**迭代一软件架构设计文档**

**作者：何梦婷，庞云奎**

目录

1.引言3

2.产品概述3

3.体系结构模型3

3.1整体架构描述3

3.1.1逻辑视角3

3.1.2开发包图4

3.2模块的职责5

3.3用户界面层的分解5

3.3.1用户界面层模块的职责5

3.3.2界面跳转的设计5

3.4业务逻辑层的分解5

3.4.1业务逻辑层模块的职责5

3.4.2业务逻辑层模块的接口规范5

3.5集成层的分解7

3.5.1集成层模块的职责7

3.5.2集成层模块的接口规范7

3.6数据层的分解7

3.6.1数据层模块的职责7

3.6.2数据层模块的接口规范8

3.7物理部署8

4.模型之间的映射8

4.1 基于接口的模型关系8

4.2 数据模型9

5.设计原理10

1.引言

1.1编制目的

本文档提供AnyQuant系统的软件架构概览，采用若干架构视图描述系统的不同方面，以便表示构造系统所需要的重要架构决策。

本报告面向开发人员、测试人员及最终用户而编写，用于指导下一循环的代码开发和测试工作和了解系统。

1.2参考资料

1）软件需求规格说明书

1.3变更记录

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 变更人 | 变更内容 | 版本 |
| 2016/3/8 | 庞云奎 | 修改接口规范 | V2.0 |
|  |  |  |  |
|  |  |  |  |

2.产品概述

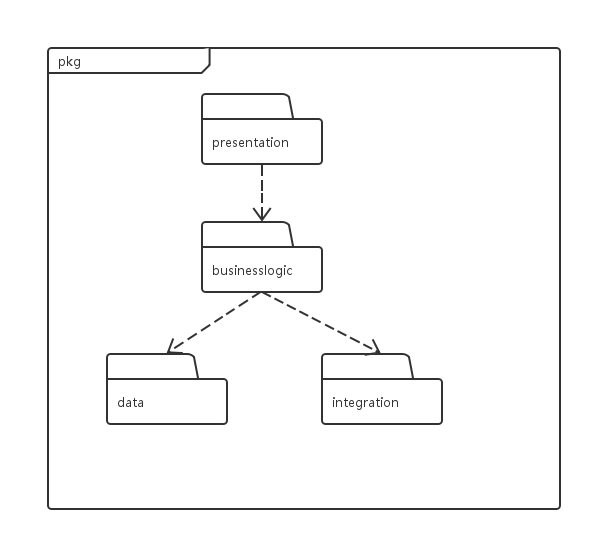
参考软件需求规格说明书中对产品的概括描述。

3. 体系结构模型

3.1整体架构描述

3.1.1逻辑视角

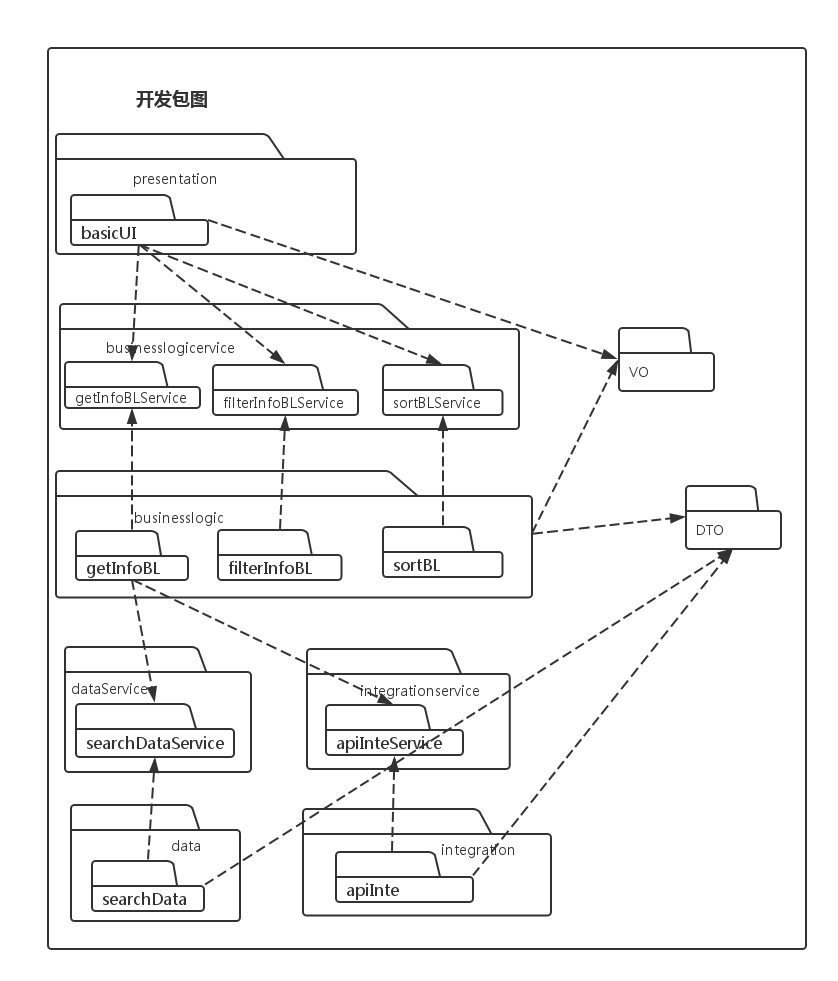
在AnyQuant系统中，选择了分层的体系风格（设计理由参见**5.设计原理**），将系统分为4层（展示层、业务逻辑层、数据层、集成层）能够很好地示意整个高层抽象展示层包含GUI界面的实现，业务逻辑层包含业务逻辑处理的实现，数据层负责数据的持久化和访问，集成层负责与外部接口的交互。

