Quantour

**软件设计文档**

南京大学软件学院2015级Clam Team

费慧通，冯超，何林洋，梁先伟

2017.03.22

目录

[更新历史 2](#_Toc448516206)

[1.引言 3](#_Toc448516207)

[1.1编制目的 3](#_Toc448516208)

[1.2参考资料 3](#_Toc448516209)

[2.产品概述 3](#_Toc448516210)

[3.逻辑视角 3](#_Toc448516211)

[4.组合视角 5](#_Toc448516212)

[4.1开发包图 5](#_Toc448516213)

[4.2物理部署 6](#_Toc448516214)

[5.接口视角 6](#_Toc448516215)

[5.1模块的职责 6](#_Toc448516216)

[5.1.1用户界面层模块的职责 7](#_Toc448516217)

[5.1.2用户界面模块设计原理 7](#_Toc448516218)

[5.2业务逻辑层的分解 7](#_Toc448516219)

[5.3数据层的分解 1](#_Toc448516220)5

[5.3.1数据层模块的职责 1](#_Toc448516221)5

[5.3.2数据层模块的接口规范 1](#_Toc448516222)5

**更新历史**

|  |  |  |  |
| --- | --- | --- | --- |
| **修改人员** | **修改日期** | **修改原因** | **版本号** |
| 梁先伟 | 2017.03.22 | 初稿 | V2.0 |
| 梁先伟 | 2017.04.10 | 修订稿 | V2.1 |
| 梁先伟 | 2017.04.16 | 修订稿 | V2.2 |
| 梁先伟 | 2017.04.19 | 最终稿 | V2.3 |
|  |  |  |  |
|  |  |  |  |

