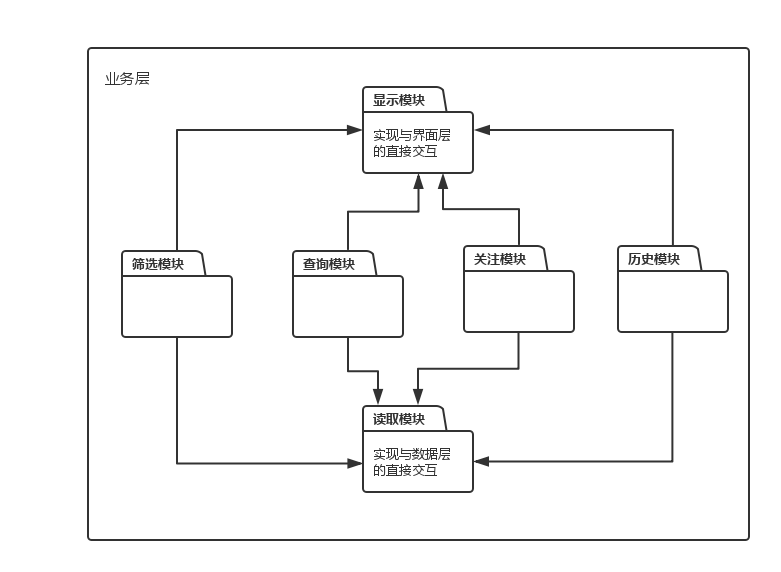


图3 业务层的分解示意图

## 3.2显示模块的分解

显示模块分为显示大盘指数、显示股票列表、显示股票信息和显示图表，实现与界面层的直接交互，也是唯一与界面层交互的模块。

### 3.2.1显示模块的职责

显示大盘指数负责显示大盘指数的重要信息；

显示股票列表负责分类显示所有股票，并显示股票的重要信息；

显示股票信息负责显示大盘指数和股票的所有详细信息；

显示图表负责显示针对一只股票的K线图、折线图和分时图。

### 3.2.2显示模块的接口规范

表1显示大盘指数的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的接口（供接口） | | |
| DisplayMarketIndexService.show | 语法 | public Iterator<MarketIndexVO> getIndex() |
| 前置条件 | 已创建一个MarketIndex领域对象，已添加大盘指数记录 |
| 后置条件 | 返回大盘指数列表的迭代器 |
| DisplayMarketIndexService.getIndex | 语法 | public Iterator<MarketIndexVO> getIndex(Date date) |
| 前置条件 | 已创建一个MarketIndex领域对象，已添加大盘指数记录 |
| 后置条件 | 返回指定日期大盘指数列表的迭代器 |
| 需要的接口（需接口） | | |
| 服务名 | | 服务 |
| IndexDataService.find() | | 返回当天所有大盘指数信息 |
| IndexDataService.find(Date date) | | 根据日期返回所有大盘指数信息 |

|  |  |  |
| --- | --- | --- |
| 提供的接口（供接口） | | |
| DisplayStockListService.  getStocks | 语法 | public Iterator<StockVO> getStocks () |
| 前置条件 | 已创建一个StockList对象，已添加股票列表记录，并且输入符合输入规则 |
| 后置条件 | 返回股票列表的迭代器 |
| DisplayStockListService.  getStocks | 语法 | public Iterator<StockVO> getStocks (Date date) |
| 前置条件 | 已创建一个StockList对象，已添加股票列表记录，并且输入符合输入规则 |
| 后置条件 | 返回指定日期股票列表的迭代器 |
| 需要的接口（需接口） | | |
| 服务名 | | 服务 |
| StockDataService.find() | | 返回当天所有股票信息 |
| StockDataService.find(Date date) | | 根据日期返回所有股票信息 |

### 3.2.3显示模块的设计原理

显示模块可以显示大盘指数或者股票列表，大盘指数供为用户提供股市的整体情况说明；股票列表展现所有股票的重要信息，为用户浏览股票提供了方便，同时也提供进一步筛选和搜索股票，符合逐层递进的思维习惯；而显示图表模块让用户对某支股票的涨跌一目了然，为用户提供了极大的参考价值。

