Quick Start Guide

AlgoTrader

Version 4.5

by Andy Flury and Robert Kolar

Table of Contents

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
- Tortoise Git
- Git for Windows
- Notepad++

[1] https://aws.amazon.com/
[2] https://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

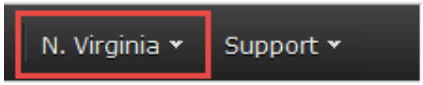
The following steps will guide through the installation of a Windows AWS Instance containing the AlgoTrader 30-day trial version

1. Open the Amazon AWS console:

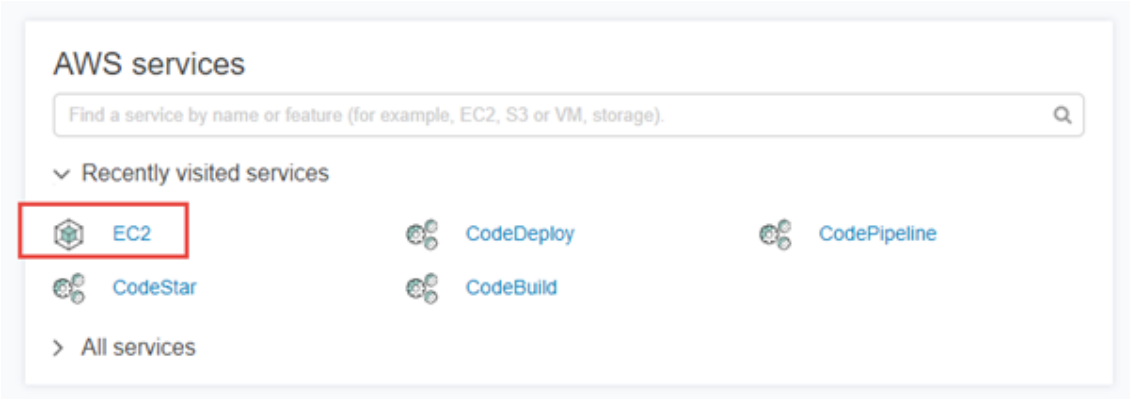
   https://console.aws.amazon.com/console/home

   and Login using the Amazon `username` and `password`.

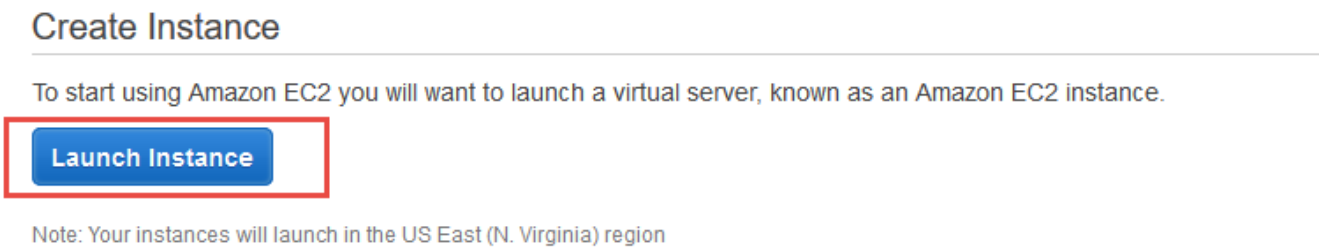
2. In the upper right-hand corner of the screen make sure the `N. Virginia` Region is selected:



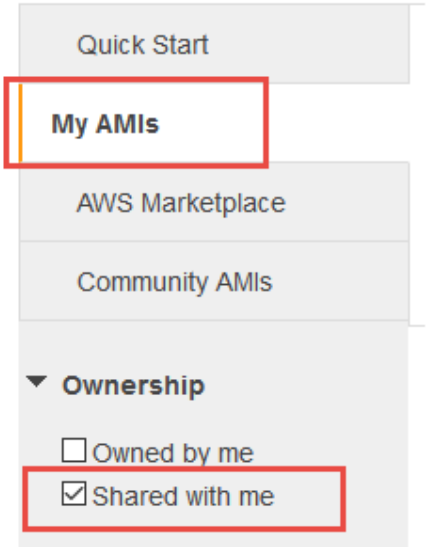
3. Select `EC2`



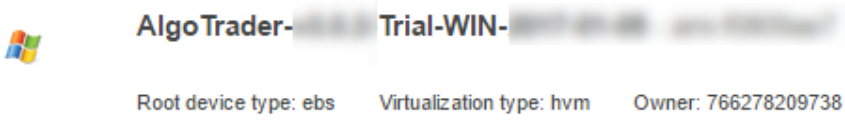
4. Click `Launch Instance` in the middle of the screen



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`



> **Important**
>
> 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

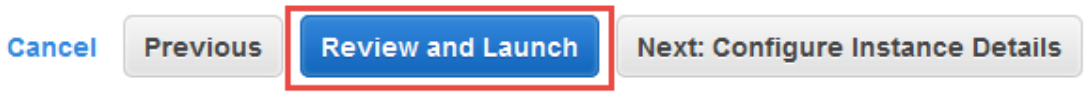
7. On the next screen select the `Instance Type`. We recommend at least instance type `t2.medium`, ideally instance type `m5.large`

Type	vCPUs	Memory (GiB)
t2.medium	2	4
m5.large	2	8

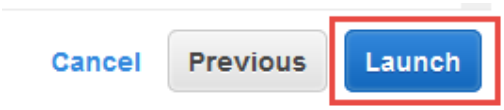
Warning

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

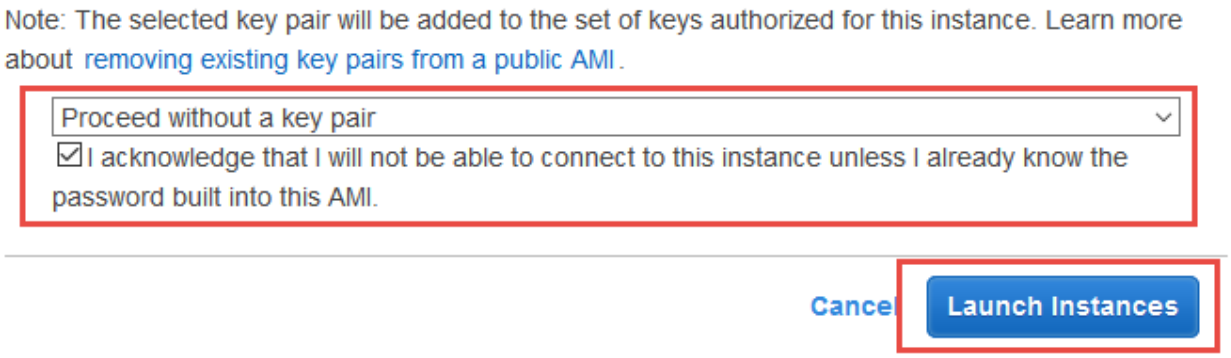
8. Click `Review and Launch` on the bottom right of the screen



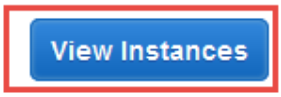
9. Click `Launch` on the bottom right of the screen



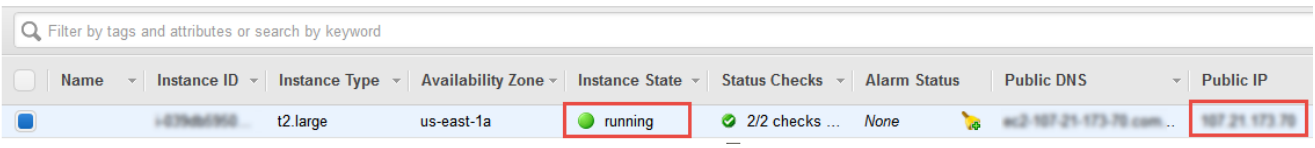
10. On the Dialog that shows select `Proceed without a key pair`, select `I acknowledge...` and click `Launch Instance`



