STOCK STATISTIC SYSTEM

**详细设计文档**

Duck Plus

2015-3-22

# 文档更新纪录：

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 作者 | 描述 | 日期 |
| v1.0 | 段正谋 |  | 2016年03月09日 |
| V1.1 | 侍硕 | 完善包图、类图、顺序图、BL、DS接口定义 | 2016年3月13日 |

**目录**（按Ctrl+目录有链接）

[文档更新纪录： 2](#_Toc445316314)

[引言： 4](#_Toc445316315)

[词汇表： 5](#_Toc445316316)

[1.产品概述 5](#_Toc445316317)

[2.系统的分层架构 6](#_Toc445316318)

[3.用户界面层的分解 7](#_Toc445316319)

[4.系统的架构设计 8](#_Toc445316320)

[5.系统的类图设计 9](#_Toc445316321)

[6.系统顺序图 10](#_Toc445316322)

[7.数据层的分解 12](#_Toc445316323)

[8.接口规范设计 14](#_Toc445316324)

# 引言：

|  |  |
| --- | --- |
| 内容和说明 | 项目 |
| 1.编写的目的（说明整个文档所有达到的目标） | 本文档提供股票数据分析系统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。 |
| 2.对象与范围（说明整个文档的内容范围和针对的读者对象） | 本文档的作者是Duck Plus团队内部的开发和管理人员，参考了rup的《软件架构文档模版》，用于指导下一循环的代码开发和测试工作。 |
| 3.参考文献（说明文档中主要的所需引用） | 1.《软件需求规格说明书》；  《软件架构文档模版》，Rational Software Corporation，2002；2.The Object Management Group (OMG), The Unified Modeling Language Specification v1.4, 2003  3.丁二玉，刘钦.计算与软件工程（卷二）[M]机械工业出版社，2012：134—182  4.Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Micheael Stal.Pattern-Oriented Software Architecture Volume 1: A system of Patterns [M]机械工业出版社,2003  5.Christopher　Fox.软件工程设计导论：过程、原理与模式（UML2.0版）[M]清华大学出版社，2007 |
| 4.名词与术语(说明文档中常用的技术缩略和先相关词条) |  |
|  |  |

# 词汇表：

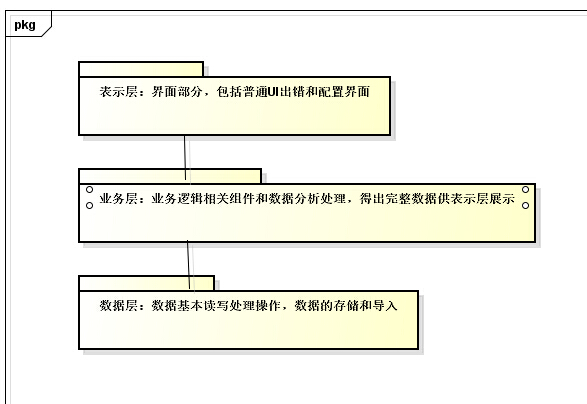
|  |  |  |
| --- | --- | --- |
| 词汇名称 | 词汇含义 | 备注 |
| \_ui | 表示某展示层 |  |
| \_bl | 表示某逻辑层 |  |
| \_data | 表示某数据层 |  |
|  |  |  |

# 1.产品概述

Stock Statistic System 是用于统计股票的一些包括开盘价、收盘价、最高价、最低价、成交量、换手率等数据的展示系统，主要由Java语言开发，从API中读取数据，解析数据，并根据合理的逻辑展示在用户界面上，为用户提供一个多方面访问数据的窗口。

# 2.系统的分层架构

图示如下：



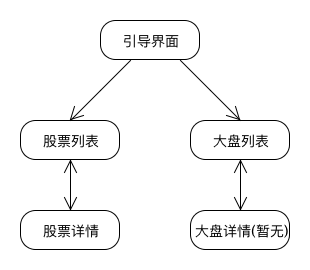
系统划分为一下3个逻辑层次：  
1）表示层：用于前台界面展示和配置的层次  
2）业务层：包含业务控制和逻辑的层次  
3）数据层：定义和储存系统中相关数据的层次  
（迭代二后可能会加入网络通信中间层，根据需求变更作出相应处理）

系统可以部署在以下3个物理层次。  
1）访问层：用于用户访问系统的层次。  
2）业务层：部署业务控制和逻辑的层次。  
3）数据层：部署和储存系统中相关数据的层次。

