

# OS.ENGINE



## MANUAL

(BOT CREATE)

O-S-A.NET 2016



# Оглавление

1. Идеология.....	3
2. Архитектура создания бота .....	4
3. Создание торгового робота.....	5
4. Возможности для торговли.....	11

# 1. Идеология

- Os.Engine - библиотека для создания роботов для трейдинга.
- Библиотека построена таким образом, что бы можно разделить уровень входа для алготрейдеров и хардкорных программистов. Низкий порог входа достигается за счет разделения слоев внутренней архитектуры. Вместо двух лет(стокШарп, уровень архитектора) - у нас два месяца(как у WealthLab, уровень ниже Джуниора).
- В данном мануале речь пойдет про нулевой уровень. Т.е. не надо будет копаться во внутренностях библиотеки, чтобы сделать простого робота.



- Пара слов об архитектуре:  
1. Тестер



- Робот никогда не знает, торгует ли он на реальном сервере, или на учебном. Архитектура вообще не позволяет написать робота, у которого не будет тестера.

## 2. Архитектура создания бота

Реализация создания бота не совсем очевидна и завязана на наследование, поэтому это почти единственное, в чём надо разобраться перед написанием кода.

Картинка, на которой изображена схема создания бота:

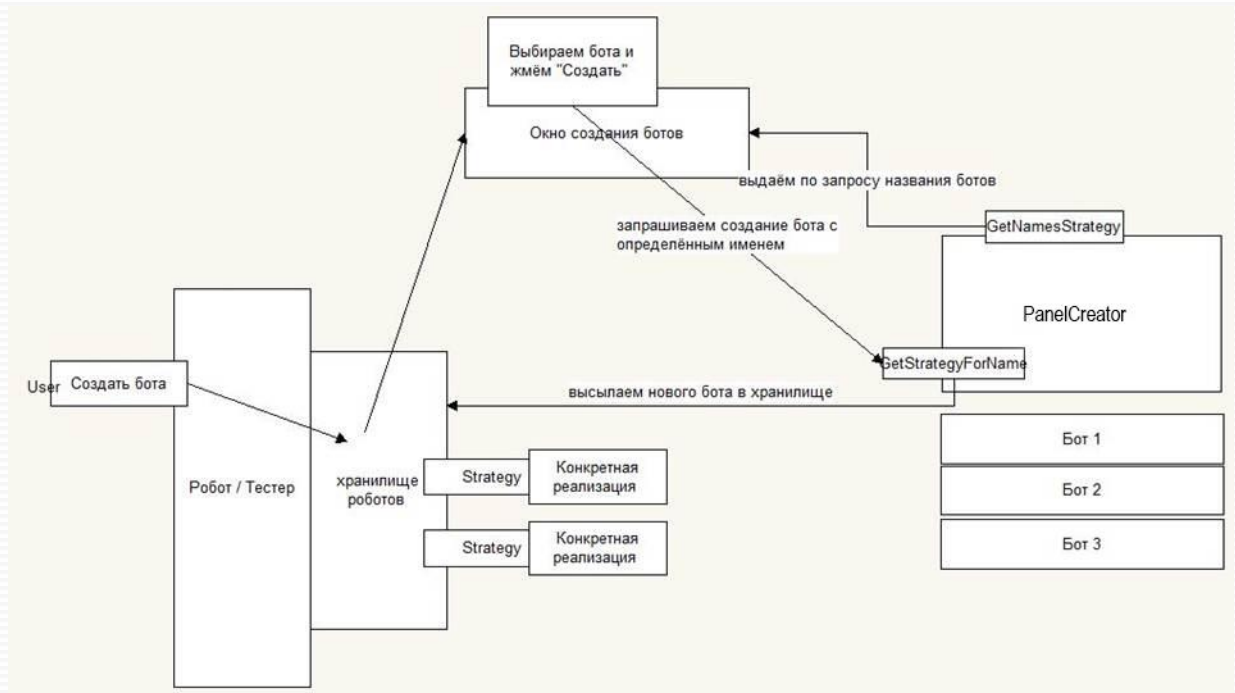


Схема работы (логика работы, когда пользователь работает с терминалом):