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

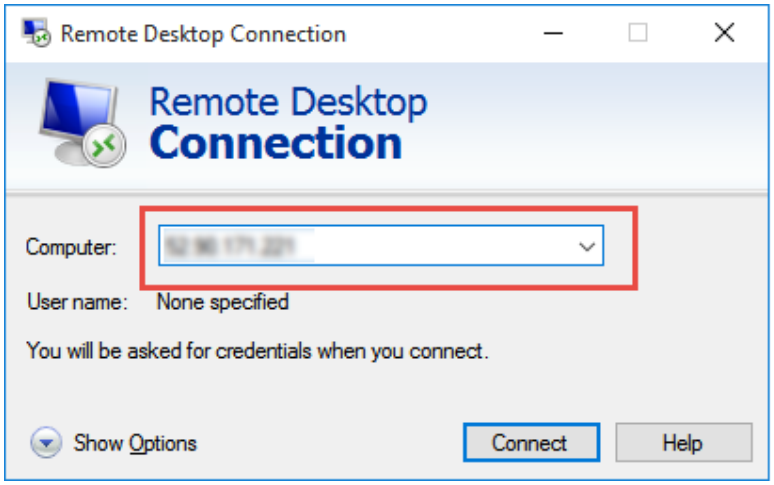


12. On the next screen, you can see the Instance starting up. Wait until it is running and note the `Public IP` address.



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

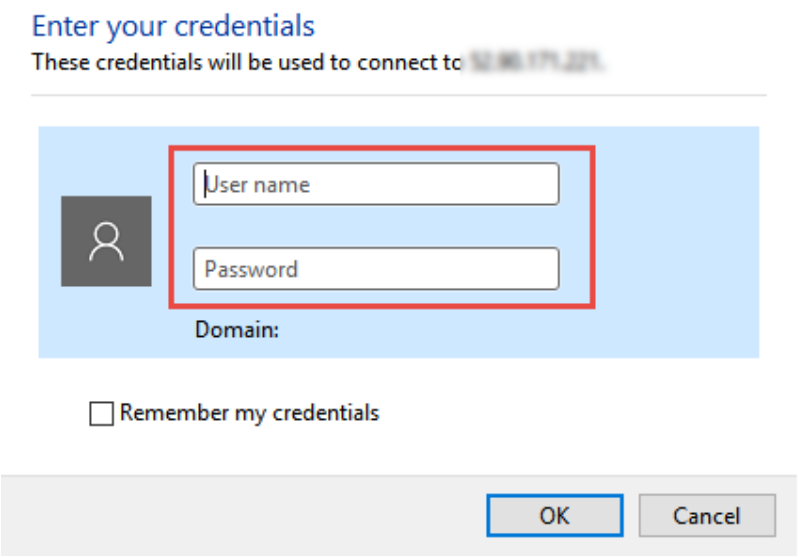
13. Use Microsoft Remote Desktop to connect to the instance by typing in the Public IP address that was noted in the previous step



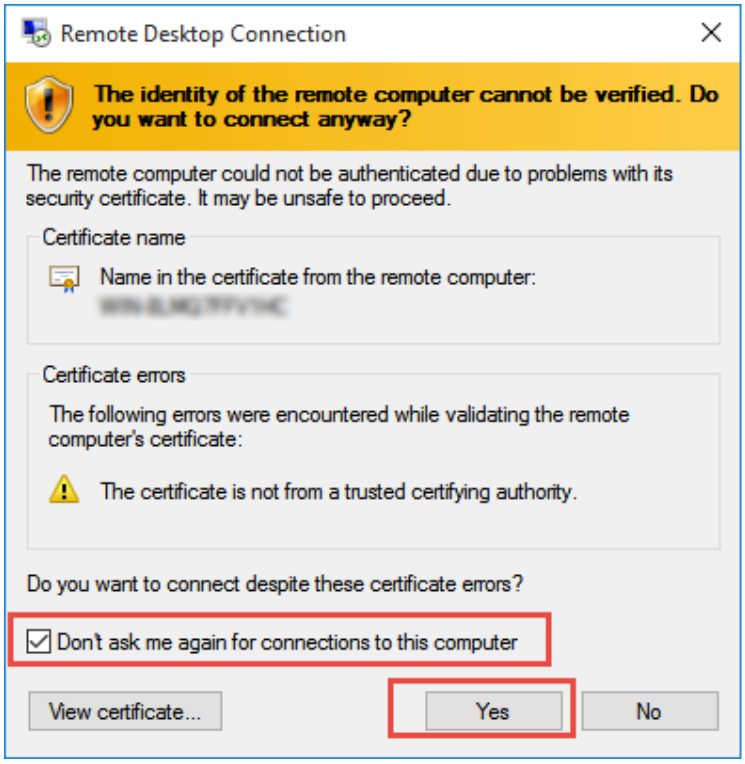
> **Note**
>
> - You need to use Remote Desktop Connection (RDP), a Web Browser will NOT work for this
> - For more information on Remote Desktop Connections please visit: *Windows*[1], *Mac*[2] and *Linux*[3]

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

[1] https://support.microsoft.com/en-us/help/4028379/windows-10-how-to-use-remote-desktop
[2] https://itunes.apple.com/us/app/microsoft-remote-desktop-10/id1295203466?mt=12
[3] http://www.rdesktop.org/



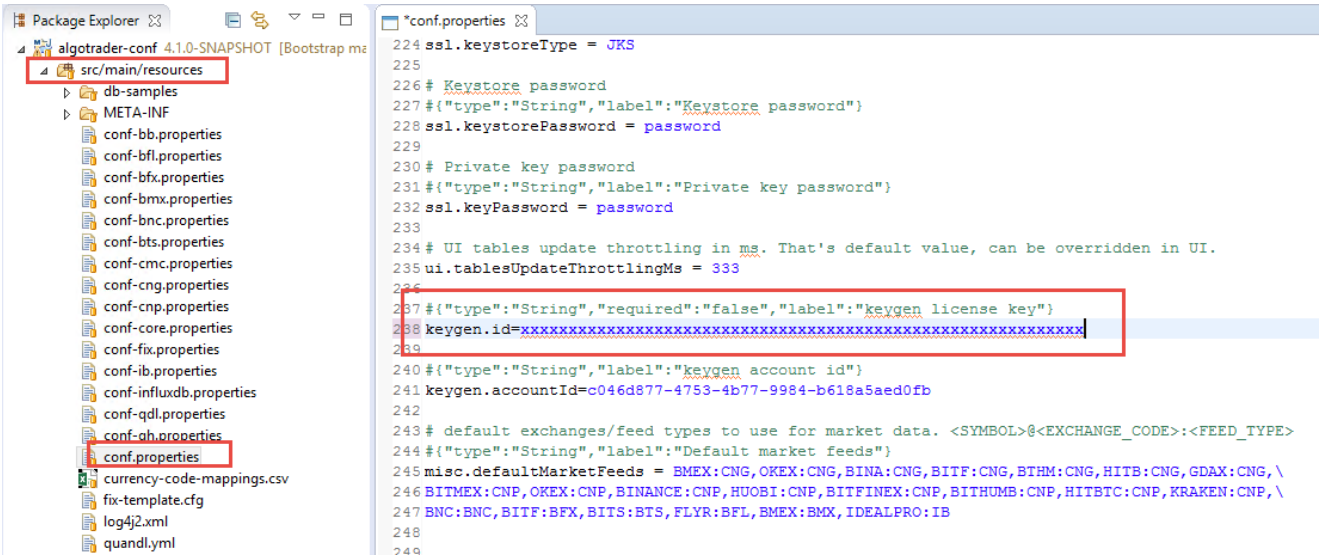
15. On the next dialog select `Don't ask...` and click `Yes`



16. 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 active Microsoft Excel using the license key. Alternatively, one can request a free 60-day trial through:

    https://www.microsoft.com/en-us/evalcenter/

17. Copy the license key that was provided in the Email after signing up for the AlgoTrader free 30-day trial into the file `/algotrader-conf/src/main/resources/conf.properties`.