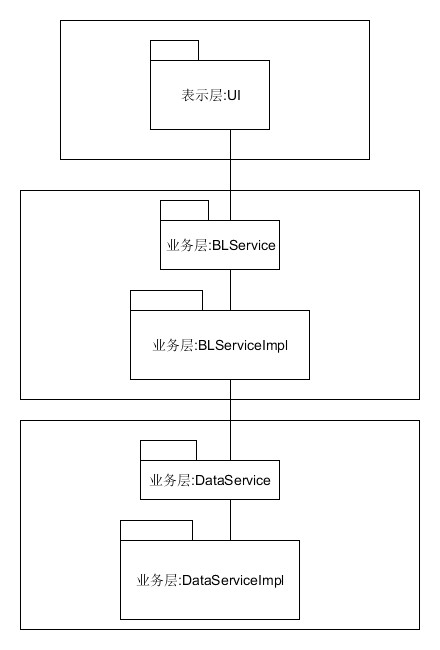
StockStatisticSystem在迭代一中只包含本地客户端

# 3.用户界面层的分解

该系统在迭代一共存在主要界面5个，分别为 引导界面、股票列表界面、股票详情界面、大盘列表界面、大盘详情界面其跳转设计如下图：

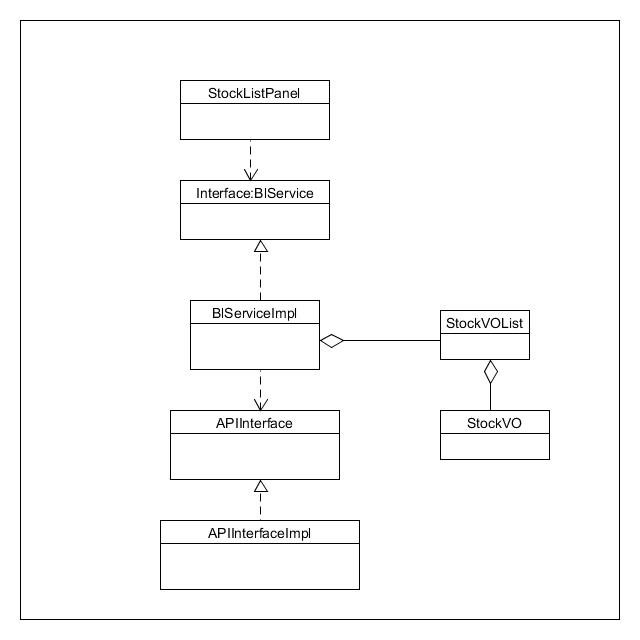


# 4.系统的架构设计

系统架构中的对象分为6类：  
1）UI对象，负责处理系统数据的展现和用户的交互。  
3）Service对象，负责提供服务的抽象接口，获取从数据端组装好的数据。  
4）ServiceImpl对象，负责对于抽象接口的实现模块。  
5）VO对象，负责封装从本地文件获取的数据并与逻辑、展示层交互。  
6）PO对象，负责与本地文件交互，获取数据。  
  
