

AlgoTrader Quick Start Guide

Algorithmic Trading Software

Version 4.0

by Andy Flury and Robert Kolar

Table of Contents

1. Introduction	1
2. Installation	3
3. Starting a Trading Strategy	9
3.1. Starting a Back Test	9
3.2. Starting Live Trading	10
4. Creating a Trading Strategy	14
4.1. AlgoTrader Strategy Wizard	14
4.2. Adding Strategy Logic	17
5. Managing data	23
5.1. Reference Data	24
5.2. Historical Data	25
6. Cryptocurrency Trading	26
7. Shutting down the System	28

Introduction

The AlgoTrader Quick Start Guide is based on the AlgoTrader 30-day trial provided via [Amazon AWS](#)¹ which can be requested [here](#)².

The AlgoTrader 30-day trial version includes a fully functional AlgoTrader installation as well as the following example strategies:

- [Box Strategy](#)³
- [Break Out Strategy](#)⁴
- [Exponential Moving Average Strategy](#)⁵
- [IPO Strategy](#)⁶
- [Pairs Trading Strategy](#)⁷
- [Random Strategy](#)⁸

The AlgoTrader 30-day trial version contains the following pre-installed components:

- Java JDK
- AlgoTrader Server
- AlgoTrader Eclipse IDE
- MySQL Database
- InfluxDB Database
- Toad for MySQL Freeware
- InteractiveBrokers Gateway
- TortoiseGit
- Git for Windows
- Notepad++

¹ <https://aws.amazon.com/>

² <http://www.algotrader.com/product/trial-version/>

³ http://doc.algotrader.ch/html/Box_Strategy.html

⁴ http://doc.algotrader.ch/html/BreakOut_Strategy.html

⁵ http://doc.algotrader.ch/html/EMA_Strategy.html

⁶ http://doc.algotrader.ch/html/IPO_Strategy.html

⁷ http://doc.algotrader.ch/html/PairsTrading_Strategy.html

⁸ http://doc.algotrader.ch/html/Random_Strategy.html

- Google Chrome
- MS Excel (not activated)



Warning

Amazon AWS usage cost will apply based on the instance type select. For further details, please visit: <https://aws.amazon.com/ec2/pricing/>



Warning

It is prohibited to reverse engineer, decompile, disassemble, or copy any parts of the AlgoTrader 30-day trial

Installation

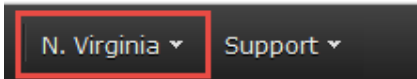
The following steps will guide through the installation of a Windows AWS Instance containing the AlgoTrader 30-day trial version

1. Open the Amazon AWS console:

<https://console.aws.amazon.com/>

and Login using the Amazon username and password.

2. In the upper right-hand corner of the screen make sure the N. Virginia Region is selected:



3. Select EC2



4. Click Launch Instance in the middle of the screen

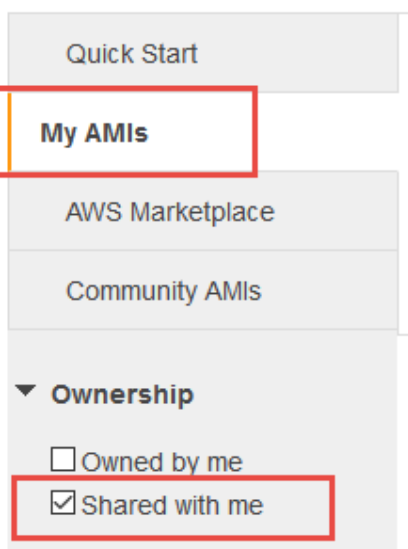
Create Instance

To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

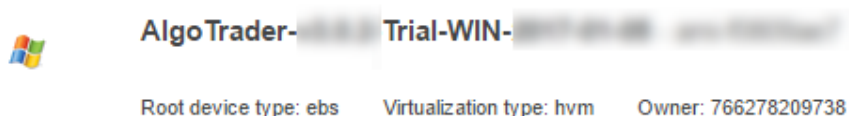



Note: Your instances will launch in the US East (N. Virginia) region

5. In the menu on the left-hand side select My AMIs and check Shared with me



6. Select the `AlgoTrader-x.x.x-Trial-WIN-xxxx.xx.xx`




 **Note**

The `AlgoTrader-x.x.x-Trial-WIN-xxxx.xx.xx` image is only available in the N. Virginia Region but not in any other Regions. It is thus necessary that the N. Virginia Region is selected in the top-right corner of the screen

7. On the next screen select the Instance Type. We recommend at least instance type `t2.medium`, ideally instance type `m4.large`

Type	vCPUs	Memory (GiB)
r4.large	2	15.25
m4.large	2	8

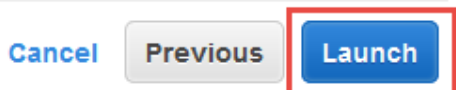
 **Warning**

Amazon AWS usage cost depends on the instance type selected. For further details, please visit: <https://aws.amazon.com/ec2/pricing/>

8. Click `Review and Launch` on the bottom right of the screen

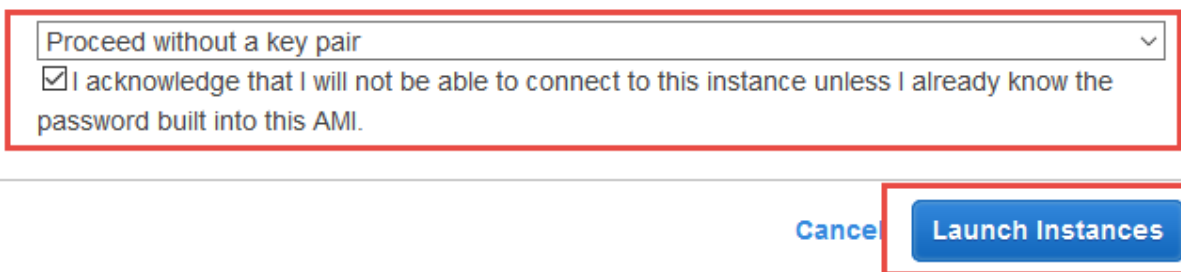


9. Click `Launch` on the bottom right of the screen



10. On the Dialog that shows select `Proceed without a key pair`, select `I acknowledge...` and click `Launch Instance`

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).



11. Click `View Instances` on the bottom right of the screen



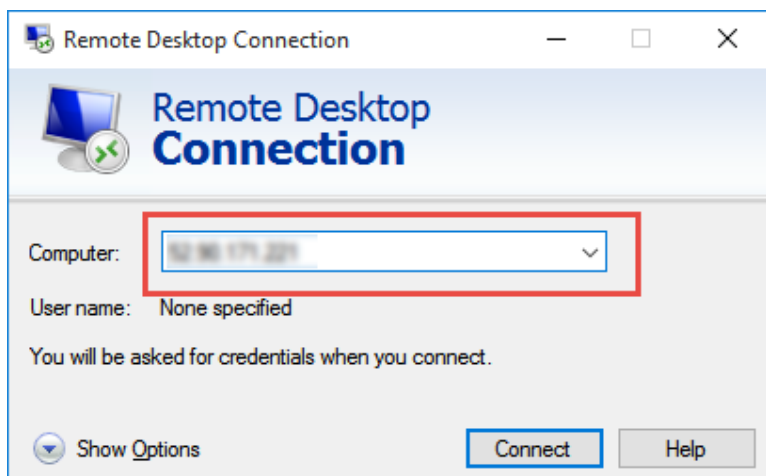
12. On the next screen, one can see the Instance starting up. Wait until it is running and note the `Public IP` address.

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS	Public IP
i-0796d952...		t2.large	us-east-1a	running	2/2 checks ...	None	ec2-107-21-173-70.com...	107.21.173.70

Note

It will take at least 4 minutes for the Instance to startup and become available.