分层体系的逻辑视角和逻辑设计方案如图。

3.1.2开发包图

AnyQuant系统的最终开发包设计如表所示

|  |  |
| --- | --- |
| 开发（物理）包 | 依赖的其他开发包 |
| basicUI | vo, filterBLService, getInfoBLService, sortBLService |
| filterBLService |  |
| filterBL | filterBLService, apiInteService , vo, dto |
| getInfoBLService |  |
| getInfoBL | getInfoBLService, vo, dto |
| sortBLService |  |
| sortBL | sortBLService, vo, dto |
| apiInteService |  |
| apiInte | dto |
| vo |  |
| dto |  |



3.2模块的职责

系统各层的职责：

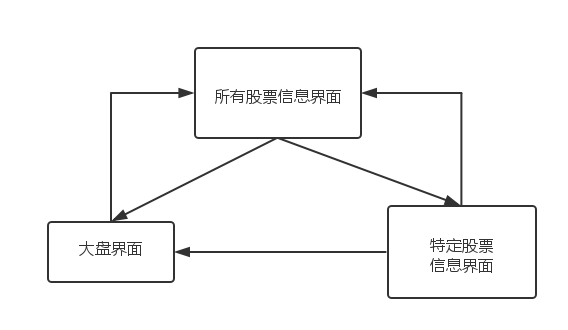
|  |  |
| --- | --- |
| 层 | 职责 |
| 用户界面层 | 用于前台的界面展示 |
| 业务逻辑层 | 包含业务控制和逻辑逻辑处理 |
| 数据层 | 用于读取和存储持久化数据 |
| 集成层 | 与外界系统交互，调用并解析AnyQuant的API |

3.3用户界面层的分解

3.3.1用户界面层模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| basicUI | 负责显示股票信息，接收与用户的交互信息 |

3.3.2界面跳转的设计

 根据需求，系统存在3个界面：大盘界面，所有股票信息界面，特定股票信息界面。界面跳转关系如图：

3.4业务逻辑层的分解

3.4.1业务逻辑层模块的职责：

|  |  |
| --- | --- |
| 模块 | 职责 |
| getInfoBL | 负责将数据从集成层传输到展示层 |
| filterInfoBL | 负责对数据进行过滤 |
| sortBL | 负责将数据按某种性质排序 |

3.4.2业务逻辑层模块的接口规范：

getInfoBL模块的接口规范如表所示：

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Check.checkNumber | 语法 | public double checkNumber(String number) |
| 前置条件 | 无 |
| 后置条件 | 判断字符串能否转化成数字，如果能返回结果，不能返回-1 |
| Share.getNameList | 语法 | public Iterator getNameList () |
| 前置条件 | 存在符合条件的股票代码 |
| 后置条件 | 返回所有符合条件的股票代码 |
| Share.getInfoList | 语法 | public ArrayList getInfoList (String strategy) |
| 前置条件 | 无 |
| 后置条件 | 返回所有股票指定的信息 |
| Share.getSpecifiedInfo | 语法 | public ArrayList getSpecifiedInfo(String name, String strategy) |
| 前置条件 | 存在传入的股票代码 |
| 后置条件 | 返回最近一个月内特定股票的指定的信息 |
| Share.recordHistory | 语法 | public void recordHistory(String name) |
| 前置条件 | 存在传入的股票代码 |
| 后置条件 | 记录一次搜索的股票代码 |
| Benchmark.getList | 语法 | public ArrayList getList() |
| 前置条件 | 无 |
| 后置条件 | 返回从2007至今大盘的详细信息，包括代码、日期、开盘价、收盘价、最高价、最低价、交易量、交易金额 |
| Benchmark.getList | 语法 | public ArrayList getList(String startDate,String endDate) |
| 前置条件 | 指定的开始到结束时间内存在大盘数据 |
| 后置条件 | 返回指定的开始到结束时间内大盘的详细信息，包括代码、日期、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）** | | |
| apiInteService.getShareList() | 获取所有股票的代码 | |
| apiInteService.getShareDetail(String name,String strategy) | 获取特定股票的指定信息 | |
| apiInteService.get Benchmark All(String startDate,String endDate) | 获得指定的开始到结束时间内大盘的所有信息 | |
| searchDataService.getSearchHistory() | 获得股票代码的搜索历史纪录 | |