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

Note

- Amazon Windows Instances tend to run a bit slow when 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.

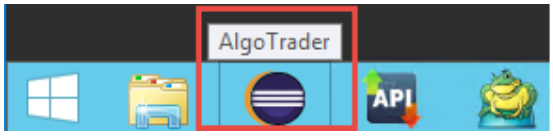
Starting a Trading Strategy

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.

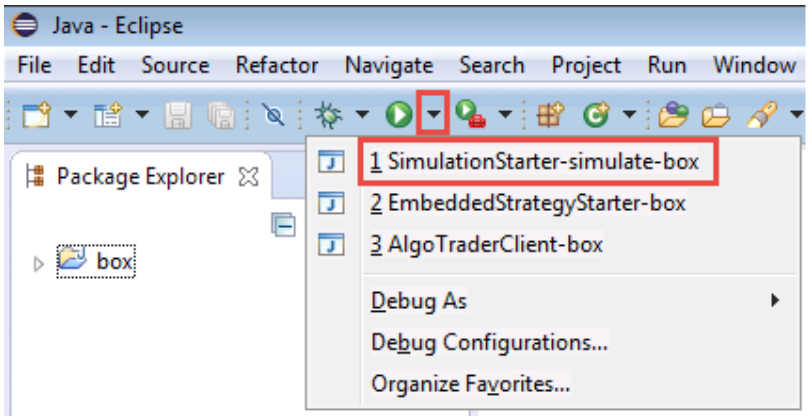


3.1. Starting a Back Test

The back-test in this example strategy runs with CSV files. If you have a historical data provider that AlgoTrader supports, you could also run vs. InfluxDB and would have to load the historical data first by running a `HistoricalDataStarter` (see *Historical Data*[7]).

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

[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
[7] http://doc.algotrader.ch/html_single/index.html#Historical_Data



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.

ALGOTrader Refresh **AlgoTrader Backtest Report**

Master Data	
Date	04.01.2016
Time	13:00:05
Execution Time (min)	0.4063
Data Set	eurusd
Net Liquid Value	2'229'714

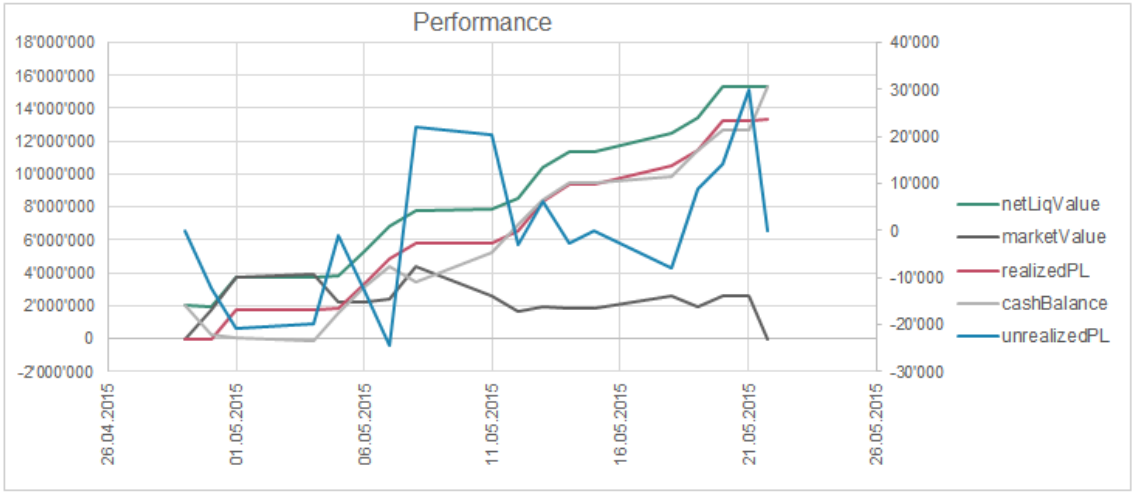
Performance	
Monthly Average Return	0.78%
Monthly Standard Deviation	0.93%
Annual Average Return	9.77%
Annual Standard Deviation	3.23%
Sharpe Ratio	3.02
Colmar Ratio	4.14

Monthly / Yearly Return	
Positive Months	10
Negative Months	4
Positive Years	2
Negative Years	1

Draw Down	
Worst Monthly Draw Down	1.03%
Best Monthly Performance	2.02%
Worst Draw Down (%)	2.36%
Worst Draw Down (days)	46

Trades	Winning	Losing	All
Count	428	365	793
%	54%	46%	100%
Total Profit	1'277'201.43	-1'047'487	229'714.09
Average Profit	2'984.12	-2'869.83	289.00
Average Profit (%)	0.23%	-0.26%	15.00%

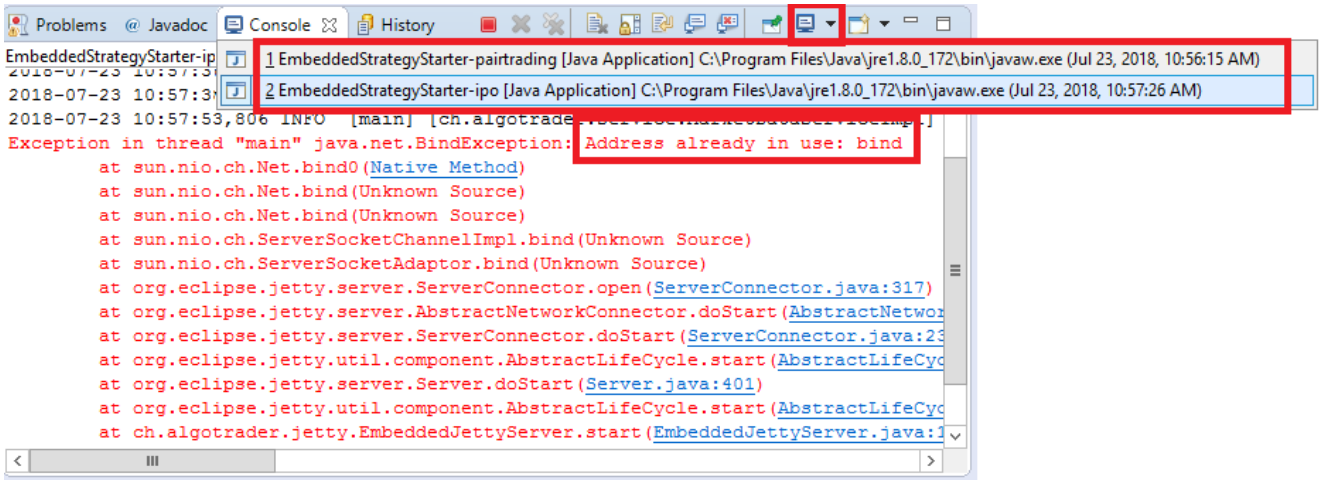
%	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Year
2'011												0.29	0.29
2'012	2.02	1.36	-0.10	1.45	-1.03	1.27	1.70	0.35	-0.31	1.96	1.25	0.78	11.20
2'013	-0.03												-0.03



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

- *Strategy Backtesting*[8]
- *Starting a Strategy in Simulation Mode*[9]

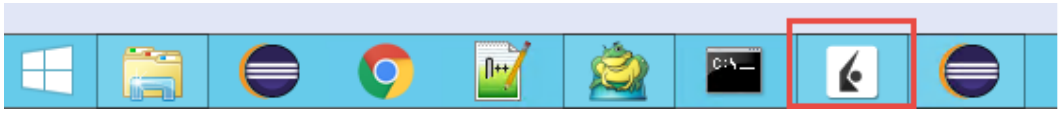
> **Note**
>
> In case the console shows an error message like "Address already in use" this means that there is already an instance of AlgoTrader running. Please check the list of currently running process and stop the existing AlgoTrader instances before starting a new one.



