Атрибуты

№ урока: 7 **Курс:** C# Professional

Средства обучения: Компьютер с установленной Visual Studio

Обзор, цель и назначение урока

Урок посвящен работе с системными и пользовательскими атрибутами. Рассматриваются понятия атрибутов, принципы их создания и применения. Урок позволяет научиться вводить в свою программу информацию декларативного характера, с помощью которой можно определить дополнительные сведения, связанные с классом, структурой, методом и т.д.

Изучив материал данного занятия, учащийся сможет:

- Понимать основы аспектно-ориентированного подхода в программировании.
- Создавать пользовательские атрибуты.
- Применять собственные и системные атрибуты.
- Использовать механизмы рефлексии для определения атрибутов.

Содержание урока

1. Общее понятие атрибутов.
2. Класс `System.Attribute`.
3. Позиционные и именованные параметры атрибутов.
4. Создание атрибутов.
5. Использование атрибутов.
6. Определение атрибутов при помощи рефлексии.
7. Системные атрибуты.

Резюме

- Коллекция `Attribute` связывает предопределенную системную информацию или заданную пользователем информацию с целевым элементом. Целевым элементом может быть: сборка, класс, конструктор, делегат, перечисление, событие, поле, интерфейс, метод, переносимый исполняемый (PE) файл, модуль, параметр, свойство, возвращаемое значение, структура или другой атрибут.
- Информация, предоставляемая атрибутом, называется также метаданными. Метаданные можно анализировать в приложении во время выполнения, для того чтобы управлять тем, как это приложение осуществляет обработку данных, или до времени выполнения внешними средствами для управления обработкой и выполнением самого приложения. Например, на платформе .NET Framework предопределены и используются типы атрибутов для управления поведением времени выполнения, и некоторые языки программирования используют типы атрибутов для представления языковых функций, не поддерживаемых непосредственно общей системой типов .NET Framework.
- Все типы атрибутов прямо или косвенно наследуются от класса `Attribute`. Атрибуты могут быть применены к любому целевому элементу; несколько экземпляров атрибута могут быть применены к одному и тому же целевому элементу; атрибуты могут наследоваться элементом, являющимся производным от целевого элемента.
- Позиционные параметры указываются в порядке, который определяется списком параметров конструктора атрибута. Позиционные параметры всегда должны быть указаны при назначении атрибута.
- Именованные параметры отсутствуют в списке параметров конструктора атрибута. Значения, задаваемые для именованных параметров, используются для инициализации полей и свойств создаваемого экземпляра атрибута. Список именованных параметров указывается через запятую после списка позиционных

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine
Tel. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com

параметров. Каждый именованный параметр определяется как: ***Имя_параметра = Значение_параметра***.

- В C# предопределено много системных (встроенных) атрибутов, но особое значение имеют следующие три: `AttributeUsage`, `Conditional`, `Obsolete`.
- `AttributeUsage` – определяет типы элементов, к которым может быть применен объявляемый атрибут, а также возможность неоднократного применения атрибута и возможность наследования атрибута производными классами. За типы элементов, к которым может быть применен атрибут отвечает позиционный параметр `AttributeTargets`, допустимые значения для этого перечисления можно детальнее посмотреть на MSDN. Именованный параметр `AllowMultiple` может принимать значения `true` или `false` и отвечает за возможность неоднократного применения атрибута. И, наконец, за возможность наследования атрибутов отвечает именованный параметр `Inherited`.
- `Conditional` – позволяет создавать условные методы, которые вызываются только в том случае, если с помощью директивы `#define` определен конкретный идентификатор, а иначе метод пропускается. Следовательно, условный метод служит альтернативной условной компиляции по директиве `#if`. Для применения данного атрибута в исходный код программы следует включить пространство имен `System.Diagnostics`. На условные методы накладываются следующие ограничения:
  - Они должны возвращать значение типа `void`.
  - Должны быть членами класса или структуры, но не интерфейса.
  - Они не могут предшествовать ключевому слову `override`.
- `Obsolete` – позволяет пометить элемент программы как устаревший. Существует несколько способов применения данного атрибута. Первый выглядит так: `[Obsolete("сообщение")]`. Применяется, когда необходимо просто пометить код, как устаревший и вывести соответствующее сообщение. Вторая форма: `[Obsolete("сообщение", ошибка)]`. Применяется, когда кроме вывода сообщения также необходимо запретить компиляцию.

Закрепление материала

- Что такое атрибут?
- Как создать пользовательский атрибут? Перечислите все требуемые шаги и ограничения.
- В чем отличие между позиционными и именованными параметрами атрибутов?
- К чему применимы атрибуты?
- Как при помощи рефлексии получить информацию об атрибутах, примененных к классу или его методам?
- Перечислите известные вам системные атрибуты. Каково их назначение?

Дополнительное задание

Создайте пользовательский атрибут `AccessLevelAttribute`, позволяющий определить уровень доступа пользователя к системе. Сформируйте состав сотрудников некоторой фирмы в виде набора классов, например, `Manager`, `Programmer`, `Director`. При помощи атрибута `AccessLevelAttribute` распределите уровни доступа персонала и отобразите на экране реакцию системы на попытку каждого сотрудника получить доступ в защищенную секцию.

Самостоятельная деятельность учащегося

Задание 1
Выучите основные конструкции и понятия, рассмотренные на уроке.

Задание 2
Создайте класс и примените к его методам атрибут `Obsolete` сначала в форме, просто выводящей предупреждение, а затем в форме, препятствующей компиляции. Продемонстрируйте работу атрибута на примере вызова данных методов.

CyberBionic systematics
CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine
Tel. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com

Задание 3
Расширьте возможности программы-рефлектора из предыдущего урока следующим образом:

1. Добавьте возможность выбирать, какие именно члены типа должны быть показаны пользователю. При этом должна быть возможность выбирать сразу несколько членов типа, например, методы и свойства.
2. Добавьте возможность вывода информации об атрибутах для типов и всех членов типа, которые могут быть декорированы атрибутами.

Задание 4
Зайдите на сайт MSDN.
Используя поисковые механизмы MSDN, найдите самостоятельно описание темы по каждому примеру, который был рассмотрен на уроке, так, как это представлено ниже, в разделе «Рекомендуемые ресурсы», описания данного урока. Сохраните ссылки и дайте им короткое описание.

Рекомендуемые ресурсы

MSDN: класс `System.Attribute`
http://msdn.microsoft.com/ru-ru/library/system.attribute.aspx

MSDN: использование атрибутов
http://msdn.microsoft.com/ru-ru/library/z0w1kczw.aspx

MSDN: общие атрибуты
http://msdn.microsoft.com/ru-ru/library/z371wyft.aspx

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com