Атрибути

Атрибутите – свързват информация към тип или член:
- **по време на проектиране (като документация);**
- **по време на изпълнение (име на атрибут в база данни);**
- **поведенчески характеристики по време на изпълнение (участие на даден член в транзакция).**

1. Дефиниране на атрибути

а) производен клас на базовия клас System.Attribute

б) суфикс на атрибутния клас Attribute

в) суфиксът се изпуска при използването.

Атрибут STAThread **(STA – Single Threaded Apartment) – модел за приложението с една нишка.**

```
class Class1
{
   [STAThread]
   static void Main (string[] args)
   {
   }
}
```

2. Видове атрибути

а) атрибути на клас;

б) атрибут на метод;

в) атрибут на поле.

Пример: Атрибут на клас – атрибут на отдалечен сървър, върху който ще се създава обект (отпада необходимостта да се пази информацията в константа или ресурсен файл).

```
using System;
public enum RemoteServers
{
   AERO,
   ECAD,
   ALICE
}
public class RemoteObjectAttribute : Attribute
{
   protected RemoteServers server;
   public string Server
   {
      get
      {   return RemoteServers.GetName
                 (typeof(RemoteServers), this.server);
      }
   }
```

Атрибут на обект върху отдалечен сървър

Enum.GetName **връща името на константа от изброимия тип**

```
   public RemoteObjectAttribute(RemoteServers Server)
   {
      this.server = Server;
   }
}

[RemoteObject (RemoteServers.ALICE)]
class MyRemoteClass
{
}
```

Съкратеното име на атрибут се използва само за присъединяване на атрибут

```
class Test
{
   [STAThread]
   static void Main (string[] args)
   {   Type type = typeof(MyRemoteClass);
       foreach(Attribute attr in type.GetCustomAttributes(true))
       {   RemoteObjectAttribute remoteAttr = attr as
                                              RemoteObjectAttribute;
           if (null != remoteAttr)
              Console.WriteLine ("Създава обекта върху {0}.",
                                 remoteAttr.Server);
       }
   }
}
```

Връща масив от тип Attribute **с всички атрибути за даден клас –** true **показва да се търси в наследниците**

Пълното име на атрибут се използва при търсенето

Резултати:

Създава обекта върху ALICE.

Пример: Атрибут на метод – методът е достъпен при транзакциите.

```
using System;
using System.Reflection;
namespace MethodAttribs
{
   public class TransactionableAttribute : Attribute
   {
      public TransactionableAttribute()
      {
      }
   }
```

Достъпност на метод при транзакция

Конструктор без параметри

```
class SomeClass
{
   [Transactionable]
   public void Foo()
   {
   }
   public void Bar()
   {
   }
   [Transactionable]
   public void Goo()
   {
   }
}
```

При добавянето на атрибута не се включват скоби (конструктор без параметри)

```
class Test
{
  [STAThread]
  static void Main(string[] args)
  {  Type type = Type.GetType("MethodAttribs.SomeClass");
     foreach (MethodInfo method in type.GetMethods())
       foreach (Attribute attr in method.GetCustomAttributes(true))
         if (attr is TransactionableAttribute)
           Console.WriteLine ("{0} е достъпен при транзакция.",
                              method.Name);
  }
 }
}
```

Статичен метод Type.GetType – **връща типа на обекта**

Връща масив от System.Reflection.MethodInfo **обекти**

MethodInfo.GetCustomAttributes **връща потребителските атрибути за метод**

Резултати:

Foo е достъпен при транзакция.
Goo е достъпен при транзакция.

Пример: Атрибут на поле – стойности на полета на клас се запазват в Registry. Дефинира се атрибут с конструктор с два параметъра: коректен регистров ключ от изброим тип и име на стойността на Registry. Търси се регистров ключ на поле.

```
using System;
using System.Reflection;
namespace FieldAttribs
{
   public enum RegHives
   {    HKEY_CLASSES_ROOT,
        HKEY_CURRENT_USER,
        HKEY_LOCAL_MACHINE,
        HKEY_USERS,
        HKEY_CURRENT_CONFIG
   }
```

```
   public class RegKeyAttribute : Attribute
   {
      public RegKeyAttribute(RegHives Hive,String ValueName)
      {
         this.Hive = Hive;
         this.ValueName = ValueName;
      }
      protected RegHives hive;
      public RegHives Hive
      {  get  {  return hive;  }
         set  {  hive = value; }
      }
      protected String valueName;
      public String ValueName
      {  get  {  return valueName;  }
         set  {  valueName = value;}
      }
   }
```

Регистров ключ

```
class SomeClass
{
   [RegKey(RegHives.HKEY_CURRENT_USER, "Foo")]
   public int Foo;
   public int Bar;
}
```

```
class Test
{   [STAThread]
    static void Main (string[] args)
    { Type type = Type.GetType("FieldAttribs.SomeClass");
      foreach (FieldInfo field in type.GetFields())
        foreach (Attribute attr in
                                  field.GetCustomAttributes(true))
        {   RegKeyAttribute rka = attr as RegKeyAttribute;
            if (null != rka)
              Console.WriteLine("{0} ще се запази в {1}\\\\{2}",
                      field.Name, rka.Hive, rka.ValueName);
        }
    }
}
```

Връща масив от FieldInfo обекти, представящ всички public полета

Резултати:

Foo ще се запази в HKEY_CURRENT_USER\\Foo

3. Параметри на атрибути

а) позиционни параметри – дефинират се в конструктора на атрибутите; определят се всеки път при използване на атрибут.