系统中的组件和组件接口：  
  


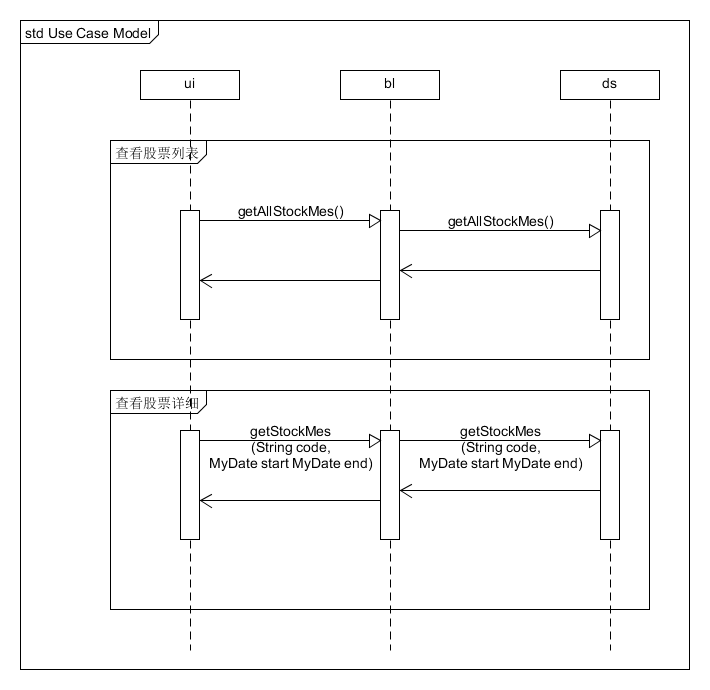
# 5.系统的类图设计

如下：



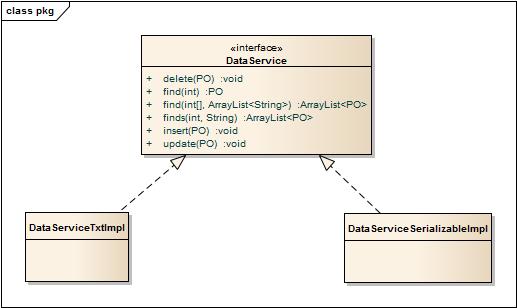
# 6.系统顺序图

设计如下：



# 7.数据层的分解

数据层主要给业务逻辑层提供数据服务，包括对于持久化数据的增、删、改、查。业务逻辑需要的服务DataService接口提供。数据层模块的具体描述如图



迭代一采用txt文件进行数据存储和交互。

接口组件设计结构如下表格：

|  |  |  |  |
| --- | --- | --- | --- |
| 接口ID | 连接组件 | 接口信息 | |
| BlService | 连接UI与BL | 语法 | Return(Response) Interface(Request) |
| 前置条件 | 用户的输入正确 |
| 后置条件 | 处理控制组件处理请求并且响应 |
| 不变量 | 用户请求信息 |
| APIInterface | 连接BL与DS | 语法 | Return(Response) Interface(Request) |
| 前置条件 | 无 |
| 后置条件 | 对应的DataService执行对应的业务逻辑 |
| 不变量 | 无 |

# 8.接口规范设计

表StockBLService的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| StockBLService. getDayOHLC\_Data(String stockCode, MyDate start,  MyDate end ) | 语法 | public List<OHLC\_VO> getDayOHLC\_Data(String stockCode, MyDate start,  MyDate end ); |
| 前置条件 | 已创建一个StockBLImpl对象 |
| 后置条件 | 调用StockBLImpl领域对象的getDayOHLC\_Data方法 |
| StockBLService. getSharingVOs(String stockCode) | 语法 | public List<TimeSharingVO> getSharingVOs(String stockCode); |
| 前置条件 | 已创建一个StockBLImpl对象 |
| 后置条件 | 调用StockBLImpl领域对象的getSharingVOs方法 |
| StockBLService. getDayDealVOs(String stockCode, MyDate start,MyDate end) | 语法 | public List<DealVO> getDayDealVOs(String stockCode, MyDate start,  MyDate end); |
| 前置条件 | 已创建一个StockBLImpl对象 |
| 后置条件 | 调用StockBLImpl领域对象的getDayDealVOs方法 |
| StockBLService. getAllStocks() | 语法 | public Iterator<StockVO> getAllStocks() |
| 前置条件 | 已创建一个StockBLImpl对象 |
| 后置条件 | 调用StockBLImpl领域对象的getAllStocks()方法 |
| StockBLService. getStocksByTime(String stockCode, MyDate start,MyDate end); | 语法 | public Iterator<StockVO> getStocksByTime(String stockCode, MyDate start,MyDate end); |
| 前置条件 | 已创建一个StockBLImpl对象 |
| 后置条件 | 调用StockBLImpl领域对象的getStocksByTime方法 |
| StockBLService. getStocksByStockCode(String code); | 语法 | Public Iterator<StockVO> getStocksByStockCode(String code); |
| 前置条件 | 已创建一个StockBLImpl对象 |
| 后置条件 | 调用StockBLImpl领域对象的getStocksByStockCode方法 |
| 需要的服务 | | |
| 服务名 | 服务 | |
| StockBLImpl. getDayOHLC\_Data(String stockCode, MyDate start,  MyDate end ) | 传输OHLC\_VO列表 | |
| StockBLImpl. getSharingVOs(String stockCode) | 传输SharingVO列表 | |
| StockBLImpl. getDayDealVOs(String stockCode, MyDate start,MyDate end) | 传输DealVO列表 | |
| StockBLImpl. getAllStocks() | 传输StockVO列表 | |
| StockBLImpl. getStocksByTime(String stockCode, MyDate start,MyDate end); | 传输StockVO列表 | |
| StockBLImpl. getStocksByTime(String stockCode, MyDate start,MyDate end); | 传输StockVO列表 | |
| StockBLService. getStocksByStockCode(String code); |  | |

