Creating a New Strategy

- Getting Started
- Installation Guide
- Getting Started How-To
- Building From Source
- Creating a New Strategy

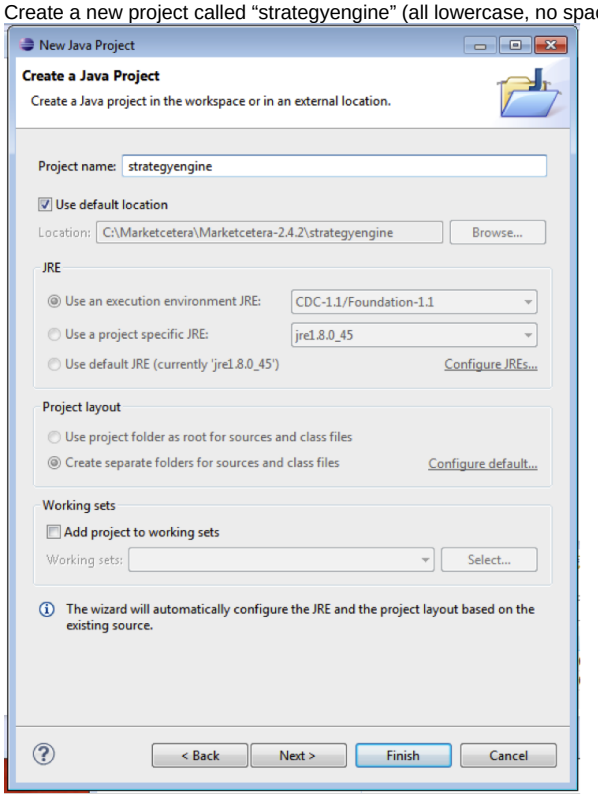
CREATING A NEW STRATEGY

Creating a New Strategy

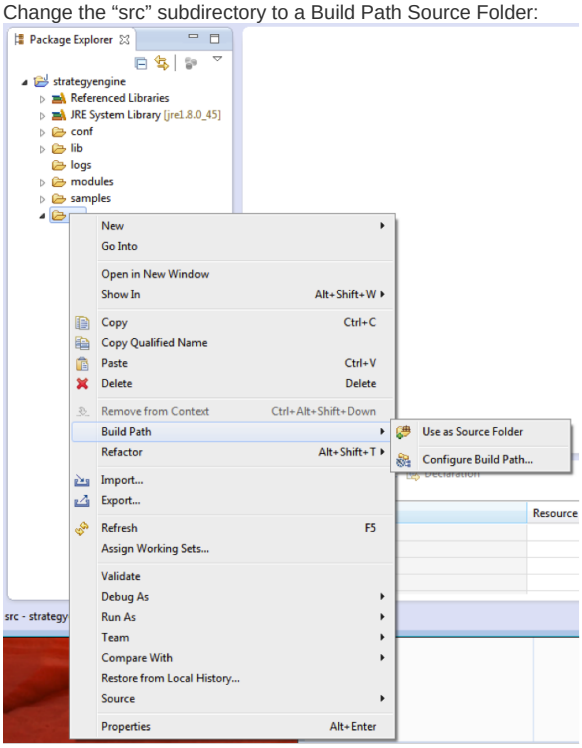
When creating a strategy, it's helpful to use a context-aware IDE like Eclipse, IntelliJ, or NetBeans when building a complex Java application. This step-by-step guide will show you one way to set this up.

Step-by-step guide

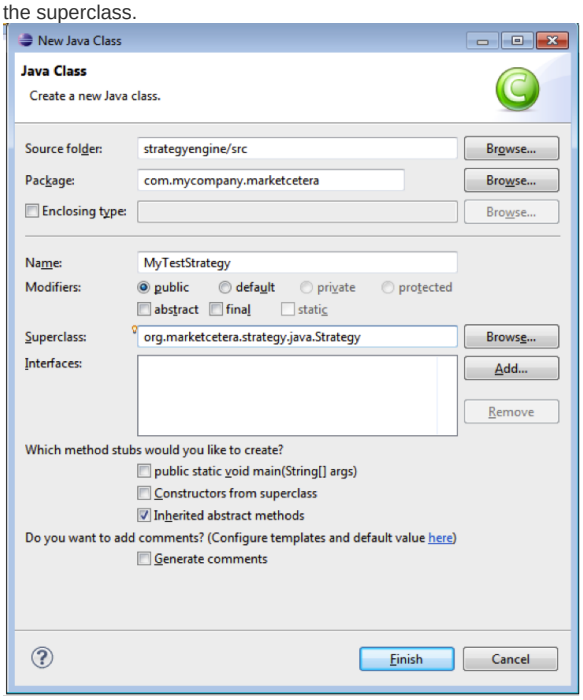
1. Download and install Marketcetera for your platform. You'll want the Strategy Engine and Photon. You won't need to Order Loader for this, but it won't hurt, either.
2. Install your favorite IDE. We'll be using Eclipse for this example, but you can use whatever you want, you'll just have to make some adjustments.
3. Start Eclipse and create a new workspace using the Marketcetera installed directory (Marketcetera-2.4.2, e.g.) as the base folder
4. Create a new project called "strategyengine" (all lowercase, no spaces, just like this):



5. Change the “src” subdirectory to a Build Path Source Folder:



6. Create a new strategy by right-clicking the “src” subfolder, selecting New, and Java Class. Note the superclass.



7. Create your strategy.

```
package com.mycompany.marketcetera;
import java.math.BigDecimal;
import org.marketcetera.strategy.java.Strategy;
import org.marketcetera.trade.Equity;
import org.marketcetera.trade.ExecutionReport;
import org.marketcetera.trade.Factory;
import org.marketcetera.trade.OrderSingle;
import org.marketcetera.trade.OrderType;
import org.marketcetera.trade.Side;
public class MyTestStrategy extends Strategy {
  public void onStart()
  {
          respondedToOrder = false;
          // this will send a message to the Photon Sink Console (warn is the default level, can be changed in config)
          warn("MyTestStrategy starting");
          orderFactory = Factory.getInstance();
          OrderSingle order = orderFactory.createOrderSingle();
          // this can be Equity, Option, Future, Currency, or ConvertibleBond
          order.setInstrument(new Equity("METC"));
          order.setOrderType(OrderType.Limit);
          order.setPrice(new BigDecimal(75.50));
          order.setQuantity(new BigDecimal(200));
          order.setSide(Side.Sell);
          // this will send the order to the default broker
          send(order);
  }
  public void onExecutionReport(ExecutionReport inExecutionReport)
  {
          warn("Received: " + inExecutionReport);
          // do something to respond, how about crossing the order?
          if(!respondedToOrder) {
                  respondedToOrder = true;
                  OrderSingle cross = orderFactory.createOrderSingle();
                  cross.setInstrument(inExecutionReport.getInstrument());
                  cross.setOrderType(inExecutionReport.getOrderType());
                  cross.setPrice(inExecutionReport.getPrice());
                  cross.setQuantity(inExecutionReport.getLeavesQuantity());
                  cross.setSide(inExecutionReport.getSide() == Side.Buy ? Side.Sell : Side.Buy);
                  send(cross);
          }
  }
  /**
   * indicates if we've responded to the order yet or not
   */
  private boolean respondedToOrder;
  /**
   * creates order objects
   */
  private Factory orderFactory;
}
```

8. Deploy the strategy from Photon or a command file