**软件架构设计文档**

# 团队名称： Neo

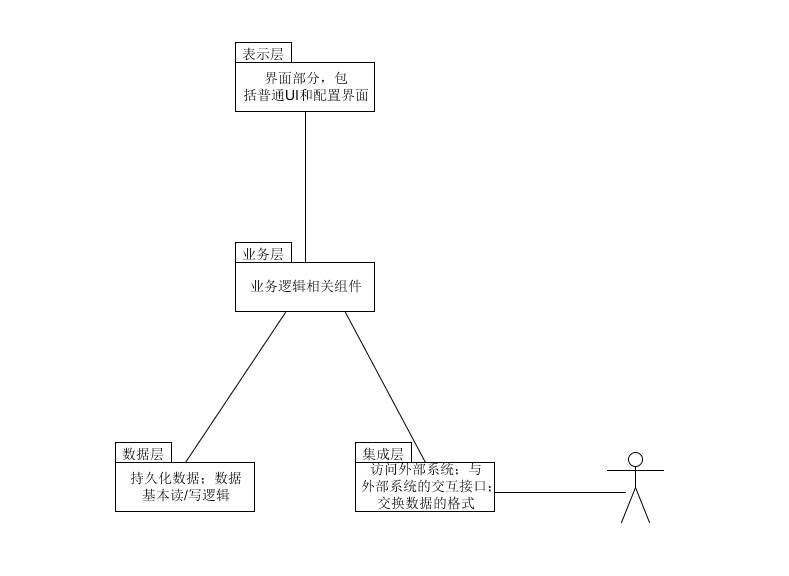
# 文档更新记录：

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 作者 | 版本描述（应该说明做了哪些修改及修改的标示等） | 日期 |
| V1.0 | 罗铉斌 | 基本架构 | 2016/3/4 |
| V1.1 | 吕德超 | 逻辑层分解和分层细化 | 2016/3/4 |
| V1.2 | 罗铉斌 | 数据层完善 | 2016/3/6 |

# 引言：

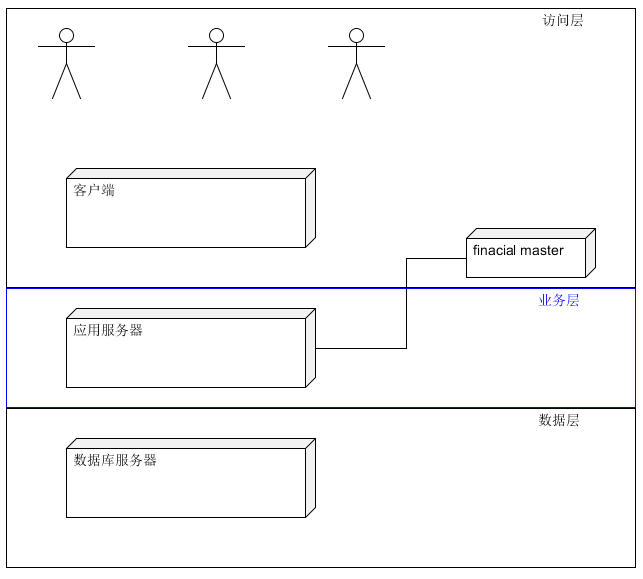
|  |  |
| --- | --- |
| 内容说明 |  |
| 1 编写目的  （说明整个文档所有达到的目标） | 本文档提供师生交流系统的软件架构概览，采用若干架构试图描述系统的不同方面，以便表示构造系统所需要的重要架构决策 |
| 2 对象与范围  （说明整个文档的内容范围和针对的读者对象） | 本文档的读者是Neo团队内部的开发和管理人员，参考了RUP的《软件架构文档模板》，用于指导下一循环的代码开法和测试工作 |
| 3 参考文献  （说明文档中主要的所需所用） | 《软件需求规格说明书》，Neo  《软件架构文档模板》，Rational Software Corporation |
| 4 名词与术语  （说明文档汇总常用的技术缩略和相关词条） |  |

# 系统的分层架构：



系统划分为以下4个逻辑层次。

1. 展示层：用于前台界面展示和配置的层次。
2. 逻辑层：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. 集成层：定义和集成与外部系统交互接口的层次。



