

# Creating Custom Strategies

Make sure you [set up your development environment](#) before following the steps in this article.

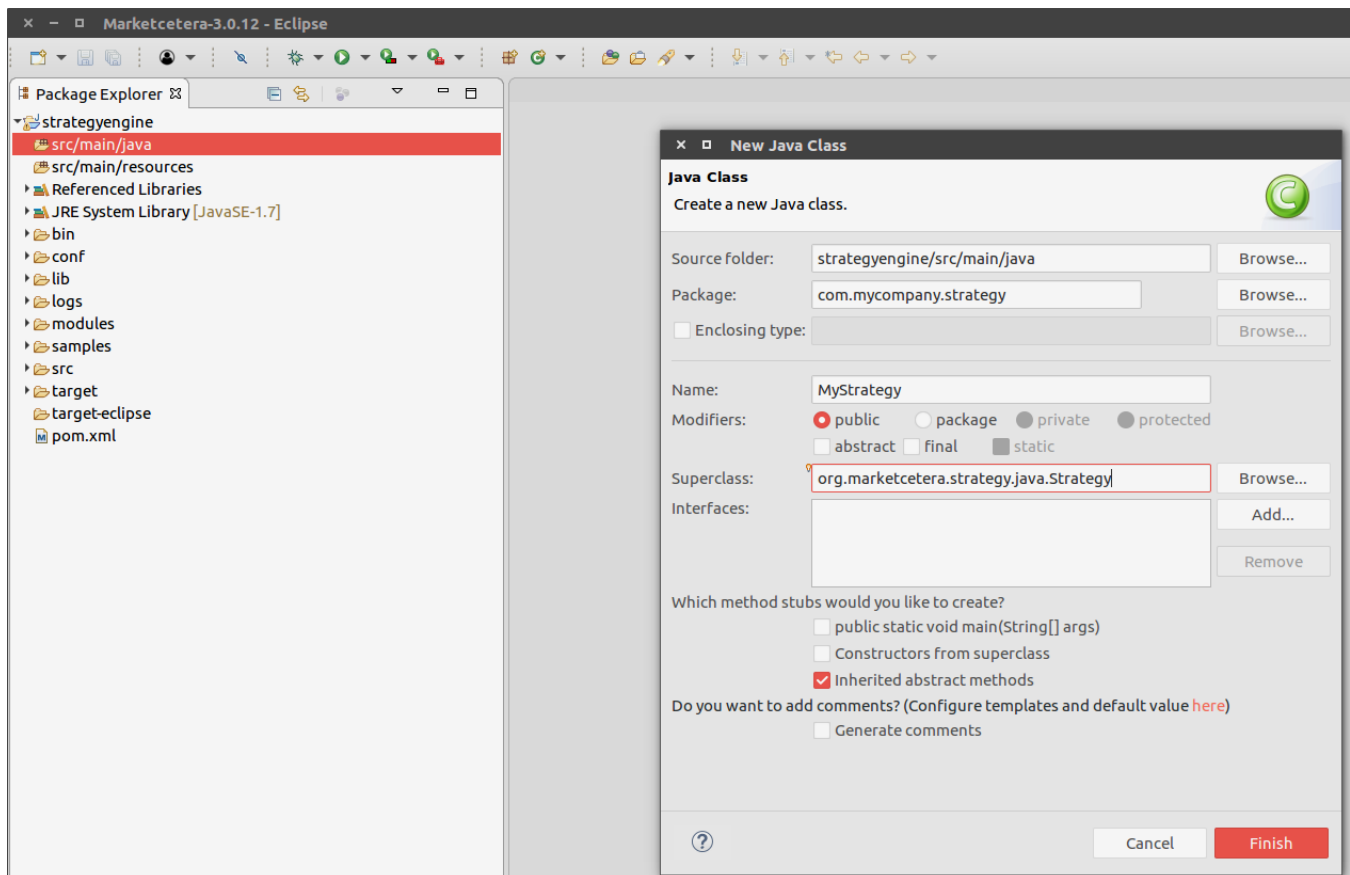
## What is a Strategy?

A strategy, in the context of this article, is a collection of classes you write that you wish to run in the Marketcetera platform. The Marketcetera Strategy Engine is open-source, so you could certainly modify or extend the base platform and recompile it, but this article focuses on implementing your strategies without modifying the base platform. The advantages of this approach include ease-of-upgrade to the base platform when new version become available, being able to modify and rerun your custom strategies without restarting the Strategy Engine, and general good separation-of-concerns from an architecture standpoint.

Your strategy can actually be composed of any number of classes, though we typically refer to "a strategy" as a single logical unit. Your strategy will have a single class that is used as an entry point. This class is identified to the platform as the place to start, and this class can refer to any number of other classes that are either part of the base platform or provided by you.

## How Do I Create a Strategy?

For this article, we'll start with the simple case and build to more complex examples. After setting up your development environment, let's create a basic strategy.



To start, our strategy is empty and doesn't do much.

## Sample Strategy

```
package com.mycompany.strategy;

import org.marketcetera.strategy.java.Strategy;

/* $License$ */

/**
 * My sample strategy.
 *
 * @author <a href="mailto:colin@marketcetera.com">Colin DuPlantis</a>
 * @version $Id$
 * @since $Release$
 */
public class MyStrategy
    extends Strategy
{
}
```

Let's add some basic behavior to the strategy and have it make a market data request on start.