# 引言

## 1.1编制目的

本文档详细完成对股票分析软件的详细设计，达到指导后续软件构造的目的，同时实现和测试人员及用户的沟通。

本文档面向开发人员、测试人员及最终用户编写，是了解该软件的导航。

## 1.2参考资料

IEEE标准；

《股票展示分析系统需求规格说明文档》 。

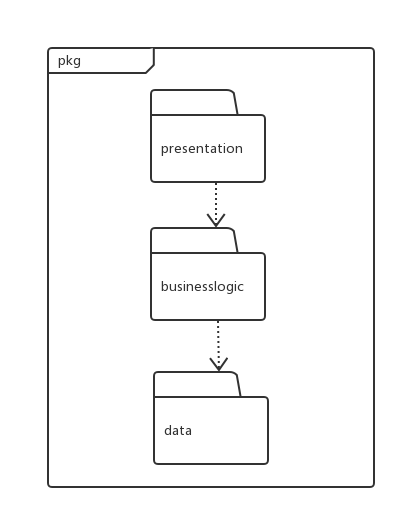
# 产品概述

# 参考《股票展示分析系统需求规格说明文档》中对产品的概括描述。

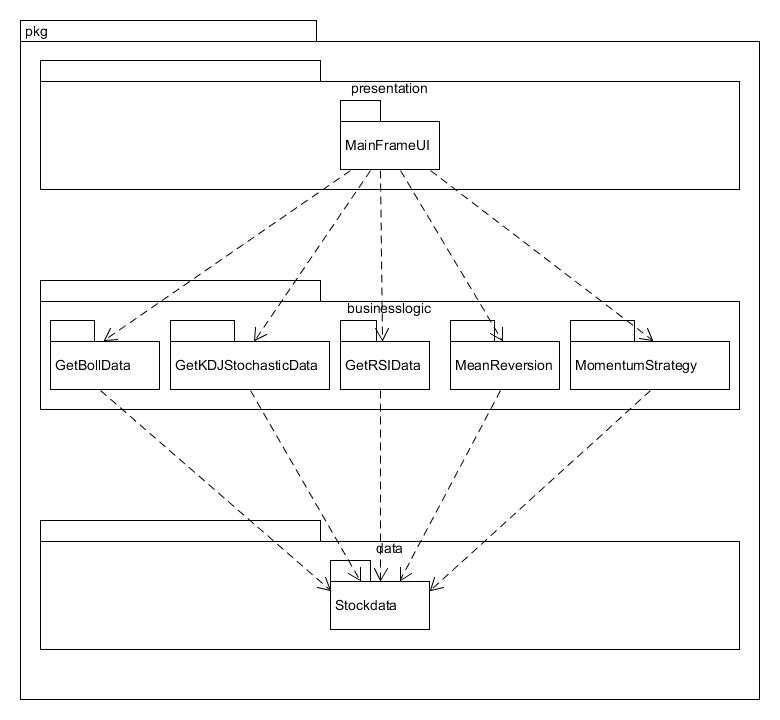
**3.逻辑视角**

股票分析软件中，我们小组选择了分层体系结构风格，将软件系统分为3层（展示层、业务逻辑层、数据层）。

展示层主要负责界面框架的实现，业务逻辑层则是对业务逻辑处理的实现，迭代一中数据层负责从文件中读取数据。分层体系结构的逻辑视角和逻辑设计方案如图3-1和图3-2所示。



**图3-1 参照体系结构风格的包图表达逻辑视角**

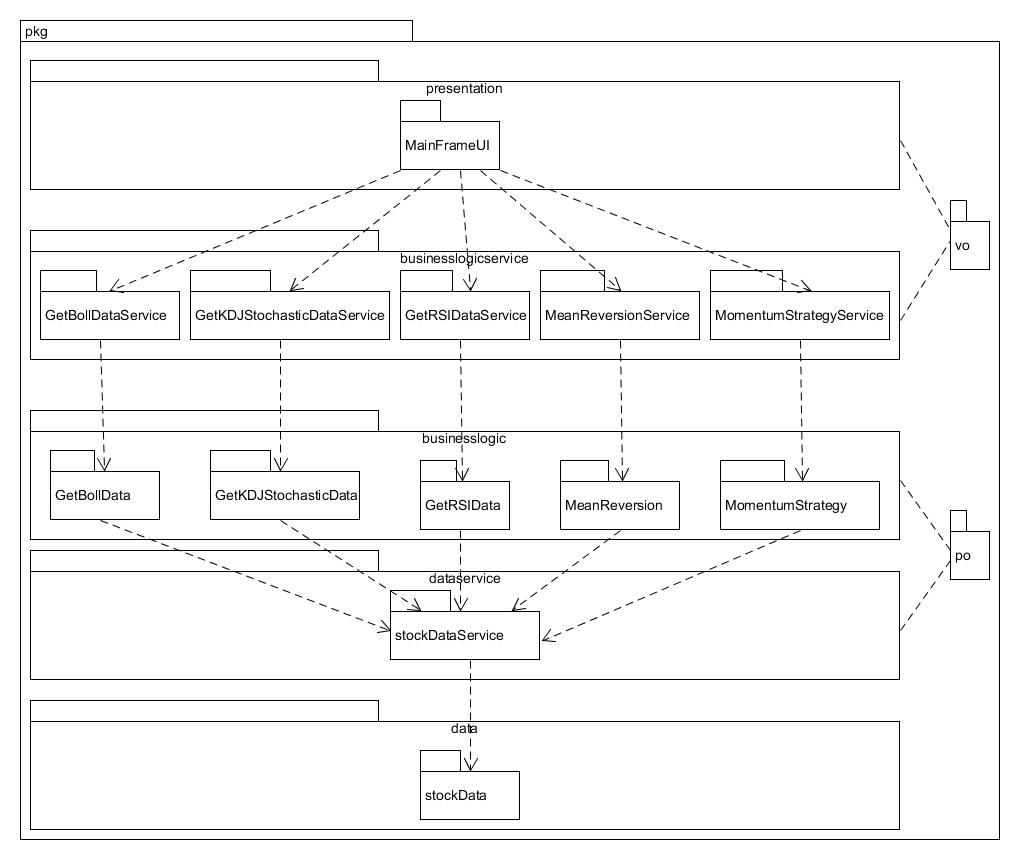
****

**图3-1 软件体系结构逻辑设计方案**

1. **组合视角**

**4.1开发包图**

**图4.1-1 开发包图**



**4.2物理部署**

# 接口视角

## **5.1模块的职责**

**表5.1-1 各层的职责**

|  |  |
| --- | --- |
| **层** | **职责** |
| 用户界面层 | 展示股票信息 |
| 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 |
| 数据层 | 从文件中获取数据并保存，并做简单的数据处理 |

**表5.1-2 层之间调用的接口**

|  |  |  |
| --- | --- | --- |
| **接口** | **服务调用方** | **服务提供方** |
| getBollDataService | 客户端展示层 | 客户端业务逻辑层 |
| GetKDJStochasticDataService |
| GetRSIDataService |
| MeanReversionService |
| MomentumStrategyService |
| StockDataService | 客户端业务逻辑层 | 客户端数据层 |