filterInfoBL模块的接口规范如表所示：

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Filter.filter | 语法 | public ArrayList filter(StrategyVO strategy, ArrayList list) |
| 前置条件 | 无 |
| 后置条件 | 返回所有符合条件的股票信息列表 |

sortBL模块的接口规范如表所示：

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| Sort.sort | 语法 | public ArrayList sort(StrategyVO strategy, ArrayList list) |
| 前置条件 | 无 |
| 后置条件 | 返回按照要求排好序的股票信息列表 |

3.5集成层的分解

3.5.1集成层模块的职责：

|  |  |
| --- | --- |
| 模块 | 职责 |
| apiInteService | 股票信息获取的集成层接口，提供股票信息的查询服务 |
| apiInte | 股票信息获取的数据层接口实现，负责与外部系统的交互 |

3.5.2集成层模块的接口规范：

apiInteService模块的接口规范如表所示：

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| apiInteService.getShareList | 语法 | Public Iterator getShareList() |
| 前置条件 | 无 |
| 后置条件 | 得到所有的股票代码 |
| apiInteService.getShareDetail | 语法 | Public ArrayList getShareDetail(String name,String strategy) |
| 前置条件 | 存在传入的股票代码 |
| 后置条件 | 得到指定的股票的指定信息 |
| apiInteService.getBenchmark | 语法 | Public ArrayList getbenchmark (String startDate,String endDate) |
| 前置条件 | 从指定的开始日期到结束日期内存在大盘信息 |
| 后置条件 | 得到从指定的开始日期到结束日期内大盘的指定信息 |

3.6数据层的分解

3.6.1数据层模块的职责：

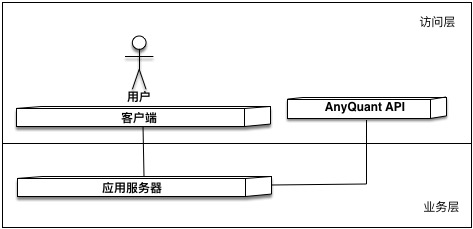
|  |  |
| --- | --- |
| 模块 | 职责 |
| searchDataService | 股票代码搜索记录的数据层接口，提供股票代码搜索记录的查询服务 |
| searchData | 股票代码搜索记录的数据层接口实现 |