## My Strategy Requests Market Data

```
package com.mycompany.strategy;

import org.marketcetera.marketdata.Content;
import org.marketcetera.marketdata.MarketDataRequestBuilder;
import org.marketcetera.marketdata.bogus.BogusFeedModuleFactory;
import org.marketcetera.strategy.java.Strategy;

/* $License$ */

/**
 * My sample strategy.
 *
 * @author <a href="mailto:colin@marketcetera.com">Colin DuPlantis</a>
 * @version $Id$
 * @since $Release$
 */
public class MyStrategy
    extends Strategy
{
    /* (non-Javadoc)
     * @see org.marketcetera.strategy.java.Strategy#onStart()
     */
    @Override
    public void onStart()
    {
        requestMarketData(MarketDataRequestBuilder.newRequest()
            .withSymbols("AAPL")
            .withProvider(BogusFeedModuleFactory.IDENTIFIER)
            .withContent(Content.LATEST_TICK).create());
    }
}
```

The Marketcetera platform automatically executes `onStart` when the strategy is started. In this case, our strategy is requesting trade market data for AAPL. Let's add a callback so we can receive the market data.

## My Strategy Receives Market Data

```
package com.mycompany.strategy;

import org.marketcetera.event.TradeEvent;
import org.marketcetera.marketdata.Content;
import org.marketcetera.marketdata.MarketDataRequestBuilder;
import org.marketcetera.marketdata.bogus.BogusFeedModuleFactory;
import org.marketcetera.strategy.java.Strategy;

/* $License$ */

/**
 * My sample strategy.
 *
 * @author <a href="mailto:colin@marketcetera.com">Colin DuPlantis</a>
 * @version $Id$
 * @since $Release$
 */
public class MyStrategy
    extends Strategy
{
    /* (non-Javadoc)
     * @see org.marketcetera.strategy.java.Strategy#onStart()
     */
    @Override
    public void onStart()
    {
        requestMarketData(MarketDataRequestBuilder.newRequest()
            .withSymbols("AAPL")
            .withProvider(BogusFeedModuleFactory.IDENTIFIER)
            .withContent(Content.LATEST_TICK).create());
    }
    /* (non-Javadoc)
     * @see org.marketcetera.strategy.java.Strategy#onTrade(org.marketcetera.event.TradeEvent)
     */
    @Override
    public void onTrade(TradeEvent inTrade)
    {
        info("Received " + inTrade);
    }
}
```

The first iteration of our sample strategy is ready to run.

## How Do I Run a Strategy?

The first and easiest way to deploy and run a strategy is to connect to your Strategy Engine from Photon and deploy the strategy from there. The Strategy Engine is a process you run locally, either on your machine or in your data center, that allows secure execution of your private strategies. Let's start the Strategy Engine and give it some commands to run to prepare it for our strategy.

On Windows, the `strategyengine.bat` file automatically executes the commands in `src/main/commands/commands.txt`. These commands include the market data provider we're going to use, the Bogus Feed. On Linux and OSX, execute `bin/strategyengine.sh`. The command file gives the Strategy Engine some data flow instructions that we'll need.