3.2. Starting Live Trading

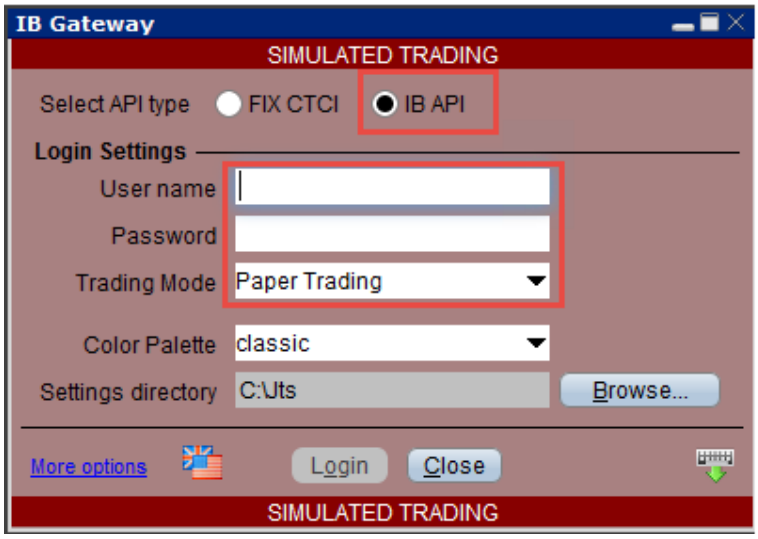
The following sections describes how to use AlgoTrader in Live Trading mode by using Interactive Brokers for market data and trading.

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



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

[8] http://doc.algotrader.ch/html/Strategy_Backtesting.html
[9] http://doc.algotrader.ch/html/Starting_AlgoTrader.html#Simulation_Mode



Warning

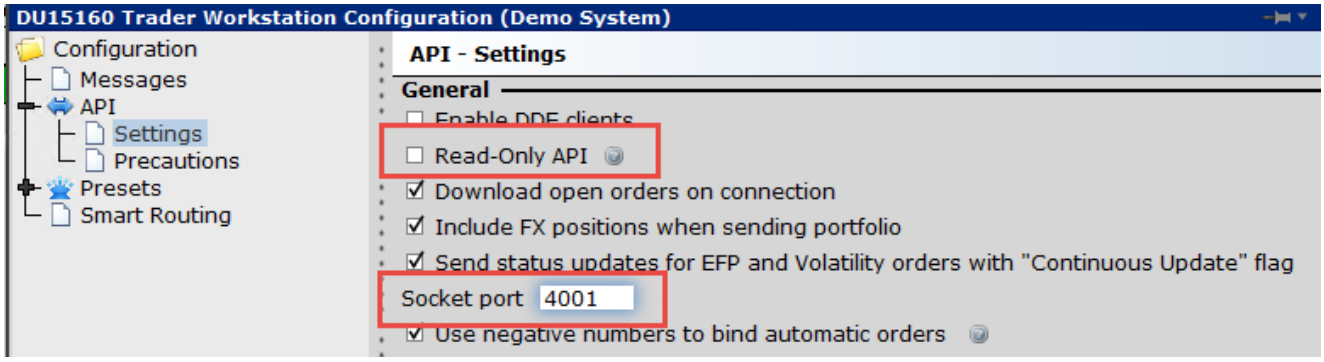
Do not use a live account until absolutely sure that the trading strategy works as expected. Until then use an IB paper trading account.

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: `pmdemo`
- 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`.



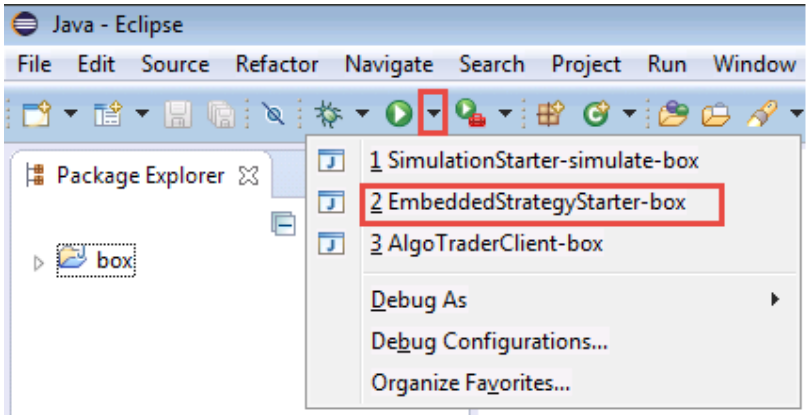
Note

In case the Interactive Broker Gateway shows a blank window this means that your screen resolution is probably lower than the size of the Interactive Brokers Gateway. In this case please try the keyboard combination **Alt+Space Bar** followed by **R**. Then repeat **Alt+Space Bar** followed by **M** for move or **S** for resize.

For further information please see *this article*[10].

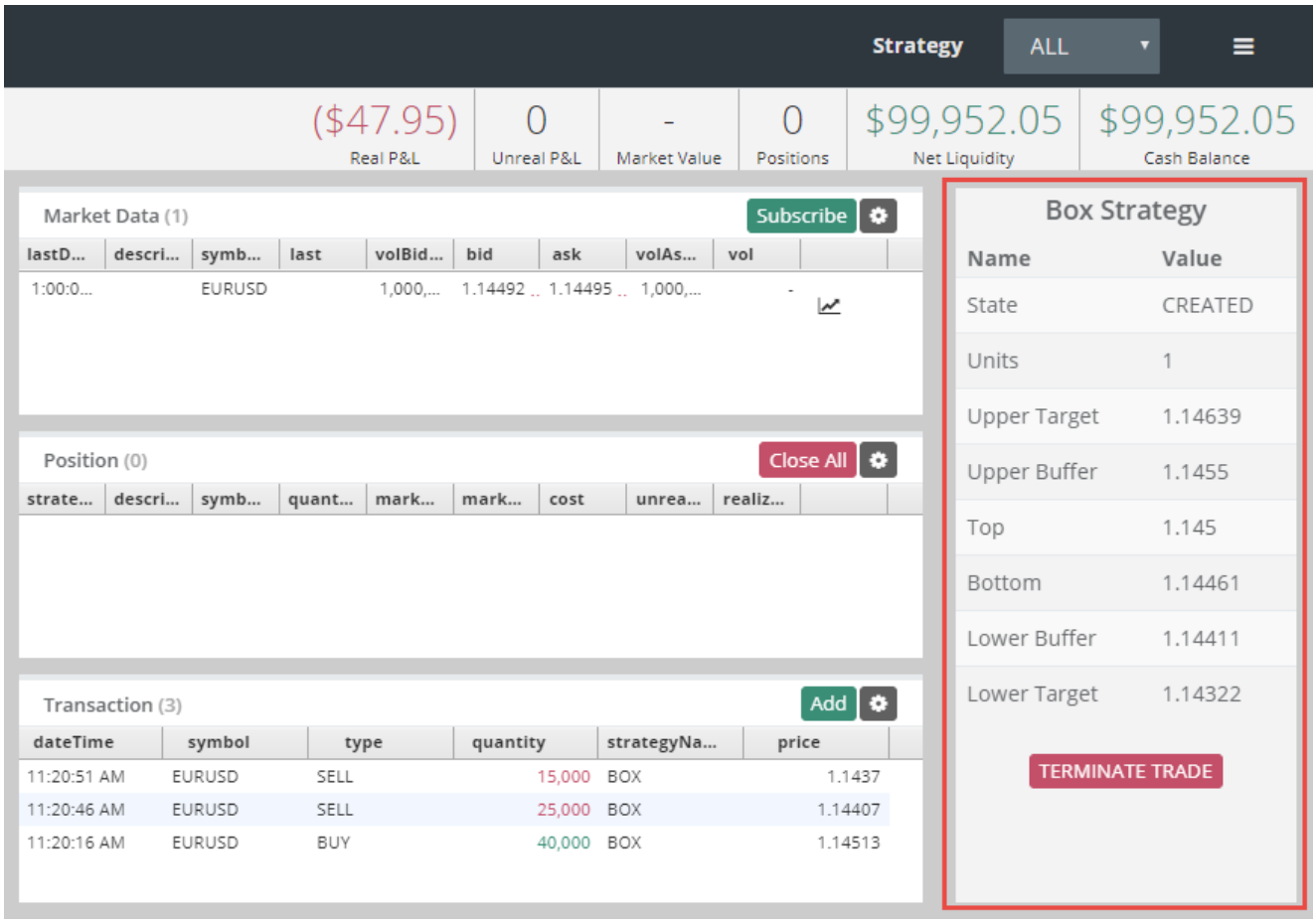
If the above does not solve your problem, try to execute **Alt + Spacebar** and tap 4 times on the down arrow on the keyboard and hit **Enter** .