# 体系结构设计

## 系统的部署在以下3个物理层次

访问层：用于用户访问系统的层次。

业务层：部署业务控制和逻辑的层次

数据层：部署和存储系统中相关数据的层次

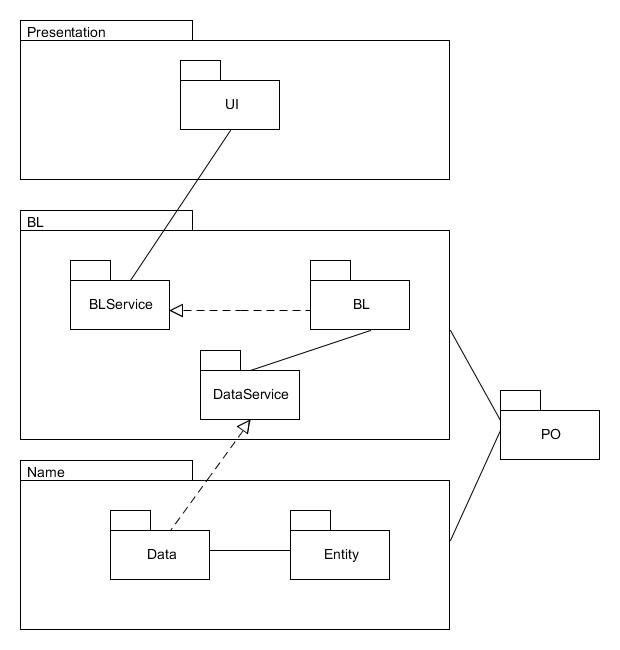
## 系统的架构设计如下。

系统架构中的对象分为7类

1. UI对象，负责处理系统数据的展示和用户的交互
2. BLService对象，负责向界面层提供抽象接口
3. BL对象，负责对BLService抽象接口的实现
4. DataService对象，负责向逻辑层提供抽象接口
5. Data对象，负责对DataService抽象接口的实现
6. Entity对象，该对象将用来从数据库中获取的数据封装成数据实体

7）PO对象，用于将数据对象打包成PO进行数据传递

## 系统中的组件和组件接口：



## 业务逻辑层的分解

##### 4.1业务逻辑层模块的职责

**业务逻辑层模块的职责**

|  |  |
| --- | --- |
| **模块** | **职责** |
| StockListBL | 负责股票列表界面所需要的服务 |
| StockMessageBL | 负责查看某支股票详细信息所需要的服务 |
| StockMarketBL | 负责实现中转接收界面所需要的服务 |

##### 4.2业务逻辑层模块的接口规范

###### 4.2.1股票查询模块

4.2.1.1 StockListBL模块

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StockListBL.getStockList | 语法 | public String[][] getStockList() |
| 前置条件 | 启动一个股票查询回合 |
| 后置条件 | 返回预选股票的列表，用二维数组展示 |
| StockListBL.updateStockList | 语法 | public String[][] updateStockList(String key) |
| 前置条件 | 用户在搜索框输入用来筛选的关键字 |
| 后置条件 | 返回筛选过后的列表 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| StockListData getStockList() | 从数据层得到股票的列表 | |

4.2.1.1 StockMessageBL模块

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StockMessagetBL.getStockMessage | 语法 | public StockVO getStockMessage(String id) |
| 前置条件 | 从股票列表中选择某支股票 |
| 后置条件 | 返回所选股票的详细信息，打包成StockPO |
| StockMessageBL.updateStockMessage | 语法 | public StockVO updateStockList(String id,String startData,String overData) |
| 前置条件 | 用户选择正确的时间段 |
| 后置条件 | 根据用户输入的时间段返回筛选过后的列表 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| StockMessageData getStockmessage(String id) | 从数据层得到指定id股票的详细信息（默认为最近三十天的历史数据） | |
| StockMessageData getStockmessage(String id,String startData,String overData) | 从数据层得到指定id，指定时间段股票的详细信息 | |

###### 4.2.2大盘查询模块

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StockMarketBL.getStockMarket | 语法 | public StockMarketVO getStockMarket(String key,Sting data) |
| 前置条件 | 启动一个股票查询回合 |
| 后置条件 | 根据关键字和时间返回当前最新的大盘信息 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| StockMessageData getStockMarket(String key,Sting data) | 得到指定证券指数指定时间的大盘信息 | |

## 数据层的分解

账户管理的接口规范

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| StockDataService.getCodeNames | 语法 | public codeNamePO getCodeName(int year,String exchange) |
| 前置条件 | API中有返回所有股票代号的接口 |
| 后置条件 | 返回一个codeNamePO |
| StockDataService.getStatisitcOfStock | 语法 | public stockStatisticPO getStatisitcOfStock(String codeName,String start,String end) |
| 前置条件 | API中有查询所有股票数据的接口 |
| 后置条件 | 返回一个相应的stockStatisticPO |
| StockDataService.getBenchmark | 语法 | public benchCodePO getBenchmark() |
| 前置条件 | API中有返回所有股票大盘指数的接口 |
| 后置条件 | 返回一个相应的benchCodePO |
| StockDataService.getStatisticOfBenchmark | 语法 | public benchmarkStatisticPO getStatisticOfBenchmark (String benchCode , String start,String end) |
| 前置条件 | API中有返回指定股票大盘数据的接口 |
| 后置条件 | 返回一个相应的benchmarkStatisticPO |
| StockDataService. getStatisticOfField | 语法 | public fieldStatisticPO getStatisticOfField () |
| 前置条件 | API中有返回可用的交易数据字段的接口 |
| 后置条件 | 返回相应fieldStatisticPO |