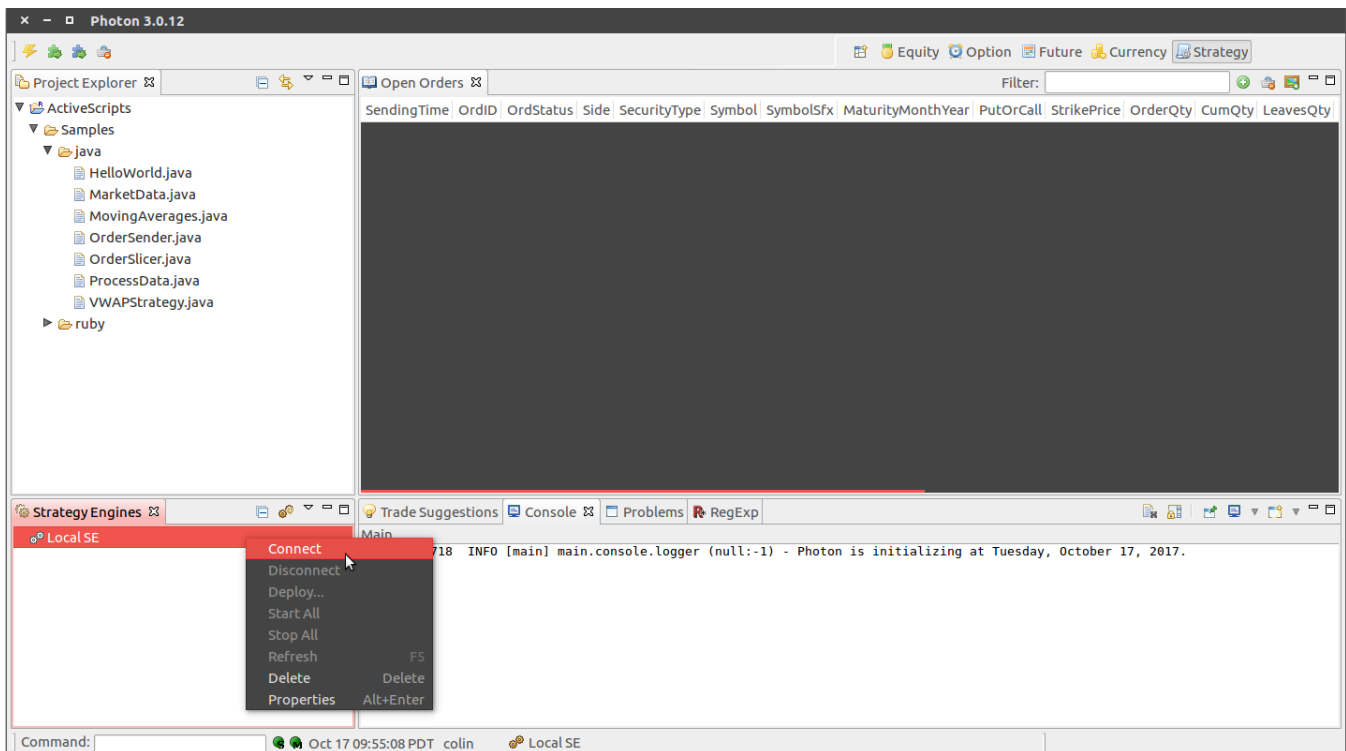
```

$ bin/strategyengine.sh
2017-10-17 09:49:29,429 INFO [main] ? (:) - Strategy Engine version '3.0.12' (build 736 17584
20171017T164928459Z)
2017-10-17 09:49:32,524 INFO [main] ? (:) - Running command 'createModule' with parameters 'metc:cep:system;
metc:cep:system:myinstance'...
2017-10-17 09:49:32,531 INFO [main] ? (:) - Completed command 'createModule' with result 'metc:cep:system:
myinstance'.
2017-10-17 09:49:32,532 INFO [main] ? (:) - Running command 'startModule' with parameters 'metc:mdata:bogus:
single'...
2017-10-17 09:49:32,546 INFO [main] ? (:) - Completed command 'startModule' with result 'true'.
2017-10-17 09:49:32,547 INFO [main] ? (:) - Running command 'createDataFlow' with parameters 'metc:mdata:bogus:
single;type=marketdata:symbols=AAPL^metc:sink:system'...
2017-10-17 09:49:32,578 INFO [main] ? (:) - Completed command 'createDataFlow' with result '1'.
2017-10-17 09:49:33,567 INFO [pool-6-thread-1] ? (:) - Data Flow ID '1' generated data 'of type org.
marketcetera.event.impl.EquityBidEventImpl and' value 'Equity Bid(ADD-6 UPDATE_FINAL) for Equity[symbol=AAPL]:
9.87 5800 Equity[symbol=AAPL] BGS1 at Tue Oct 17 09:49:32 PDT 2017 level1: 1'
2017-10-17 09:49:33,567 INFO [pool-6-thread-1] ? (:) - Data Flow ID '1' generated data 'of type org.
marketcetera.event.impl.EquityAskEventImpl and' value 'Equity Ask(ADD-5 UPDATE_FINAL) for Equity[symbol=AAPL]:
9.89 1310 Equity[symbol=AAPL] BGS1 at Tue Oct 17 09:49:32 PDT 2017 level2: 6'
2017-10-17 09:49:33,569 INFO [pool-6-thread-1] ? (:) - Data Flow ID '1' generated data 'of type org.
marketcetera.event.impl.EquityBidEventImpl and' value 'Equity Bid(ADD-10 UPDATE_FINAL) for Equity[symbol=AAPL]:
9.88 2100 Equity[symbol=AAPL] BGS1 at Tue Oct 17 09:49:33 PDT 2017 level1: 1'
2017-10-17 09:49:34,546 INFO [pool-6-thread-1] ? (:) - Data Flow ID '1' generated data 'of type org.
marketcetera.event.impl.EquityBidEventImpl and' value 'Equity Bid(ADD-15 UPDATE_FINAL) for Equity[symbol=AAPL]:
9.89 5743 Equity[symbol=AAPL] BGS1 at Tue Oct 17 09:49:34 PDT 2017 level1: 6'
2017-10-17 09:49:34,550 INFO [pool-6-thread-1] ? (:) - Data Flow ID '1' generated data 'of type org.
marketcetera.event.impl.EquityBidEventImpl and' value 'Equity Bid(ADD-15 UPDATE_FINAL) for Equity[symbol=AAPL]:
9.89 4433 Equity[symbol=AAPL] BGS1 at Tue Oct 17 09:49:34 PDT 2017 level1: 6'
2017-10-17 09:49:34,551 INFO [pool-6-thread-1] ? (:) - Data Flow ID '1' generated data 'of type org.
marketcetera.event.impl.EquityAskEventImpl and' value 'Equity Ask(ADD-8 UPDATE_FINAL) for Equity[symbol=AAPL]:
9.90 9690 Equity[symbol=AAPL] BGS1 at Tue Oct 17 09:49:33 PDT 2017 level2: 2'

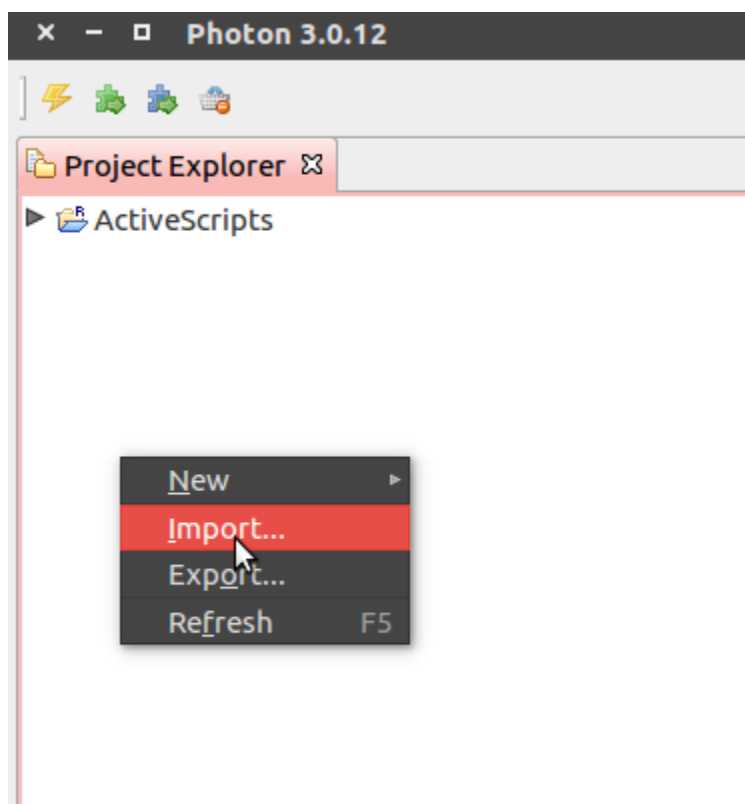
```