3.6.2数据层模块的接口规范：

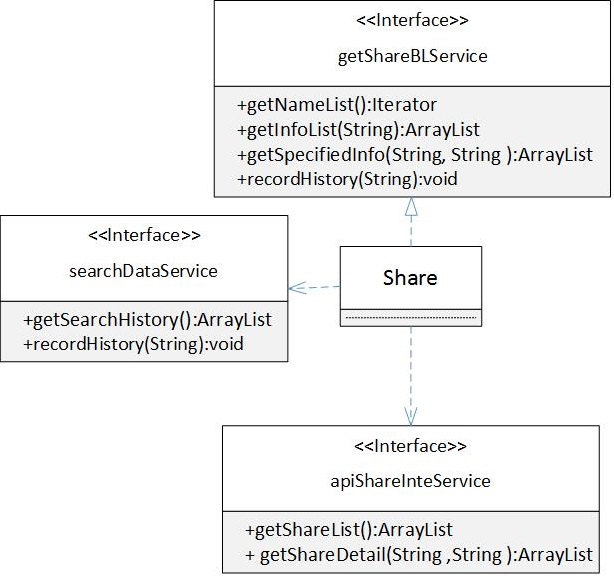
searchDataService模块的接口规范如表所示：

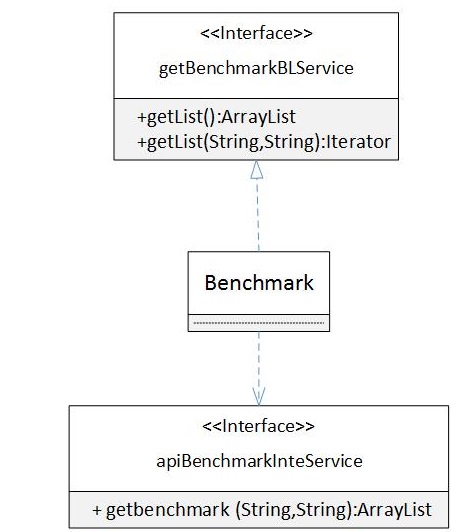
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| searchDataService.getSearchHistory | 语法 | Public Iterator getSearchHistory() |
| 前置条件 | 无 |
| 后置条件 | 得到所有的股票代码搜索纪录 |
| searchDataService.recordHistory | 语法 | Public void recordHistory(String name) |
| 前置条件 | 无 |
| 后置条件 | 记录一次搜索的股票代码 |

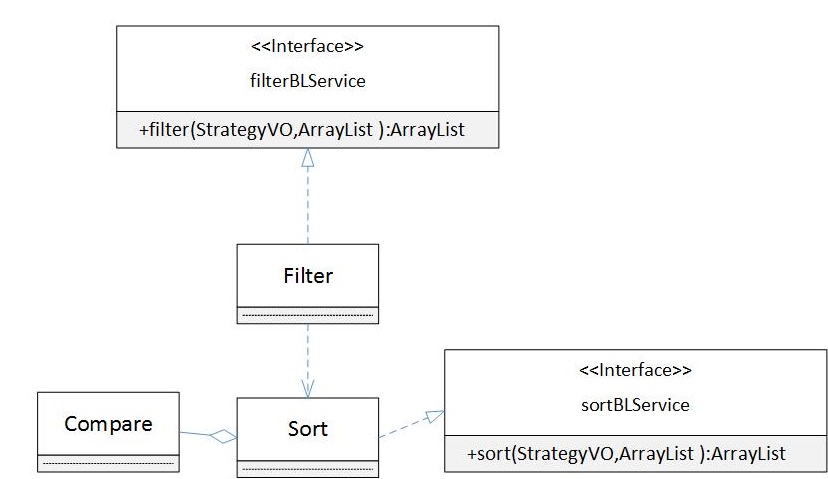
3.7物理部署：



4.模型之间的映射：

4.1 基于接口的模型关系：





4.2数据模型：

4.2.1DTO数据格式：

* StockDTO类包含股票的详细数据：股票代码、日期、开盘价、收盘价、最高价、最低价、交易量、交易金额、后复权价、市净率、市盈率
* StockOverviewDTO类包含股票的代码、日期、开盘价、收盘价

4.2.2持久化数据：

搜索历史纪录保存在.ser文件中，采用序列化存储方式

5.设计理由：

|  |  |
| --- | --- |
| 难点 | 系统需要的数据来源既有外部网络，也有持久化数据，以后还可能使用数据库 |
| 架构决策 | 对数据多来源的区分处理 |
| 假设 | 对本地持久存储数据和外部临时数据分开处理 |
| 可替代方法 | 选项1：数据层处理所有的数据  选项2：数据层处理持久化数据，集成层处理外部数据 |
| 选取的选项 | 选项2 |
| 理由 | 选项2能使系统结构更加清楚。在后期迭代引入数据库时，可能要增加dao包负责与数据库的数据交互，如果将持久化数据和外部数据分开处理，可减少模块变动带来的迷惑性，即dao只负责数据库，而不负责外部数据和序列化文件 |

|  |  |
| --- | --- |
| 难点 | 迭代二的需求目前不清楚 |
| 架构决策 | 应对需求变更 |
| 假设 | 迭代二的需求可能有较多变动 |
| 可替代方法 | 选项1：根据迭代一的需求构建接口，不考虑扩展  选项2：设计可扩展的接口，添加策略 |
| 选取的选项 | 选项2 |
| 理由 | 选项2虽然可能增加实现复杂度，但可以更好地应对变更 |

|  |  |
| --- | --- |
| 难点 | 在数据处理时，要处理数据的不同属性（如股票对象的开盘价、收盘价等），但处理逻辑基本相同 |
| 架构决策 | 代码复用 |
| 假设 | 数据的属性较多，要处理9种不同的属性 |
| 可替代方法 | 选项1：对每种数据属性采用专有的处理方式  选项2：实现compare接口，对不同属性抽象处理，抽出共同使用的逻辑部分 |
| 选取的选项 | 选项2 |
| 理由 | 选项2虽然可能增加控制耦合，但提高了代码复用性 |