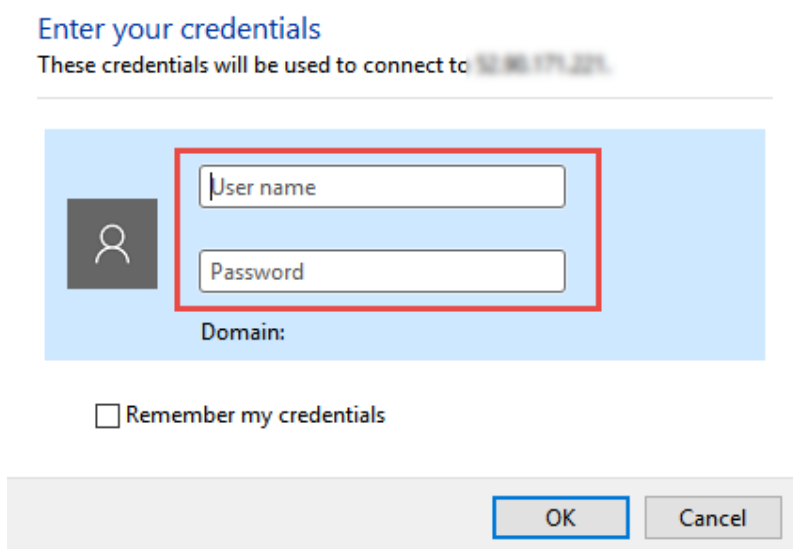
13. Use Microsoft Remote Desktop to connect to the instance by typing in the `Public IP` address that was noted in the previous step



Note

- You need to use Remote Desktop Connection (RDP), a Web Browser will NOT work for this
- For more information on Remote Desktop Connections please visit: [Windows](#)¹, [Mac](#)² and [Linux](#)³

14. Specify `User name` and `Password` that was provided in the Email after signing up for the AlgoTrader free 30-day trial.

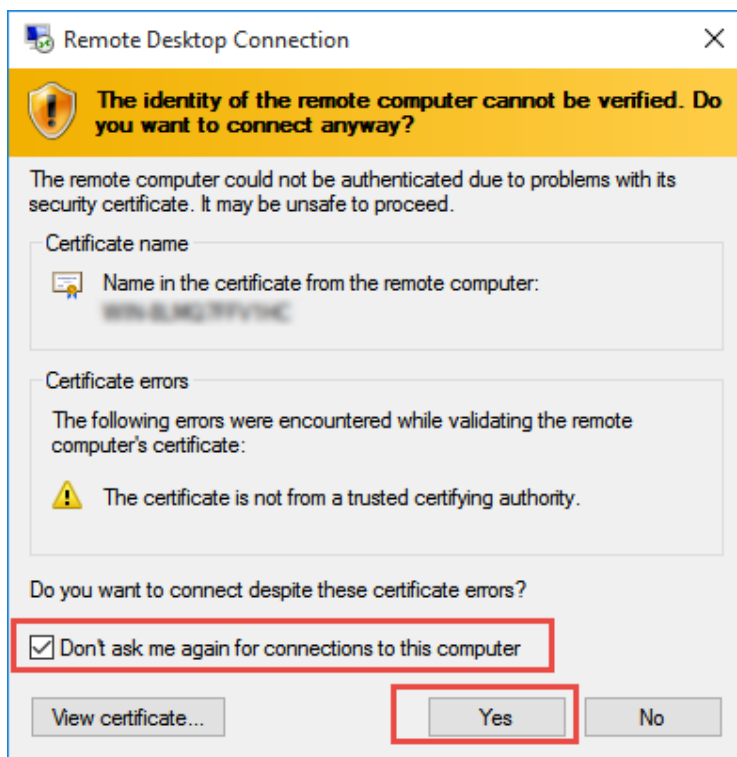


15. On the next dialog select `Don't ask...` and click `Yes`

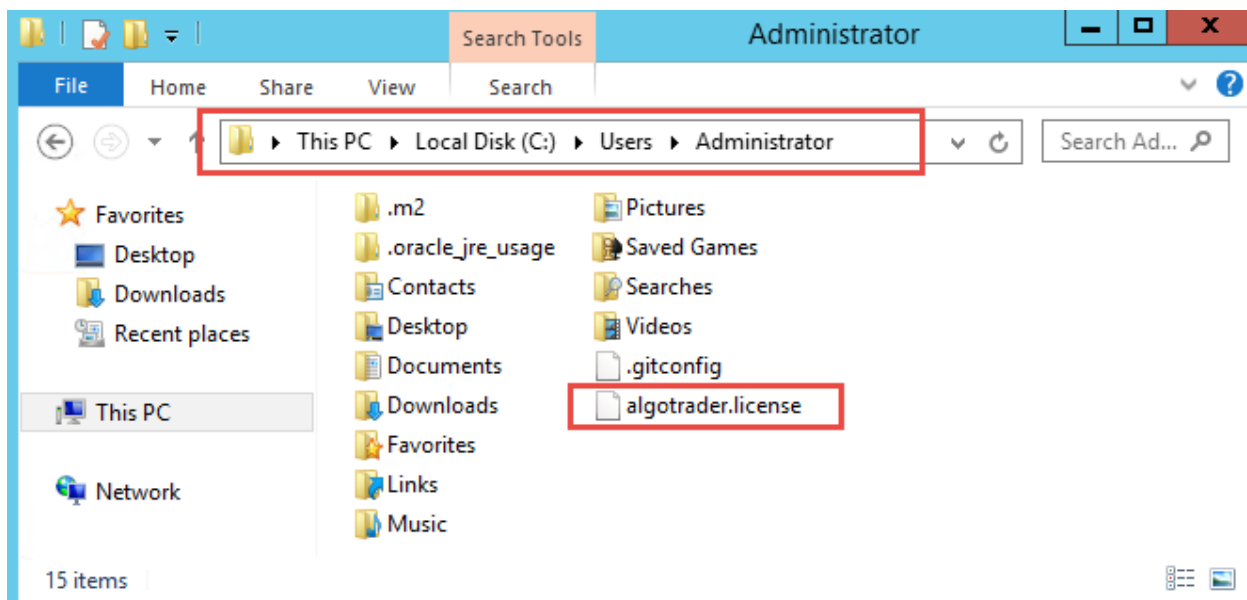
¹ <http://windows.microsoft.com/en-us/windows7/products/features/remote-desktop-connection>

² <https://itunes.apple.com/en/app/microsoft-remote-desktop/id715768417>

³ <http://www.rdesktop.org/>



16. You are now connected to the Amazon AWS Instance containing AlgoTrader. Now copy the `algotrader.license` file that was attached to the Email (that was received when signing up for the AlgoTrader free 30-day trial) to `C:\Users\Administrator` on the Amazon Instance:



17. Microsoft Excel is pre-installed on the machine to view the AlgoTrader Excel based Back Test Report. However due to licensing restrictions Microsoft Excel has not been activated yet. If a Microsoft Office license is available, please activate Microsoft Excel using the license key. Alternatively, one can request a free 60-day trial through:

<https://www.microsoft.com/en-us/evalcenter/evaluate-office-professional-plus-2013>

18.The Amazon Instance including AlgoTrader is now ready for usage!



Note

- Amazon Windows Instances tend to run a bit slow after they are first created. Responsiveness will however increase after some time.
- The performance of the Amazon Instance also depends on the instance type selected in step 7.

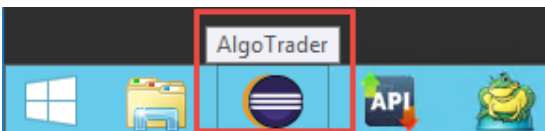
Starting a Trading Strategy

This section gives a quick introduction on how to start a trading strategy by discussing the [Box example strategy](#)¹.

