Getting started ...

myOp€nTr@der

Welcome to MyOpenTrader ...

MyOpenTrader is an open-source complex event based trading engine. In the simplest way, users download & configure MOT, create their own strategy and start the engine. Trading happens automatically and "hopefully" the strategies do the rest.

System requirements:

In order to use MyOpenTrader (MOT), you have to fulfil the following requirements:

- A database for storing the meta/data (currently only mysql is supported!)
- Java 1.7++

Optional:

- An Interactive Brokers Account, with its API accessible from where you intend to run MOT. (Development and Testing can be done with the embedded tick Generator)
- An ActiveMQ broker for message processing. (Use the embedded Apollo broker instead for all development & testing)

Installation

Make sure you properly install MyOpenTrader on your host (as outlined in the system requirements). More details on how to install each of the components can be found here:

- Configuring Interactive Brokers Traders Workstation
- Setting up ActiveMQ Broker
- Setting up the database

Starting MyOpenTrader

First of all, lets start the Embedded Message bus:

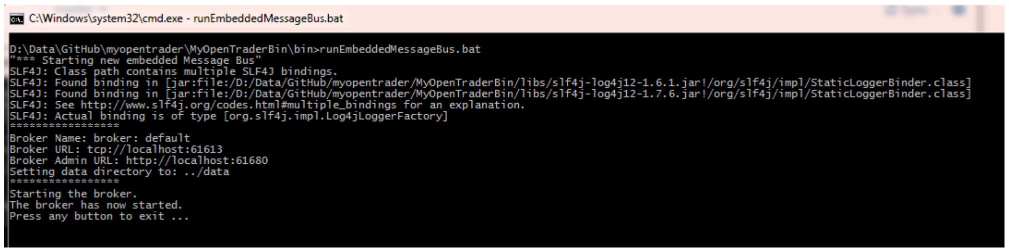
What is the Embedded Message Broker?

For your backtesting, it is important that you have the ability to replay all of your previous days ticks. MyOpenTrader has a built in functionality, that makes sure you can replay your previous days.

Starting / Stopping the Embedded Message Broker:

In your <MyOpenTraderBin/bin> folder, you will see the two scripts: runEmbeddedMessageBus.bat or .sh

Simply run this script to start the replayer.



C:\Windows\system32\cmd.exe - runEmbeddedMessageBus.bat

```
D:\Data\GitHub\myopentrader\MyOpenTraderBin\bin>runEmbeddedMessageBus.bat
"*** Starting new embedded Message Bus"
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/D:/Data/GitHub/myopentrader/MyOpenTraderBin/libs/slf4j-log4j12-1.6.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/D:/Data/GitHub/myopentrader/MyOpenTraderBin/libs/slf4j-log4j12-1.7.6.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
*****************
Broker Name: broker: default
Broker URL: tcp://localhost:61613
Broker Admin URL: http://localhost:61680
Setting data directory to: ../data
*****************
Starting the broker.
The broker has now started.
Press any button to exit ...
```

Configuring the Embedded Message Broker:

The command line utility has no command line properties

Once the message bus is up & running, we can start publishing the first random ticks:

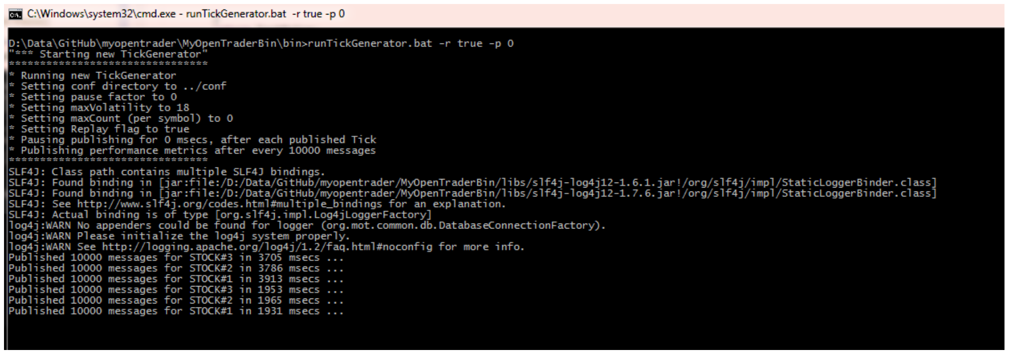
What is the Tick Generator?

MyOpenTrader includes a random tick generator, which can be used for development purposes or in case, there is no real live feed of market data available.

Starting / Stopping the Tick Generator:

In your <MyOpenTraderBin/bin> folder, you will see the two scripts: runTickGenerator.bat or .sh

Simply run this script to start the generator. The tickGenerator will spit out performance statistics with each 10k ticks being published.