### 5.1.1用户界面层模块的职责

如表5.1.1-1所示为用户界面层模块的职责。

**表5.1.1-1 用户界面层模块的职责**

|  |  |
| --- | --- |
| **模块** | **职责** |
| MainFrame | 分为三个部分，分别负责展示个股K线图+均线图、两只股票数据比较和市场行情相关数据。 |

**5.1.2用户界面模块设计原理**

用户界面主要利用利用JavaFX实现。

**5.2业务逻辑层的分解**

**5.2.1 业务逻辑层模块的职责**

业务逻辑层模块的职责如下表所示：

|  |  |
| --- | --- |
| **模块** | **职责** |
| getBollDataService | 实现布林线指标（Boll）策略 |
| GetKDJStochasticDataService | 实现随机指标（KDJ）策略 |
| GetRSIDataService | 实现相对强弱指标（RSI）策略 |
| MeanReversionService | 实现均值回归策略，并计算出统计功能可视化图的各项数据 |
| MomentumStrategyService | 实现动量策略，并计算出统计功能可视化图的各项数据 |

**5.2.2 业务逻辑层模块的接口规范**

**5.2.2.1** getBollData**模块**

**表5.2.2.1** getBollData**模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| getBollData.getBollData | 语法 | | **public** OHLCSeriesCollection getBollData(String condition, String begin, String end); |
| 前置条件 | | Data层成功拿到数据 |
| 后置条件 | | 对指定时间段内数据进行计算，得到布林线所需的三轨线数据，并对计算所得数据进行存储 |
| getBollData.getAverageData | 语法 | | **public** TimeSeriesCollection getAverageData(String Code, String Begin, String End); |
| 前置条件 | | Data层成功拿到数据 |
| 后置条件 | | 对数据进行均线所需计算，得到中轨线数据，并对计算所得数据进行进行存储 |
| getBollData.GetBollBackTestGraphData | 语法 | | **public** DefaultCategoryDataset GetBollBackTestGraphData(String section, ArrayList<String> stockPool, String begin, String end); |
| 前置条件 | | Data层成功拿到数据 |
| 后置条件 | | 对策略和基准的累计收益率进行计算，并进行存储 |
| getBollData.getParameter | 语法 | | **public** MeanReversionVO getParameter(); |
| 前置条件 | | 已经对某一段时间内的布林线指标进行过计算，并得到保存的数据 |
| 后置条件 | | 将数据返回到界面层 |
| getBollData.getSuggest | 语法 | | **public** ArrayList<String> getSuggest(); |
| 前置条件 | | 已经对某一段时间内的布林线指标进行过计算，并得到保存的数据 |
| 后置条件 | | 根据计算出的数据，得到对于股民的建议 |
| **需要的服务（需接口）** | | | |
| 服务名 | | 服务 | |
| stockData. getStockByNameAndDate(String Name,String Begin,String End) | | 获得指定名称股票的指定日期内的数据 | |
| stockData.JudgeIfTheLast(**int** Code, String Begin) | | 判断是否到了股票发行的那一天 | |
| stockData.getStockOpenBySection(String section, String begin, String end) | | 根据股票板块获得开盘价 | |
| stockData.getStockAdjCloseBySection(String section, String begin, String end); | | 根据股票板块获得复权收盘价 | |
| GraphUtil.getStockData(String condition, String begin, String end) | | 根据输入情况读取出文件中所有符合条件的股票数据 | |
| GraphUtil.GetOrigin(String Code,String Begin) | | 获得所选择股票的指定日期的上一个有效日期 | |
| MovingAverage.getAveData(ArrayList<Double> adjCloses, **int** days, **int** beforeDays) | | 计算几天内的均线图所需的均线数据 | |
| ParameterCalculation.getStandardDeviation(ArrayList<Double> data) | | 计算所需的标准差数据 | |
| ParameterCalculation.getMaxDrawdownLevel(ArrayList<Double> valueCollection) | | 计算所需的最大回撤率数据 | |
| ParameterCalculation.getBetaCoefficient(ArrayList<Double> marketIncome, ArrayList<Double> strategicIncome) | | 计算所需的贝塔系数数据 | |
| ParameterCalculation.getAlphaCoefficient(ArrayList<Double> marketIncome, ArrayList<Double> strategicIncome) | | 计算所需的阿尔法系数数据 | |
| ParameterCalculation.getSharpeRatio() | | 计算所需的夏普比率数据 | |