To start any of the other strategies please consult the relevant parts in the documentation:

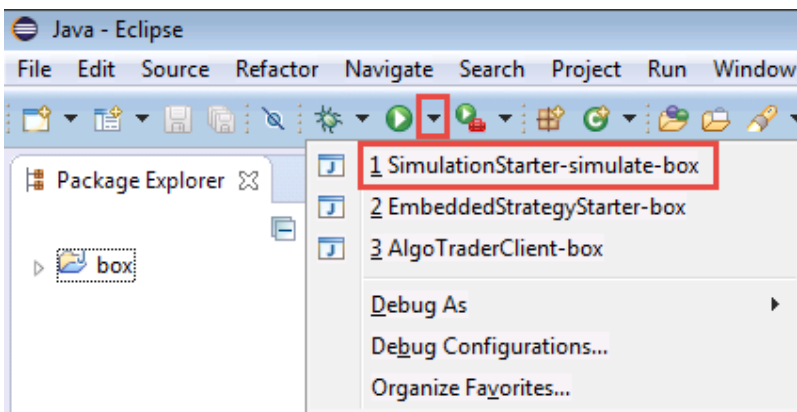
- [Break Out Strategy](#)²
- [Exponential Moving Average Strategy](#)³
- [IPO Strategy](#)⁴
- [Pairs Trading Strategy](#)⁵
- [Random Strategy](#)⁶

First one needs to start the AlgoTrader Eclipse IDE (Integrated Development Environment) using the Eclipse icon in the task bar.



3.1. Starting a Back Test

Launch the `SimulationStarter-simulate-box` by first clicking the downward facing arrow next to the green start icon.



The system will now perform a back test based on historical data

Once the back test has finished, the Excel based back test report will open automatically.

¹ http://doc.algotrader.ch/html/Box_Strategy.html

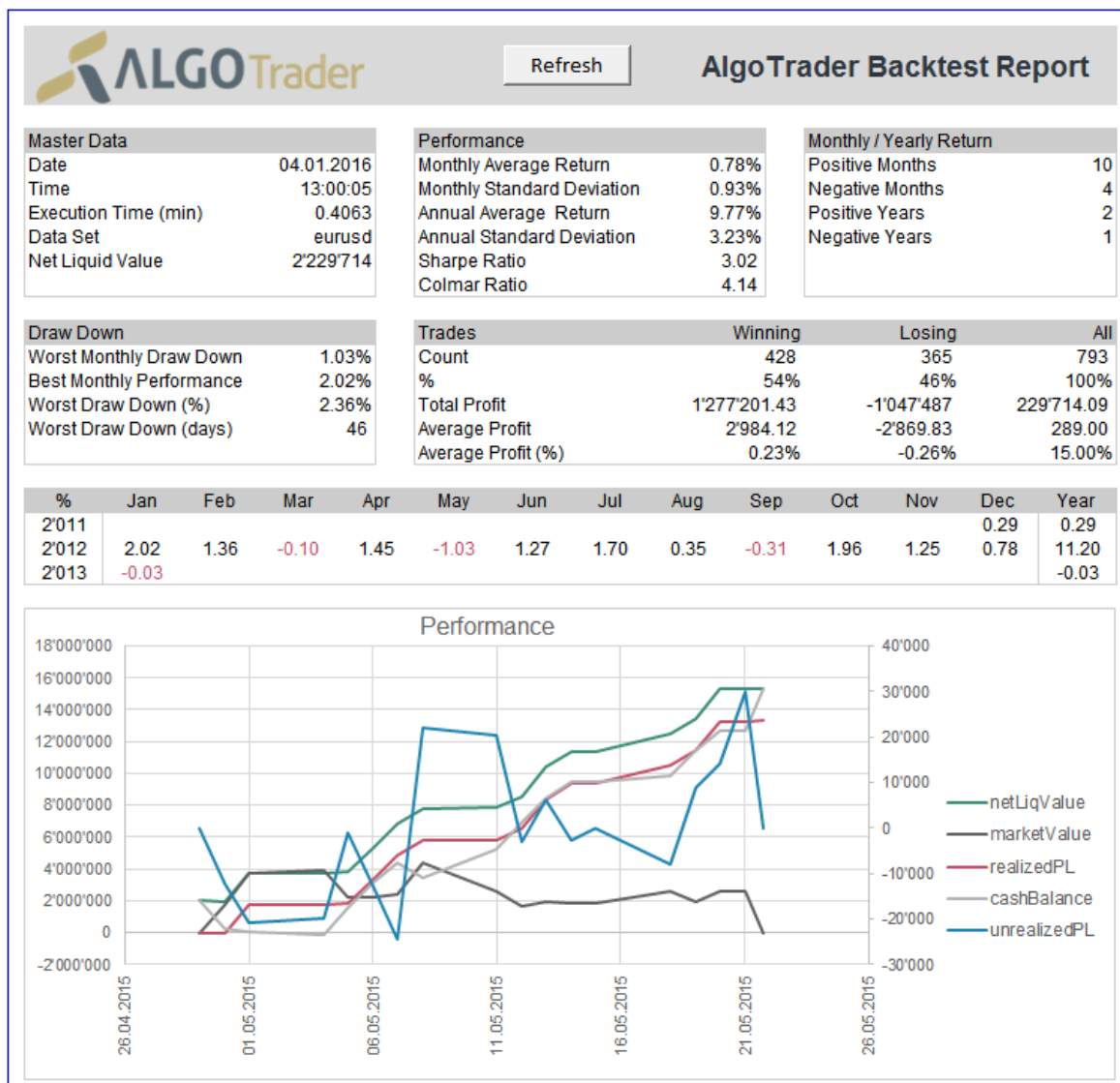
² http://doc.algotrader.ch/html/BreakOut_Strategy.html

³ http://doc.algotrader.ch/html/EMA_Strategy.html

⁴ http://doc.algotrader.ch/html/IPO_Strategy.html

⁵ http://doc.algotrader.ch/html/PairsTrading_Strategy.html

⁶ http://doc.algotrader.ch/html/Random_Strategy.html



For further information regarding back testing, visit the following chapters in the documentation:

- [Strategy Backtesting](#)⁷
- [Starting a Strategy in Simulation Mode](#)⁸

3.2. Starting Live Trading

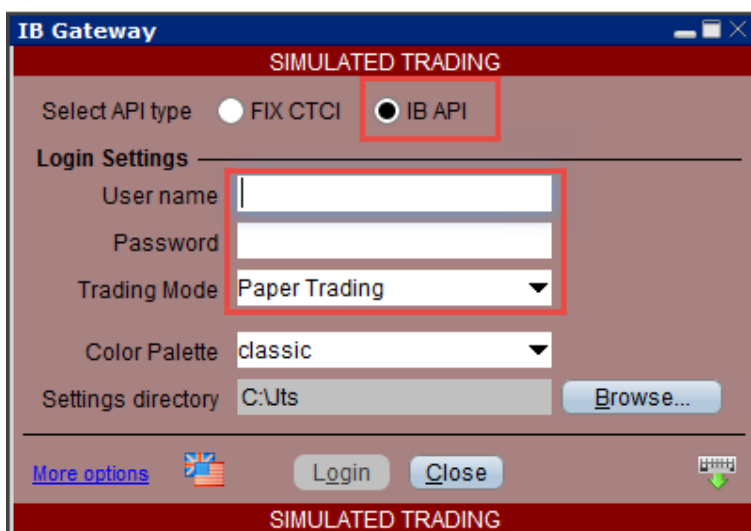
Before starting the strategy in live trading mode, the Interactive Brokers Gateway needs to be started using the API icon in the task bar.



⁷ http://doc.algotrader.ch/html/Strategy_Backtesting.html

⁸ http://doc.algotrader.ch/html/Starting_AlgoTrader.html#Simulation_Mode

Then select IB API, enter the Interactive Brokers username and password and select Paper Trading:



Warning

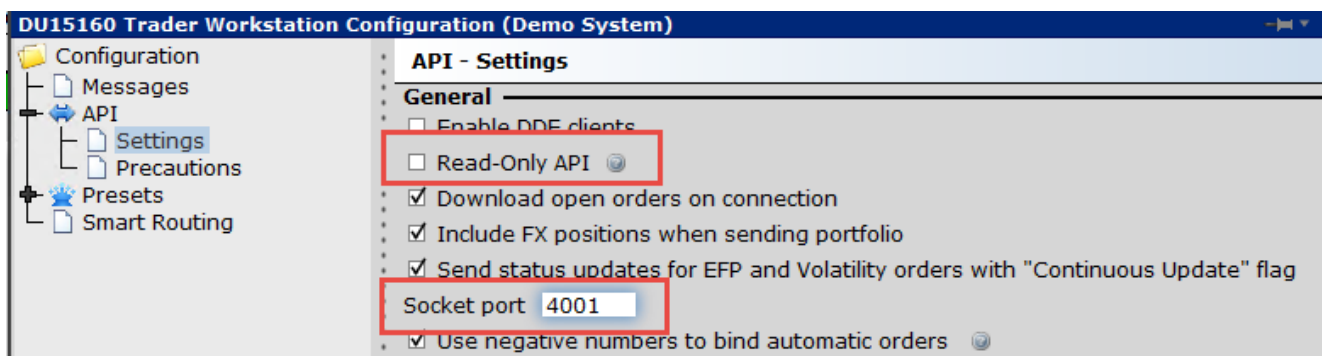
Do not use a live account until absolutely sure that the trading strategy works as expected. Until then use an IB paper trading account.

