**.NET程序集初级-特性**

**特性Attribute运行程序员将以类似注解的方式将自己开发的特性供其他应用程序开发的类型（类、结构体、接口等）使用。使用了特性的类型将额外多了特性赋予的信息或者限制。**

* **特性是什么？**

特性本身是继承了基类Atrribute的类，但它是封闭的，无法继承。特性就如同使用了它的其他类型或者程序集的身份标签，该标签告诉编译器该类型或者该程序集有什么额外的信息，并将这些信息打包到程序集的元数据中。

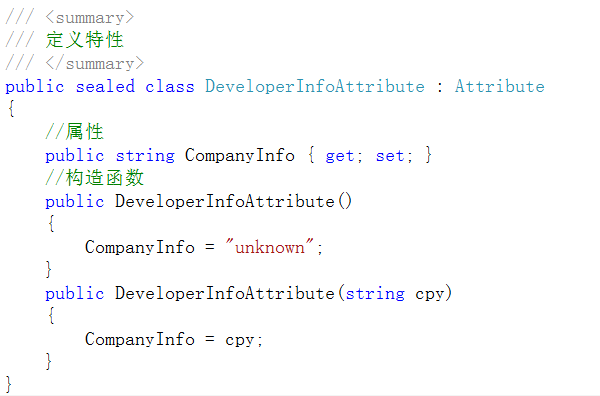
* **特性必须以\*Atrribute命名**

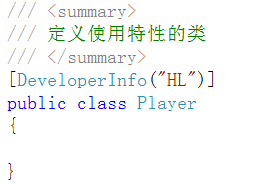
特性本质上是类，它继承了基类Atrribute。例如自定义一个命名为DeveloperInfoAttribute的特性。但在C#编程（唯一例子）里面，可以使用DeveloperInfo简化名称使用特性（C#编译器实际上会补全为DeveloperInfoAttribute）。

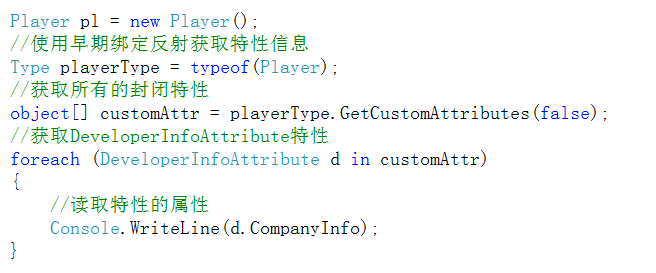
* **特性能做什么？**

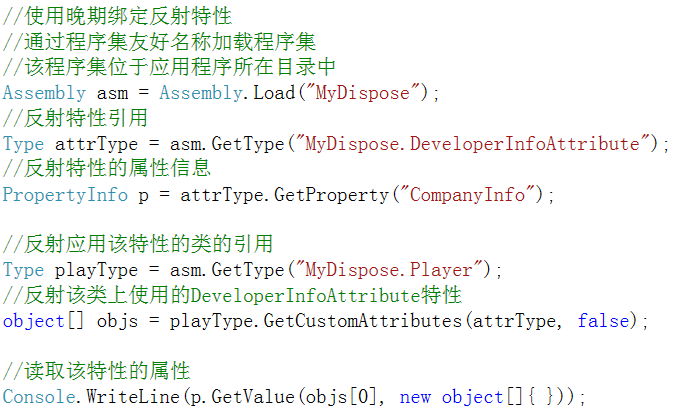
**特性可以应用于类型或者程序集，可以做到：**

* **类型级别特性：**
* 增加特殊的属性：特性可以增加使用该特性的自定义属性。但是该属性只能通过反射获取。

****

****

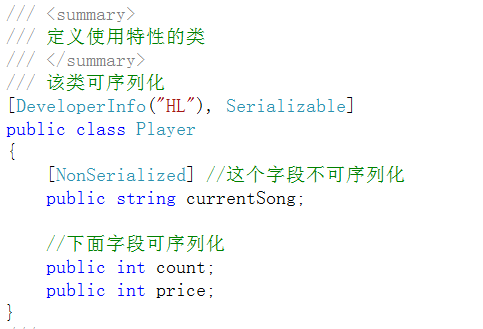
****

****

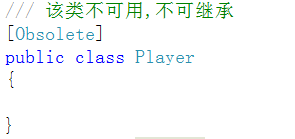
注意：Type.GetCustomAttributes()会返回对象，因此，只需在类型上操作而不需要创建实例。