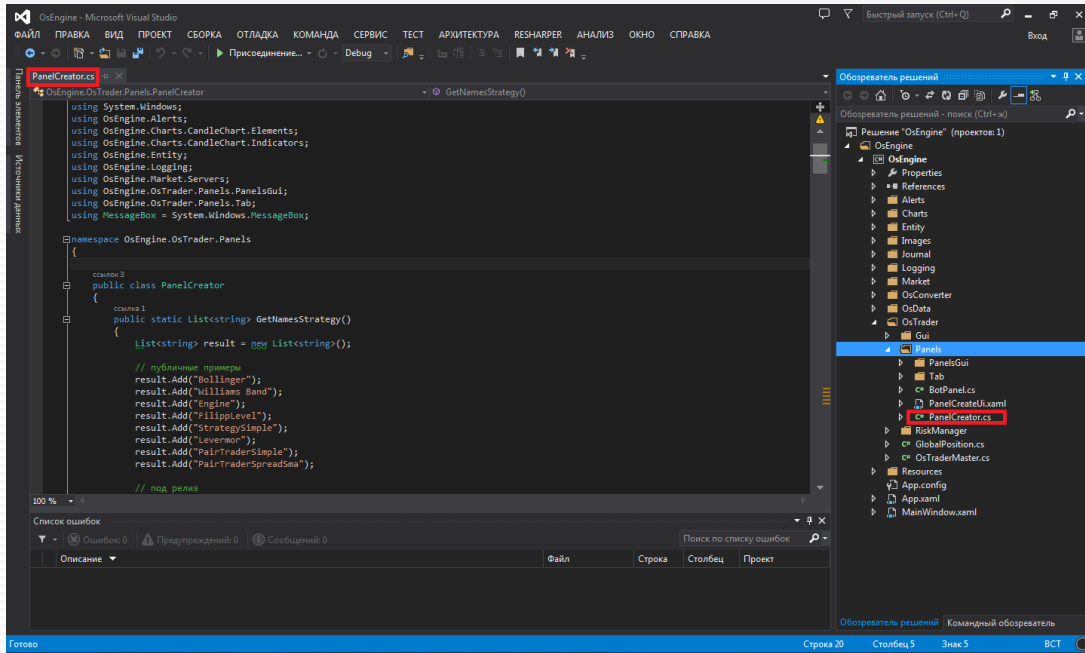
1. Из пользовательского интерфейса происходит сигнал о том, что пользователь хочет создать бота (панель).
2. Хранилище ботов создаёт меню создания ботов.
3. Меню, во время создания, запрашивает названия всех стратегий созданных в системе, у класса **PanelCreator**.
4. Пользователь выбирает тип бота и жмёт "Создать", в это время мы обращаемся к классу **PanelCreator** и просим создать нам бота с таким названием.
5. Робот отправляется в хранилище ботов и появляется у пользователя в интерфейсе.

Если данная схема не совсем понятна, ничего страшного. На практике все гораздо проще, так как вся логика создания бота (панели) находится в одном месте. Так же существуют простые примеры, для изучения.

### 3. Создание торгового робота

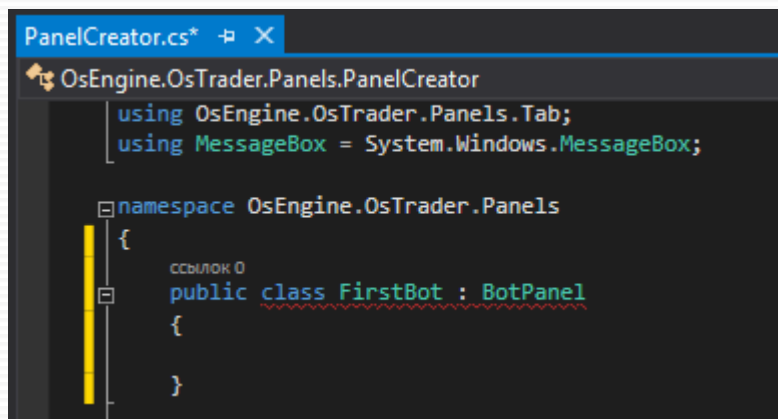
Открываем библиотеку и идём сразу в Os.Trader/Panels/PanelCreator

Всё остальное игнорируем, это нам не нужно. Это для программистов хардкорщиков и архитекторов платформы. **PanelCreator** - для создания ботов. Открываем файл.



Для того чтобы создать своего бота нужно:

1. Создать публичный класс и назначить его наследником класса **BotPanel**.



### 3. Создание торгового робота

- В созданном классе необходимо создать два метода GetNameStrategyType и ShowIndividualSettingsDialog :

```
ссылка 0
public class FirstBot : BotPanel
{
    ссылка 43
    public override string GetNameStrategyType()
    {
        return "FirstBot";
    }

    ссылка 43
    public override void ShowIndividualSettingsDialog()
    {
        MessageBox.Show("У данной стратегии пока нет настроек");
    }
}
```

- 3. Затем нам надо найти класс PanelCreator, а в нем метод GetNamesStrategy и добавить в него информацию о своем классе. После добавления её сюда, библиотека сможет ссылаться на имя вашего робота:

```
PanelCreator.cs*
OsEngine.OsTrader.Panels.PanelCreator

ссылка 3
public class PanelCreator
{
    ссылка 1
    public static List<string> GetNamesStrategy()
    {
        List<string> result = new List<string>();

        result.Add("FirstBot"); добавить строчку

        // публичные примеры
        result.Add("Bollinger");
        result.Add("Williams Band");
    }
}
```