Note

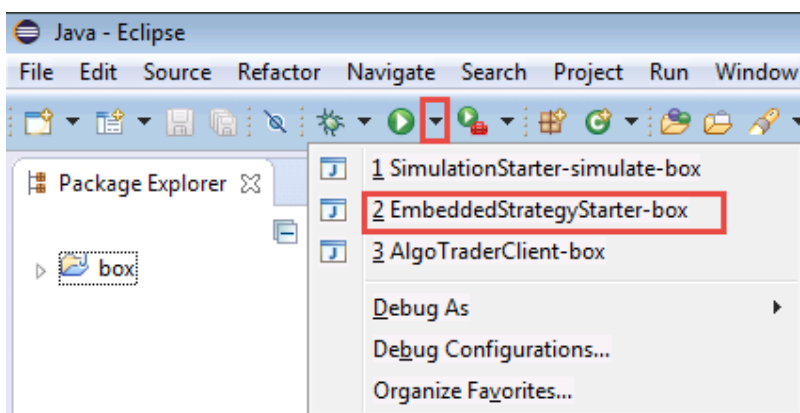
If no Interactive Brokers account is available yet, one can use the following demo account which is available to everybody. This account however provides faux market data and does not provide historical data.

- Username: edemo
- Password: demouser

Once the Interactive Broker Gateway has started, go to `Configure / Settings`, then select `API / Settings`, make sure that `Read-Only API` is disabled and `Socket port` is set to 4001.



Now launch the `EmbeddedStrategyStarter-box` by first clicking the downward facing arrow next to the green start icon. This will start the AlgoTrader server as well as the Box strategy and will connect to the InteractiveBrokers Gateway.



Once the system is started up it will automatically open the HTML5 frontend (within the Chrome browser).

The screenshot displays the AlgoTrader HTML5 frontend. At the top, there's a 'Strategy' dropdown set to 'ALL'. Below this, a summary row shows: Real P&L: (\$47.95), Unreal P&L: 0, Market Value: -, Positions: 0, Net Liquidity: \$99,952.05, and Cash Balance: \$99,952.05. The main content area is divided into three sections: 'Market Data (1)' with a table for EURUSD, 'Position (0)', and 'Transaction (3)'. A red box highlights the 'Box Strategy' details on the right side.

Name	Value
State	CREATED
Units	1
Upper Target	1.14639
Upper Buffer	1.1455
Top	1.145
Bottom	1.14461
Lower Buffer	1.14411
Lower Target	1.14322

At the bottom of the 'Box Strategy' panel, there is a red button labeled 'TERMINATE TRADE'.

For further information on live trading, please visit [Starting a Strategy in Live Trading Mode](http://doc.algotrader.ch/html/Starting_AlgoTrader.html#Live_Trading_Mode)⁹

⁹ http://doc.algotrader.ch/html/Starting_AlgoTrader.html#Live_Trading_Mode

For further information on the AlgoTrader Web Frontend please visit the AlgoTrader documentation regarding the [HTML5 client](#)¹⁰

¹⁰ http://doc.algotrader.ch/html/Client.html#HTML5_Client

Creating a Trading Strategy

This section will give a quick introduction on how to create a trading strategy by discussion the EMA (Exponential Moving Average) Strategy

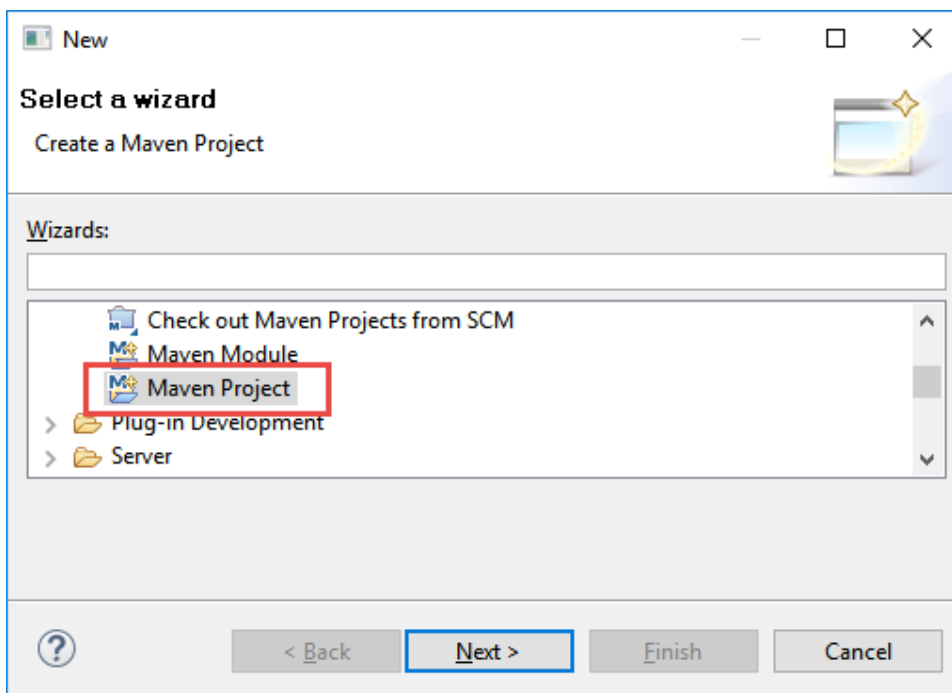


Note

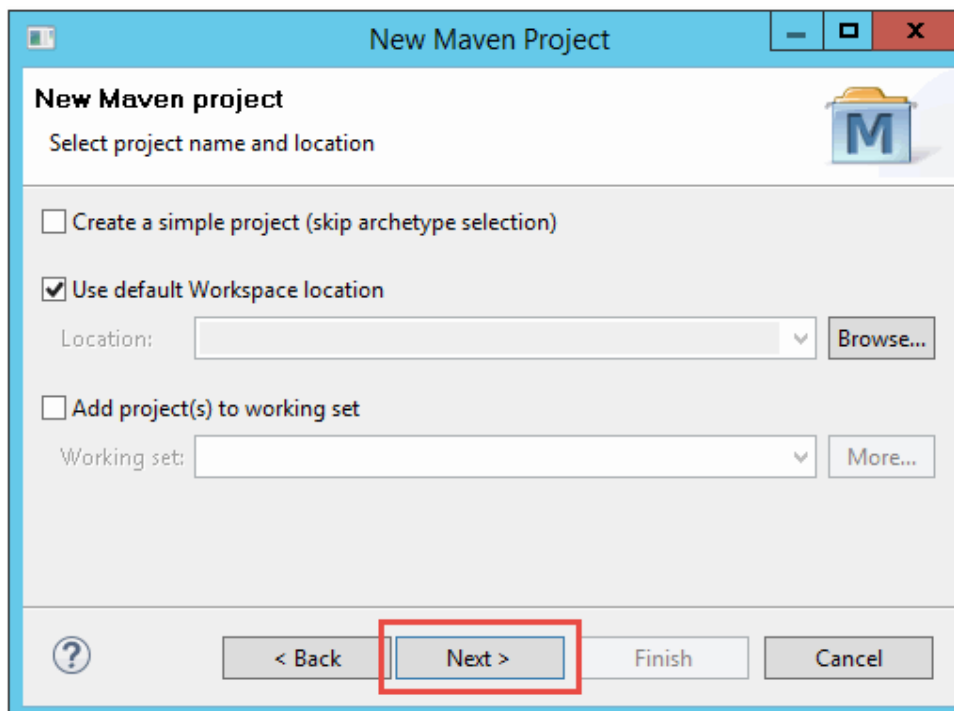
The AlgoTrader 30-day free trial already contains the final EMA strategy with all artifacts.