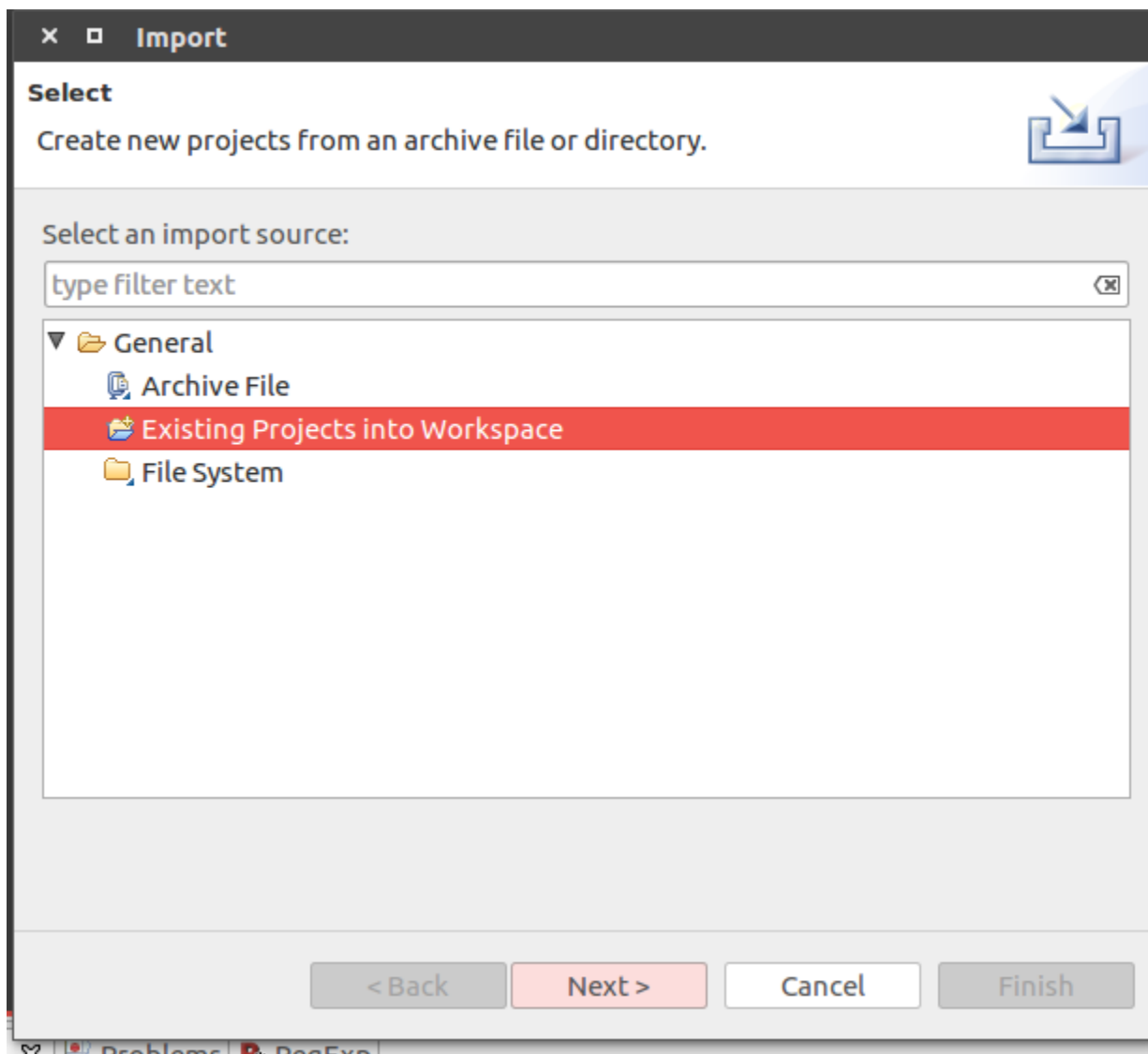
The Strategy Engine is ready for us to connect and deploy our strategy. Start Photon and open the Strategies perspective. Make sure you log in to Photon with the same credentials you specified when you installed the Strategy Engine (located in `strategyengine/conf/user.properties`).

Connect to the Strategy Engine you just started.



Import the strategyengine project we created with Maven to the Photon workspace.







×

□

Import

Import Projects



 Some projects cannot be imported because they already exist in the workspace

☒ Select root directory:

Browse...

☐ Select archive file:

Browse...

Projects:

☐ ActiveScripts (/opt/Marketcetera-3.0.12/photon/workspace/Acti

☒ strategy (/opt/Marketcetera-3.0.12/strategyengine)

Select All

Deselect All

Refresh

☐ Copy projects into workspaceWorking sets

☐ Add project to working sets

Working sets: 

▲▼

Select...

