|  |
| --- |
| BUFF TREASURE 股票数据分析系统 |
| **软件设计文档** |
| 版本号：2.1.3 |

|  |
| --- |
| 王帅惟，朱俊毅，万年杰，田原  2017-5-12 |

# 引言

## 1.1 目的

本文档简要描述了BUFF TREASURE股票分析软件的架构设计，主要描述逻辑层与数据层接口规范以及关键部位的实现设计。

## 1.2 参考资料

《BUFF TREASURE需求规格说明文档》

# 2. 中层设计

## 2.1 模块划分与职责

本软件采用分层模式，分为界面层、逻辑层与数据层。本文档仅记录逻辑层与数据层的设计情况。

### 2.1.1 逻辑层模块划分

|  |  |
| --- | --- |
| 模块名 | 职责 |
| forum | 提供论坛相关的接口 |
| functionMap | 提供策略回测时参数映射以及筛选与排序的接口 |
| industryandbench | 提供行业与板块相关接口 |
| market | 提供大盘信息的获取功能 |
| realtime | 提供实时数据获取的接口 |
| statistics | 提供个股分析与统计的接口 |
| strategy | 提供策略回测的接口 |
| thermometer | 提供市场温度计的接口 |
| tool | 提供一些常用的统计函数的实现接口 |
| singlestock | 提供个股相关的所有接口 |
| allStock | 提供获取所有股票相关信息的接口 |

### 2.1.2 数据层模块划分

|  |  |
| --- | --- |
| 模块名 | 职责 |
| allStock | 提供股票代码与名称的映射以及个股所属板块与行业的功能 |
| content | 提供股票评论的持久化 |
| forum | 提供论坛点赞交互的持久化 |
| hotStockAndBoard | 提供实时热门股票和热门板块的持久化 |
| marketIndex | 提供大盘指数的持久化 |
| RTStock | 提供实时推荐股票的持久化 |
| singleStock | 提供个股信息的持久化 |
| singleStockRT | 个股实时信息的持久化 |
| user | 用户信息的持久化 |
| thermoter | 市场温度的信息的持久化 |
| strategy | 所有用户的策略栈持久化 |

## 2.2 逻辑层模块接口说明

### 2.2.1 singleStock模块

|  |  |  |
| --- | --- | --- |
| 提供的接口 | | |
| singleStockbl.getDailyData(code, callback) | 语法 | exports.getWeeklyData = (code, callback) => {} |
| 前置条件 | callback 形如 (err, docs) => { } |
| 后置条件 | 无 |
| singleStockbl.getWeeklyData(code, callback) | 语法 | exports.getWeeklyData = (code, callback) => {} |
| 前置条件 | callback 形如 (err, docs) => { } |
| 后置条件 | 无 |
| singleStockbl.getMonthlyData(code, callback) | 语法 | exports.getMonthlyData = (code, callback) => {} |
| 前置条件 | callback 形如 (err, docs) => { } |
| 后置条件 | 无 |
| singleStockbl.getRTInfo(code,callback) | 语法 | exports.getRTInfo = (code, callback) |
| 前置条件 | callback 形如 (err, stockRTInfo) => { } |
| 后置条件 | 无 |
| singleStockbl.getLatestStockRTInfo(code, callback) | 语法 | exports.getMonthlyData = (code, callback) => {} |
| 前置条件 | callback 形如 (err, docs) => { } |
| 后置条件 | 无 |
| singleStockbl.getCompanyInfo(code, callback) | 语法 | exports.getCompanyInfo = (code, callback) => {} |
| 前置条件 | callback 形如 (err, companyInfo) => { } |
| 后置条件 | 无 |
| singleStockbl.getHotStocks(callback) | 语法 | exports.getHotStocks = (callback) => {} |
| 前置条件 | callback 形如 (err,hotStocks) => { } |
| 后置条件 | 无 |
| singleStockbl.updateHotStocks(callback) | 语法 | exports.updateHotStocks = (callback) => {} |
| 前置条件 | callback 形如 (err) => { } |
| 后置条件 | 无 |

### 2.2.2 AllStock模块

|  |  |  |
| --- | --- | --- |
| 提供的接口 | | |
| allStockbl.getAllStockCodeAndName（callback） | 语法 | exports.getAllStockCodeAndName = (callback) => {} |
| 前置条件 | callback 形如 (err, allstocks) => { } |
| 后置条件 | 无 |