4.1. AlgoTrader Strategy Wizard

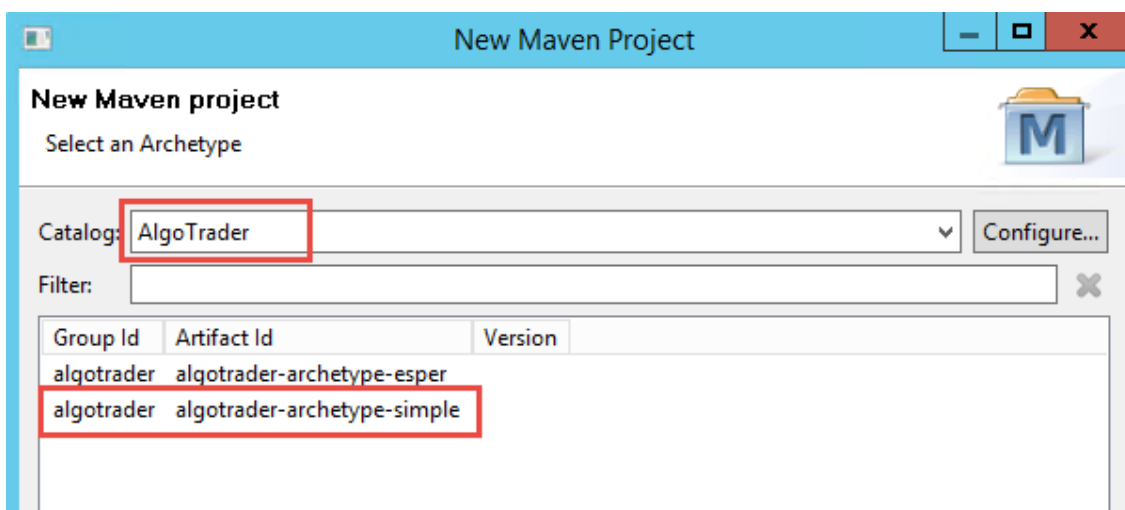
The AlgoTrader Strategy Wizard provides an easy way to automatically create all artifacts necessary for an AlgoTrader based trading strategy. The Wizard can be started via the `File / New / Other` which will bring up the following screen where the Maven Project wizard can be selected:



On the next screen please click `Next`.



On the next screen please select the Catalog AlgoTrader select `algotrader-archetype-simple` and click Next.



On the next screen, the following items have to be entered:

Group Id

The maven group id (e.g. `algotrader`), all lower-case, can contain periods

Artifact Id

The maven artifact id (e.g. `algotrader-ema`), all lower-case, can contain dashes

Version

The maven version (e.g. `1.0.0-SNAPSHOT`), `x.y.z`, plus optionally `-SNAPSHOT`

Package

The java package name (`ch.algotrader.strategy`), all lower-case, can contain periods.

name

The name of the strategy (e.g. `ema`), all lower-case, no periods, no dashes

serviceName

The name of the strategy service (e.g. `EMA`), first letter upper-case or all upper-case, do not include `Service` at the end (e.g. do not specify `EMAService`)

**Note**

For Spring Auto-Wiring to work the package name needs to be `ch.algotrader.strategy`. If a different package is assigned services (e.g. `OrderService` and `LookupService`) will not be available.

New Maven project
Specify Archetype parameters

Group Id:

Artifact Id:

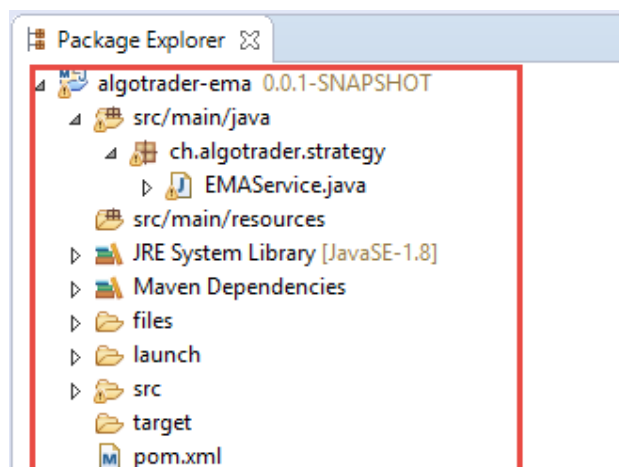
Version:

Package:

Properties available from archetype:

Name	Value
name	ema
serviceName	EMA

When clicking `Finish` the Strategy Wizard will create a new Eclipse project called `algotrader-ema`.

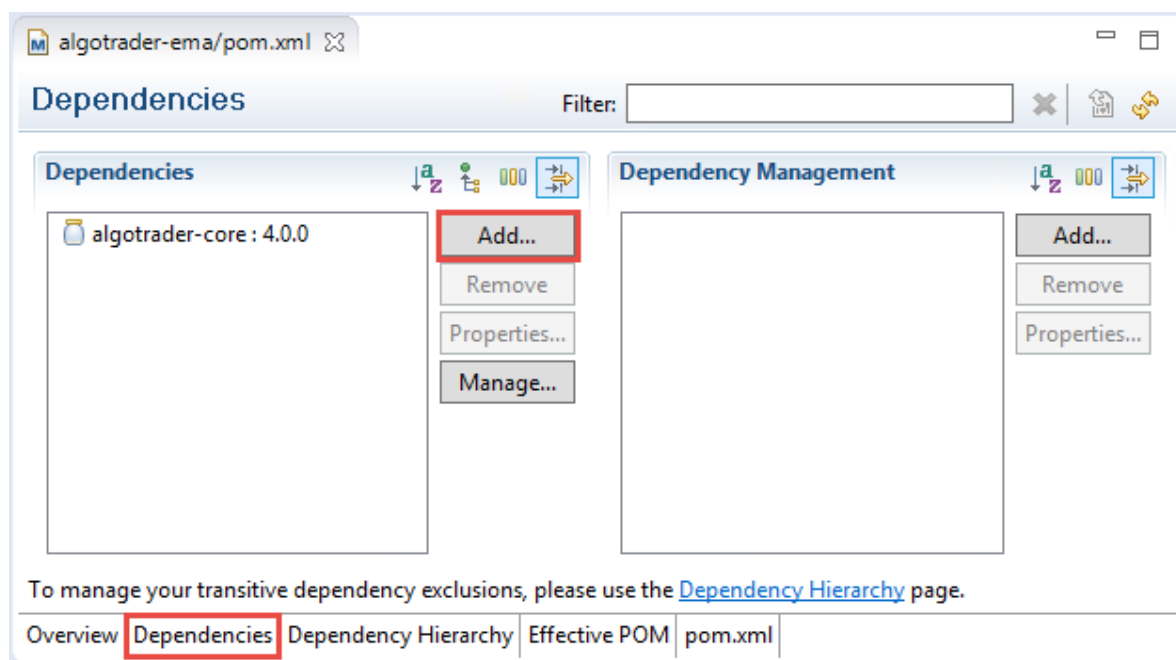


4.2. Adding Strategy Logic

The Strategy Wizard also generated some boiler plate code that needs to be replaced with the actual logic of the EMA strategy.