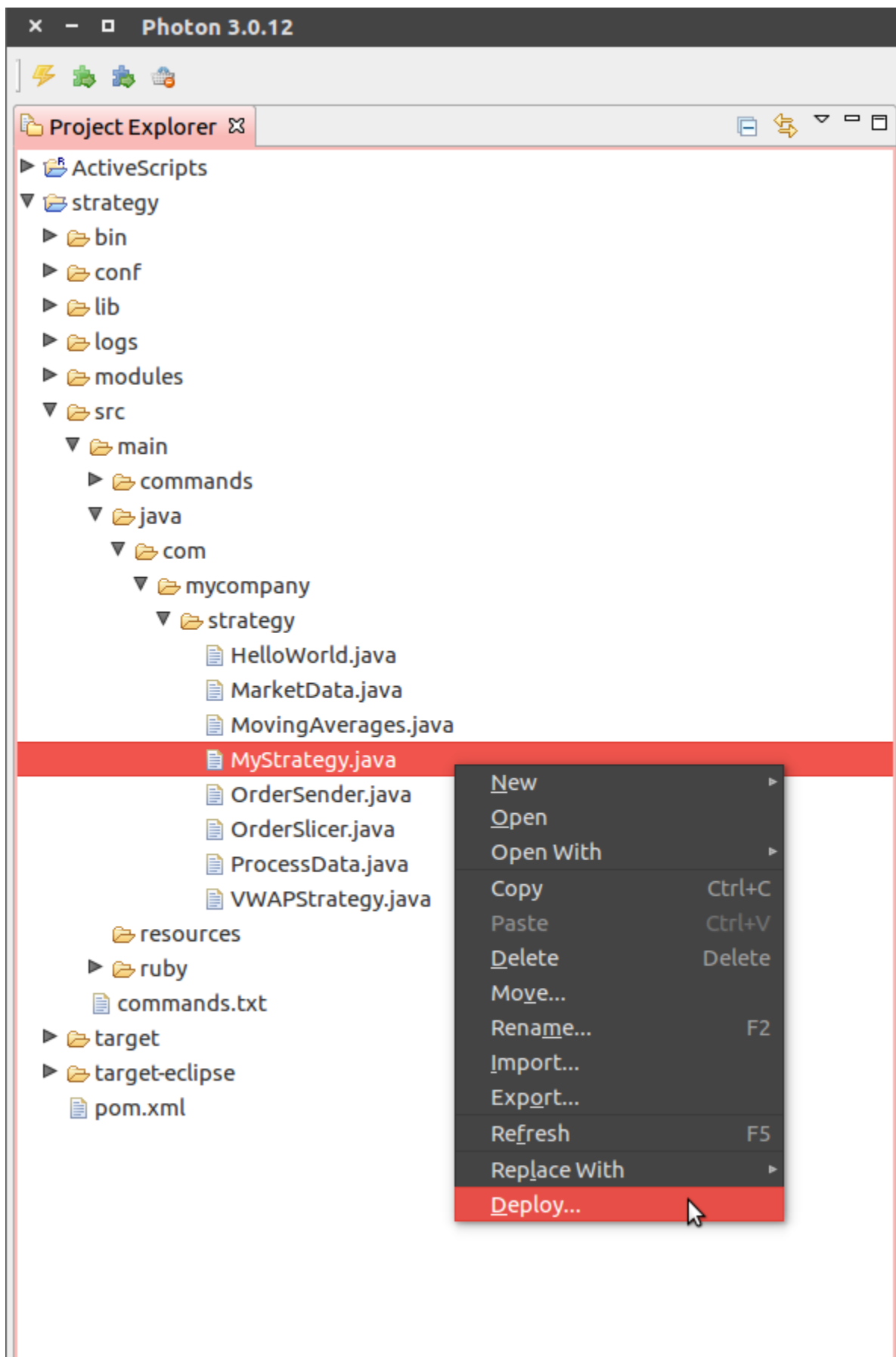
< Back

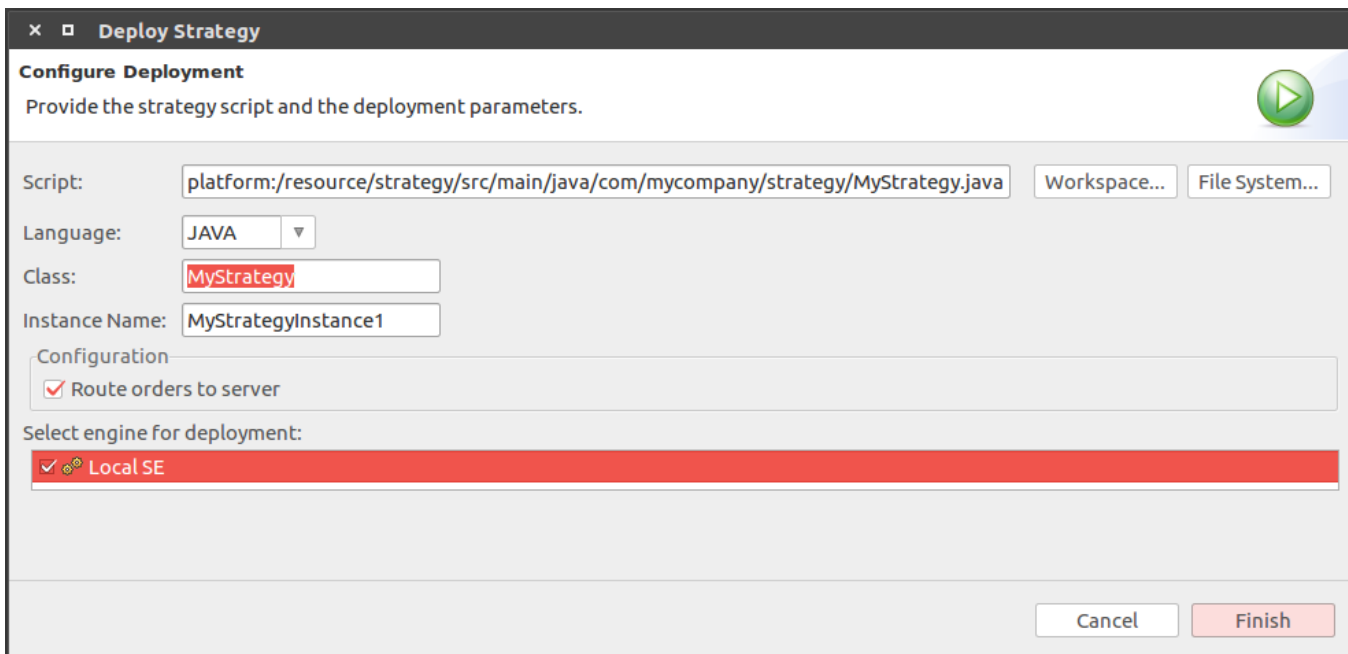
Next >

Cancel

