OS.ENGINE

INDICATORS

O-S-A.NET

Оглавление

О наследовании

Для того чтобы самому создавать индикаторы, необходимо понимать концепцию и способы наследования.

Везде, кроме места создания(BotPanel) где бы не применялись индикаторы в Os.Engine, они не применяются по конечному классу в котором реализованы. Они используются через Интерфейс IIndicatorCandle.

Поэтому, если Вы не знакомы с наследованием – лучшим решением будет почитать учебные материалы на тему.

1. Работа индикатора в архитектуре Os.Engine

Где работает индикатор:

PanelCreator

Создание роботом. Хранение ссылки и использование в логике робота

ChartMaster

Создание пользователем. Хранение. Прогрузка свечками и доступ пользователю для настройки

ChartPainter

Прорисовка

2. Индикаторы и PanelCreator

Во время создания робота, можно создавать индикаторы и использовать их во время торгов. При этом созданные индикаторы передаются в ChartMaster для хранения и прогрузки свечками.

Чтобы создать индикатор в боте, надо:

1. В классе наследнике BotPanel реализующем логику робота, необходимо объявить поле индикатора, например MovingAverage
2. Присвоить этому индикатору значение.
3. Передать индикатор в ChartMaster для хранения и расчётов

```
private MovingAverage Sma; // объявление индикатора
```

1 способ создания, с передачей в качестве параметра уникального имени индикатора. Это позволяет роботу сохранять настройки индикатора после выключения / включения программы

```
Sma = new MovingAverage(name +"Sma") {Lenght = 15,TypeCalculationAverage = MovingAverageTypeCalculation.Exponential}; // создание с задаванием
```

2 способ создания, без параметров. Робот при этом не будет сохранять параметры, если их изменять вручную. И каждый раз во время загрузки будет вызывать параметры заданные из кода.

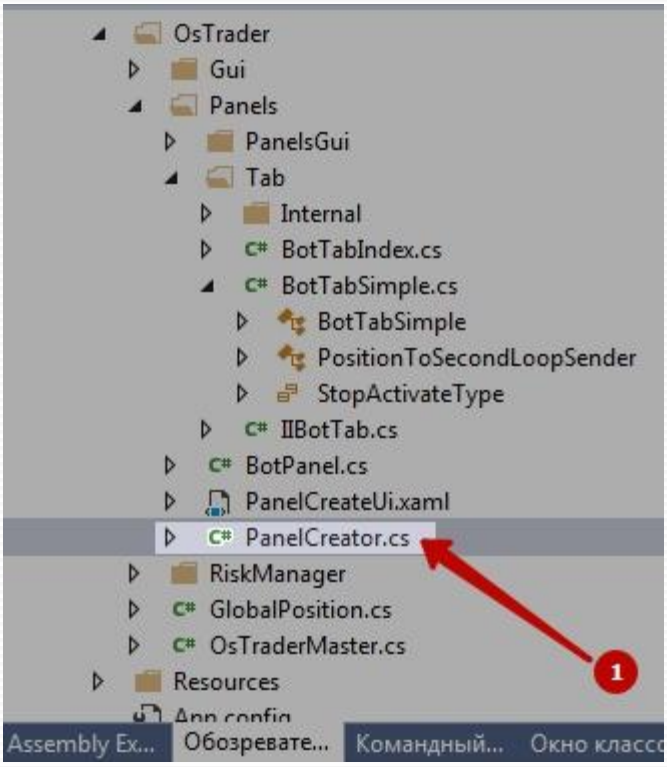
```
Sma = new MovingAverage() {Lenght = 15,TypeCalculationAverage = MovingAverageTypeCalculation.Exponential}; // создание с задаванием
```

```
CreateCandleIndicator(Sma,"Prime"); // процесс передачи индикатора на хранение в ChartMaster. В качестве второго параметра передаётся название области чарта на которой будет прорисовываться индикатор. Prime – область со свечками
```

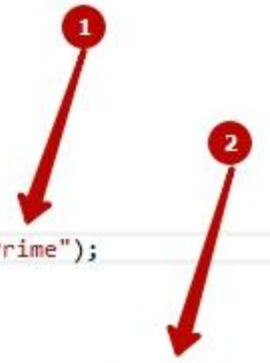
Если указано “Prime” , то индикатор в роботе будет располагаться на основном окне чарта вместе со свечами.

