OS.ENGINE

INDICATORS

O-S-A.NET

Table of content

About inheritance

In order to create indicators by yourself, you need to understand the concept and methods of inheritance.

In all cases, except the place of creation (BotPanel) where Os.Engine indicators are applied, they are not applied to the target class in which they are implemented. They are used through the IIndicatorCandle Interface.

Therefore, if you are not familiar with inheritance, the best solution would be to read educational materials on the topic.

1. Work of the indicator in Os.Engine architecture

Where the indicator works:

PanelCreator

Creation by the robot. Storing the links and using the logic of the robot

ChartMaster

Creation by the user. Storage. Loading by candles and user access for configuration

ChartPainter

Tracing

2. Indicators and PanelCreator

During the creation of the robot, you can create indicators and use them when trading. However, once created, indicators are transmitted to ChartMaster for storing and loading by candles.

In order to create an indicator in the bot, you need:

1. In the descendant class of BotPanel which implements the logic of the robot, it is necessary to declare an indicator field, such as MovingAverage
2. Assign a value to this indicator.
3. Transmit the indicator into a ChartMaster for storage and calculations

```
private MovingAverage Sma; // declaring indicator
```

1 method of creation, with the transfer of the unique name of the indicator as a parameter. This allows the robot to save the indicator settings after turning the program off and on

```
Sma = new MovingAverage(name +"Sma") {Lenght = 15,TypeCalculationAverage = MovingAverageTypeCalculation.Exponential}; // creation with indication
```

2 ways of creation, without parameters. The robot will not save the parameters if you change them manually. Each time in the process of loading it will call the parameters specified in the code.

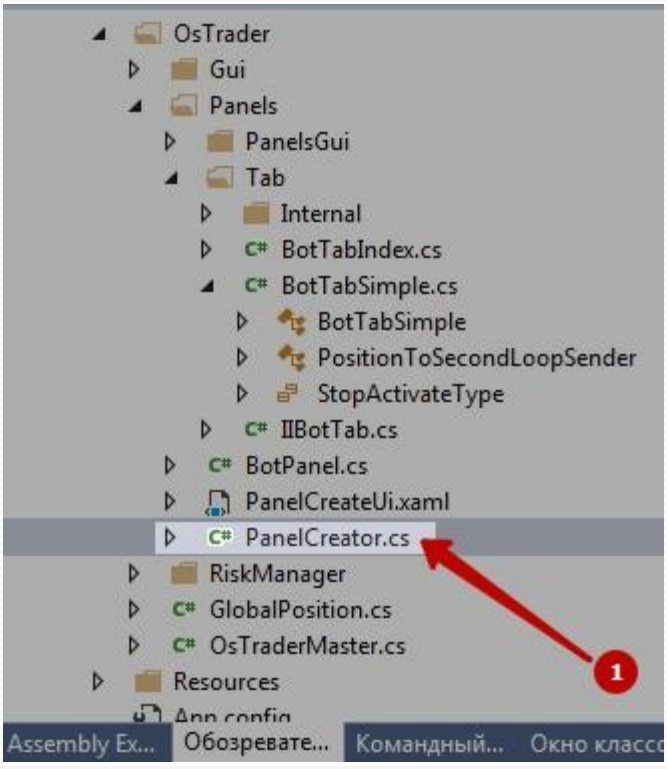
```
Sma = new MovingAverage() {Lenght = 15,TypeCalculationAverage = MovingAverageTypeCalculation.Exponential}; // creation with indication
```

```
CreateCandleIndicator(Sma,"Prime"); // process of transmission of the indicator to ChartMaster for storage. The second parameter is the name of the chart area on which the indicator will be reflected. Prime is the area with candles
```

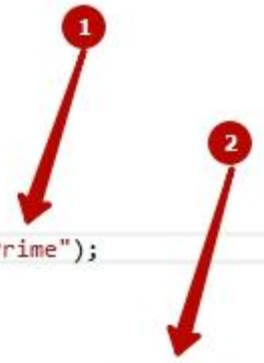
If “Prime” is specified, the indicator in the robot will be located on the main chart window along with candles.

If you specify any other name, an individual area will be created to display the indicator under the main chart.

An example of using the indicator as part of creating a robot.
When creating a robot using PanelCreator