**5.2.2.2** getKDJStochasticData**模块**

**表5.2.2.2** getKDJStochasticData**模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| getKDJStochasticData.getKDJStochasticData | 语法 | **public** DefaultCategoryDataset getKDJStochasticData(String condition, String begin, String end); |
| 前置条件 | Data层已经获得所有的数据 |
| 后置条件 | 计算最后一个计算周期的未成熟随机值RSV、K值、D值、J值，并对数据进行存储 |
| getKDJStochasticData.GetKDJStochasticBackTestGraphData | 语法 | **public** DefaultCategoryDataset GetKDJStochasticBackTestGraphData(String section, ArrayList<String> stockPool, String begin, String end); |
| 前置条件 | Data层已经获得所有的数据 |
| 后置条件 | 对策略和基准的累计收益率进行计算，并进行存储 |
| getKDJStochasticData.getParameter | 语法 | **public** MeanReversionVO getParameter(); |
| 前置条件 | 已经对某一段时间内的随机指标进行过计算，并得到保存的数据 |
| 后置条件 | 将数据返回到界面层 |
| getKDJStochasticData.getSuggest | 语法 | **public** ArrayList<String> getSuggest(); |
| 前置条件 | 已经对某一段时间内的随机指标进行过计算，并得到保存的数据 |
| 后置条件 | 根据计算出的数据，得到对于股民的建议 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| stockData. getStockByNameAndDate(String Name,String Begin,String End) | 获得指定名称股票的指定日期内的数据 | |
| stockData.JudgeIfTheLast(**int** Code, String Begin) | 判断是否到了股票发行的那一天 | |
| stockData.getStockOpenBySection(String section, String begin, String end) | 根据股票板块获得开盘价 | |
| stockData.getStockAdjCloseBySection(String section, String begin, String end); | 根据股票板块获得复权收盘价 | |
| GraphUtil.getStockData(String condition, String begin, String end) | 根据输入情况读取出文件中所有符合条件的股票数据 | |
| GraphUtil.GetOrigin(String Code,String Begin) | 获得所选择股票的指定日期的上一个有效日期 | |
| ParameterCalculation.getMaxDrawdownLevel(ArrayList<Double> valueCollection) | 计算所需的最大回撤率数据 | |
| ParameterCalculation.getBetaCoefficient(ArrayList<Double> marketIncome, ArrayList<Double> strategicIncome) | 计算所需的贝塔系数数据 | |
| ParameterCalculation.getAlphaCoefficient(ArrayList<Double> marketIncome, ArrayList<Double> strategicIncome) | 计算所需的阿尔法系数数据 | |
| ParameterCalculation.getSharpeRatio() | 计算所需的夏普比率数据 | |

**5.2.2.3** GetRSIData**模块**

**表5.2.2.3** GetRSIData**模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| GetRSIData.getRSIGraphData | 语法 | **public** DefaultCategoryDataset getRSIGraphData(String condition, String begin, String end); |
| 前置条件 | Data层已经获得所有的数据 |
| 后置条件 | 对某一段时间内的相对强弱指标进行计算，并对计算结果进行存储 |
| GetRSIData.GetRSIBackTestGraphData | 语法 | **public** DefaultCategoryDataset GetRSIBackTestGraphData(String section, ArrayList<String> stockPool, String begin, String end); |
| 前置条件 | Data层成功拿到数据 |
| 后置条件 | 对策略和基准的累计收益率进行计算，并进行存储 |
| GetRSIData.getParameter | 语法 | **public** MeanReversionVO getParameter(); |
| 前置条件 | 已经对某一段时间内的相对强弱指标进行过计算，并得到保存的数据 |
| 后置条件 | 将数据返回到界面层 |
| GetRSIData.getSuggest | 语法 | **public** String getSuggest(); |
| 前置条件 | 已经对某一段时间内的相对强弱指标进行过计算，并得到保存的数据 |
| 后置条件 | 根据计算出的数据，得到对于股民的建议 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| stockData. getStockByNameAndDate(String Name,String Begin,String End) | 获得指定名称股票的指定日期内的数据 | |
| stockData.JudgeIfTheLast(**int** Code, String Begin) | 判断是否到了股票发行的那一天 | |
| stockData.getStockOpenBySection(String section, String begin, String end) | 根据股票板块获得开盘价 | |
| stockData.getStockAdjCloseBySection(String section, String begin, String end); | 根据股票板块获得复权收盘价 | |
| GraphUtil.getStockData(String condition, String begin, String end) | 根据输入情况读取出文件中所有符合条件的股票数据 | |
| GraphUtil.GetOrigin(String Code,String Begin) | 获得所选择股票的指定日期的上一个有效日期 | |
| ParameterCalculation.getMaxDrawdownLevel(ArrayList<Double> valueCollection) | 计算所需的最大回撤率数据 | |
| ParameterCalculation.getBetaCoefficient(ArrayList<Double> marketIncome, ArrayList<Double> strategicIncome) | 计算所需的贝塔系数数据 | |
| ParameterCalculation.getAlphaCoefficient(ArrayList<Double> marketIncome, ArrayList<Double> strategicIncome) | 计算所需的阿尔法系数数据 | |
| ParameterCalculation.getSharpeRatio() | 计算所需的夏普比率数据 | |