## 3.3筛选模块的分解

筛选模块分为筛选股票、搜索股票，实现对股票列表的筛选和排序操作。

### 3.3.1筛选模块的职责

筛选股票负责根据不同的筛选项对股票进行筛选，并对筛选结果进行排序。

搜索股票负责根据名称或代码来搜索股票。

### 3.3.2筛选模块的接口规范

表3 筛选股票的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| FilterStockService.  getQualifiedStock | 语法 | public Itreator<StockPO> getQualifiedStock (String item，T low,T high) | |
| 前置条件 | 用户从股票列表中输入筛选条件，启动一次股票筛选任务 | |
| 后置条件 | 系统选出所有符合用户输入筛选条件的股票 | |
| FilterStockService.  sort | 语法 | public Itreator<StockPO> sort(String item) | |
| 前置条件 | 系统筛选出所有符合条件的股票 | |
| 后置条件 | 所有被筛选出的股票按照符合条件的由大到小程度进行排序并显示 | |
| FilterStockService.  sort | 语法 | public Itreator<StockPO> sort(String item, Compare c) | |
| 前置条件 | 系统筛选出所有符合条件的股票 | |
| 后置条件 | 所有被筛选出的股票按照排序规则进行排序 | |
| **需要的服务（需接口）** | | | |
| 服务名 | | | 服务 |
| FilterStockDataService.findQualifiedStock (T t) | | | 根据筛选条件进行查找对应的所有股票对象 |

表4 搜索股票的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| SearchStockService.getQualifiedStock | 语法 | public Itreator<String> getQualifiedStock (T t) |
| 前置条件 | 用户从股票列表中输入关键字，启动一次股票搜索任务 |
| 后置条件 | 系统选出所有符合用户输入搜索条件的股票 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| SearchStockDataService.findQualifiedStock (T t) | 根据搜索条件进行查找对应的所有股票对象 | |

表5 筛选股票的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| FilterStockDataService.  findQualifiedStock | 语法 | public StockPO findQualifiedStock (T item) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 查找并返回 StockPO 结果 |

表6搜索股票的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| SearchStockDataService.findQualifiedStock | 语法 | public StockPO findQualifiedStock (T item) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 查找并返回 StockPO 结果 |

### 3.3.3筛选模块的设计原理

为了增强用户体验，系统提供对股票列表的筛选操作，由于搜索股票属于一种特殊的筛选，所以讲其并入筛选模块。仅仅对股票列表进行筛选显然并不能很好地展现筛选效果，所以在显示结果前，对筛选结果进行排序。关于排序，参照经典文件资源管理器采用的方式，系统提供根据不同数据域进行排序，并可实现倒序。

## 3.4查询模块的分解

查询模块分为查看大盘指数和查看股票详细信息，负责实现对股票（大盘指数）详细信息的查询。

### 3.4.1查询模块的职责

查看大盘指数负责获得指定日期大盘指数的详细信息。

查看详细信息负责获得指定日期股票的详细信息。

### 3.4.2查询模块的接口规范

表7 查看大盘指数的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的接口（供接口） | | | |
| InquireMarketIndex  Service.getIndex | 语法 | public IndexPO getIndex(String code) | |
| 前置条件 | 已选择一个大盘指数 | |
| 后置条件 | 返回当天大盘指数 | |
| InquireMarketIndex  Service.getIndex | 语法 | public IndexPO getIndex(String code, Date date) | |
| 前置条件 | 已选择一个大盘指数，且已修改显示日期（默认为当天） | |
| 后置条件 | 返回指定日期的大盘指数 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| IndexDataService.find(String code) | | | 根据代码返回当天大盘指数信息 |
| IndexDataService.find(String code, Date date) | | | 根据代码返回指定日期大盘指数信息 |