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 front-end (within the Chrome browser).

[10] https://www.techsupportalert.com/content/how-use-keyboard-move-or-resize-window-too-big-screen.htm



Note

In case the frame on the right side of the screen named `Box Strategy` does not show up, please click `Shift + Reload`

For further information on live trading, please visit *Starting a Strategy in Live Trading Mode*[11]

For further information on the AlgoTrader Web Front-end please visit the AlgoTrader documentation regarding the *HTML5 client*[12]

[11] http://doc.algotrader.ch/html/Starting_AlgoTrader.html#Live_Trading_Mode
[12] http://doc.algotrader.ch/html/Client.html#HTML5_Client

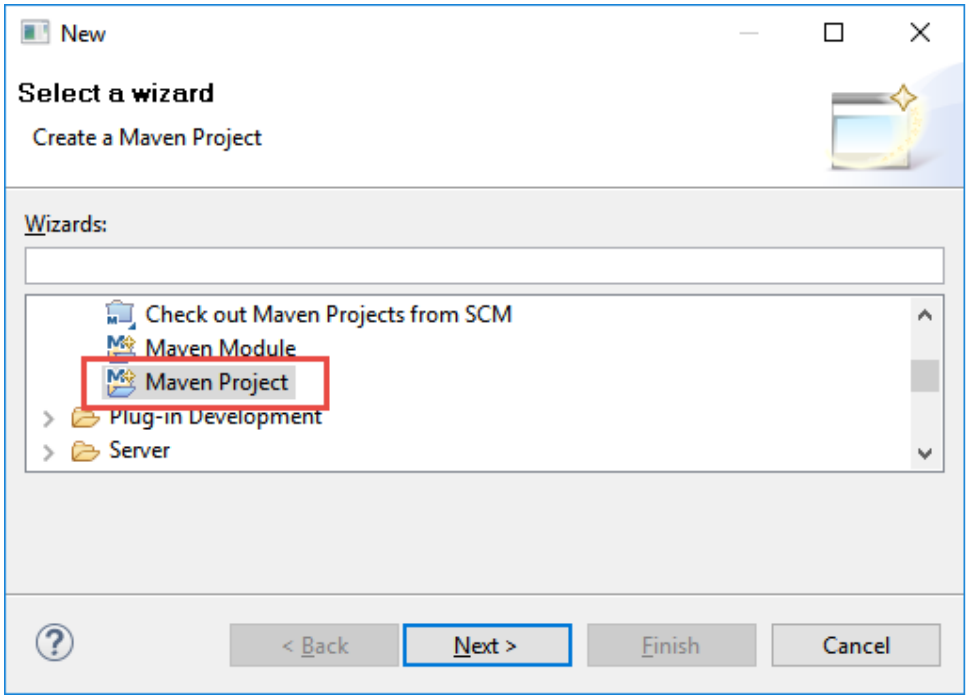
Creating a Trading Strategy

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

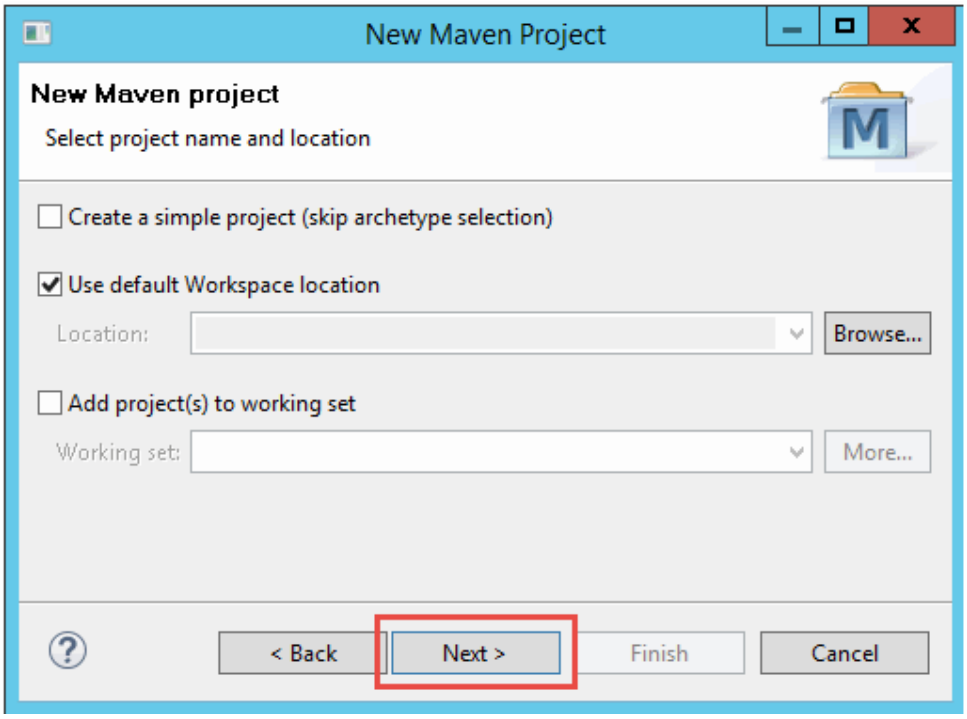
> **Note**
>
> The AlgoTrader 30-day free trial already contains the final EMA strategy with all artifacts. In case you want to follow below steps please delete the existing EMA strategy first.