**5.2.2.4** MeanReversion**模块**

**表5.2.2.1** MeanReversion**模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| MeanReversion.getMeanReversionGraphData | 语法 | | **public** DefaultCategoryDataset getMeanReversionGraphData(String section, ArrayList<String> stockPool, **int** shares, **int** holdPeriod, **int** formingPeriod, String begin, String end); |
| 前置条件 | | Data层成功拿到数据 |
| 后置条件 | | 根据股票池，持有期，形成期等信息对均值回归策略以及基准的累计收益率进行计算，并对结果进行存储 |
| MeanReversion.GetMeanReturnRateGraphData | 语法 | | **public** DefaultCategoryDataset GetMeanReturnRateGraphData(String section, ArrayList<String> stockPool, **int** shares, **int** holdPeriod, **int** formingPeriod, String begin,String end); |
| 前置条件 | | Data层成功拿到数据 |
| 后置条件 | | 根据股票池，持有期，形成期等信息对均值回归策略的超额收益率进行计算，并计算对结果进行存储 |
| MeanReversion.GetMeanWinningPercentageGraphData | 语法 | | **public** DefaultCategoryDataset GetMeanWinningPercentageGraphData(String section, ArrayList<String> stockPool, **int** shares, **int** holdPeriod, **int** formingPeriod, String begin, String end); |
| 前置条件 | | Data层成功拿到数据 |
| 后置条件 | | 根据股票池，持有期，形成期等信息对均值回归策略的股票胜率进行计算，并计算对结果进行存储 |
| MeanReversion.GetDistributionHistogramGraphData | 语法 | | **public** DefaultCategoryDataset GetDistributionHistogramGraphData(String section, ArrayList<String> stockPool, **int** shares, **int** holdPeriod, **int** formingPeriod, String begin, String end); |
| 前置条件 | | Data层成功拿到数据 |
| 后置条件 | | 根据股票池，持有期，形成期等信息对均值回归策略的收益率分布进行计算，并计算对结果进行存储 |
| MeanReversion.getParameter | 语法 | | **public** MeanReversionVO getParameter(); |
| 前置条件 | | 计算获得年化收益率,基准年化收益率,最大回撤,阿尔法,贝塔,夏普比率，并对结果进行存储 |
| 后置条件 | | 将数据返回到界面层 |
| MeanReversion.getCalculationCycle | 语法 | | **public** ArrayList<MeanReturnRateVO> getCalculationCycle(); |
| 前置条件 | | 获得相对强弱计算周期,获得超额收益,获得1年内胜率，并对结果进行存储 |
| 后置条件 | | 将数据返回到界面层 |
| MeanReversion.getDistributionHistogram | 语法 | | **public** DistributionHistogramVO getDistributionHistogram(); |
| 前置条件 | | 获得正收益次数,负收益次数,胜率，并对结果进行存储 |
| 后置条件 | | 将数据返回到界面层 |
| **需要的服务（需接口）** | | | |
| 服务名 | | 服务 | |
| stockData. getStockByNameAndDate(String Name,String Begin,String End) | | 获得指定名称股票的指定日期内的数据 | |
| stockData. getStockByCodeAndDate(String Name,String Begin,String End) | | 获得指定股票号股票的指定日期内的数据 | |
| stockData.getStockOpenBySection(String section, String begin, String end) | | 根据股票板块获得开盘价 | |
| stockData.getStockAdjCloseBySection(String section, String begin, String end); | | 根据股票板块获得复权收盘价 | |
| MeanReversionUtil.heapSort(**double**[] array) | | 对偏离度进行排序，并选出最高的五个 | |
| MovingAverage.getAveData(ArrayList<Double> adjCloses, **int** days, **int** beforeDays) | | 计算几天内的均线图所需的均线数据 | |
| GraphUtil.GetOrigin(String Code,String Begin) | | 获得所选择股票的指定日期的上一个有效日期 | |
| ParameterCalculation.getMaxDrawdownLevel(ArrayList<Double> valueCollection) | | 计算所需的最大回撤率数据 | |
| ParameterCalculation.getBetaCoefficient(ArrayList<Double> marketIncome, ArrayList<Double> strategicIncome) | | 计算所需的贝塔系数数据 | |
| ParameterCalculation.getAlphaCoefficient(ArrayList<Double> marketIncome, ArrayList<Double> strategicIncome) | | 计算所需的阿尔法系数数据 | |
| ParameterCalculation.getSharpeRatio() | | 计算所需的夏普比率数据 | |

