Algo-trading market Client LLD

## **MarketRequest class**

#### Functionality

Provides parsing for JSON requests and calls the RequestOperations class for performing actions of the request object.

#### Request JSON fields

* All Requests:
  + “type” - request type, one of the following : {“buy”, “sell”, “queryBuySell”, “queryUser”, “queryMarket”, “cancelBuySell”}
* Buy/Sell requests:
  + "commodity" - commodity ID.
  + “amount” - buy/sell amount.
  + “price” - buy/sell price.
* Cancel buy/sell request:
  + “id” - request ID to query.
* Query buy/sell request:
  + “id” - request ID to query.
* Query user request:
  + No arguments
* Query user requests:
  + No arguments
* Query market request:
  + "commodity" - commodity ID.
* Query all market requests:
  + No arguments

**Files:**

* Data layer:
* *DataEntries*

AllDataRequest.cs: Holds a record of pending request.

AllMarketRequest.cs: Holds for all the commodities their ask and bid prices

MarketBuySell.cs: Stores the buy or sell response from the server.

MarketCommodityOffer.cs: Stores Ask price and Bid price.

MarketItemQuery.cs: Stores the Data of active buy or sell request.

MarketUserData.cs: Stores the current state of the user.

MarketUserRequests.cs

* *Loggers*

HistoryLogger.cs: In charge of writing to the history file.

WriteHistory(int requestId, string action, int commodity, int price, int amount): Writes a line to the history file.

ReadHistory(): Returns a collection of records from the history file.

*MarketRequests*

BuyRequest.cs: Stores the buy request related information.

SellRequest.cs: Stores the sell request related information.

CancelBuySellRequest.cs: Stores a cancel request related information.

QuerybuySellRequest.cs: Stores an information about ongoing request.

QueryMarketRequest.cs: Stores the information about the market.

QueryUserRequest.cs: Stores the information about the user.

QueryUserRequests.cs: Stores all the pending requests the user sent to the server.

MarketClientClass.cs: In charge of communicating with the server.

Logic layer:

AMA.cs: Contains 2 timers –

One is activated in case the user wish to run the AMA (our recommended algorithm), while the other activated in case the user wants to activate the UserAMA (which contains only the user commands).

In addition, it's contain a 'counter' field which prevents from overloading the server by sleep function.

UserAsksLink.cs: In case the user choose the UserAMA, he insert the system commands. Those commands saved in UserAsksLink class which contains fields that stores the command info. After that it sent to the relevant functions.

* WPF\_App

MainWindow.xaml: the GUI interface the user works with.

BuyButton\_Click(object sender, RoutedEventArgs e): Commit a buy request.

SellButton\_Click(object sender, RoutedEventArgs e): Commit a sell request.

CancelRequestButton\_Click(object sender, RoutedEventArgs e): Deletes a pending from the server.

* Presentation layer: (not in usage)

Presentation.cs: In charge of communicating with the user, deal with illegal input, communicates with the logic layer- and represent the results to the user.

Details about:

Terminology, Trading framework and Actors can be found here:

<https://goo.gl/h5MyZu>