### 3. Создание торгового робота

4. В этом же классе **PanelCreator** находим метод **GetStrategyForName** и добавляем механизм создания бота:

```
ссылка 2
public static BotPanel GetStrategyForName(string nameClass, string name)
{
    BotPanel bot = null;
    // примеры и бесплатные боты

    if (nameClass == "FirstBot")
    {
        bot = new FirstBot(name);
    }

    if (nameClass == "Williams Band")
    {
        bot = new StrategyBillWilliams(name);
    }
}
```

добавить конструкцию

5. Затем возвращаемся в наш созданный класс **FirstBot** и добавим в него конструктор, в дальнейшем мы сможем подписаться на событие завершения свечек или тиков, а также начнем собирать логику самого бота

```
namespace OsEngine.OsTrader.Panels
{
    ссылка 2
    public class FirstBot : BotPanel
    {
        ссылка 43
        public override string GetNameStrategyType()
        {
            return "FirstBot";
        }

        ссылка 43
        public override void ShowIndividualSettingsDialog()
        {
            MessageBox.Show("У данной стратегии пока нет настроек");
        }

        ссылка 1
        public FirstBot(string name) : base(name)
        {
        }
    }
}
```

создаем конструктор

### 3. Создание торгового робота

5. Теперь нам нужно создать вкладку для данного бота **BotTabSimple**

```
namespace OsEngine.OsTrader.Panels
{
    ссылка 2
    public class FirstBot : BotPanel
    {
        ссылка 43
        public override string GetNameStrategyType()
        {
            return "FirstBot";
        }

        ссылка 43
        public override void ShowIndividualSettingsDialog()
        {
            MessageBox.Show("У данной стратегии пока нет настроек");
        }

        ссылка 1
        public FirstBot(string name) : base(name)
        {
            TabCreate(BotTabType.Simple);
            _tab = TabsSimple[0];
        }

        private BotTabSimple _tab;
    }
}
```

**добавляем реализацию**

**создаем переменную BotTabSimple**

#### ВАЖНО

- **BotTabSimple** - является базовой реализацией вкладки или другими словами стандартом для написания множества базовых стратегий, чаще всего это и нужно. У BotTabSimple может быть всего один источник (инструмент), над которым мы можем производить торговые операции.
- Также существуют другие реализации такие как **BotTabIndex**
- **BotTabIndex** – это еще один вид реализации вкладки. Эта реализация позволяет подгружать множественные источники (инструменты) и строить из них индекс (синтетику)
- У одной панели (бота) могут быть несколько вкладок (табов). Например: 3 BotTabSimple и 1 BotTabIndex

После всех этих операций мы можем уже компилировать программу, и на выходе мы получили пустой шаблон робота. Но это еще не полноценный бот.



### 3. Создание торгового робота

6. Теперь нам осталось подписаться на событие завершения свечек **CandleFinishedEvent** и создать метод **TradeLogic** для обработки логики:

```
ссылка 2
public class FirstBot : BotPanel
{
    ссылка 43
    public override string GetNameStrategyType()
    {
        return "FirstBot";
    }

    ссылка 43
    public override void ShowIndividualSettingsDialog()
    {
        MessageBox.Show("У данной стратегии пока нет настроек");
    }

    ссылка 1
    public FirstBot(string name) : base(name)
    {
        TabCreate(BotTabType.Simple);
        _tab = TabsSimple[0];

        _tab.CandleFinishedEvent += TradeLogic; подписываемся на событие завершения свечи
    }

    private BotTabSimple _tab;

    ссылка 1
    private void TradeLogic(List<Candle> candles) торговая логика
    {
    }
}
```

Теперь мы можем вписать торговую логику в наш метод **TradeLogic**. В этот метод мы уже передаем наши свечи (candles) инструмента, с которыми мы будем обрабатывать нашу логику.

Вот простое техническое задание для нашего примера:

1. если свечей меньше чем 5 - выходим из метода.
2. если уже есть открытые позиции – закрываем и выходим.
3. если позиций нет, то смотрим в какую сторону заходить. Если закрытие последней свечи выше закрытия предыдущей – покупаем.
4. если закрытие последней свечи ниже закрытия предыдущей, продаем.