**5.2.2.5** MomentumStrategy**模块**

**表5.2.2.5** MomentumStrategy**模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| MomentumStrategygetMStrategyComparedGraph | 语法 | | **public** DefaultCategoryDataset getMStrategyComparedGraph(String Begin, String End, **int** existTime, **int** holdTime) **throws** ParseException; |
| 前置条件 | | Data层成功拿到数据 |
| 后置条件 | | 根据股票池，持有期，形成期等信息对动量策略以及基准的累计收益率进行计算，并对结果进行存储 |
| MomentumStrategy.getMStrategyWinningGraph | 语法 | | **public** DefaultCategoryDataset getMStrategyWinningGraph(**boolean** isHold, **int** Time, String Begin, String End) **throws** ParseException; |
| 前置条件 | | Data层成功拿到数据 |
| 后置条件 | | 根据股票池，不同持有期/形成期等信息对动量策略的额外收益率进行计算，并计算对结果进行存储 |
| MomentumStrategy.getMStrategyExtraProfitGraph | 语法 | | **public** DefaultCategoryDataset getMStrategyExtraProfitGraph(**boolean** isHold, **int** Time, String Begin, String End) **throws** ParseException; |
| 前置条件 | | Data层成功拿到数据 |
| 后置条件 | | 根据股票池，不同持有期/形成期等信息对动量策略的策略胜率进行计算，并计算对结果进行存储 |
| MomentumStrategy.getMStrategyYieldGraph | 语法 | | **public** DefaultCategoryDataset getMStrategyYieldGraph(String Begin, String End, **int** holdTime, **int** existTime) **throws** ParseException; |
| 前置条件 | | Data层成功拿到数据 |
| 后置条件 | | 根据股票池，不同持有期/形成期等信息对动量策略的收益率分布进行计算，并计算对结果进行存储 |
| MomentumStrategy.getParameter | 语法 | | **public** MeanReversionVO getParameter(); |
| 前置条件 | | 计算获得年化收益率,基准年化收益率,最大回撤,阿尔法,贝塔,夏普比率，并对结果进行存储 |
| 后置条件 | | 将数据返回到界面层 |
| MomentumStrategy.getCalculationCycle | 语法 | | **public** ArrayList<MeanReturnRateVO> getCalculationCycle(); |
| 前置条件 | | 获得相对强弱计算周期,获得超额收益,获得1年内胜率，并对结果进行存储 |
| 后置条件 | | 将数据返回到界面层 |
| MomentumStrategy.getDistributionHistogram | 语法 | | **public** DistributionHistogramVO getDistributionHistogram(); |
| 前置条件 | | 获得正收益次数,负收益次数,胜率，并对结果进行存储 |
| 后置条件 | | 将数据返回到界面层 |
| **需要的服务（需接口）** | | | |
| 服务名 | | 服务 | |
| stockData. getStockByNameAndDate(String Name,String Begin,String End) | | 获得指定名称股票的指定日期内的数据 | |
| stockData.getDate(String begin, String end); | | 根据股票日期获得日期数据 | |
| stockData.getStockOpenBySection(String section, String begin, String end) | | 根据股票板块获得开盘价 | |
| stockData.getStockAdjCloseBySection(String section, String begin, String end); | | 根据股票板块获得复权收盘价 | |