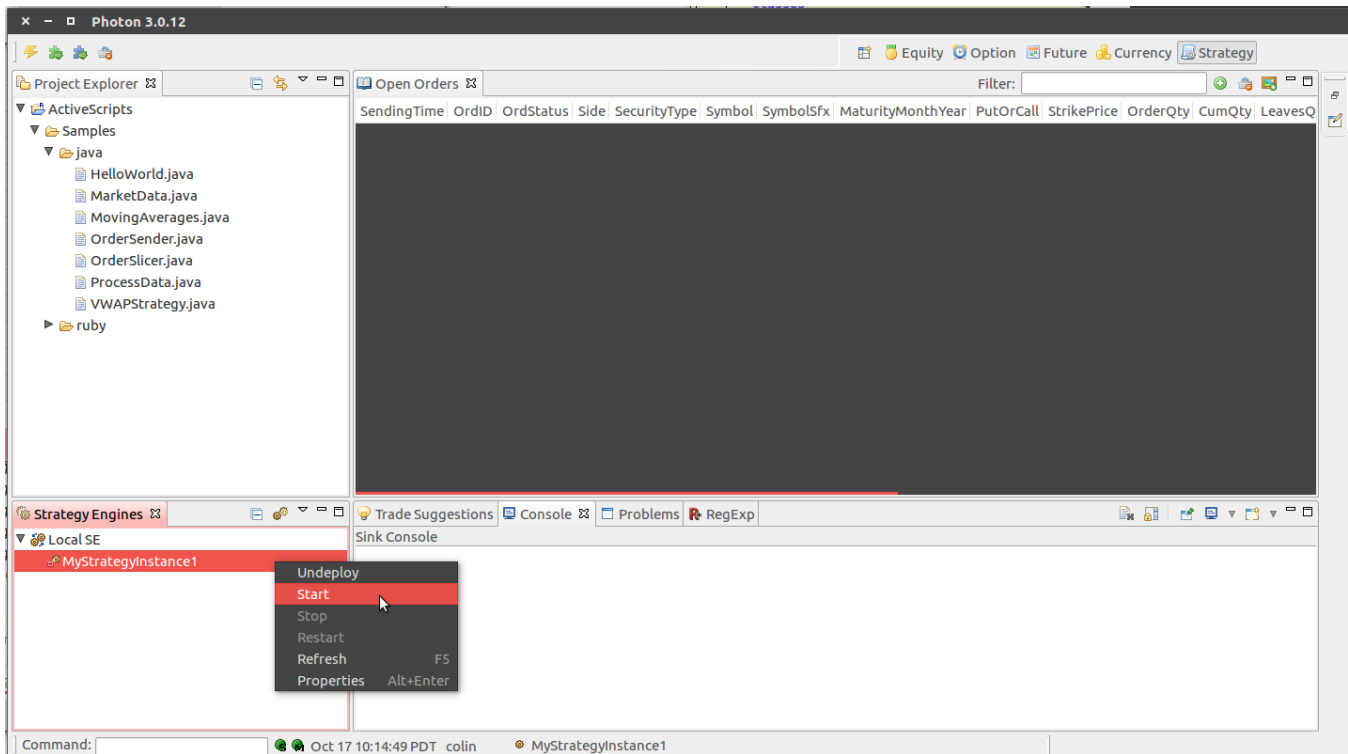
Finish

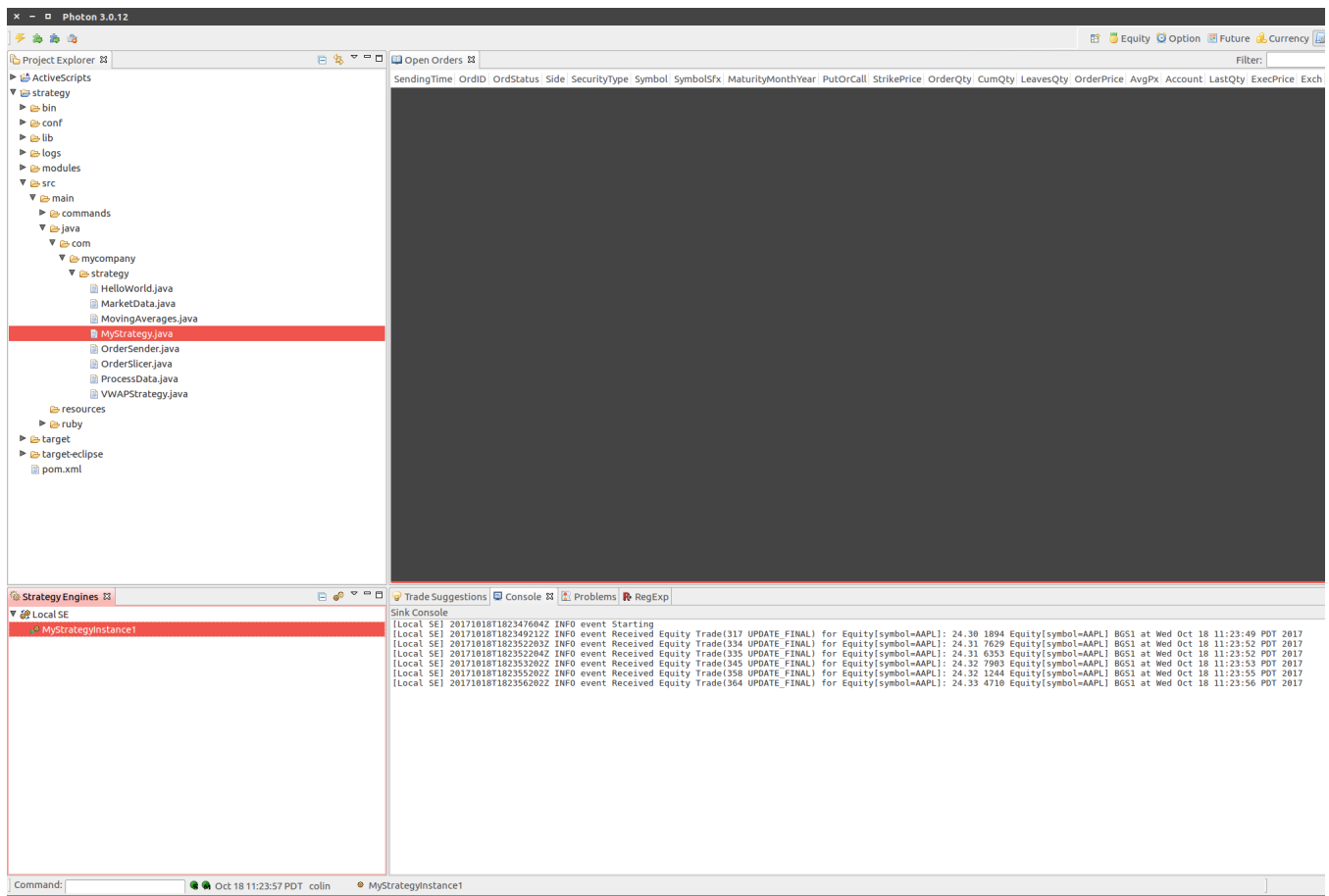
Deploy the sample strategy to the Strategy Engine.