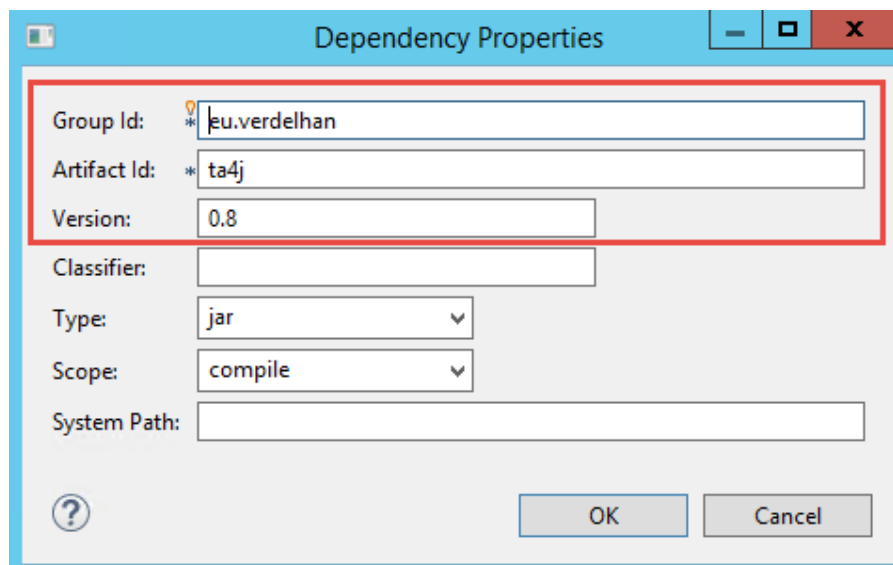
AlgoTrader strategies are regular Java programs. Due to this any type of java library or add-ons can be used. The EMA strategy is based on the [TA4J¹](https://github.com/mdeverdelhan/ta4j) library which contains a collection of over 100 technical indicators.

As a first step the library needs to be configured in order to be available to the trading strategy by adding the dependency to the `pom.xml` file. Double click the `pom.xml` file to open it, go to the `Dependencies` Tab and click `Add`:



Then add the following items:

¹ <https://github.com/mdeverdelhan/ta4j>



- Group Id: eu.verdelhan
- Artifact Id: ta4j
- Version: 0.8

Then click `OK` and save the file.

Now, double click the `EMAService.java` file which contains the main logic of the EMA strategy.

The header of the `EMAService.java` is already generated and no further changes are necessary. It contains the java class name (`EMAService`) as well as the name of the interface it is derived from (`StrategyService`). Also, it contains an `@Component` annotation which marks the Java class as a Spring bean and automatically gets references to necessary service like the `OrderService` and the `LookupService`.



Note

For Spring Auto-Wiring to work the package name needs to be `ch.algotrader.strategy`. If a different package is assigned services (e.g. `OrderService` and `LookupService`) will not be available.

```
@Component
public class EMAService extends StrategyService {
```

The next part of the `EMAService.java` contains settings the strategy will use. Three of them are already generated by the Wizard but a few more need to be added.

```
private final long accountId = 1;
```

```
private final long securityId = 25;
private final long orderQuantity = 10000;
private final int emaPeriodShort = 10;
private final int emaPeriodLong = 20;

private TimeSeries series;
private DifferenceIndicator emaDifference;
```

- The `accountId` defines the id of the account the strategy will use for trading.
- The `securityId` will define the id of the instrument the strategy will trade.
- The `orderQuantity` is the number of contracts the strategy will trade.
- The `emaPeriodSort` is the look back period of the shorter EMA indicator.
- The `emaPeriodLong` is the look back period of the longer EMA indicator.

In addition, the following two fields need to be defined:

- The `TimeSeries` object used by the exponential moving average indicators
- The `DifferenceIndicator` which will contain the different between the short and the long EMA

Next, the Java Constructor for the `EMAService` class needs to be created:

```
public EMAService() {

    setStrategyName("EMA");

    this.series = new TimeSeries(Period.minutes(1));
    this.series.setMaximumTickCount(this.emaPeriodLong);

    ClosePriceIndicator closePriceIndicator = new ClosePriceIndicator(this.series);
    EMAIndicator emaShort = new EMAIndicator(closePriceIndicator, this.emaPeriodShort);
    EMAIndicator emaLong = new EMAIndicator(closePriceIndicator, this.emaPeriodLong);
    this.emaDifference = new DifferenceIndicator(emaShort, emaLong);
}
```

- First the `EMAService` constructor sets the name of the Strategy used during the back test.
- Next the `TimeSeries` object is initialized to a length of one Bar. In addition, the number of bars the Time Series is set (in this case 20 Bars).
- Next a `ClosePriceIndicator` is created which causes the system to look at closing prices of Bar events.

- Then both the short and the long EMA indicator need to be created by associating them with the `ClosePriceIndicator` and setting the `lookback` period (in this case 10 and 20).
- Last the `DifferenceIndicator` needs to be created which contains the difference between the short EMA and the long EMA indicator.

Next, the `onStart` (an `AlgoTrader` Live Cycle Method) method needs to be created which will be called once the strategy starts up.

```
@Override
public void onStart(final LifecycleEventVO event) {
    getSubscriptionService().subscribeMarketDataEvent(getStrategyName(), this.securityId);
}
```

For further details please visit the `AlgoTrader` documentation regarding [Life Cycle Events](#)².

The `onStart` methods calls `subscribeMarketDataEvent` of the `SubscriptionService` by passing the `strategyName` and the `securityId` of the instrument the strategy wants to receive market data for. The `SubscriptionService` is automatically made available to the strategy through Spring Auto Wiring.

Next, the `onBar` method needs to be created which will be invoked on every incoming `Bar`:

```
@Override
public void onBar(BarVO bar) {

    addBar(bar);

    int i = this.series.getEnd();
    Decimal currentValue = this.emaDifference.getValue(i);
    Decimal previousValue = this.emaDifference.getValue(i - 1);

    if (currentValue.isPositive() && previousValue.isNegativeOrZero()) {
        sendOrder(Side.BUY);
    } else if (currentValue.isNegative() && previousValue.isPositiveOrZero()) {
        sendOrder(Side.SELL);
    }
}
```

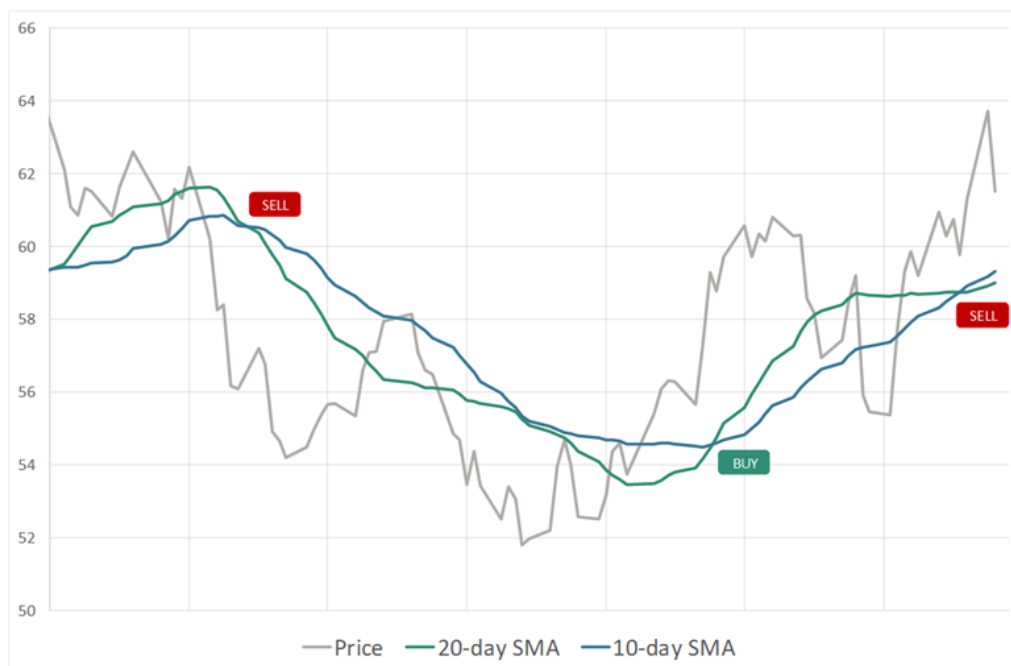
- The method first calls the `addBar` method which will add the incoming `Bar` to the Time Series defined above
- Next, the index `i` of the last element of the Time Series is retrieved
- Then the value of the last and the second-last element of the `DifferenceIndicator` is retrieved

Then the actual trading rules need to be defined:

² http://doc.algotrader.ch/html/Strategy_Life_Cycle_Events.html

- If the current value of the `DifferenceIndicator` is positive and the previous value was negative or zero a `BUY` order is sent. In other words, if the short EMA crossed above the long EMA a `BUY` order is sent.
- If the current value of the `DifferenceIndicator` is negative and the previous value was positive or zero a `SELL` order is sent. In other words, if the short EMA crossed below the long EMA a `SELL` order is sent.