4.1. AlgoTrader Strategy Wizard

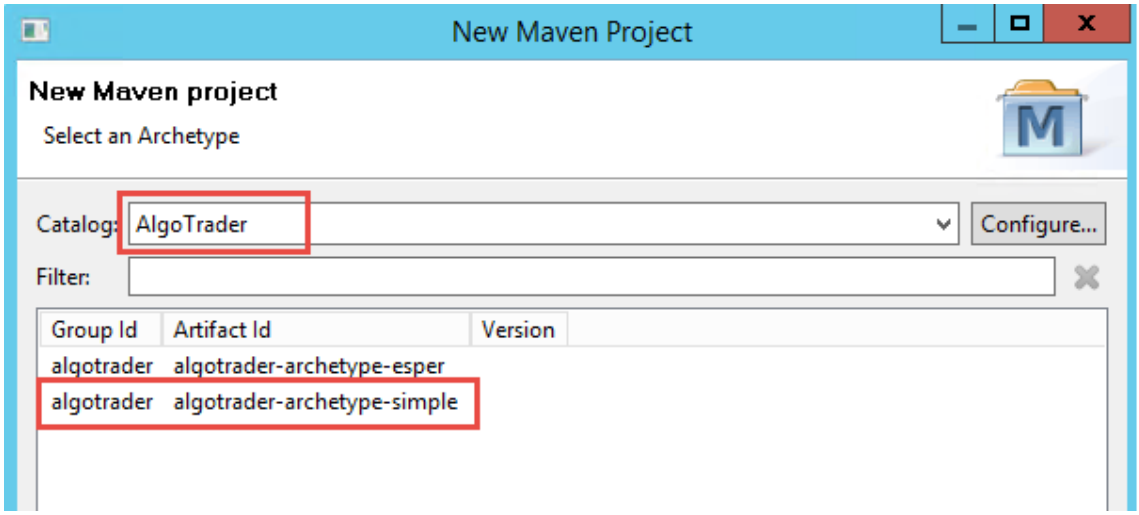
The AlgoTrader Strategy Wizard provides an easy way to automatically create all artifacts necessary for an AlgoTrader based trading strategy. The Wizard can be started via the `File / New / Other` which will bring up the following screen where the Maven Project wizard can be selected:



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. `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.

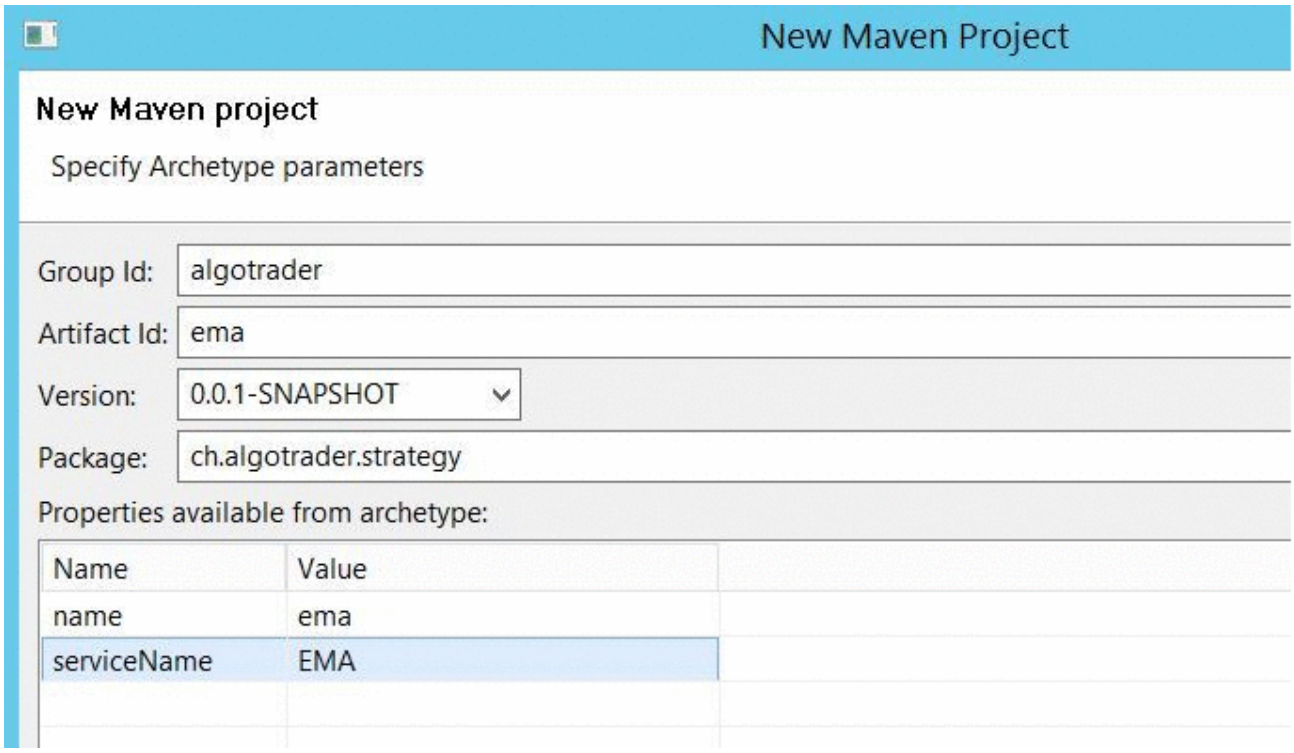
name

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

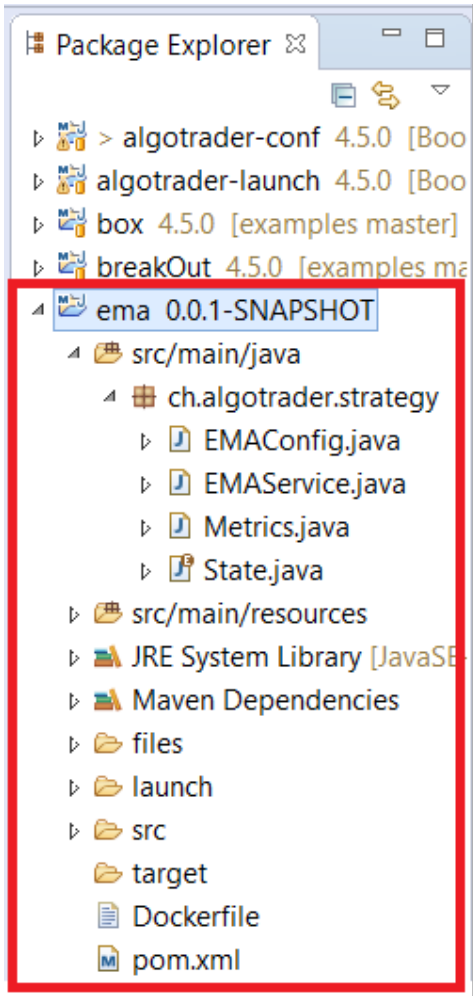
serviceName

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.



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



4.2. Adding Strategy Logic

The Strategy Wizard also generated 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 *TA4J*[1] library which contains a collection of over 100 technical indicators.

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`.

[1] https://github.com/mdeverdelhan/ta4j-origins

> **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 = 204;
private final long securityId = 860;
private final BigDecimal orderQuantity = new BigDecimal("0.002");
private final int emaPeriodShort = 10;
private final int emaPeriodLong = 20;
private final String defaultFeedType = "BNC";

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.
- The `defaultFeedType` indicates we want to get market data from Binance by default.

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 difference 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 BaseTimeSeries();
        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 `lookbackPeriod` (in this case 10 and 20).
- Last the `DifferenceIndicator` needs to be created which contains the difference between the sort EMA and the long EMA indicator.

Next, update the `onStart` (an AlgoTrader Live Cycle Method) method, which will be called when the strategy starts up.