表8 查看详细信息的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| InquireDetailedInfoService.getStockInfoByID | 语法 | public StockVO getStockByID(String id) | |
| 前置条件 | 启动一次查看股票详细信息操作 | |
| 后置条件 | 在一个股票详细信息查看操作中，获得该股票的开盘价、收盘价、最高价、最低价、交易量、交易金额 | |
| InquireDetailedInfoService.getStockInfoByName | 语法 | public StockVO getStockInfoByName(String name) | |
| 前置条件 | 启动一次查看股票详细信息操作 | |
| 后置条件 | 在一个股票详细信息查看操作中，获得该股票的开盘价、收盘价、最高价、最低价、交易量、交易金额 | |
| **需要的服务（需接口）** | | | |
| 服务名 | | | 服务 |
| InquireDetailedInfoDataService.findStockInfoByName(String name) | | | 根据股票名称查找股票持久化对象 |
| InquireDetailedInfoDataService.findStockInfoByID(String name) | | | 根据股票代码查找股票持久化对象 |

表9 查看大盘指数的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的接口（供接口） | | |
| IndexDataService.find | 语法 | public IndexPO find(String code) |
| 前置条件 | 无 |
| 后置条件 | 查找并返回当天大盘指数（IndexPO） |
| IndexDataService.find | 语法 | public IndexPO find(String code, Date date) |
| 前置条件 | 无 |
| 后置条件 | 查找并返回指定日期的大盘指数 |

表10 查找股票信息的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| InquireDetailedInfoDataService.findStockByID（String id） | 语法 | public StockPO findStockByID(String id) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 根据股票代号查找并返回StockPO结果 |
| InquireDetailedInfoDataService.findStockByName(String name) | 语法 | public StockPO findStockByName(String name) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 根据股票名称查找并返回StockPO结果 |

表11 绘制股票图表的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| ShowCurrentDataService.showCurrentData(String id) | 语法 | public CurrentStockVO showCurrentData (String id) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 根据股票代号查找并返回CurrentStockVO结果用于绘制图表 |
| ShowStockNewsService.showStockNews(String id) | 语法 | public List<stockNewsVO> showStockNews (String id) |
| 前置条件 | 无 |
| 后置条件 | 根据股票代号查找并返回股票的信息列表 |
| ShowStockIDNameService.getStockIdAndName(String id) | 语法 | public StockIDNameVO getStockIdAndName (String id) |
| 前置条件 | 无 |
| 后置条件 | 根据股票代号查找并返回StockIDNameVO |
| ShowStockIDNameService. getAllStockIdAndName (String id) | 语法 | public List< StockIDNameVO > getAllStockIdAndName (String id) |
| 前置条件 | 无 |
| 后置条件 | 根据股票代号查找并获取StockIDNameVO列表 |

### 3.4.3查询模块的设计原理

对于用户而言，仅提供股票列表中的信息时远远不够的，系统提供对大盘指数和股票详细信息的查询，及对大盘指数和股票历史信息的查询。

## 3.5关注模块的分解

关注模块分为关注股票和取消关注，实现对自选股的管理。

### 3.5.1关注模块的职责

关注股票及取消关注负责实现对自选股的管理。

### 3.5.2关注模块的接口规范

表12 关注模块的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| ManageSelfSelectStockService.addSelfSelectStock | 语法 | public void addSelfSelectStock(String id) | |
| 前置条件 | 启动一次添加自选股操作 | |
| 后置条件 | 在一个自选股添加操作中，将该股票加入用户的自选股列表中 | |
| ManageSelfSelectStockService.removeSelfSelectStock | 语法 | public void removeSelfSelectStock(String name) | |
| 前置条件 | 启动一次删除自选股操作 | |
| 后置条件 | 在一个自选股删除操作中，从用户的自选股列表中删除该股票 | |
| **需要的服务（需接口）** | | | |
| 服务名 | | | 服务 |
| ManageSelfSelectStockDataService.addSelfSelectStock(String userID,String stockID) | | | 根据用户账户将当前股票加入用户的自选股列表中 |
| ManageSelfSelectStockDataService.removeSelfSelectStock(String userID,String stockID) | | | 根据用户账户将当前股票从用户的自选股列表中移除 |

