C# Professional

Атрибуты.

После урока обязательно

Повторите этот урок в видео формате на ITVDN.com

Проверьте как Вы усвоили данный материал на TestProvider.com

Атрибуты.

Attributes

Существует два типа атрибутов:

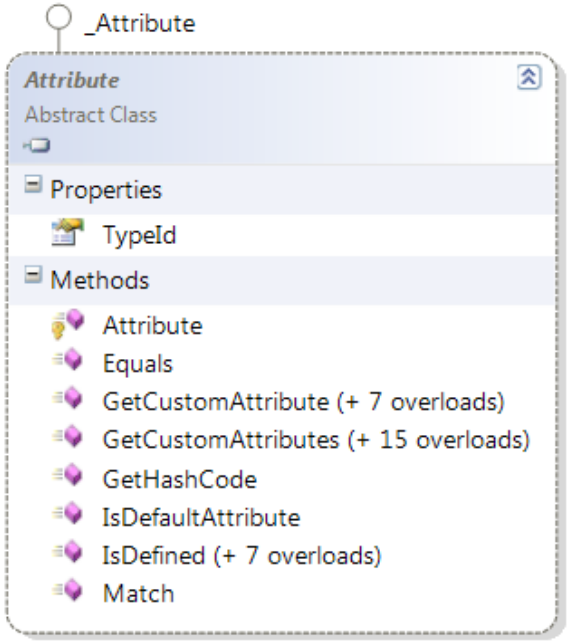
Предопределенные атрибуты (идут в поставке FCL), и пользовательские атрибуты, создаваемые пользователем для добавления в код дополнительных сведений. С точки зрения разработчика оба типа имеют одинаковый синтаксис.

Пользовательский атрибут представляет собой обычный класс, унаследованный от класса `Attribute`.

Такой атрибут может быть использован для любого метода, свойства, класса или сборки при использовании следующего синтаксиса:

```
[ИмяАтрибута(необязательный параметр, дополнительные пары имя=значение)]
```

Класс



`Attribute` - ключевой класс при работе с атрибутами.

Все системные классы-атрибуты наследуются от данного класса. При создании собственных атрибутов, также необходимо наследоваться от данного класса.

Custom Attributes

Правила создания пользовательского атрибута:

1. Имя атрибута должно содержать суффикс `Attribute` *(Данное правило является рекомендацией, но его лучше придерживаться).*

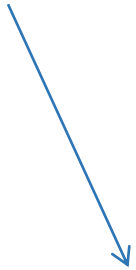
2. Класс-атрибут обязан наследоваться от системного класса `System.Attribute`

3. Класс-атрибут обязан быть декорирован атрибутом `[AttributeUsageAttribute]`

Параметры

```
[MyAttribute("MyPositionalParam", MyNamedParam = 123)]
```

Позиционные параметры:
аргументы конструктора класса атрибута
(всегда указываются перед именованными)



Именованные параметры:
все открытые нестатические поля и свойства
класса атрибута, доступные для записи

Атрибут

Системный атрибут `AttributeUsage`, необходим для создания пользовательских атрибутов.

Позиционные параметры:

- **`validOn`** – имеет тип `AttributeTargets`, указывает, к чему можно применять данный атрибут.

Именованные параметры:

- **`AllowMultiple`** – имеет тип `bool`, разрешает или запрещает множественное применение атрибута (Значение по умолчанию - `false`).
- **`Inherited`** – имеет тип `bool`, разрешает или запрещает наследование атрибута в производных классах (Значение по умолчанию - `true`).

Атрибут

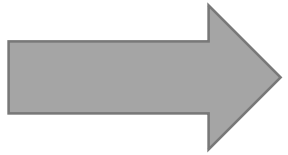
Системный атрибут `ObsoleteAttribute`, позволяет отмечать устаревшие элементы системы.

Позиционные параметры:

- **`message`** – имеет тип `string`, содержит строку сообщения, которая показывается при попытке использования устаревшего элемента.
- **`error`** – имеет тип `bool`, если установлен в `true` – использование элемента, помеченного атрибутом, будет порождать ошибку на этапе компиляции.

Атрибут

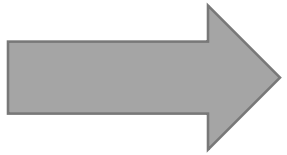
```
[OnMethodBoundaryAspect]
void TargetCode()
{
        DoSomething()
};
```



```
{
    OnEntry();
    try
    {
        TargetCode();
        OnSuccess();
    }
    catch (Exception ex)
    {
        OnException();
    }
    finally
    {
        OnExit();
    }
}
```

Атрибут

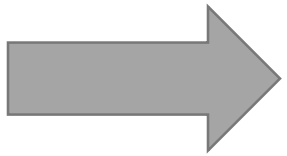
```
[OnExceptionAspect]
void TargetCode()
{
        DoSomething()
};
```



```
{
    try
    {
        TargetCode();
    }
    catch (Exception ex)
    {
        OnException();
    }
}
```

Атрибут

```
[LocationInterceptionAspect ]
public double X { get; set; }
```



```
public string MyProperty
{
    get
    {
        OnGetValue();
        return _myProperty;
    }
    set
    {
        OnSetValue();
        _myProperty = value;
    }
}
```

Кортежи

Кортеж — упорядоченный набор значений фиксированной длины.

Кортежи доступны в C# 7.0 и более поздних версиях. Они предоставляют краткий синтаксис для группирования нескольких элементов данных в упрощенную структуру данных.

Типы кортежей являются структурными типами, а элементы кортежа — публичными полями.

Кортежи могут использоваться для возврата множественных значений из методов, для группировки логически связанных переменных без необходимости определять новый тип, как структурная альтернатива анонимным типам.

```
(int, string) t1 = (3, "Hello World!");
```

Q&A

Информационный видеосервис для разработчиков программного обеспечения