Модуль 13: Атрибуты

Обзор

- Обзор атрибутов
- Создание пользовательских атрибутов
- Получение значения атрибута

Обзор атрибутов

- Понятие атрибутов
- Применение атрибутов
- Использование стандартных атрибутов
- Использование атрибута Conditional

Понятие атрибутов

- **Атрибуты - это:**
  - Описательные тэги в программном коде, передающие информацию во время выполнения программы
  - Хранятся вместе с метаданными элемента
- **.NET Framework содержит множество встроенных атрибутов**
  - Среда выполнения содержит код, проверяющий значения атрибутов и меняет свое поведение в соответствии с этими значениями

Применение атрибутов

- **Синтаксис: Для использования атрибута необходимо указать его имя в квадратных скобках**

```
[attribute(positional_parameters,named_parameter=value, ...)]
element
```

- **Можно указать несколько атрибутов для одного элемента:**
  - Заключить каждый из атрибутов в отдельные квадратные скобки
  - Использовать одни квадратные скобки и перечислить атрибуты через запятую
  - В некоторых случаях необходимо явно указать имя элемента, которому принадлежит атрибут

Использование стандартных атрибутов

- **В .NET определено большое количество стандартных атрибутов**
  - Пример: Использование атрибута Conditional

Использование атрибута Conditional

- **Используется как инструмент отладки**
  - Производит условную компиляцию вызовов метода в зависимости от значения параметра, определяемого программным путем
  - Не производит условную компиляцию самих методов
- **Ограничения на методы:**
  - Должны возвращать тип **void**
  - Не должны быть объявлены как **override**
  - Не должны быть методами наследуемыми от интерфейса

```
using System.Diagnostics;
...
class MyClass
{
  [Conditional ("DEBUGGING")]
  public static void MyMethod( )
  {
    ...
  }
}
```

Применение атрибута DllImport

- **С атрибутом DllImport можно:**
  - Вызвать неуправляемый код DLLs из C# -приложения
  - Указать метку метода в неуправляемом коде DLL

```
using System.Runtime.InteropServices;
...
public class MyClass( )
{
  [DllImport("MyDLL.dll", EntryPoint="MyFunction")]
  public static extern int MyFunction(string param1);
  ...
  int result = MyFunction("Hello Unmanaged Code");
  ...
}
```

Создание пользовательских атрибутов

- Определение области действия пользовательского атрибута
- Создание класса атрибута
- Обработка пользовательского атрибута
- Использование нескольких атрибутов

Определение области действия пользовательского атрибута

- Для определения области действия используйте тэг атрибута AttributeUsage

```
[AttributeUsage(AttributeTargets.Method)]
public class MyAttribute: System.Attribute
{ ... }
```

- Для определения нескольких элементов необходимо использовать оператор «|»

```
[AttributeUsage(AttributeTargets.Class | AttributeTargets.Struct)]
public class MyAttribute: System.Attribute
{ ... }
```

- Спецификация используемости атрибута

```
[AttributeUsage(доступные_элементы, AllowMultiple=true_или_false,
  Inherited=наследуемость )]
```

Создание класса атрибута

- **Наследование класса атрибута**
  - Все классы атрибутов должны наследоваться от System.Attribute
  - Добавляйте к имени класса атрибута слово “Attribute”
- **Компоненты класса атрибута**
  - Для каждого класса атрибута определите один конструктор, устанавливающий обязательную информацию
  - Создайте свойства для передачи дополнительных именованных параметров.

Обработка пользовательского атрибута

Процесс компиляции

1. **Поиск класса атрибута**
2. **Проверка области действия атрибута**
3. **Проверка конструктора атрибута**
4. **Создание экземпляра объекта**
5. **Проверка именованных параметров**
6. **Установка для поля или свойства значения именованного параметра**
7. **Сохраняется текущее состояние класса атрибута**

Использование нескольких атрибутов

- **Для одного элемента можно определить несколько атрибутов**
  - Определяйте каждый атрибут по отдельности
- **Для одного элемента можно определить несколько экземпляров одного и того же атрибута**
  - Используйте AllowMultiple = true

Получение значения атрибута

- Проверка метаданных класса
- Запрос информации об атрибуте

Запрос информации об атрибуте

- **Для получения информации об атрибуте:**
  - Используйте **GetCustomAttributes** для получения всей информации об атрибуте в виде массива

```
System.Reflection.MemberInfo typeInfo;
typeInfo = typeof(MyClass);
object[ ] attrs = typeInfo.GetCustomAttributes(false);
```

  - Производите итерации по всем элементам массива и проверяйте хранимые в нем значения

Проверка метаданных класса

- **Для запроса информации о метаданных класса:**
  - Используйте класс MemberInfo пространства имен System.Reflection
  - Заполните объект MemberInfo, используя System.Type
  - Создайте объект System.Type, используя оператор typeof
- **Пример**

```
System.Reflection.MemberInfo typeInfo;
typeInfo = typeof(MyClass);
```

Лабораторная работа 13: Создание и использование атрибутов