表13 关注模块的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| ManageSelfSelectStockDataService.findSelfSelectStock（String userid） | 语法 | public List<StockPO> findSelfSelectStock(String userid) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 根据用户账号查找并返回该用户所有自选股的StockPO |
| ManageSelfSelectStockDataService.addSelfSelectStock(String name) | 语法 | public void addSelfSelectStock(String name) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 根据股票名称将该股票加入该用户的自选股列表中 |
| ManageSelfSelectStockDataService.removeSelfSelectStock(String name) | 语法 | public void removeSelfSelectStock(String name) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 根据股票名称从用户自选股列表中删除该股票 |

### 3.5.3关注模块的设计原理

通常用户仅仅关注位数不多的几只股票，为了是用户可以更方便地获得这些股票最近的数据，系统提供自选股功能，用户可以对自选股进行管理。

## 3.6历史模块的分解

历史模块包括关注历史。

### 3.6.1历史模块的职责

关注历史负责展现用户所有关注过的股票列表，并提供对它们的管理。

### 3.6.2历史模块的接口规范

表14 关注历史的接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的接口（供接口） | | | |
| FollowHistoryService.  getFollow | 语法 | | public Iterator<StockPO> getFollow() |
| 前置条件 | | 启动一个查询关注历史回合 |
| 后置条件 | | 返回当前关注列表的迭代器 |
| FollowHistoryService.  GetFollowed | 语法 | | public Iterator<StockPO > getFollowed() |
| 前置条件 | | 启动一个查询关注历史回合 |
| 后置条件 | | 返回历史关注列表（不包括当前关注列表）的迭代器 |
| FollowHistoryService.  getFollowNum | 语法 | | public int getFollowNum() |
| 前置条件 | | 启动一个查询关注历史回合 |
| 后置条件 | | 返回当前关注股票数量 |
| FollowHistoryService.  getFollowedNum | 语法 | | public int getFollowedNum() |
| 前置条件 | | 启动一个查询关注历史回合 |
| 后置条件 | | 返回历史关注股票数量 |
| FollowHistoryService.  clearFollow | 语法 | | public boolean clearFollow() |
| 前置条件 | | 启动一个查询关注历史回合 |
| 后置条件 | | 取消当前所有关注，并将关注股票全部添加到历史关注列表 |
| FollowHistoryService.  clearFollowed | 语法 | | public boolean clearFollowed() |
| 前置条件 | | 启动一个查询关注历史回合 |
| 后置条件 | | 清空历史关注列表 |
| FollowHistoryService.  clearAll | 语法 | | public boolean clearAll() |
| 前置条件 | | 启动一个查询关注历史回合 |
| 后置条件 | | 取消所有关注，并清空关注列表 |
| 需要的服务（需接口） | | | |
| 服务名 | | 服务 | |
| FollowHistoryDataService.getFollow | | 返回当前关注列表 | |
| FollowHistoryDataService.getFollowed | | 返回历史关注列表 | |
| FollowHistoryDataService.clearFollow | | 取消当前所有关注，并将所有关注股票添加到历史关注列表 | |
| FollowHistoryDataService.clearFollowed | | 清空历史关注列表 | |
| FollowHistoryDataService.clearAll | | 取消所有关注，并清空关注列表 | |

表15 关注历史的接口规范（数据层）

|  |  |  |
| --- | --- | --- |
| 提供的接口（供接口） | | |
| FollowHistoryDataService.getFollow | 语法 | public List<StockPO> getFollow() |
| 前置条件 | 无 |
| 后置条件 | 返回当前关注列表 |
| FollowHistoryDataService.getFollowed | 语法 | public List<StockPO> getFollowed() |
| 前置条件 | 无 |
| 后置条件 | 返回历史关注列表（不包括当前关注列表） |
| FollowHistoryDataService.clearFollow | 语法 | public boolean clearFollow() |
| 前置条件 | 无 |
| 后置条件 | 取消当前所有关注，并将关注股票全部添加到历史关注列表 |
| FollowHistoryDataService.clearFollowed | 语法 | public boolean clearFollowed() |
| 前置条件 | 无 |
| 后置条件 | 清空历史关注列表 |
| FollowHistoryDataService.clearAll | 语法 | public boolean clearAll() |
| 前置条件 | 无 |
| 后置条件 | 取消所有关注，并清空关注列表 |