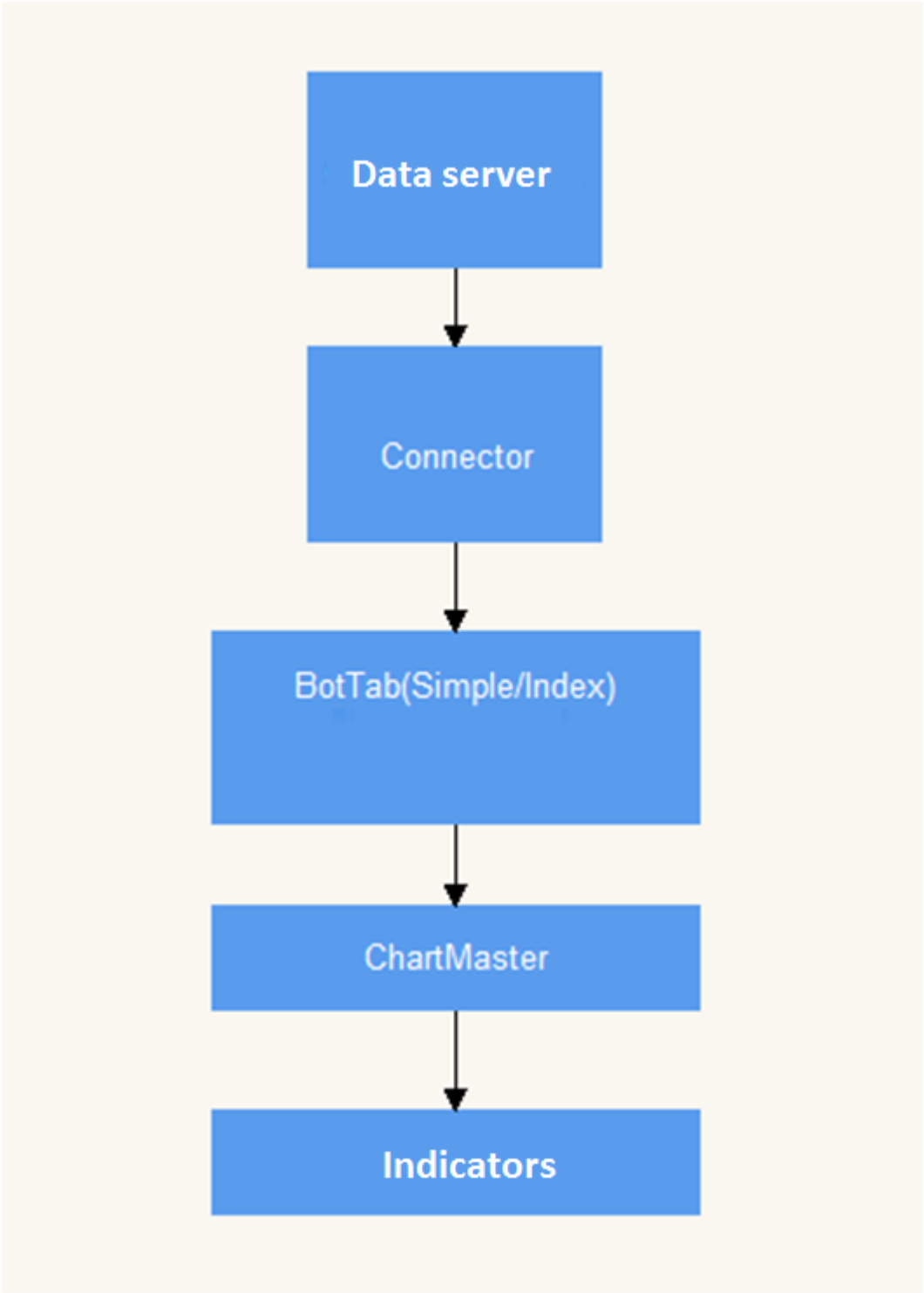
```
public class SmaStochastic : BotPanel
{
    /// <summary>
    /// конструктор
    /// </summary>
    ссылка 1
    public SmaStochastic(string name)
        : base(name)
    {
        TabCreate(BotTabType.Simple);
        _tab = TabsSimple[0];

        _sma = new MovingAverage(name + "Sma", false);
        _sma = (MovingAverage)_tab.CreateCandleIndicator(_sma, "Prime");
        _sma.Save();

        _stoc = new StochasticOscillator(name + "ST", false);
        _stoc = (StochasticOscillator)_tab.CreateCandleIndicator(_stoc, "StocArea");
        _stoc.Save();
```



After initialization of the indicator, and its transfer to _tab.CreateIndicarot as soon as candles appear in the robot, the indicator will be counted and reflected.

The data will be available in the Values array. So it is possible to take the last value: _sma.Values[_sma.Values.Count - 1]