The trading logic is depicted in the following chart also.



The `addBar` method simply converts `AlgoTrader BarVO` events into `TA4j` objects and adds them to the `Time Series`

```
private void addBar(BarVO bar) {

    Tick tick = new Tick(new DateTime(bar.getDateTime().getTime()),
        bar.getOpen().doubleValue(),
        bar.getHigh().doubleValue(),
        bar.getLow().doubleValue(),
        bar.getClose().doubleValue(),
        bar.getVol().doubleValue());

    this.series.addTick(tick);
}
```

As the last item, the `sendOrder` method needs to be created which will take care of constructing an order object and handing it over to the `OrderService`:

```
private void sendOrder(Side side) {
```

```
MarketOrderVO order = MarketOrderVOBuilder.create()
    .setStrategyId(getStrategy().getId())
    .setAccountId(this.accountId)
    .setSecurityId(this.securityId)
    .setQuantity(new BigDecimal(this.orderQuantity))
    .setSide(side)
    .build();

getOrderService().sendOrder(order);
}
```

The `sendOrder` method creates a `MarketOrder` by using the `MarketOrderVOBuilder` and assigns the `strategyId`, the `accountId`, the `securityId`, the `orderQuantity`, the order side (BUY or SELL) and finally calls `build` to create the `MarketOrder` object. The order object is then handed over to the `OrderService` which will execute the order. The `OrderService` is automatically made available to the strategy through Spring Auto Wiring.

For further details on how order are please visit the AlgoTrader documentation regarding [Order Management](#)³

The implementation of the trading strategy is now finished a first back test can be started according to [Chapter 3, Starting a Trading Strategy](#).

The EMA strategy is an example strategy based on Java code only. For details on how to build a trading strategy using Esper please visit the AlgoTrader documentation regarding [Strategy Development](#)⁴

³ <http://doc.algotrader.ch/html/OrderManagement.html>

⁴ http://doc.algotrader.ch/html/Strategy_Development.html

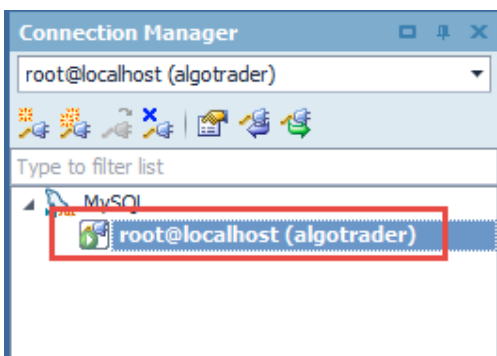
Managing data

During live trading, all relevant information like orders, positions and transactions are stored in the MySQL database.

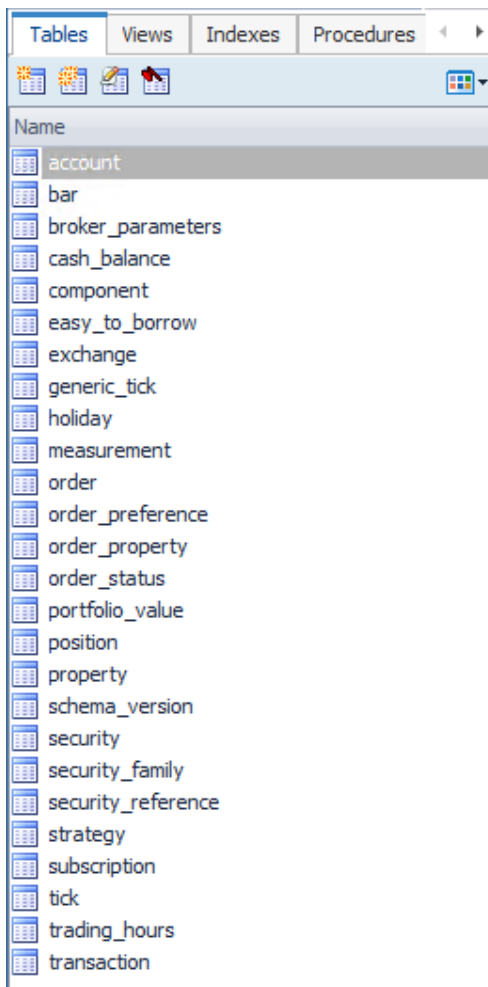
To view database data please open TOAD for MySQL.



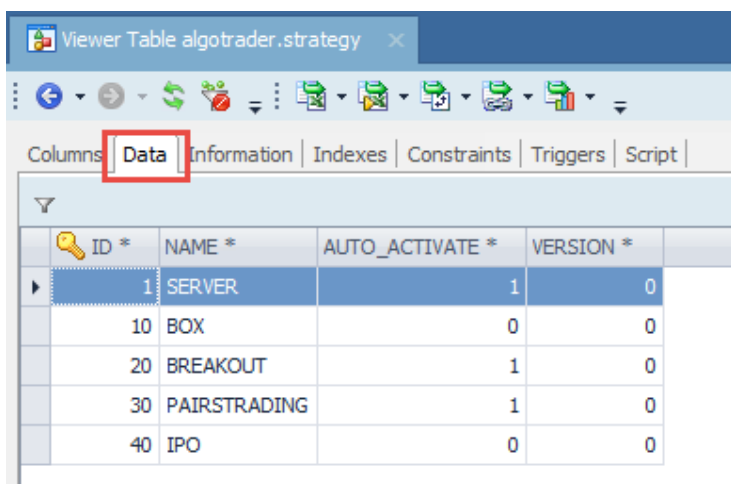
On the left-hand side of the application double click on `root@localhost (algotrader)`.



You now see all AlgoTrader tables listed below



To view the contents of a table, double-click its name and go to the Data tab



5.1. Reference Data

Reference Data like instrument definitions, strategies etc. are also stored in Mysql (withing the tables like security, security_family, strategy, etc.)

Before an AlgoTrader based trading strategy can trade a particular instrument in needs to be defined in the database.



Note

- The AlgoTrader fee 30-day trial version is pre-configured with sample reference data for commonly used FX pairs, US Equities and Futures.
- Per default AlgoTrader uses an in-memory H2 database instead of MySQL during Back Testing. Same as with the MySQL data the AlgoTrader 30-day trial version already contains sample reference data for the in-memory H2 database.

For further details visit the AlgoTrader documentation regarding [Reference Data](#)¹.

5.2. Historical Data

For Back Testing AlgoTrader can use historical data provided by `.csv` files. For the EMA strategy the AlgoTrader Strategy Wizard created a sample historical bar data file `/algotrader-ema/files/bardata/fx/EURUSD.csv`. The file name (`EURUSD`) needs to match the `symbol` column in the database table `security` of the instrument the strategy is going to trade.

For further details on naming conventions and the location of historical data `.csv` files see the AlgoTrader documentation regarding [Market Data File Format](#)².

As a more sophisticated alternative to providing historical data through `.csv` files, the Time Series database [InfluxDB](#)³ can be used for storage and retrieval of historical data. For further details on downloading, storing and using InfluxDB data for back testing please visit the AlgoTrader documentation on [Historical Data](#)⁴.

¹ http://doc.algotrader.ch/html/Reference_Data.html

² http://doc.algotrader.ch/html/Market_Data.html#Market_Data_File_Format

³ <https://www.influxdata.com/time-series-platform/influxdb/>

⁴ http://doc.algotrader.ch/html/Historical_Data.html

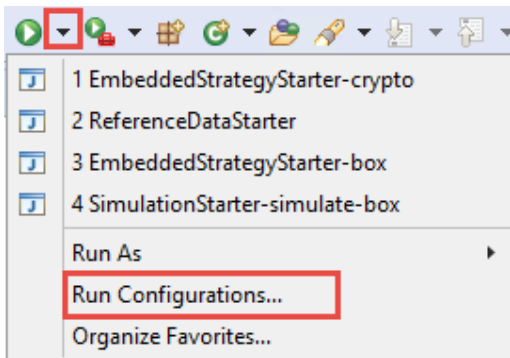
Cryptocurrency Trading

The AlgoTrader 30-day trial version can also be used to trade Bitcoin and other Cryptocurrencies via [Coinigy](#)¹

To setup a connection to Coinigy the following steps have to be taken:

- Sign-up for a Coinigy account on [Coinigy Sign up](#)²
- Enable two factor authentication (2FA) on the account following the [2FA Instructions](#)³
- In the API accounts settings add the API keys from all of the exchanges where an account is setup according to these [Instructions](#)⁴
- In the account preferences generate a new Coinigy API key and Secret Key and use it in the settings below
- In the account preferences click the button 'Click to reveal my Private Channel ID (Websocket API)' and use it in the settings below

First the Coinigy API key, Secret Key and Private Channel ID noted above needs to be added by clicking the downward facing arrow next to the green start icon and then selecting `Run Configurations`.



