属性パラメータ

名前付きパラメータ

前回の「属性」の章をよく理解すれば、属性がクラスやメンバにコンパイル時に関連付けられ
これをリフレクション API で調べることによって、実行時に
動的にクラスの性質を知る手段が、言語レベルで確立されたととがわかります

しかし、属性の種類によっては、以前の初期化方法では冗長になることもあります
例えば、そのクラスの著作者をあらわす属性を作成したとしましょう
その属性が公開され、標準化されているならばともかく
そうでなければ、著作者はほとんど同じ人物、または同じ社名になるでしょう
それならば、デフォルトの値を決め、必要な時だけ選択できれば便利というものです

前回のサンプルプログラムのような、属性クラスのコンストラクタが要求する
アタッチする時に必ず指定しなければならない属性値を**位置パラメータ**と呼びます
位置パラメータは、省略することはできません

これに対して、省略可能な**名前付きパラメータ**と呼ばれる属性値があります
名前付きパラメータとは、属性クラスのコンストラクタで設定されない
readonly、static、const ではないメンバ、または static ではないプロパティを表します

これらの名前付きパラメータのデフォルト値は、コンストラクタで代入します
では、デフォルト値以外を代入したい時はどのようにすれば良いのでしょうか
名前付きパラメータを指定するには、位置パラメータの次に**代入式**を指定します

[Attribute(parameter , member = value , ...)] ...

位置パラメータは確実に指定しなければならないので、最初に指定します
その後、必要であれば、メンバ名を指定して属性クラスのメンバに代入を行えるのです
属性クラスは、コンストラクタで何らかの値が代入されたかを調べ
メンバが初期化されていなければ、デフォルトの値を与えるなどの処置が行えます

```
using System;

enum KittyName { RENA , YUKI , MIMI }

class KittyAttribute : Attribute {
 public readonly KittyName name;
 public String strTitle;
 public KittyAttribute(KittyName name) {
  this.name = name;
  if (strTitle == null) strTitle = "Kitty on your lap";
 }
}

[Kitty(KittyName.RENA)] class Kitty1 {}
[Kitty(KittyName.YUKI , strTitle = "Hiza no ue no pertner")] class Kitty2 {}

class Test {
 public static void Main() {
  WriteAtt(typeof(Kitty1));
  WriteAtt(typeof(Kitty2));
 }
 private static void WriteAtt(Type t) {
  foreach(Object tmp in t.GetCustomAttributes(false)) {
   KittyAttribute attrKitty = tmp as KittyAttribute;
   if (attrKitty != null) {
    Console.WriteLine("作品 : " + attrKitty.strTitle);
    Console.WriteLine("名前 : " + attrKitty.name);
   }
  }
 }
}
```

このプログラムを実行すれば、次のような結果になります

作品 : Kitty on your lap
名前 : RENA
作品 : Hiza no ue no pertner
名前 : YUKI

プログラムを見ると KittyAttribute 属性クラスが定義されています
この属性は、コンストラクタが要求する name 位置パラメータと
コンストラクタの引数で要求されない strTitle 名前付きパラメータがあります
strTitle メンバは、省略された場合、デフォルトの "Kitty on your lap" が割り当てられます