
TradingMate Documentation

Release 0.1.0

Alberto Cardellini

Jan 12, 2020

CONTENTS

1	Introduction	1
	Python Module Index	7
	Index	9

INTRODUCTION

TradingMate is an autonomous trading system that uses customised strategies to trade in the London Stock Exchange market. This documentation provides an overview of the system, explaining how to create new trading strategies and how to integrate them with TradingMate. Explore the next sections for a detailed documentation of each module too.

1.1 System Overview

TradingMate is a portfolio manager with the goal to help traders to record their trades and deals in the stock market providing useful statistics and metrics to help backtest and review trade performance. It offers a simple user interface that provide information about the current asset value, the overall profit/loss indicator and the trade history.

1.1.1 TradingMate

TradingMate is the main entity used to initialise all the components. It is the link between the user interface and the data.

TODO

1.2 Modules

Each section of this document provides the source code documentation of each component of TradingMate.

1.2.1 TradingMate

1.2.2 Model

The `Model` module contains the business logic and the data management of TradingMate.

Holding

class `Model.Holding.Holding` (*symbol, quantity, open_price=None*)

add_quantity (*value*)

Add or subtract (if value is negative) the value to the holding quantity

Portfolio

DatabaseHandler

class `Model.DatabaseHandler.DatabaseHandler` (*config, trading_log_path*)

Handles the IO operation with the database to handle persistent data

add_trade (*trade*)

Add a trade to the database

get_db_filepath ()

Return the database filepath

get_trades_list ()

Return the list of trades stored in the db

get_trading_log_name ()

Return the trading log database name

read_data (*filepath=None*)

Read the trade history from the json database and return the list of trades

- **filepath**: optional, if not set the configured path will be used

remove_last_trade ()

Remove the last trade from the trade history

write_data (*filepath=None*)

Write the trade history to the database

StockPriceGetter

1.2.3 UI

The `UI` module contains the components that compose the User Interface of TradingMate.

DataInterface

class `UI.DataInterface.DataInterface` (*client, data_callback*)

Thread that periodically requests the most recent data from TradingMate server notify the parent object through a callback function

task ()

The task done by this thread - override in subclasses

TradingMateClient

```
class UI.TradingMateClient.TradingMateClient (server)
    Client interface to the TradingMate business logic

    delete_last_trade_event (portfolio_id)
        Request last trade deletion to the server

    get_portfolios ()
        Get the loaded portfolios

    get_settings_event ()
        Request server to fetch TradingMate settings

    is_portfolio_auto_refreshing (portfolio_id)
        Return True if portfolio has data auto refresh enabled, False otherwise

    manual_refresh_event (portfolio_id)
        Request server to refresh portfolio data

    new_trade_event (new_trade, portfolio_id)
        Push new trade notification to the server

    open_portfolio_event (filepath)
        Request server to open a portfolio

    save_portfolio_event (portfolio_id, filepath)
        Request server to save a portfolio

    save_settings_event (settings)
        Request server to save the settings

    set_auto_refresh_event (value, portfolio_id)
        Set server to automatically update data for the portfolio

    stop ()
        Handle stop event

    unsaved_changes ()
        Request if open portfolios have unsaved changes and return True
```

GTK

The `gtk` module contains the `gtk` components and widgets of the graphical interface

UIHandler

ConfirmDialog

MessageDialog

PortfolioPage

SettingsWindow

1.2.4 Utils

The `Utils` module contains all the utility components.

ConfigurationManager

```
class Utils.ConfigurationManager.ConfigurationManager (config_path)
    Class that loads the configuration and credentials json files exposing static methods to provide the configurable parameters

    get_alpha_vantage_api_key ()
        Get the alphavantage api key

    get_alpha_vantage_base_url ()
        Get the alphavantage API base URI

    get_alpha_vantage_polling_period ()
        Get the alphavantage configured polling period

    get_credentials_path ()
        Get the filepath of the credentials file

    get_editable_config ()
        Get a dictionary containing the editable configuration parameters

    get_trading_database_path ()
        Get the filepath of the trading log file

    save_settings (config)
        Save the edited configuration settings
```

TaskThread

```
class Utils.TaskThread.TaskThread
    Thread that executes a task every N seconds

    cancel_timeout ()
        Cancel the timeout and run the task

    enable (enabled)
        Disable/enable this thread

    run ()
        Method representing the thread's activity.

        You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

    setInterval (interval)
        Set the number of seconds we sleep between executing our task

    shutdown ()
        Stop this thread

    task ()
        The task done by this thread - override in subclasses
```

Trade

class `Utils.Trade.Trade` (*date_string, action, quantity, symbol, price, fee, sdr, notes*)

Utils

class `Utils.Utils.Actions`
An enumeration.

class `Utils.Utils.Messages`
An enumeration.

class `Utils.Utils.Markets`
An enumeration.

1.3 Changelog

All notable changes to this project will be documented in this file.

The format is based on [Keep a Changelog](#), and this project adheres to [Semantic Versioning](#).