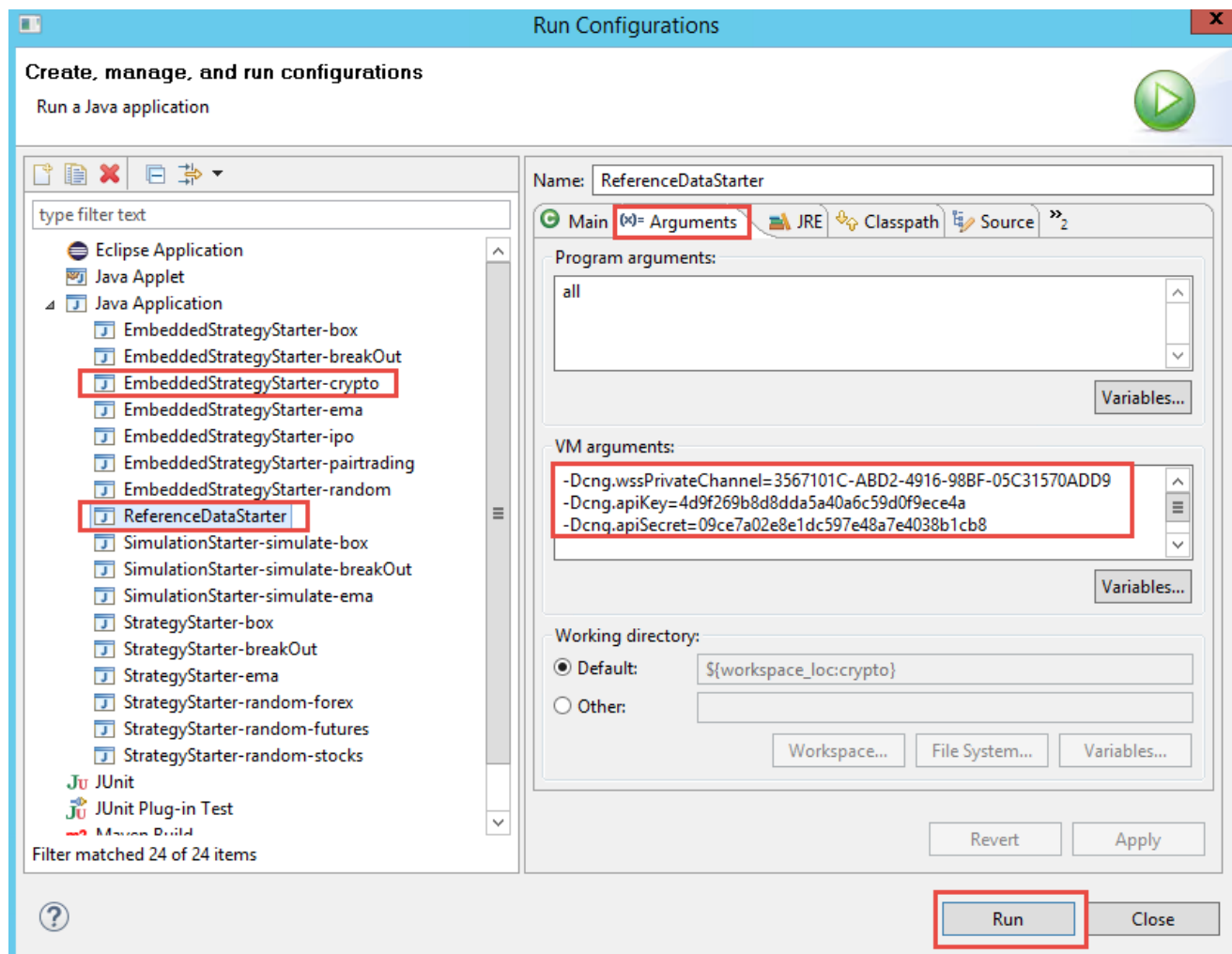
Then update both the `EmbeddedStrategyStarter-crypto` and the `ReferenceDataStarter` by selecting them on the left-hand side and going to the `Arguments` Tab

¹ <https://www.coinigy.com/>

² <https://www.coinigy.com/auth/signup>

³ <https://coinigy.freshdesk.com/support/solutions/articles/1000085817-how-do-i-enable-two-factor-authentication-2fa-on-my-account->

⁴ <https://coinigy.freshdesk.com/support/solutions/articles/1000087506-how-do-i-add-an-api-key-to-coinigy->



Here the Coinigy API key, Secret Key and Private Channel ID noted above needs to be added. Then click Apply.

In order to populate the database with Coinigy Accounts, Exchanges, Security Families and Securities select the `ReferenceDataStarter` and click Run.

Now the strategy can be started by selecting the `EmbeddedStrategyStarter-crypto` and click Run.

This will start the strategy and subscribe to the BTC.USD pairs on the following three exchanges:

- BITFINEX
- BTC-E
- POLONIEX

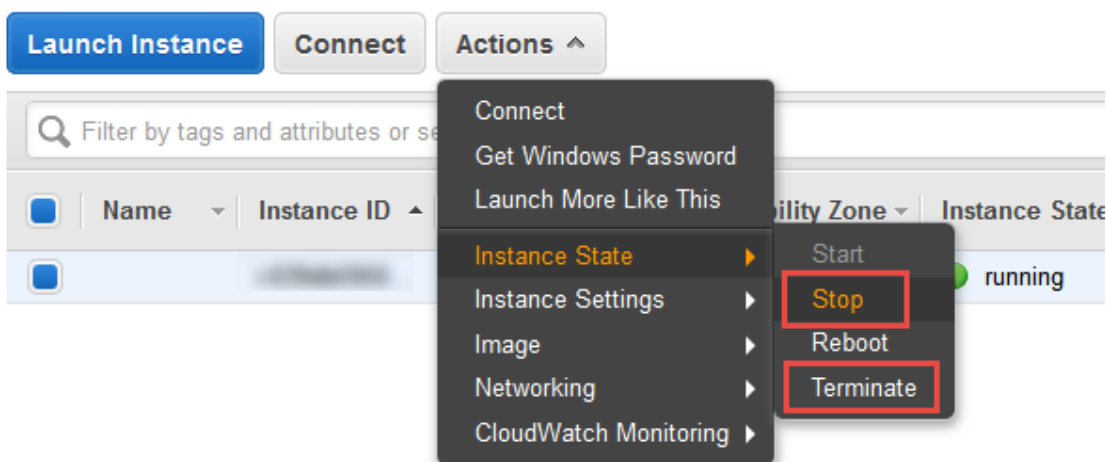
Other than that the strategy has no further logic and it is left up to the user to add trading rules according to [Chapter 4, Creating a Trading Strategy](#).

Shutting down the System

You can shut down the system by clicking on the Start Menu in the lower left-hand corner of the Windows Desktop and then select `Power Options` in the upper right-hand corner:



Alternatively, the system can be shutdown via the Amazon AWS Console <https://console.aws.amazon.com/> by first selecting the Amazon Instance and the under `Actions` select `Instance State` and then either `Stop` or `Terminate`



Note

- If `Stop` is clicked the instance can be restarted at a later point in time. In the stopped state the Amazon Instance will still incur disc space using costs as mentioned in: <https://aws.amazon.com/ec2/pricing/>
- If `Terminate` is clicked the instance cannot be restarted. In the terminated state, no further Amazon instance costs apply