Если указать любое другое название, то будет создана индивидуальная область для отображения индикатора под основным чартом.

Пример использования индикатора в рамках создания робота.
При создании робота через 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();
```

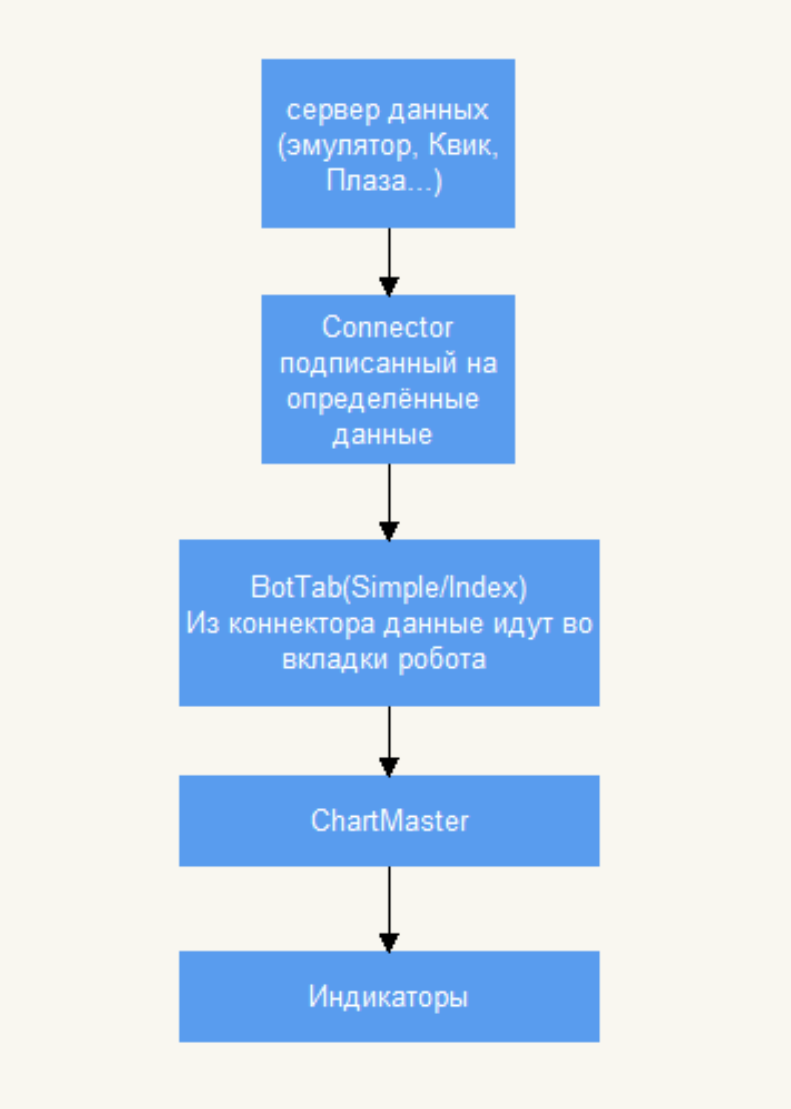


После инициализации индикатора, и передачи его в в _tab.CreateIndicarot, как только в роботе появятся свечи – наш индикатор посчитается и прорисуется.

Данные будут доступны в массиве Values. Вот так можно взять последнее значение:

_sma.Values[_sma.Values.Count - 1]

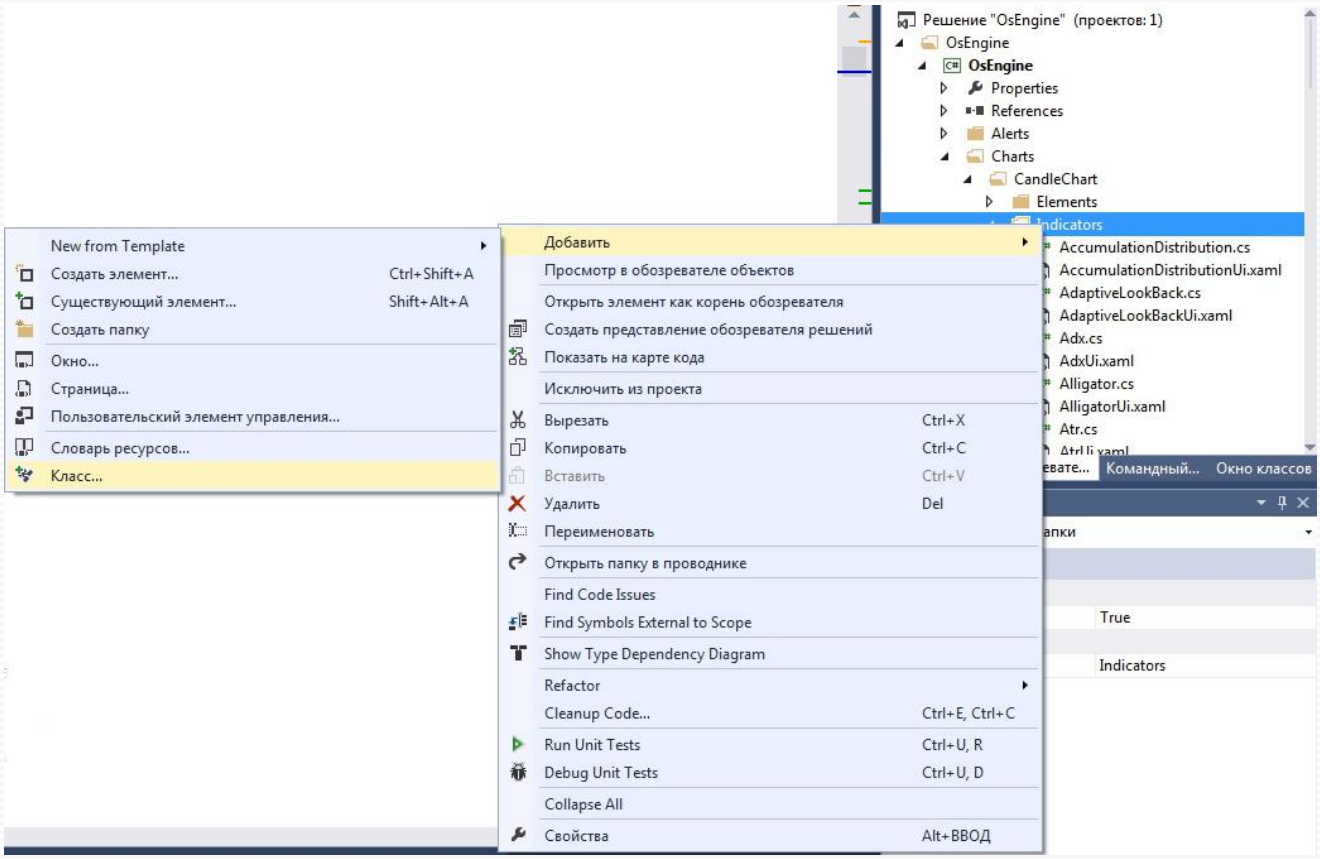
3. Путь свечек к хранилищу индикаторов



4. Создание индикатора

Создание индикатора начинается с создание класса. Для этого надо перейти в Indicators, нажать правую кнопку мыши и выбрать Добавить > Класс

После создания класса он появится в списке ниже.



После этого появится шаблон создания индикаторов.

Для начала модифицируем строку

public class НазваниеКласса : IIndicatorCandle

ChartPainter прорисовывает индикаторы на заданной области. И делает это в зависимости от типа прорисовки индикатора, который задаётся в Классе индикатора. В поле: TypeIndicator

Оно может принимает значения:

- Line /// Линия
- Column /// Столбец
- Point /// Точка

Рассмотрим создание индикатора в OS Engin на примере ATR

Создаем два конструктора для индикатора с сохранением параметров и без

