How to automate analytics reporting

2015-05-04

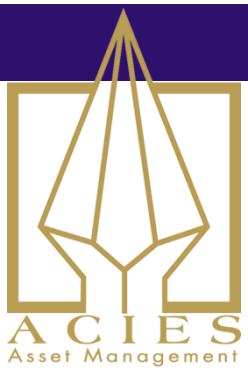
I often come across situations where it would be useful to automate calculations. Perhaps you need a daily dashboard screen with analytics or view the status of a trading model. There are lots of things that could be useful to calculate on an automated basis. Let me show you a simple way to achieve this.

It's no surprise to anyone that I prefer RightEdge. The reason I like it so much is the flexibility. It's certainly not built for automated reporting, but I make heavy use of it for that purpose anyhow. Let's take a look at a little trick that makes reporting from RightEdge very easy.

Reporting Current Calculated Values

So let's assume that you want to calculate some analytic on the markets you follow. Whether you're dealing with 5 markets of 500 doesn't matter. It doesn't even matter if you're looking to calculate 5 or 500 values per market. The method here can be used either way. I use this method for updating the weekly futures report for instance. I also use it to calculate momentum rankings for a few thousand stocks every day.

What we want to do in this example is to trigger code only on the very last day, where we gather the information we're looking for and report it to a database. If you're familiar with RightEdge, you may be asking how we know which the last day is. That's not entirely clear in that platform. Unless of course you're using the system class.



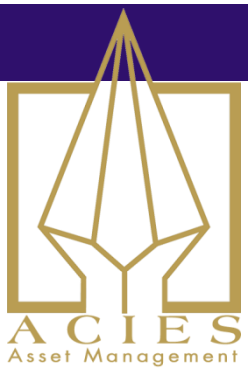
Using RightEdge, the key part to understand is the two levels. When you start a new system, the pre-generated code will give you two classes. The top one is the system level, which most people never use. The bottom part is the symbol level, where most people put all their code.

```
public class MySystem : MySystemBase
{

}

public class MySymbolScript : MySymbolScriptBase
{

}
```

Granted that the class names are pretty bad. The system level class is MySystem and the symbol level class is MySymbolScript. Here's the important part: When you do a simulation on many markets, there will be one single MySystem class and one MySymbolScript class per market. Having that top level system class can be extremely useful.

We'll use MySymbolScript to calculate whatever we need to calculate for each market. Then we'll do the reporting when the system as a whole is complete. As an example, let's add a simple moving average to the symbol class. This is the number we want to report in this example. You add this like you'd normally do it.

```
public class MySymbolScript : MySymbolScriptBase
{
        public SMA movingAverage;

        public override void Startup()
        {
                movingAverage  = new SMA(50,Close);
                movingAverage.ChartSettings.ShowInChart=true;
        }
```




```
        public override void NewBar()
        {
                // This is where you'd normally put your trading code.
        }
}
```

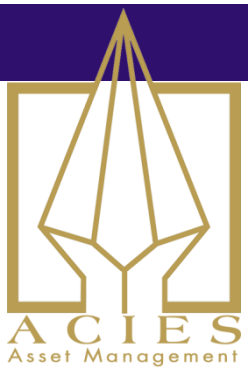
In the code above, I just added a 50 day moving average.

Next we move up to the top, to the system level logic. By default, you'll only see a Startup() method. We don't need that. What we want to do is run code before the system shuts down. So let's override the Shutdown() method.

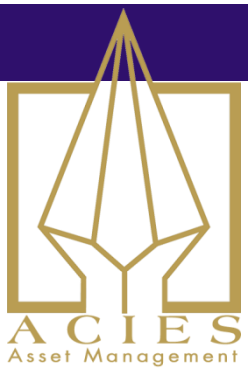
```
        public override void Shutdown()
        {

        }
```

Adding the above code to the System level class will let you run something just when the whole system is done. Here we can add our reporting logic. The system level class has access to a collection called SymbolScripts. This contains all the individual MySymbolScript classes, so you can iterate through them, get the info you need and report it.

The full source follows on the next page. By adapting this code to fit your own needs and environment, you can use RightEdge not only as a simulation platform, but as a reporting engine. Of course, if you don't like databases (who doesn't like databases?) you could always output to a text file or whatever format you so prefer.




```
#region Using statements
using System;
using System.Drawing;
using System.Collections.Generic;
using System.Linq;
using RightEdge.Common;
using RightEdge.Common.ChartObjects;
using RightEdge.Indicators;
using MySql.Data.MySqlClient;  // Don't forget the references if you're using database reporting.
#endregion

#region System class
public class MySystem : MySystemBase
{

	public override void Shutdown()
	{
		foreach(MySymbolScript s in SymbolScripts) // iterate through all individual symbols
		{
			reportSMA(s.Bars.Current.BarStartTime,  s.Symbol.Name,
s.movingAverage.Current); //send the date, symbol name and analytic value
		}
	}

	private string conString =
"SERVER=serverIP;PORT=3306;DATABASE=dbName;Uid=user;Pwd=pwd;"; // replace with your connection
string
	private void reportSMA(DateTime tradeDate, string ticker, double movingAverage)
	{
		string query = @"insert ignore into yourTable (trade_date, ticker,  movingAverage) values ('"
			+ tradeDate.ToString("yyyy-MM-dd") + "', '" + ticker + "', "
			+ movingAverage + ") on duplicate key update
movingAverage=values(movingAverage)"; // adapt query to your own database structure

				using (MySqlConnection connection = new MySqlConnection(conString))
				{
					MySqlCommand command = new MySqlCommand(query,
connection);
					if (connection.State != System.Data.ConnectionState.Open)
						connection.Open();
```




```
                                        command.ExecuteNonQuery();
                                    }
        }
}
#endregion

public class MySymbolScript : MySymbolScriptBase
{
        public SMA movingAverage;

        public override void Startup()
        {
                movingAverage  = new SMA(50,Close);
                movingAverage.ChartSettings.ShowInChart=true;
        }

        public override void NewBar()
        {
                // This is where you'd normally put your trading code.
        }
}
```