1.3.1 [2.1.0] - 2020-01-12

Changed

- Replaced TK user interface with GTK+ 3
- Tickers prices are fetched using `alpha-vantage` Python module
- **`alpha_vantage_polling_period`** configuration parameter is used to wait between each AV call
- AlphaVantage http requests are thread safe

Added

- Status bar showing portfolio filepath
- Button to open a new window tailing the current application log file

1.3.2 [2.0.0] - 2019-12-14

Changed

- Issue37 - Improved installation process and dependencies setup
- Updated default `.credentials` configured path
- Re-design of system architecture and API
- Edited Portfolios are not saved automatically and a warning is displayed

Added

- Added Pipfile to manage python dependencies
- Added FEE action
- Added notes field in trade
- Support load of multiple portfolios
- Save As and Save buttons per portfolio

1.3.3 [1.0.0] 2019-05-03

Added

- Initial release

PYTHON MODULE INDEX

m

`Model.DatabaseHandler`, 2

`Model.Holding`, 2

u

`UI.DataInterface`, 2

`UI.TradingMateClient`, 3

`Utils.ConfigurationManager`, 4

`Utils.TaskThread`, 4

`Utils.Trade`, 5

`Utils.Utils`, 5

INDEX

A

Actions (class in *Utils.Utils*), 5

add_quantity() (*Model.Holding.Holding* method), 2

add_trade() (*Model.DatabaseHandler.DatabaseHandler* method), 2

C

cancel_timeout() (*Utils.TaskThread.TaskThread* method), 4

ConfigurationManager (class in *Utils.ConfigurationManager*), 4

D

DatabaseHandler (class in *Model.DatabaseHandler*), 2

DataInterface (class in *UI.DataInterface*), 2

delete_last_trade_event() (*UI.TradingMateClient.TradingMateClient* method), 3

E

enable() (*Utils.TaskThread.TaskThread* method), 4

G

get_alpha_vantage_api_key() (*Utils.ConfigurationManager.ConfigurationManager* method), 4

get_alpha_vantage_base_url() (*Utils.ConfigurationManager.ConfigurationManager* method), 4

get_alpha_vantage_polling_period() (*Utils.ConfigurationManager.ConfigurationManager* method), 4

get_credentials_path() (*Utils.ConfigurationManager.ConfigurationManager* method), 4

get_db_filepath() (*Model.DatabaseHandler.DatabaseHandler* method), 2

get_editable_config() (*Utils.ConfigurationManager.ConfigurationManager* method), 4

get_portfolios() (*UI.TradingMateClient.TradingMateClient* method), 3

get_settings_event() (*UI.TradingMateClient.TradingMateClient* method), 3

get_trades_list() (*Model.DatabaseHandler.DatabaseHandler* method), 2

get_trading_database_path() (*Utils.ConfigurationManager.ConfigurationManager* method), 4

get_trading_log_name() (*Model.DatabaseHandler.DatabaseHandler* method), 2

H

Holding (class in *Model.Holding*), 2

I

is_portfolio_auto_refreshing() (*UI.TradingMateClient.TradingMateClient* method), 3

M

manual_refresh_event() (*UI.TradingMateClient.TradingMateClient* method), 3

Markets (class in *Utils.Utils*), 5

Messages (class in *Utils.Utils*), 5

Model.DatabaseHandler (module), 2

Model.Holding (module), 2

N

new_trade_event() (*UI.TradingMateClient.TradingMateClient* method), 3

O

open_portfolio_event() (*UI.TradingMateClient.TradingMateClient* method), 3

R

`read_data()` (*Model.DatabaseHandler.DatabaseHandler method*), 2
`remove_last_trade()` (*Model.DatabaseHandler.DatabaseHandler method*), 2
`run()` (*Utils.TaskThread.TaskThread method*), 4

S

`save_portfolio_event()` (*UI.TradingMateClient.TradingMateClient method*), 3
`save_settings()` (*Utils.ConfigurationManager.ConfigurationManager method*), 4
`save_settings_event()` (*UI.TradingMateClient.TradingMateClient method*), 3
`set_auto_refresh_event()` (*UI.TradingMateClient.TradingMateClient method*), 3
`setInterval()` (*Utils.TaskThread.TaskThread method*), 4
`shutdown()` (*Utils.TaskThread.TaskThread method*), 4
`stop()` (*UI.TradingMateClient.TradingMateClient method*), 3

T

`task()` (*UI.DataInterface.DataInterface method*), 2
`task()` (*Utils.TaskThread.TaskThread method*), 4
`TaskThread` (*class in Utils.TaskThread*), 4
`Trade` (*class in Utils.Trade*), 5
`TradingMateClient` (*class in UI.TradingMateClient*), 3

U

`UI.DataInterface` (*module*), 2
`UI.TradingMateClient` (*module*), 3
`unsaved_changes()` (*UI.TradingMateClient.TradingMateClient method*), 3
`Utils.ConfigurationManager` (*module*), 4
`Utils.TaskThread` (*module*), 4
`Utils.Trade` (*module*), 5
`Utils.Utils` (*module*), 5

W

`write_data()` (*Model.DatabaseHandler.DatabaseHandler method*), 2