```
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 уникальное имя

// canDelete можно ли пользователю удалить индикатор с графика вручную

// TypeIndicator тип отображения индикатора

// ColorBase цвет индикатора

// PaintOn нужно ли прорисовывать индикатор на графике

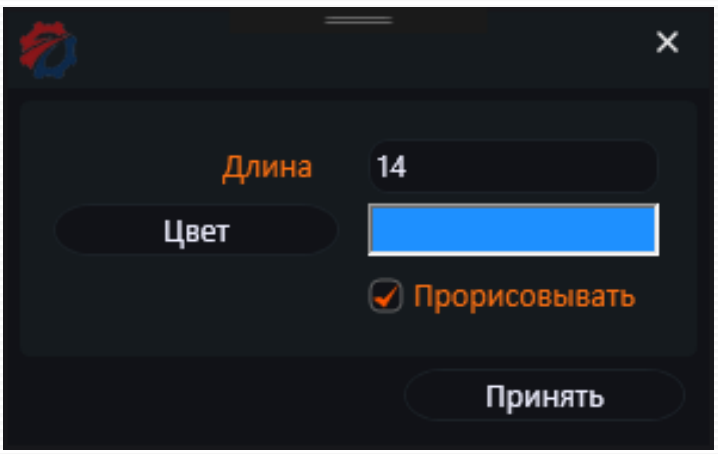
Позже объявляем переменные используемые в индикаторе

Метод Save() отвечает за сохранения параметров введенных пользователем. Все параметры хранятся в новых строках.

Метод Load() загрузка сохраненных параметров при запуске индикатора

Метод ShowDialog() отвечает за вызов графического интерфейса для ввода параметров индикатора пользователем

Для создания графического интерфейса индикатора нужно на Indicators нажать правой кнопкой мыши и выбрать Добавить > Окно и выбрать WPF



Не забываем перегрузить массивы со значениями. Если этого не сделать – прорисовываться ничего не будет. Именно из них ChartPainter получает информацию о том что и как нужно прорисовать.

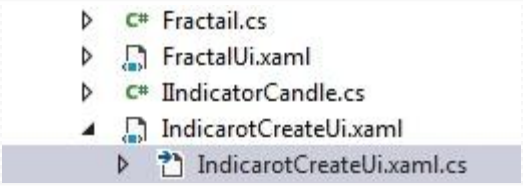
```
ссылок 43
List<List<decimal>> IIndicatorCandle.ValuesToChart
{
    get
    {
        List<List<decimal>> list = new List<List<decimal>>();
        list.Add(Values);
        return list;
    }
}

/// <summary> ...
ссылок 41
List<Color> IIndicatorCandle.Colors
{
    get
    {
        List<Color> colors = new List<Color>();
        colors.Add(ColorBase);
        return colors;
    }

}
```

Лучше пользоваться явной реализацией интерфейса, указывая названия интерфейса перед свойством. Чтобы уберечь пользователя со стороны слоя создания роботов от лишних свойств.

Для индикатора типа Column – в поле Colors должно быть двойное кол-во цветов от кол-ва массивов с данными. Один для растущего столбика, второй для падающего.

В остальных случаях, кол-во цветов должно совпадать с кол-вом массивов данных.

Затем необходимо перейти в класс IndicarotCreateUi и добавить свой индикатор в стандартный набор.



В конструкторе IndicarotCreateUi
_gridViewIndicators.Rows.Add(“Название индикатора, которое будет выводиться”);

В событии gridViewIndicators_SelectionChanged
дописываем

```
if (_gridViewIndicators.SelectedCells[0].Value.ToString() == "Название индикатора")
{
TextBlockDescription.Text = «Краткое описание индикатора";
}
```

В событии ButtonAccept_Click дописываем

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

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

5. Создание индикатора и ChartMaster

После создания класса, необходимо научиться сохранять и загружать индикатор.

Для этого необходимо в метод Load прописать строчки

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