### 3. Создание торгового робота

Получаем:

```
ссылка 1
private void TradeLogic(List<Candle> candles)
{
    // 1. если свечей меньше чем 5 - выходим из метода.
    if (candles.Count < 5)
    {
        return;
    }

    // 2. если уже есть открытые позиции - закрываем и выходим.
    if (_tab.PositionsOpenAll != null && _tab.PositionsOpenAll.Count != 0)
    {
        _tab.CloseAllAtMarket();
        return;
    }

    // 3. если закрытие последней свечи выше закрытия предыдущей - покупаем.
    if (candles[candles.Count - 1].Close > candles[candles.Count - 2].Close)
    {
        _tab.BuyAtMarket(1);
    }

    // 4. если закрытие последней свечи ниже закрытия предыдущей, продаем.
    if (candles[candles.Count - 1].Close < candles[candles.Count - 2].Close)
    {
        _tab.SellAtMarket(1);
    }
}
```

Все, наш простой бот готов. Если мы его запустим он будет работать:



Текущий пример раскрывает только базовые конструкции которые необходимы для написания любого робота.

Более сложные примеры можно рассмотреть самостоятельно в примерах, которые доступны вместе с библиотекой.

## 4. Возможности для торговли

Рассмотрим каким стандартным арсеналом возможностей мы обладаем для работы с рынком:

### Подписка на события:

- |                               |   |
|-------------------------------|---|
| • BestBidAskChangeEvent       | – изменился лучший бид/аск.                     |
| • CandleFinishedEvent         | – завершилась последняя свечка.                 |
| • CandleUpdateEvent           | – обновилась последняя свечка.                  |
| • FirstTickToDayEvent         | – утренняя сессия стартовала. Пошли первые тики |
| • LogMessageEvent             | – исходящее сообщения для лога.                 |
| • MarketDepthUpdateEvent      | – пришел  |
| • NewTickEvent                | – пришли новые тики.                            |
| • OrderUpdateEvent            | – в системе обновился какой-то ордер.           |
| • PositionClosingFailEvent    | – закрытие позиции не произошло.                |
| • PositionClosingSuccessEvent | – позиция успешно закрыта.                      |
| • PositionOpeningFailEvent    | – открытие позиции не произошло.                |
| • PositionOpeningSuccessEvent | – позиция успешно открыта.                      |
| • ServerTimeChangeEvent       | – изменилось время сервера.                     |

### Методы торговли:

- |                           |                         |  |
|---------------------------|-------------------------|--|
| • BuyAtAceberg            | SellAtAceberg           | - покупка, продажа айсбергом                               |
| • BuyAtAcebergToPosition  | SellAtAcebergToPosition | - добавить в позицию новую айсберг заявку                  |
| • BuyAtLimit              | SellAtLimit             | - покупка, продажа по определенной цене                    |
| • BuyAtLimitToPosition    | SellAtLimitToPosition   | - добавить в позицию новую лимитную заявку                 |
| • BuyAtMarket             | SellAtMarket            | - покупка, продажа по любой цене                           |
| • BuyAtMarketToPosition   | SellAtMarketToPosition  | - добавить в позицию новую маркет заявку                   |
| • BuyAtStop               | SellAtStop              | - покупка, продажа по пробитию цены                        |
| • BuyAtStopCancel         | SellAtStopCancel        | - отменить все заявки по пробитию цены                     |
|                           |                         |  |
| • CloseAllAtMarket        |                         | - закрыть все позиции по рынку                             |
| • CloseAllOrderInSystem   |                         | - отозвать все открытые роботом ордера из системы          |
| • CloseAllOrderToPosition |                         | - отозвать все ордера из системы, связанные с этой сделкой |
| • CloseAtAceberg          |                         | - закрыть позицию айсбергом по определенной цене           |
| • CloseAtLimit            |                         | - закрыть позицию по определенной цене                     |
| • CloseAtMarket           |                         | - закрыть позицию по текущей цене                          |
| • CloseAtProfit           |                         | - выставить профит-ордер для позиции                       |
| • CloseAtStop             |                         | - выставить стоп-ордер для позиции                         |
| • CloseAtTrailingStop     |                         | - выставить трейлинг стоп – ордер для позиции              |
| • CloseOrder              |                         | - отозвать ордер   |

## 4. Возможности для торговли

### Доступные данные:

- PositionAll - все позиции принадлежащие боту
- PositionOpenLong - взять все позиции лонг
- PositionOpenShort - взять все позиции шорт
- PositionsLast - последняя открытая позиция
- PositionsCloseAll - все закрытые позиции бота
- PositionsOpenAll - все открытые позиции бота
  
- PriceBestAsk - лучшая цена покупки инструмента
- PriceBestBid - лучшая цена продажи инструмента
  
- PriceCenterMarketDepth - цена центра стакана
- Securiti - инструмент для торговли
- ServerStatus - статус сервера
- Trades - все тики по инструменту
- MarketDepth - данные стакана