### 3.6.3历史模块的设计原理

考虑到后续迭代很有可能添加策略、购买股票等功能，而过去关注过的股票、做出的策略有很强的参考意义，现阶段将关注历史独立出来，形成一个新的模块是很有意义的，可以方便地添加后续迭代。

## 3.7读取模块的分解

读取模块包括读取数据和解析数据。

### 3.7.1读取模块的职责

读取模块负责从网页获取股票数据，实现与外界的交互。

### 3.7.1读取模块的接口规范

表16 读取模块的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| ReadStockDataService.  getData（String url） | 语法 | public String getData(String url) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 根据网页url获取网页上的数据 |
| ReadStockDataService.  parseJson(String jsonStr,String key) | 语法 | public String[] parseJson(String jsonStr,String key) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 将json格式的字符串解析，并放入一个字符串数组中 |

### 3.7.2读取模块的设计原理

读取模块实现系统与外界的交互，负责读取股票数据。

## 3.8界面模块的设计

### 3.8.1系统界面



图4 自选股界面



图5 行情界面

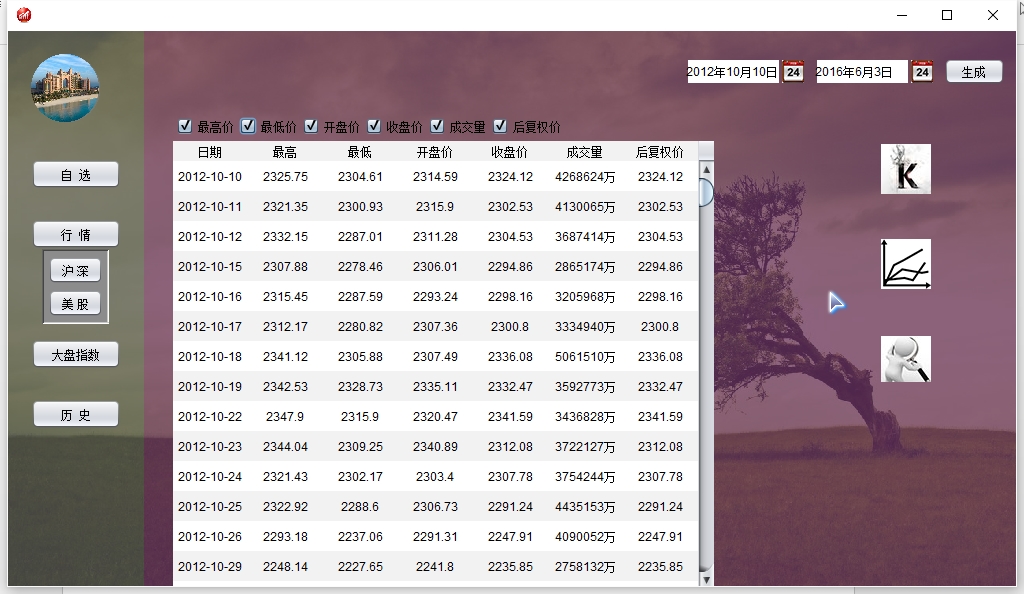


图6 大盘指数界面

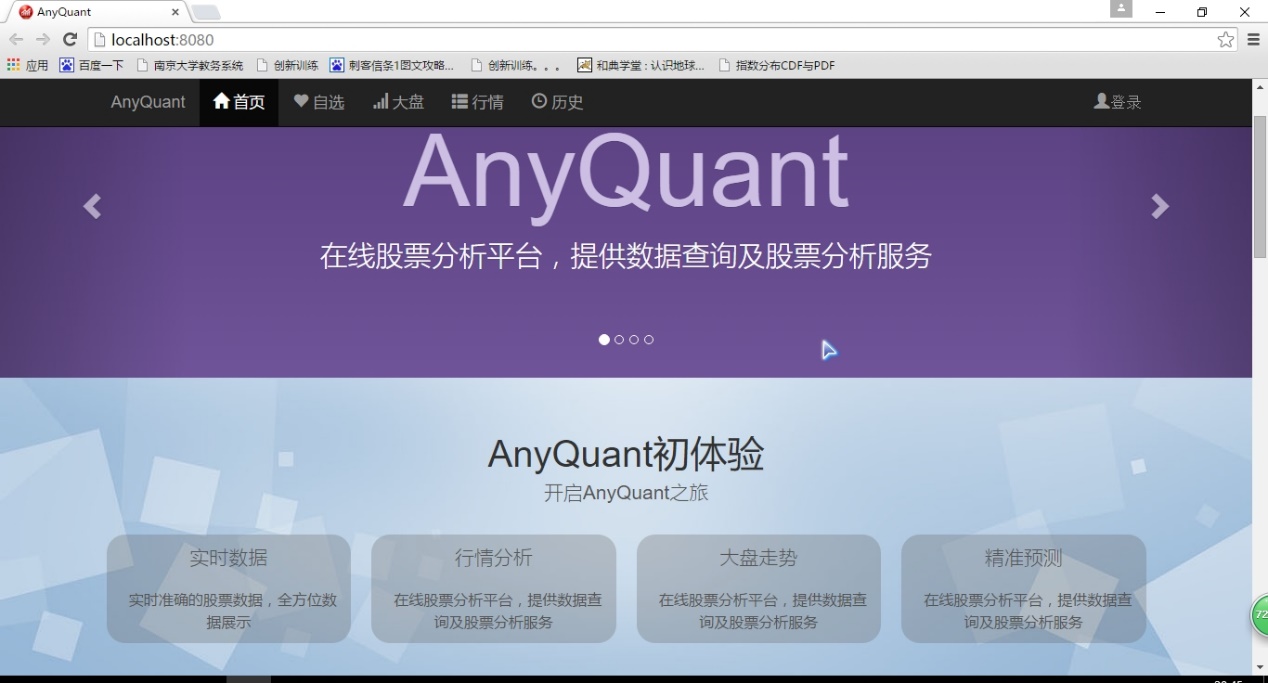


图7 首页界面（Web版）

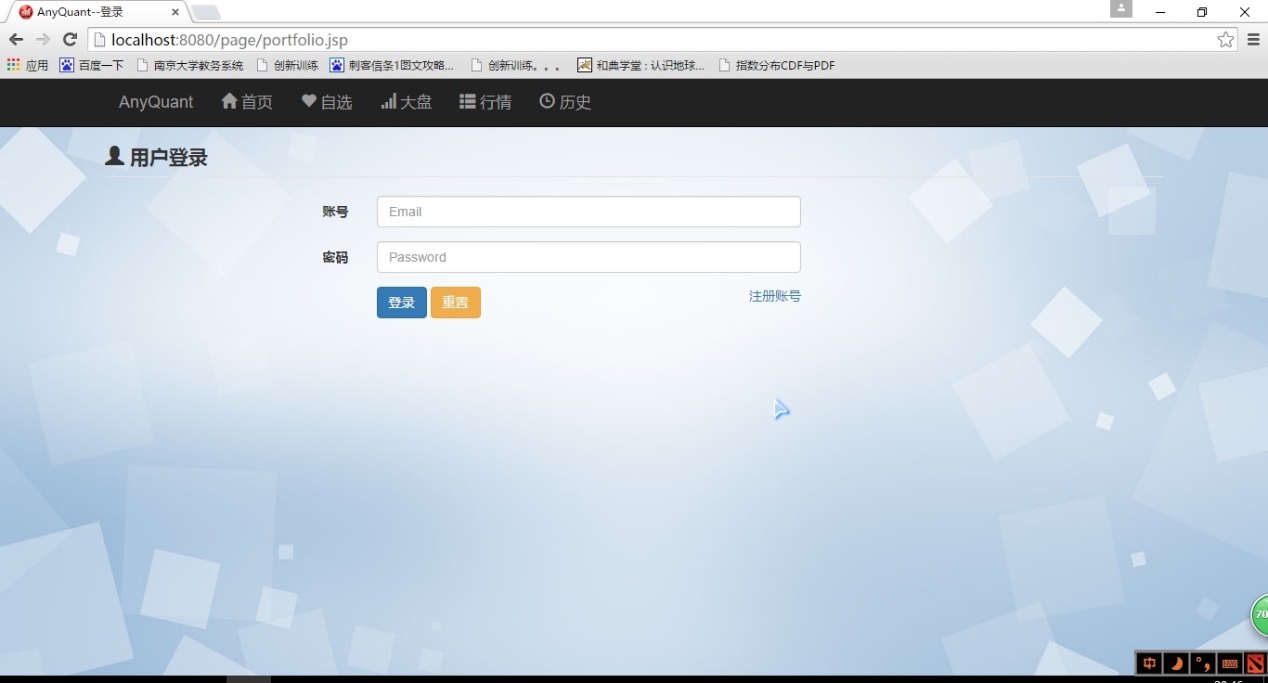


图8 登录界面（Web版）

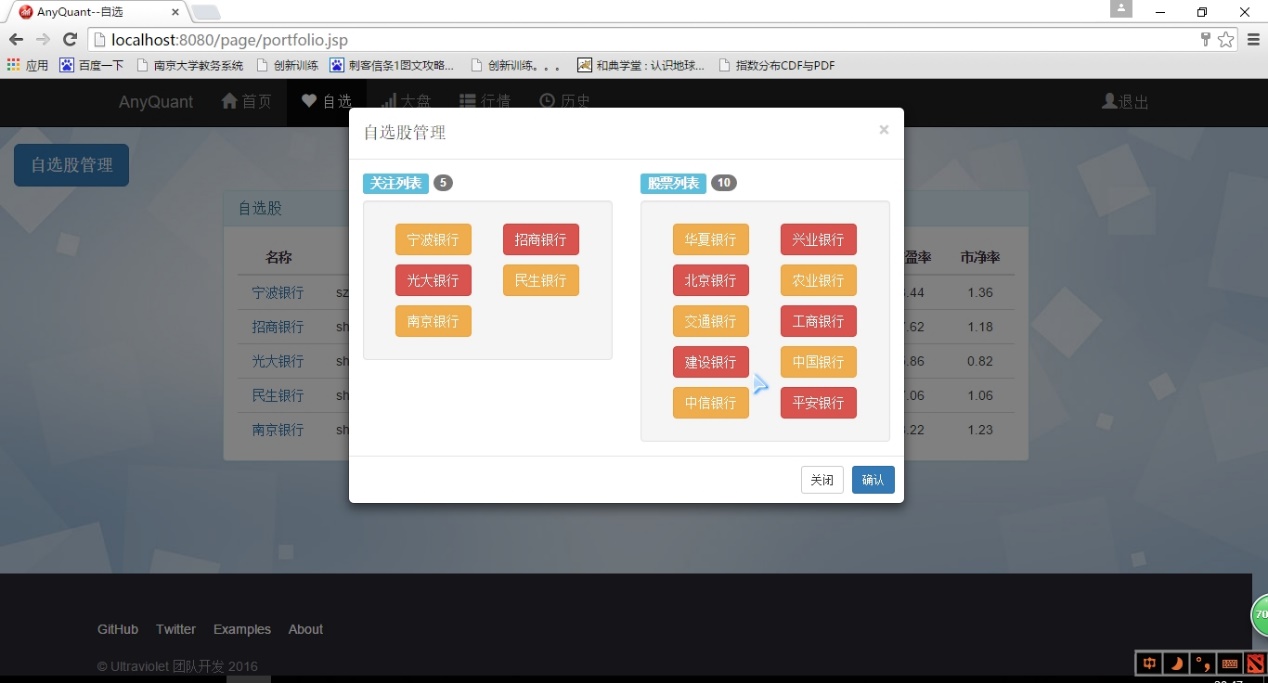


图9 自选股管理界面（Web版）



图10 股票K线图界面（Web版）