**5.3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，包括对数据的增、删、改、查。

**5.3.1数据层模块的职责**

数据层模块的职责如表5.3.1-1所示。

|  |  |
| --- | --- |
| **模块** | **职责** |
| stockData | 负责对文件中的股票数据进行读取 |

**表5.3.1-1 数据层模块的职责**

**5.3.2数据层模块的接口规范**

**5.3.2.1 stockData模块**

**表5.3.2.1 stockData模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| StockData.getStockByCodeAndDate | 语法 | **public** ArrayList<stockPO> getStockByCodeAndDate(**int** Code,String Begin,String End); |
| 前置条件 | 文件中存储着股票所有的数据 |
| 后置条件 | 读取出文件中符合用户所选择编号和起始日期的股票数据信息 |
| StockData.getStockByNameAndDate | 语法 | **public** ArrayList<stockPO> getStockByNameAndDate(String Name,String Begin,String End); |
| 前置条件 | 文件中存储着股票所有的数据 |
| 后置条件 | 读取出文件中用户所选择名称和起始日期的股票数据信息 |
| StockData.getStockByDate | 语法 | **public** ArrayList<stockPO> getStockByDate(String Begin) |
| 前置条件 | 文件中存储着股票所有的数据 |
| 后置条件 | 读取出文件中用户所选择起始日期的股票数据信息 |
| StockData.getVolumeByDateAndCode | 语法 | **public** **int** getVolumeByDateAndCode(**int** Code,String Begin) |
| 前置条件 | 文件中存储着股票所有的数据 |
| 后置条件 | 获得指定日期股票的交易量，若该日期不是工作日，返回0；否则返回1号股票的交易量，即非0； |
| StockData.getVolumeByDateAndName | 语法 | **public** **int** getVolumeByDateAndName(String Name,String Begin) |
| 前置条件 | 文件中存储着股票所有的数据 |
| 后置条件 | 获得指定日期股票的交易量，若该日期不是工作日，返回0；否则返回1号股票的交易量，即非0； |
| StockData.JudgeIfTheLast | 语法 | **public** **int** JudgeIfTheLast(**int** Code, String Begin) |
| 前置条件 | 文件中存储着股票所有的数据 |
| 后置条件 | 根据日期和股票号得到当日交易量，若该日期不是工作日，返回0；否则返回股票1号的交易量，即非0；是股票最开始一天则返回-1； |
| StockData.getCodeAndName | 语法 | **public** ArrayList<stockPO> getCodeAndName() |
| 前置条件 | 文件中存储着股票所有的数据 |
| 后置条件 | 读取出文件中所有的股票标号和名称 |
| StockData.getCodeAndNameByPlate | 语法 | **public** ArrayList<stockPO> getCodeAndNameByPlate(String plate); |
| 前置条件 | 文件中存储着股票所有的数据 |
| 后置条件 | 根据用户所选择的板块读取对应板块的股票数据信息 |
| StockData.getStockOpenBySection | 语法 | **public** ArrayList<Double> getStockOpenBySection(String section, String begin, String end); |
| 前置条件 | 文件中存储着股票所有的数据 |
| 后置条件 | 根据用户做选择的股票板块获得对应的股票开盘价数据 |
| StockData.getStockAdjCloseBySection | 语法 | **public** ArrayList<Double> getStockAdjCloseBySection(String section, String begin, String end); |
| 前置条件 | 文件中存储着股票所有的数据 |
| 后置条件 | 根据用户做选择的股票板块获得对应的股票复权收盘价数据 |
| StockData.getDate | 语法 | **public** ArrayList<String> getDateBySection(String begin, String end); |
| 前置条件 | 文件中存储着股票所有的数据 |
| 后置条件 | 根据用户做选择的日期获得对应的股票日期数据 |
| StockData.getCodeByName | 语法 | **public** **int** getCodeByName(String name); |
| 前置条件 | 文件中存储着股票所有的数据 |
| 后置条件 | 根据用户所选择的股票名称得到该股票对应的股票号 |
| StockData.getStockByOneDay | 语法 | **public** stockPO getStockByOneDay(String Name,Date day); |
| 前置条件 | 文件中存储着股票所有的数据 |
| 后置条件 | 根据用户所选择的股票名称和日期得到该股票的数据 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| Data.getAllStock(StringbeginDate, String endDate, String stockID) | 获得所有年份和交易所的所有股票数据 | |