C:\Windows\system32\cmd.exe - runTickGenerator.bat -r true -p 0

```
D:\Data\GitHub\myopentrader\MyOpenTraderBin\bin>runTickGenerator.bat -r true -p 0
"*** Starting new TickGenerator"
********************************
* Running new TickGenerator
* Setting conf directory to ../conf
* Setting pause factor to 0
* Setting maxVolatility to 18
* Setting maxCount (per symbol) to 0
* Setting Replay flag to true
* Pausing publishing for 0 msecs, after each published Tick
* Publishing performance metrics after every 10000 messages
********************************
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/D:/Data/GitHub/myopentrader/MyOpenTraderBin/libs/slf4j-log4j12-1.6.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/D:/Data/GitHub/myopentrader/MyOpenTraderBin/libs/slf4j-log4j12-1.7.6.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
log4j:WARN No appenders could be found for logger (org.mot.common.db.DatabaseConnectionFactory).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Published 10000 messages for STOCK#3 in 3705 msecs ...
Published 10000 messages for STOCK#2 in 3786 msecs ...
Published 10000 messages for STOCK#1 in 3913 msecs ...
Published 10000 messages for STOCK#3 in 1953 msecs ...
Published 10000 messages for STOCK#2 in 1965 msecs ...
Published 10000 messages for STOCK#1 in 1931 msecs ...
```

Configuring the Tick Generator:

The command line utility has a couple of parameters, that you can pass into:

Flag	Used for:
-c	(Optional) Specify a configuration directory (use the ../conf directory as default)
-s	(Optional) Specify a list of symbols to generate ticks for (default is: STOCK#1, STOCK#2, STOCK#3")
-p	(Optional) Pause factor. Forces the generator to pause for x msecs after each tick (default is 500)
-m	(Optional) Set the max volatility of the tick prices (defaults to 18)
-t	(Optional) Define a max count of ticks you want to publish (default: unlimited or until stopped)

-r	(Optional) Set the "replay" flag. Replayed ticks are NOT written to the database - useful when backtesting your strategies (Default: false)
-h	(Optional) Display the command line help functionality

And last, but surely not least, we need to process all of the incoming ticks. Lets start the MyOpenTrader Core Engine:

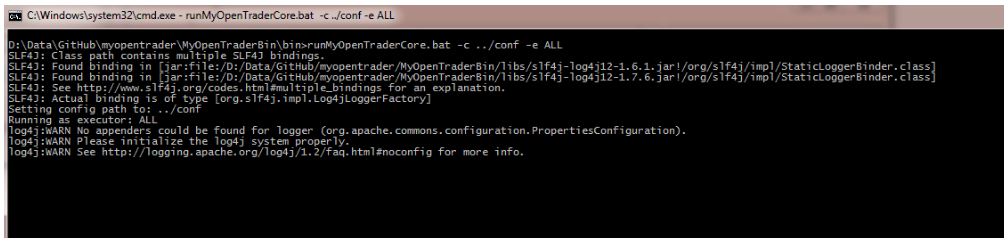
What is the MyOpenTrader Core Engine?

As you may have learnt, myOpenTrader consists out of 3 parts, the feeder, the core engine and the web application. Each part can be started individually for testing / debugging purposes.

Starting / Stopping the MyOpenTrader Core Engine:

In your **<MyOpenTraderBin/bin>** folder, you will see the two scripts: *runMyOpenTraderCore.bat or .sh*

Simply run this script to start the engine


C:\Windows\system32\cmd.exe - runMyOpenTraderCore.bat -c ../conf -e ALL

```
D:\Data\GitHub\myopentrader\MyOpenTraderBin\bin>runMyOpenTraderCore.bat -c ../conf -e ALL
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/D:/Data/GitHub/myopentrader/MyOpenTraderBin/libs/slf4j-log4j12-1.6.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/D:/Data/GitHub/myopentrader/MyOpenTraderBin/libs/slf4j-log4j12-1.7.6.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Setting config path to: ../conf
Running as executor: ALL
log4j:WARN No appenders could be found for logger (org.apache.commons.configuration.PropertiesConfiguration).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
```

Configuring the MyOpenTrader Core Engine:

The command line utility has a couple of parameters, that you can pass into:

Flag	Used for:
-c	(Optional) Specify a configuration directory (use the ../conf directory as default)
-e	(**Required**) Specify which executor to run (see Executor for more details) - default is ALL
-h	(Optional) Display the command line help functionality

Et voila - the engine is up & running, processing all of your ticks. You are ready to have a go at developing your first strategies.

Additional Documentation can be found here:

- What is MyOpenTrader?
- Getting started ...
- System requirements
- Installation Guide
- Usage Guide
- Development Guide
- Copyright information - GPLv3