### 2.2.3 user模块

|  |  |  |
| --- | --- | --- |
| 提供的接口 | | |
| userbl.login(username,password, callback) | 语法 | exports.login = (username, password, callback) => {} |
| 前置条件 | 无 |
| 后置条件 | 无 |
| userbl.signUp(username,password, email, callback) | 语法 | exports.signUp = (username, password, email, callback) => {} |
| 前置条件 | 无 |
| 后置条件 | 新增一条用户记录 |
| userbl.addToSelfSelectStock (userName, stock, callback) | 语法 | exports.addToSelfSelectStock = (userName, stock, callback) => {} |
| 前置条件 | callback (err, doc) => {} |
| 后置条件 | 在对应用户的自选股中增加一条记录 |
| userbl.getSelfSelectStock(userName, callback) | 语法 | exports.getSelfSelectStock = (userName, callback) => {} |
| 前置条件 | callback 形如(err, docs) => {} docs是数组格式 |
| 后置条件 | 无 |
| userbl.saveStrategy(userName,strategy, callback) | 语法 | exports.saveStrategy = (userName, strategy, callback) => {} |
| 前置条件 | 无 |
| 后置条件 | 无 |
| userbl.loadStrategy(userName,strategyName, callback) | 语法 | exports.loadStrategy = (userName, strategyName, callback) => {} |
| 前置条件 | 无 |
| 后置条件 | 无 |
| userbl.getUnreadMessageCount(userName, callback) | 语法 | exports.getUnreadMessageCount = (userName, callback) => {} |
| 前置条件 | 无 |
| 后置条件 | 无 |
| userbl.getAllMessages = (userName, callback) | 语法 | exports.getAllMessages = (userName, callback) => {} |
| 前置条件 | 无 |
| 后置条件 | 无 |
| userbl.markAsRead(userName,time, callback) | 语法 | exports.markAsRead = (userName, time, callback) => {} |
| 前置条件 | 无 |
| 后置条件 | 无 |
| userbl. getOneMessageContent = (userName, time, callback) | 语法 | exports.getOneMessageContent = (userName, time, callback) => {} |
| 前置条件 | 无 |
| 后置条件 | 无 |

### 2.2.4 tool模块

|  |  |  |
| --- | --- | --- |
| 提供的接口 | | |
| statisticsTool.getCOV = (a, b) | 语法 | exports.getCOV = (a, b) => {} |
| 前置条件 | 无 |
| 后置条件 | 无 |
| statisticsTool.getAverage(a) | 语法 | exports.getAverage = (a) => {} |
| 前置条件 | 无 |
| 后置条件 | 无 |
| statisticsTool.getVariance(a) | 语法 | exports.getVariance = (a)=> {} |
| 前置条件 | 无 |
| 后置条件 | 无 |
| statisticsTool.getSTD(a) | 语法 | exports.getSTD = (a) => {} |
| 前置条件 | 无 |
| 后置条件 | 无 |
| statisticsTool.getCoefficientOfVariation(a) | 语法 | exports.getCoefficientOfVariation = (a) => {} |
| 前置条件 | 无 |
| 后置条件 | 无 |

### 2.2.5 Industryandbench模块

|  |  |  |
| --- | --- | --- |
| 提供的接口 | | |
| industrybl.getIndustryByCode(code, callback) | 语法 | exports.getIndustryByCode = (code, callback) => {} |
| 前置条件 | 无 |
| 后置条件 | 无 |
| industrybl.getBoardsByCode(code, callback) | 语法 | exports.getBoardsByCode = (code, callback) => {} |
| 前置条件 | 无 |
| 后置条件 | 无 |
| industrybl.getAllIndustries(callback) | 语法 | exports.getAllIndustries = (callback) => {} |
| 前置条件 | 无 |
| 后置条件 | 无 |
| industrybl.getAllBoards(callback) | 语法 | exports.getAllBoards = (callback) => {} |
| 前置条件 | 无 |
| 后置条件 | 无 |

### 2.2.6 market模块

|  |  |  |
| --- | --- | --- |
| 提供的接口 | | |
| marketbl.getMarketRTInfo(code, callback) | 语法 | exports.getMarketRTInfo = (code, callback) => {} |
| 前置条件 | 无 |
| 后置条件 | 无 |
| marketbl.getDailyMarketIndex(code, callback) | 语法 | exports.getDailyMarketIndex = (code, callback) => {} |
| 前置条件 | 无 |
| 后置条件 | 无 |
| marketbl.getWeeklyMarketIndex(code, callback) | 语法 | exports.getWeeklyMarketIndex = (code, callback) => {} |
| 前置条件 | 无 |
| 后置条件 | 无 |
| marketbl.getMonthlyMarketIndex(code, callback) | 语法 | exports.getMonthlyMarketIndex = (code, callback) => {} |
| 前置条件 | 无 |
| 后置条件 | 无 |