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

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, update the `onBar` method, which will be invoked on every incoming Bar:

```
@Override
```

[2] http://doc.algotrader.ch/html_single/index.html#Strategy_Life_Cycle_Events

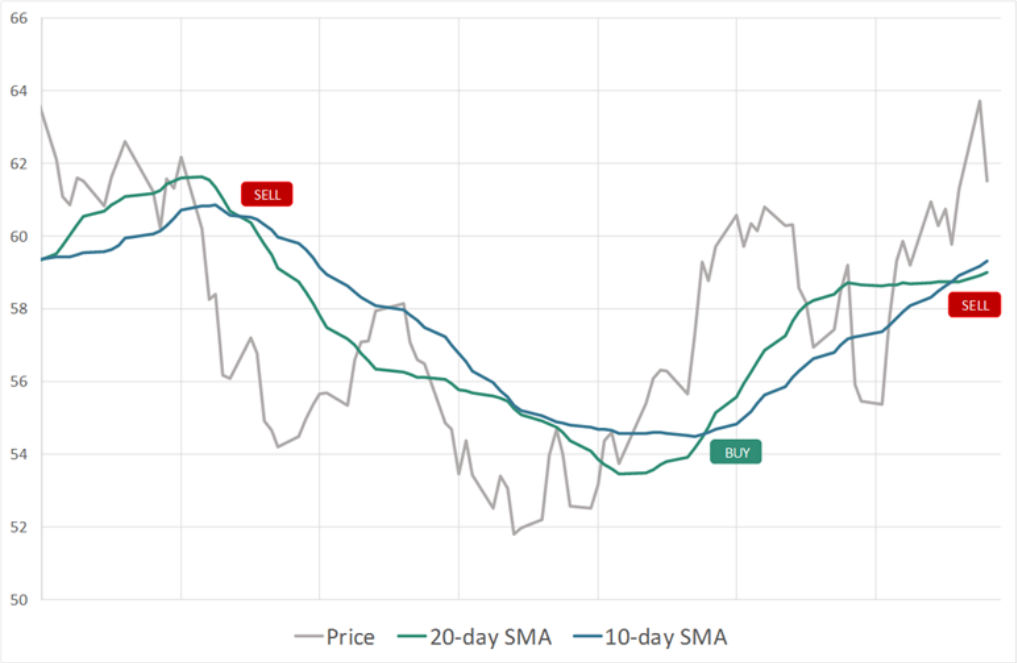
```
public void onBar(BarVO bar) {

    this.series.addTick(toTick(bar));

    int i = this.series.getEndIndex();
    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 `addTick` 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:

- 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.



As the last item, create the `sendOrder` method, 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(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]

In addition the following Java import statements need to be added to the top:

```
import org.ta4j.core.BaseTimeSeries;
import org.ta4j.core.Decimal;
import org.ta4j.core.TimeSeries;
import org.ta4j.core.indicators.EMAIndicator;
import org.ta4j.core.indicators.helpers.ClosePriceIndicator;
import org.ta4j.core.indicators.helpers.DifferenceIndicator;
import static ch.algotrader.util.TA4JUtil.toTick;
```

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]

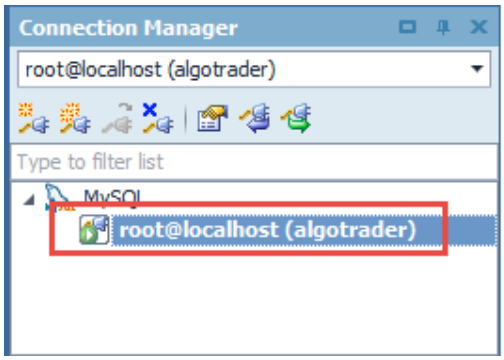
[3] http://doc.algotrader.ch/html/OrderManagement.html
[4] http://doc.algotrader.ch/html/Strategy_Development.html

Managing data

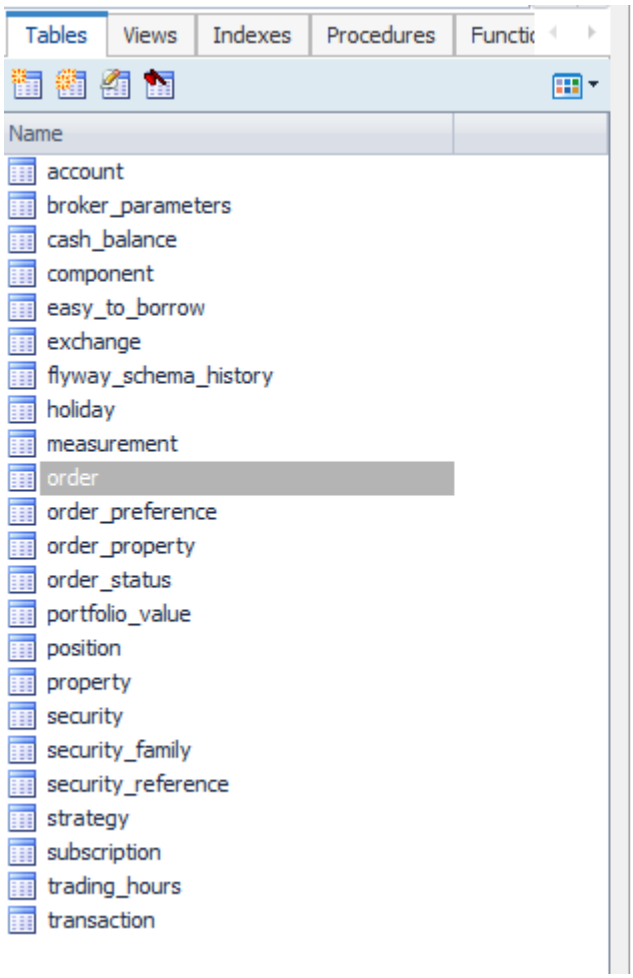
During live trading, all relevant information like orders, positions and transactions are stored in the MySql database.

To view database data please open `TOAD for MySql`.

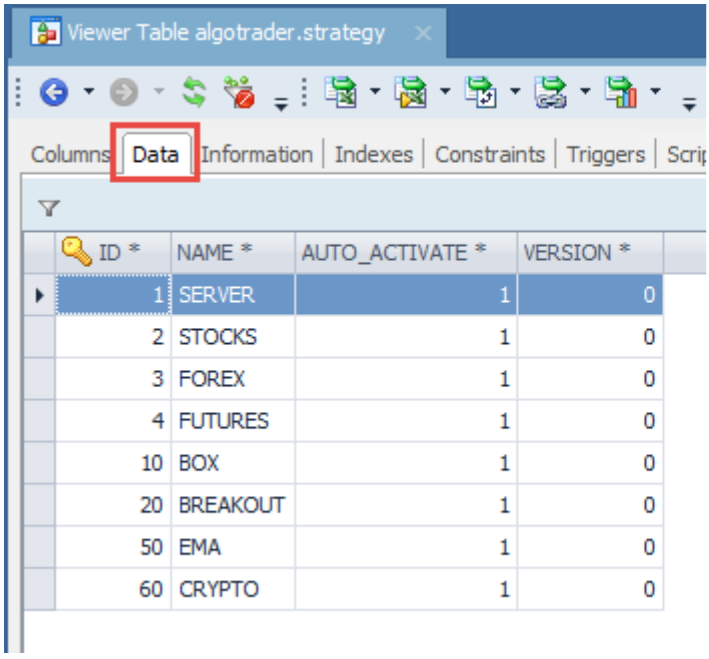
On the left-hand side of the application double click on `root@localhost (algotrader)`.



You now see all AlgoTrader tables listed below



To view the contents of a table (e.g. the `strategy` table), double-click its name and go to the `Data` tab



ID *	NAME *	AUTO_ACTIVATE *	VERSION *
1	SERVER	1	0
2	STOCKS	1	0
3	FOREX	1	0
4	FUTURES	1	0
10	BOX	1	0
20	BREAKOUT	1	0
50	EMA	1	0
60	CRYPTO	1	0

The table `security` contains a list of all available instruments that can be traded with the system. This table can for example be used to find the `securityId` for the `BTC/USDT` pair traded on Binance.

ID *	CLASS	SYMBOL	DESCRIPTION
1229	Forex...	BTCUSD	BTC/USD@BITF
1487	Forex...	BTCUSD	BTC/USD@FLYR
1497	Forex...	BTCUSD	BTC/USD@BMEX
1549	Forex...	BTCUSD	BTC/USD@BITS
860	Forex...	BTCUSDT	BTC/USDT@BINA

5.1. Reference Data

Reference Data like instrument definitions, strategies etc. are also stored in MySQL (in 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 & European Equities, Futures & Cryptocurrencies.
- 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].

[1] http://doc.algotrader.ch/html/Reference_Data.html
[2] http://doc.algotrader.ch/html/Market_Data.html#Market_Data_File_Format

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].

[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 the following exchange adapters:

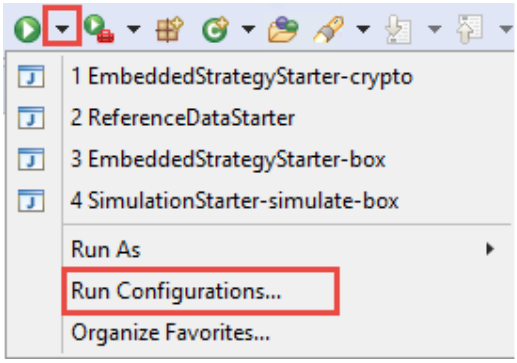
- *Binance*[1]
- *Bitfinex*[2]
- *Bitstamp*[3]
- *Bitflyer*[4]
- *BitMEX*[5]
- *Coinigy*[6]

This chapter describes how to setup trading with the Binance exchange.

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

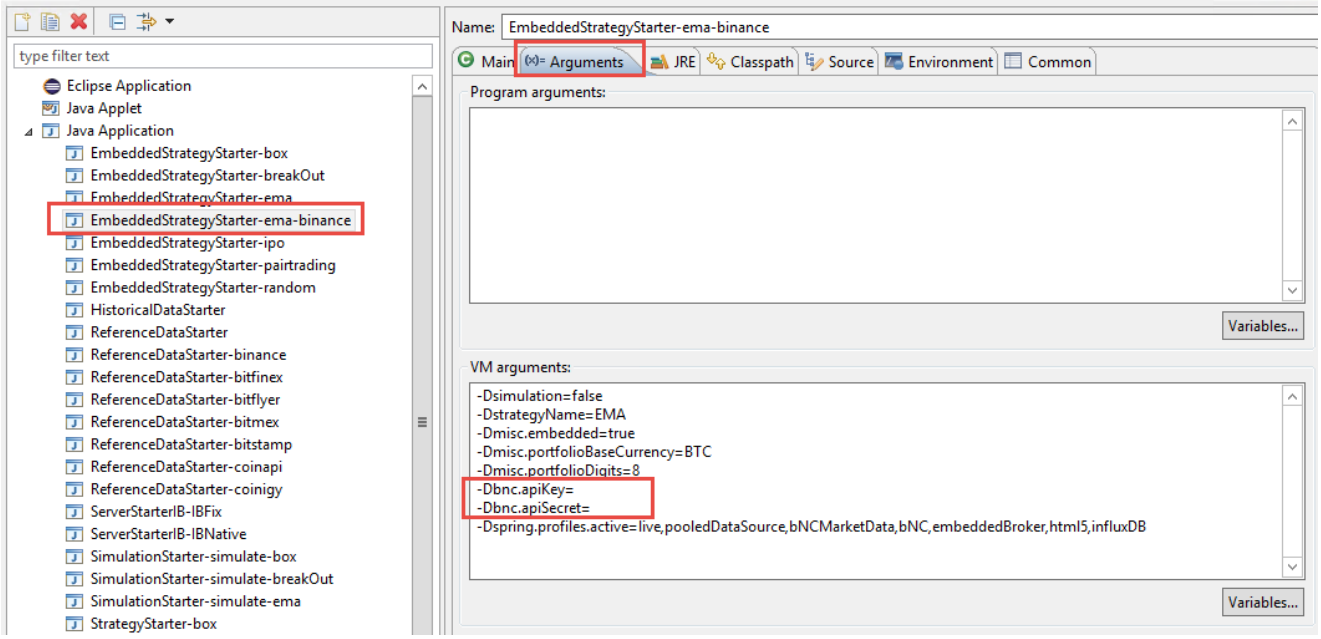
- Sign-up for a Binance account on *Register with Binance*[7]
- Enable two factor authentication (2FA) on the account following the 2FA instructions (either SMS or Google Authenticator) on the *Account Page*[8]
- On the account page generate a new Binance API key and Secret Key and use it in the settings below.

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



[1] http://binance.com
[2] http://bitfinex.com
[3] https://bitstamp.net
[4] https://bitflyer.com
[5] https://bitmex.com
[6] https://www.coinigy.com
[7] https://www.binance.com/register.html
[8] https://www.binance.com/userCenter/myAccount.html

In this example we are going to use the existing EMA strategy to trade via Binance. For this purpose the Binance API key and Secret Key need to be added to the Eclipse launch configuration `EmbeddedStrategyStarter-ema-binance` by selecting it on the left-hand side and going to the `Arguments` Tab



Here the Binance API key and Secret Key noted above need to be added. Then click `Apply`.

Per default the EMA strategy trades the `EUR/USD` currency pair through Interactive Brokers. To now switch the strategy to trade through Binance we need to update the settings at the top of the `EMAService`:

- Update the `accountId` to match the Binance account in the database.
- Update the `securityId` (`securityId` 860 represents the `BTC/USDT` cryptocurrency pair on Binance).
- Update the `orderQuantity` to a small enough number
- Update the `defaultFeedType` `BNC` in order for the strategy to subscribe for market data through the Binance adapter.
- Update the order quantity in the `SendOrder` method to match the new data type set above.