The strategy is deployed to the Strategy Engine, now start it.





## How Do I Update My Strategy?

So far, our strategy doesn't yet do anything terribly interesting. It requests market data, but doesn't really do anything with it yet. Let's modify the strategy to create an order when it receives a trade event.

To modify the strategy, go back to your IDE (I'm using Eclipse) and make the changes there. Using an IDE gives you auto-completion and access to all the Marketcetera libraries.

## My Strategy Sends Orders

```
package com.mycompany.strategy;

import java.math.BigDecimal;

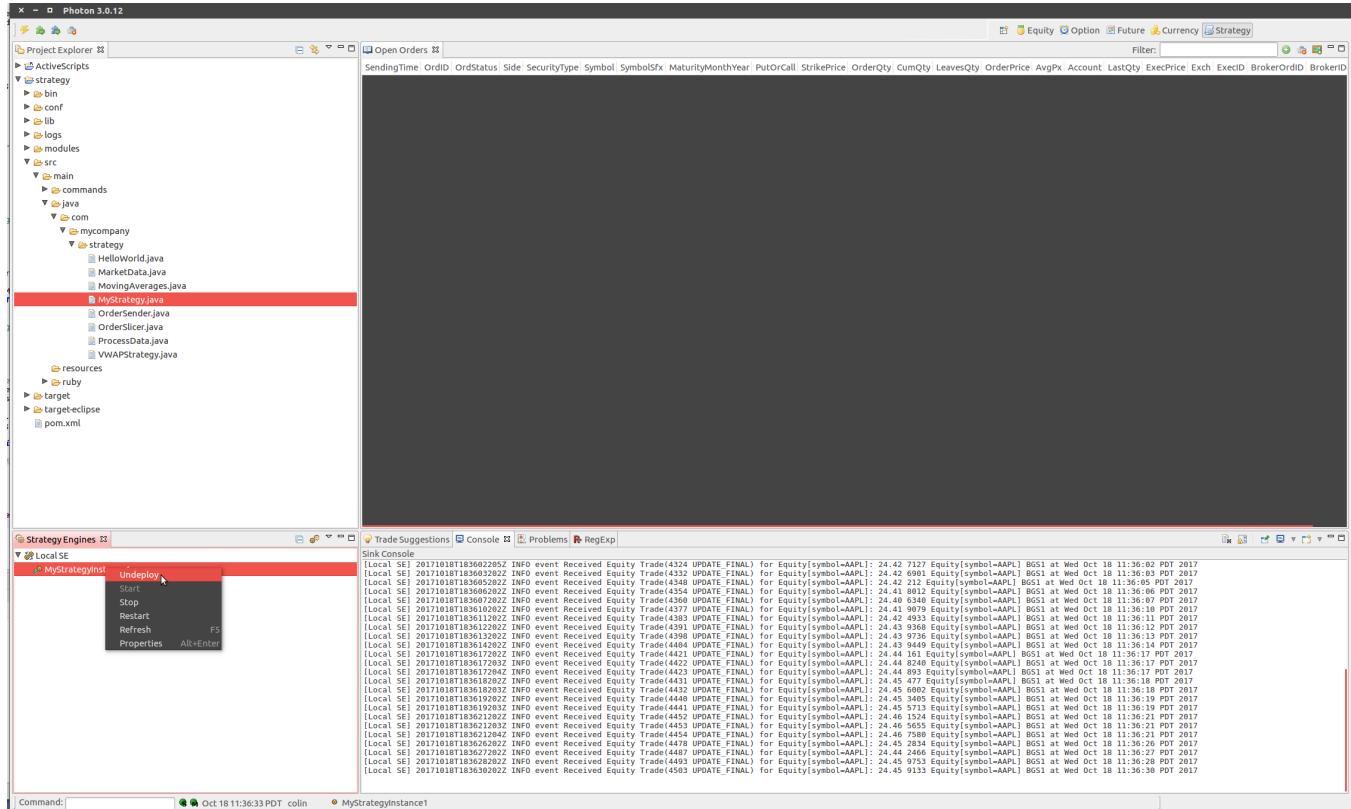
import org.marketcetera.event.TradeEvent;
import org.marketcetera.marketdata.Content;
import org.marketcetera.marketdata.MarketDataRequestBuilder;
import org.marketcetera.marketdata.bogus.BogusFeedModuleFactory;
import org.marketcetera.strategy.java.Strategy;
import org.marketcetera.trade.Factory;
import org.marketcetera.trade.OrderSingle;
import org.marketcetera.trade.OrderType;
import org.marketcetera.trade.Side;
import org.marketcetera.trade.TimeInForce;
import org.marketcetera.util.log.SLF4JLoggerProxy;

/* $License$ */

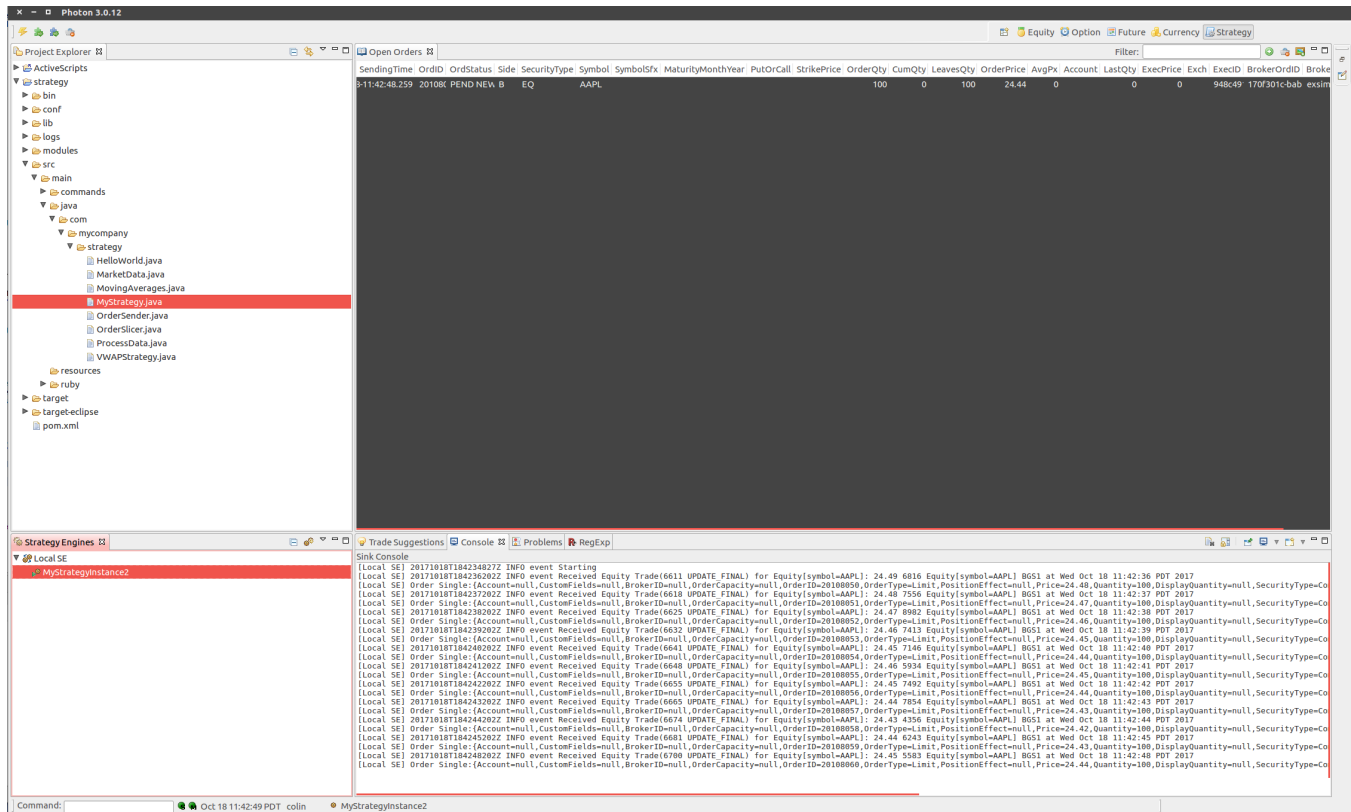
/**
 * My sample strategy.
 *
 * @author <a href="mailto:colin@marketcetera.com">Colin DuPlantis</a>
 * @version $Id$
 * @since $Release$
 */
public class MyStrategy
    extends Strategy
{
    /* (non-Javadoc)
     * @see org.marketcetera.strategy.java.Strategy#onStart()
     */
    @Override
    public void onStart()
    {
        info("Starting");
        requestMarketData(MarketDataRequestBuilder.newRequest()