### 2.2.7 realtime模块

|  |  |  |
| --- | --- | --- |
| 提供的接口 | | |
| marketRT.obtainMarketRTInfoByCode(code, callback) | 语法 | exports.obtainMarketRTInfoByCode = (code, callback) => {} |
| 前置条件 | 无 |
| 后置条件 | 无 |
| singleStockRT.obtainRTInfo(code, callback) | 语法 | exports.obtainRTInfo(code, callback) => {} |
| 前置条件 | 无 |
| 后置条件 | 无 |
| singleStockRT.updateAllStockRTInfo(callback) | 语法 | exports.updateAllStockRTInfo = (callback) => {} |
| 前置条件 | 无 |
| 后置条件 | 无 |
| singleStockRT.getHotBoard(callback) | 语法 | exports.getHotBoard = (callback) => {} |
| 前置条件 | 无 |
| 后置条件 | 无 |
| singleStockRT.updateHotBoard(callback) | 语法 | exports.updateHotBoard = (callback) => {} |
| 前置条件 | 无 |
| 后置条件 | 无 |

### 2.2.8 statistics模块

|  |  |  |
| --- | --- | --- |
| 提供的接口 | | |
| industryCorrelationbl.getIndustryCorrelationResult(code,holdingPeriod, callback) | 语法 | exports.getIndustryCorrelationResult = (code, holdingPeriod, callback) => {} |
| 前置条件 | 无 |
| 后置条件 | 无 |
| singleStockPredict.SVMAnalyze(userName, code, stockName, open\_price, holdingDays, time, callback) | 语法 | exports.SVMAnalyze = (userName, code, stockName, open\_price, holdingDays, time, callback) => {} |
| 前置条件 | 无 |
| 后置条件 | 无 |
| singleStockRT.updateAllStockRTInfo(callback) | 语法 | exports.updateAllStockRTInfo = (callback) => {} |
| 前置条件 | 无 |
| 后置条件 | 无 |
| singleStockRT.RFCAnalyze(userName, code, stockName, holdingDays, time, callback) | 语法 | exports.RFCAnalyze = (userName, code, stockName, holdingDays, time, callback) => {} |
| 前置条件 | 无 |
| 后置条件 | 无 |
| singleStockRT.CNNAnalyze(userName, code, stockName, holdingDays, isMarket=false, iterationNum=4, learningWay='ALL', time, callback) | 语法 | exports.CNNAnalyze = (userName, code, stockName, holdingDays, isMarket=false, iterationNum=4, learningWay='ALL', time, callback) => {} |
| 前置条件 | 无 |
| 后置条件 | 无 |

### 2.2.8 statistics模块

|  |  |  |
| --- | --- | --- |
| 提供的接口 | | |
| strategybl.getBackResults (beginDate, endDate, stockPoolConditionVO, rank, filter, tradeModelVO, envSpecDay, setProcess, callback) | 语法 | exports.getBackResults = function (beginDate, endDate, stockPoolConditionVO, rank, filter, tradeModelVO, envSpecDay, setProcess, callback) |
| 前置条件 | 无 |
| 后置条件 | 无 |