* 使类型可序列化：

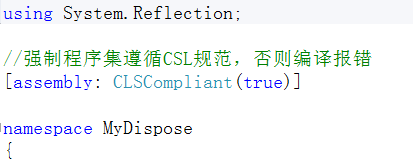
类型序列化，是指可以将类型对象实例的状态（字段等）序列化为可永久保存的数据流，如XML。



* 标记类型不可用，不可继承

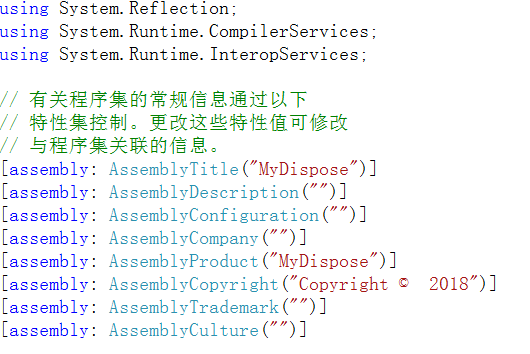


* 程序集级别特性：
* 将当前.cs文件程序集中的类型强制遵循CLS规范



* 其他特性

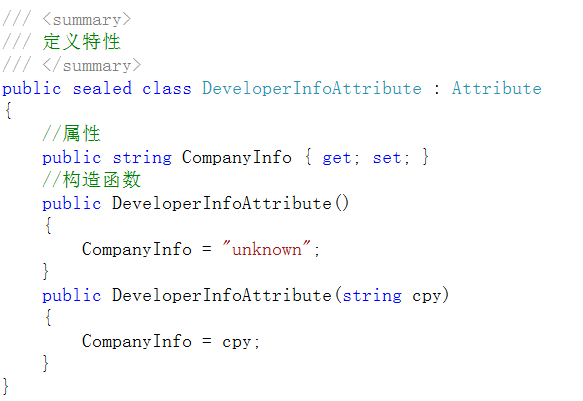
在项目的Properties下的AssemblyInfo.cs文件有其他特性的设置



* **如何自定义特性？**

特性本身就是类，一般它是封闭的。特性必须继承基类Attribute，

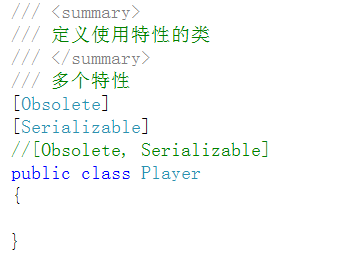
同时，它的命名必须是：特性友好名称+Attribute。



* **如何使用特性？**

将定义特性的.cs源文件或者程序集导入项目中，然后在添加特

性的类型或者程序集上方，以下面格式使用。

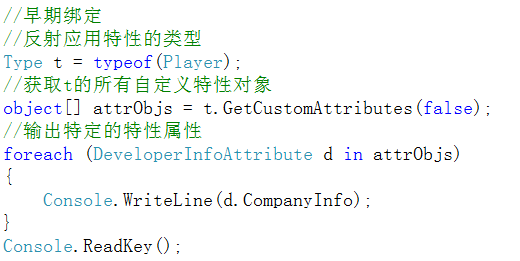
****

* **如何反射特性？**

反射特性时，只需要使用该特性的类型引用即可通过方法Type. GetCustomAttributes()获取特性对象数组。

* 早期绑定：

由于早期绑定已引用该特性的程序集，因此可以强制类型转换：



* 晚期绑定

由于晚期绑定的程序集动态加载，因此无法强制类型转换：