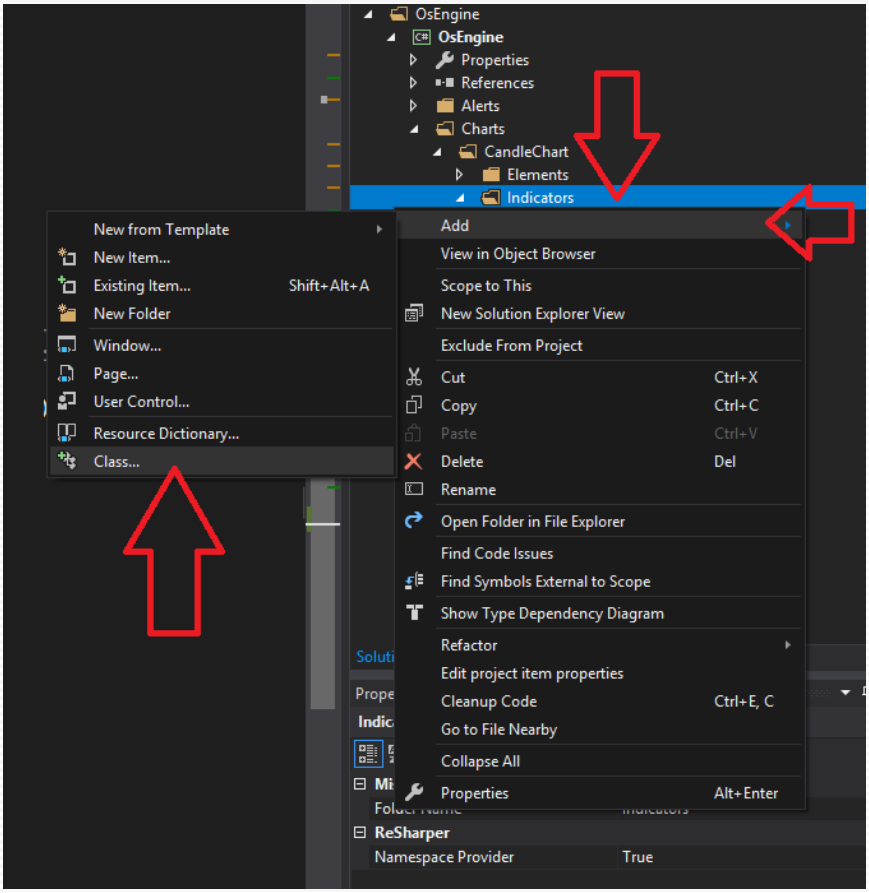
3. Path of candles to the repository of indicators



4. Creating an indicator

Creating an indicator starts with creating a class. To do this, go to Indicators, right-click and select Add > Class

After the class is created, it will appear in the list below.



After this, there will appear a template of indicators.

To begin, modify the string

public class ClassName : IIndicatorCandle

ChartPainter reflects the indicators in a given area. And it does this depending on the type of reflection of the indicator, which is specified in the Class of the indicator. In the field: TypeIndicator

It may takes values:

- Line
- Column
- Point

Let us consider creating an indicator in OS Engin on the example of ATR

We create two constructors for the indicator with and without the preservation of parameters

```
public Atr(string uniqName, bool canDelete)
{Name = uniqName;
   Lenght = 14;
   TypeIndicator = IndicatorOneCandleChartType.Line;
   TypeCalculationAverage = MovingAverageTypeCalculation.Simple;
   ColorBase = Color.DodgerBlue;
   PaintOn = true;
   CanDelete = canDelete;
   Load();
 }
// uniqName  - unique name
// canDelete   whether the user can manually remove the indicator from the chart
// TypeIndicator - type of indicator display
// ColorBase   - color of the indicator
// PaintOn      should the indicator be reflected on the chart
```

Later we declare the variables used in the indicator

The Save() method is responsible for saving parameters entered by the user. All parameters are stored in new rows.

The Load() method loads saved parameters when the indicator starts

The ShowDialog() method is responsible for the involvement of the graphical interface to enter indicator parameters by the user

In order to create the graphical interface of the indicator, right-click on Indicators and select Add > Window, then select WPF

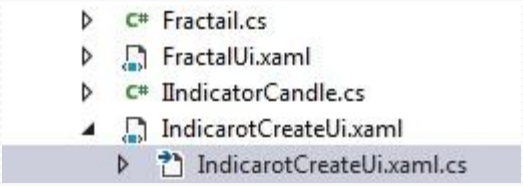
Arrays with values are to be overloaded. If this is not done, nothing will be reflected. From them, ChartPainter receives information about what and how to reflect.

It is better to use an explicit implementation of the interface, specifying the interface name before the property. In order to protect the user from unnecessary properties on the part of the robot creation layer.

For an indicator of the Column type, the Colors field must contain a double number of colors from the number of arrays with data. One for the growing column, the second for the falling.

In other cases, the number of colors must match the number of data arrays.

Then it is required to go to IndicarotCreateUi class and add the indicator as standard.



In IndicarotCreateUi constructor
_gridViewIndicators.Rows.Add ("Name of the indicator to be displayed”);

In the gridViewIndicators_SelectionChanged event
add

```
if (_gridViewIndicators.SelectedCells[0].Value.ToString() == "Indicator name")
{
TextBlockDescription.Text = "Brief description of the indicator";
}
```

In the ButtonAccept_Click event add

```
if (_gridViewIndicators.SelectedCells[0].Value.ToString() == "Indicator name")
{
string name = "";

for (int i = 0; i < 30; i++)
    {
    if (_chartMaster.IndicatorIsCreate(_chartMaster.Name + "Indicator name" + i) ==
    false)
                {
                    name = "Indicator name" + i;
                    break;
                }
    }
    IndicatorCandle = new PriceChannel(_chartMaster.Name + name, true);
            _chartMaster.CreateIndicator(IndicatorCandle, areaName);
        }
```

5. Creating an indicator and ChartMaster

After creating the class, it is required to learn how to save and load the indicator.

For this purpose it is necessary to register lines in the Load method

```
if (indicator[0] == "IndicatorName")
{
CreateIndicator(new ClassName(indicator[1], Convert.ToBoolean(indicator[3])),
    indicator[2]);
}
```