## 2.3 数据层模块接口说明

### 2.3.1 allStock模块

1. exports.allStockDB = {
2. /\*\*
3. \* 获得所有股票的代码和名称
4. \* @param callback
5. \*/
6. getAllStockCodeAndName: (callback) => {},
8. /\*\*
9. \* 根据股票代码获得股票名称
10. \* @param code
11. \* @param callback
12. \*/
13. getNameByCode: (code, callback) => { },
15. /\*\*
16. \* 获得沪深300所有股票代码和名称
17. \* @param callback
18. \*/
19. getHS300StockCodeAndName: (callback) => {  },
21. /\*\*
22. \* 获得同行业的所有股票的代码和名称
23. \* @param industry
24. \* @param callback
25. \*/
26. getStocksByIndustry: (industry, callback) => {   },
28. /\*\*
29. \* 获得属于某些板块的所有股票的代码和名称
30. \* @param bench 板块的列表，如果只有一个板块，则是只包含一个元素的列表
31. \* @param callback
32. \*/
33. getStocksByBench: (bench, callback) => {},
35. /\*\*
36. \* 根据所给的股票代号获得这支股票所在的行业
37. \* ！！！一支股票只有一个行业！！！！
38. \* @param code
39. \* @param callback
40. \*/
41. getIndustryByCode: (code, callback) => { },
43. /\*\*
44. \* 根据所给的股票代号获得这支股票所在的板块
45. \* ！！！一支股票可能从属于多个板块，所以所属板块是一个数组！！！！
46. \* @param code
47. \* @param callback
48. \*/
49. getBoardsByCode: (code, callback) => {},
51. /\*\*
52. \* 获得所有行业
53. \* @param callback
54. \*/
55. getAllIndustry: (callback) => { },
57. /\*\*
58. \* 获得所有板块
59. \* @param callback
60. \*/
61. getAllBoards: (callback) => { },
63. /\*\*
64. \* 获得随机(目前就是前500只股票的代码和名称
65. \* @param callback
66. \*/
67. getRandom500StockCodeAndName: (callback) => { },
69. /\*\*
70. \* 获得沪深A股所有股票代码和名称
71. \* @param callback
72. \*/
73. getHSAStockCodeAndName: (callback) => {   },
75. /\*\*
76. \* 获得中小板所有股票代码和名称
77. \* @param callback
78. \*/
79. getSMEBoardCodeAndName: (callback) => {  },
81. /\*\*
82. \* 获得创业板所有股票代码和名称
83. \* @param callback
84. \*/
85. getGEMBoardStockCodeAndName: (callback) => { },
87. /\*\*
88. \* 获得自定义股票池的股票代码和名称
89. \* @param industries {Array}
90. \* @param boards {Array}
91. \* @param callback
92. \*/
93. getDIYStockPool: (industries, boards, callback) => { }
94. };

2.3.2 user模块

1. exports.userDB = {
2. getPasswordByName: (name, callback) => { },
4. registerUser: (user, callback) => {   },
6. /\*\*
7. \* 添加某只股票进某个用户的自选股列表
8. \* @param userName
9. \* @param stock
10. \* @param callback
11. \*/
12. addToSelfSelectStock: (userName, stock, callback) => {  },
14. /\*\*
15. \* 获得某个用户的所有自选股
16. \* @param userName
17. \* @param callback
18. \*/
19. getSelfSelectStock: (userName, callback) => { },
21. /\*\*
22. \* 保存策略
23. \* @param userName {String} 用户名
24. \* @param strategy 策略的具体参数
25. \* @param callback
26. \*/
27. saveStrategy: (userName, strategy, callback) => { },
29. /\*\*
30. \* 覆盖原来的重名策略
31. \* @param userName
32. \* @param strategy
33. \* @param callback
34. \*/
35. overrideStrategy: (userName, strategy, callback) => { },
37. /\*\*
38. \* 获得某个用户所有已保存的策略
39. \* @param userName
40. \* @param callback
41. \*/
42. getAllStrategy: (userName, callback) => { },
44. /\*\*
45. \* 获得用户所有的消息队列
46. \* @param userName
47. \* @param callback
48. \*/
49. getAllMessage: (userName, callback) => {  },
51. /\*\*
52. \* 添加未读消息
53. \* @param userName
54. \* @param message
55. \* @param callback
56. \*/
57. addUnreadMessage: (userName, message, callback) => {  },
59. /\*\*
60. \* 为了实现将未读消息标记为已读的功能，需要重写原来的整个消息队列
61. \* @param userName
62. \* @param message
63. \* @param callback
64. \*/
65. overrideMessage: (userName, message, callback) => { }
66. };

2.3.3 singleStock模块

1. exports.singleStockDB = {
2. /\*\*
3. \* 获得具体某一支股票的每一天的数据（按日期的升序排列）
4. \* @param code
5. \* @param callback
6. \*/
7. getStockInfoByCode: **function** (code, callback) {   },
9. /\*\*
10. \* 获得具体某一天的所有的股票的数据
11. \* @param date 日期必须大于1990年12月19号，小于昨天的日期
12. \* @param callback
13. \*/
14. getStockInfoByDate: **function** (date, callback) { },
16. /\*\*
17. \* 获得日期区间范围内的  股票信息列表（按日期的升序排列）
18. \* 包含两端
19. \* @param code
20. \* @param beginDate
21. \* @param endDate
22. \* @param p {Array} 投影参数
23. \* @param callback
24. \*/
25. getStockInfoInRangeDate: **function** (code, beginDate, endDate, p, callback) {  },
27. /\*\*
28. \* 获得某一支股票某一天的数据
29. \* @param code
30. \* @param date
31. \* @param callback
32. \*/
33. getStockInfoByCodeAndDate: **function** (code, date, callback) { },
35. /\*\*
36. \* 这个方法
37. \* 不建议调用
38. \* TODO  最后删掉
39. \* @deprecated
40. \* @param date s
41. \* @param callback
42. \*/
43. getStockInfoByDateTwo: **function** (date, callback) {  }
44. };