表BenchMarkBLService的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| BenchMarkBLService. getAllBenchMarks(); | 语法 | public Iterator<BenchMarkVO> getAllBenchMarks(); |
| 前置条件 | 已创建一个BenchMarkBLImpl领域对象 |
| 后置条件 | 调用BenchMarkBLImpl领域对象的getAllBenchMarks();方法 |
| BenchMarkBLService. getRecentBenchMarks(String BenchMarkCode); | 语法 | public Iterator<BenchMarkVO> getRecentBenchMarks(String BenchMarkCode); |
| 前置条件 | 已创建一个BenchMarkBLImpl领域对象 |
| 后置条件 | 调用BenchMarkBLImpl领域对象的getRecentBenchMarks(String BenchMarkCode);方法 |
| BenchMarkBLService. getBenchMarkByTime(String BenchMarkCode, MyDate start,  MyDate end); | 语法 | public Iterator<BenchMarkVO> getBenchMarkByTime(String BenchMarkCode, MyDate start,  MyDate end); |
| 前置条件 | 已创建一个BenchMarkBLImpl领域对象 |
| 后置条件 | 调用BenchMarkBLImpl领域对象的getBenchMarkByTime(String BenchMarkCode, MyDate start,  MyDate end);方法 |
| 需要的服务 | | |
| 服务名 | 服务 | |
| BenchMarkBLImpl. getAllBenchMarks(); | 传输所有大盘代码列表 | |
| BenchMarkBLImpl. getRecentBenchMarks(String BenchMarkCode); | 传输指定大盘近一个月的数据列表 | |
| BenchMarkBLImpl. getBenchMarkByTime(String BenchMarkCode, MyDate start,  MyDate end); | 传输指定大盘指定时间段的数据列表 | |

APIInterface的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| APIInterface. getAllStocks() | 语法 | public List<String> getAllStocks();  public List<String> getAllStocks(int year);  public List<String> getAllStocks(Exchange exchange);  public List<String> getAllStocks(int year, Exchange exchange); |
| 前置条件 | 已创建一个APIInterfaceImpl对象 |
| 后置条件 | 调用APIInterfaceImpl领域对象的getAllStocks()方法 |
| APIInterface.getStockMes(String stockCode); | 语法 | public StockPO getStockMes(String stockCode);  public List<StockPO> getStockMes(String stockCode, MyDate start, MyDate end); |
| 前置条件 | 已创建一个APIInterfaceImpl对象 |
| 后置条件 | 调用APIInterfaceImpl领域对象的getStockMes(String stockCode);方法 |
| APIInterface.getAllStockMes(); | 语法 | public List<StockPO> getAllStockMes(); |
| 前置条件 | 已创建一个StockBLImpl对象 |
| 后置条件 | 调用StockBLImpl领域对象的getAllStockMes();方法 |
| APIInterface. getAllBenchMarks(); | 语法 | public List<String> getAllBenchMarks(); |
| 前置条件 | 已创建一个StockBLImpl对象 |
| 后置条件 | 调用StockBLImpl领域对象的getAllBenchMarks();方法 |
| StockBLService. getBenchMes(String benchCode); | 语法 | public BenchMarkPO getBenchMes(String benchCode);  public List<BenchMarkPO> getBenchMes(String benchCode, MyDate start,  MyDate end); |
| 前置条件 | 已创建一个StockBLImpl对象 |
| 后置条件 | 调用StockBLImpl领域对象的getBenchMes(String benchCode);  方法 |
| APIInterface. getAllBenchMes(); | 语法 | public List<BenchMarkPO> getAllBenchMes(); |
| 前置条件 | 已创建一个StockBLImpl对象 |
| 后置条件 | 调用StockBLImpl领域对象的getAllBenchMes();方法 |
| 需要的服务 | | |
| 服务名 | 服务 | |
| APIInterfaceImpl. getAllStocks() | 传输股票代码列表 | |
| APIInterfaceImpl. APIInterface.getStockMes(String stockCode); | 传输StockVO列表 | |
| APIInterfaceImpl. getAllStockMes(); | 传输StockVO列表 | |
| APIInterfaceImpl. getAllBenchMarks(); | 传输大盘代码列表 | |
| APIInterfaceImpl. getBenchMes(String benchCode); | 传输BenchMarkVO列表 | |
| APIInterfaceImpl. getBenchMes(String benchCode, MyDate start,  MyDate end); | 传输BenchMarkVO列表 | |
| APIInterfaceImpl. getAllBenchMes(); | 传输BenchMarkVO列表 | |