```
@Component
public class EMAService extends StrategyService {

    private final long accountId = 204;
    private final long securityId = 860;
    private final BigDecimal orderQuantity = new BigDecimal("0.002");
    private final int emaPeriodShort = 10;
    private final int emaPeriodLong = 20;
    private final String defaultFeedType = "BNC";
```

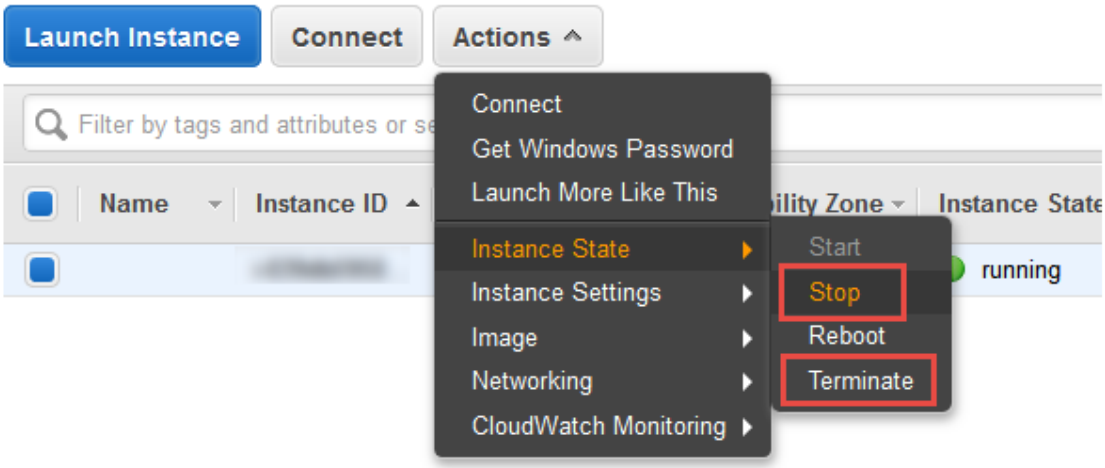
```
private void sendOrder(Side side) {

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

    getOrderService().sendOrder(order);

}
```

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

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/console/home* 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