б) наименовани параметри

- **не се дефинират в конструктора на атрибута;**
- **те са нестатични полета и свойства;**
- **установяват се при създаване екземпляр на атрибута;**
- **използват синтаксиса:**

<име_на_поле_или_свойство> = <стойност>

Наименован параметър може да бъде всяко поле, което не е readonly, static или const и всяко свойство, което включва нестатичен достъп set.

Пример: Нека RegKeyAttribute.ValueName е позиционен параметър, а RegKeyAttribute.Hive – незадължителен наименован параметър – премахваме го от дефиницията на конструктора.

```
public RegKeyAttribute (String ValueName)
{
   this.ValueName = ValueName;
}
```

Потребителят добавя атрибута по два начина:

I начин:

```
[RegKey("Foo",Hive=RegHives.HKEY_LOCAL_MACHINE)]
public int Foo;
```

Резултати:

Foo ще се запази в HKEY_LOCAL_MACHINE\\Foo

II начин:

```
[RegKey("Foo")]
public int Foo;
```

Резултати:

Foo ще се запази в HKEY_CLASSES_ROOT\\Foo

Взима се подразбиращата се стойност 0 на полето Hive.

За да не се вземе подразбиращата се стойност 0, трябва да се промени изброимият тип:

```
public enum RegHives
{  HKEY_CLASSES_ROOT=1,
   HKEY_CURRENT_USER,
   HKEY_LOCAL_MACHINE,
   HKEY_USERS,
   HKEY_CURRENT_CONFIG
}
```

Трябва да се инициализира стойността на Hive, за да не се предефинира от потребителя чрез наименован параметър:

```
public RegKeyAttribute(String ValueName)
{  if(this.Hive == 0)
      this.Hive = RegHives.HKEY_CURRENT_USER;
   this.ValueName = ValueName;
}
```

4. Правила:

а) първо се определят позиционните параметри, а след това наименованите в произволен ред;

б) позиционен параметър не може да се именува;

в) наименованите параметри са полета с public достъп или свойства с нестатичен set метод;

г) параметрите на атрибутите могат да бъдат:

- **bool, byte, char, double, float, int, long, short, string;**
- **System.Type;**
- **object;**
- **enum и тип, в който е вграден, с public достъп;**
- **едномерен масив от горните типове.**

д) конструкторът на атрибут не може да има клас като параметър (атрибутите се присъединяват по време на проектирането – все още не са създадени обекти на класове).

5. Предефинирани атрибути

.NET атрибут	Отнася се за	Описание
AttributeUsage	клас	Определя правилното използване на друг атрибутен клас.
CLSCompliant	всички	Показва съвместимост на програмния елемент със CLS (Common Language Specification).
Conditional	метод	Компилаторът ще компилира метод, за който е дефиниран дадения низ.
DllImport	метод	Определя местоположението на DLL, съдържащ реализацията на външен метод.
MTAThread	Метод (Main)	Подразбира се многонишков модел за приложението.
NonSerialized	поле	Полетата не ще бъдат сериализирани.
Obsolete	всички без асембли, модул, параметър и return	Остарял елемент, който ще бъде премахнат в бъдещите версии на продукта.
ParamArray	параметър	Единствен параметър може да се разглежда като params.

.NET атрибут	Отнася се за	Описание
Serializable	клас, структура, изброим тип, делегат	Всички public и private полета могат да се сериализират.
STAThread	Метод (Main)	Подразбира се еднонишков модел за приложението.
StructLayout	клас, структура	Определя Auto, Explicit или Sequential данни.
ThreadStatic	поле (static)	Реализира локално нишково съхранение (TLS) – всяка нишка има свое собствено копие на статично поле, което не се споделя от нишките.

Пример: Атрибутът Conditional определя, че даден метод ще се компилира в код, само ако е дефинирана препроцесорната директива "DEBUG".

```
using System.Diagnostics;
...
[Conditional ("DEBUG")]
public void SomeDebugFunc()
{
        Console.WriteLine("SomeDebugFunc");
}
```

Пример: Атрибут Obsolete – ако вторият параметър е true, компилаторът ще изведе съобщение за грешка при извикване на метода; ако е false – кодът се компилира без предупреждения и грешки; първият параметър е част от диагностиката на компютъра при генериране на грешка.

```
using System;
...
[Obsolete ("Не използвайте OldFunc, използвайте вместо
 нея NewFunc", true)]
public void OldFunc()
{
    Console.WriteLine("Стара");
}
public void NewFunc()
{
    Console.WriteLine("Нова");
}
```