2.3.3 strategy模块

1. exports.strategyDB ={
2. saveStrategy :(strategy,callback)=>{  },
4. removeStrategy: (strategyKey, callback) => {   },
6. getRandomStrategy: (callback) => {  },
8. getAllStrategy: (callback) => { },
10. /\*\*
11. \*
12. \* @param env  是哪种市场环境 传递原来定义的标准字符串
13. \* @param cap   选择是那种能力
14. \* @param num   选择最好的几个策略
15. \* @param callback
16. \*/
17. getBestStrategyByEnv: (env, cap , num ,callback) => {},
19. /\*\*
20. \* 形如err,doc
21. \*/
22. getStrategyByID:(strategyKey,callback)=>{  }
24. };

2.3.3 stockRTInfo模块

1. exports.stockRTInfoDB = {
3. /\*\*
4. \* 更新股票实时信息
5. \* @param code {String} 股票代码
6. \* @param rtInfo 股票实时信息，形如：
7. \* {
8. code: String,                         // 股票代码
9. now\_price: Number,                    // 现价
10. change\_price: Number,                 // 涨跌额
11. change\_rate: Number,                  // 涨跌幅（已经乘了100，单位为"%"）
12. yesterday\_close: Number,              // 昨收
13. today\_open: Number,                   // 今开
14. high: Number,                         // 最高
15. low: Number,                          // 最低
16. volume: Number,                       // 成交量（单位为"万手"）
17. volume\_of\_transaction: Number,        // 成交额（单位为"万"）
18. marketValue: Number,                  // 总市值（单位为"亿"）
19. floatMarketValue: Number,             // 流通市值（单位为"亿"）
20. turnOverRate: Number,                 // 换手率（已经乘了100，单位为"%"）
21. PB\_ratio: Number,                     // 市净率
22. amplitude: Number,                    // 振幅（已经乘了100，单位为"%"）
23. PE\_ratio: Number,                     // 市盈率
24. \* }
25. \* @param callback
26. \*/
27. updateRTInfo: (code, rtInfo, callback) => { },
29. /\*\*
30. \* 新增实时信息，只会在初始化RTInfo这个Collection时调用，调用一次后，不会再被调用
31. \* @deprecated
32. \* @param rtInfo
33. \* @param callback
34. \*/
35. addRTInfo: (rtInfo, callback) => { },
37. /\*\*
38. \* 获取所有的实时股票数据
39. \* @param callBack
40. \* 形如 (err,doc)
41. \*/
42. getAllRTInfo:(callback)=>{  }
43. };

2.3.3 hotStockAndBoard模块

1. exports.hotDB = {
3. /\*\*
4. \* 获得热门板块
5. \* @param callback
6. \*/
7. getHotBoards: (callback) => {},
9. /\*\*
10. \* 更新热门板块
11. \* @param hotBoards
12. \* @param callback
13. \*/
14. updateHotBoards: (hotBoards, callback) => { },
16. /\*\*
17. \* 获得热门股票
18. \* @param callback
19. \*/
20. getHotStocks: (callback) => { },
22. /\*\*
23. \* 更新热门股票
24. \* @param hotStocks
25. \* @param callback
26. \*/
27. updateHotStocks: (hotStocks, callback) => { }
28. };

## 3.1 逻辑层设计描述

### 3.1.1 总体描述

此次逻辑层的设计相对简单，但对算法的效率提出了比较高的要求。所有的返回数据以及异常处理均使用了Node.js的异步特性，有利于改善用户体验；使用内存缓存来尽可能地提升数据访问速度。

## 3.2 数据层设计描述

### 3.2.1 总体描述

采用了mongoDB作为这次工程的数据库，充分的利用了mongoDB非关系型数据库的特性，在数据持久化上极为灵活。

增加了冗余存储以提高数据获取的速度，充分的利用投影，以增快数据库IO的速度。