            .withSymbols("AAPL")
            .withProvider(BogusFeedModuleFactory.IDENTIFIER)
            .withContent(Content.LATEST_TICK).create());
    }
    /* (non-Javadoc)
     * @see org.marketcetera.strategy.java.Strategy#onTrade(org.marketcetera.event.TradeEvent)
     */
    @Override
    public void onTrade(TradeEvent inTradeEvent)
    {
        info("Received " + inTradeEvent);
        // place an order just under the most recent trade
        OrderSingle newOrder = Factory.getInstance().createOrderSingle();
        newOrder.setInstrument(inTradeEvent.getInstrument());
        newOrder.setOrderType(OrderType.Limit);
        newOrder.setPrice(inTradeEvent.getPrice().subtract(ONE_PENNY));
        newOrder.setQuantity(new BigDecimal(100));
        newOrder.setSide(Side.Buy);
        newOrder.setTimeInForce(TimeInForce.ImmediateOrCancel);
        SLF4JLoggerProxy.info(this,
            "Created order {}",
            newOrder);

        send(newOrder);
    }
    /**
     * constant value used to represent 0.01
     */
    private static final BigDecimal ONE_PENNY = new BigDecimal("0.01");
}
```

In Photon, undeploy your strategy as we're going to replace it with an updated version.



Deploy the strategy again as above. The strategy source was changed in the IDE and we're deploying that changed strategy. Start it again as above.



This describes how to author and run a strategy.

## Related articles

- [Creating Custom Strategies](#)
- [Creating Custom Modules](#)
- [Starting/Stopping a Strategy from a Strategy](#)
- [Executing a Pre-Compiled Strategy](#)
- [Creating a Strategy Data Flow](#)