C# 学习笔记

1 Attribute 特性

2 生成器

3 Action

4 Lambda

5 NuGet 依赖包管理工具

6 单元测试框架 XUnit

SimpleUnitDemo.Services

SimpleUnitDemo.Tests

7 扩展方法

静态方法 参数 this 类型名 参数名

8 结构和类的区别

结构不支持实现继承

结构是值类型 继承自 System.ValueType

类派生自System.Object

9 ref 关键字

函数参数传递分为值传递和引用传递

引用类型通过引用传递，值类型通过值传递，ref强制值类型通过引用传递

10 virtual override abstract

抽象方法：纯虚方法

虚方法：virtual

重写方法：override

11 泛型

null 只能用于引用类型 0 用于值类型

default 关键字

T doc = default(T) 如果T是值类型，就初始化为0，如果是引用类型，就初始化为null。

泛型约束：

where T:class 必须是引用类型

where T:struct 必须是值类型

where T:new() T 必须有一个默认构造函数

where T:IFoo 必须实现接口IFoo

where T:Foo 必须派生自基类Foo

where T1:T2

IComparable

int CompareTo(object obj)

12 枚举

foreach

示例代码

foreach(var p in persons)

{

Console.WriteLine(p);

}

编译器翻译：

IEnumerator<Person> enumator = persons.GetEnumerator();

while (enumator.MoveNext())

{

Person p = enumator.Current;

Console.WriteLine(p);

}

yield语句 包含yield语句的方法是一个迭代块,迭代块必须声明为返回IEnumator或其接口，不能包含return 语句，可以包含yield return 语句

# 13 委托

C++ 函数指针，委托是类型安全的类，定义了返回类型和参数的类型

匿名方法：

Func<string,string> a = delegate(string param)

{

return “result”;

}

lambda 表达式

参数=>方法

de

# 14 params 可变参数

# 15 GenericArguments

GenericParameters

# 16 Attribute

特性（属性）

属性类派生自System.Attribute 至少有一个公共构造函数

属性类有两种类型的参数：

定位参数：必须指定的参数，属性类的构造函数参数

命名参数：可选，使用属性时必须使用参数的名字。通过包含非静态字段或属性来定义命名参数

例如 AttributeUsage

有一个定位参数AllowOn 指定可以将属性赋值给的程序元素（类，方法，属性，参数）

System.Attributes.AttributeTargets

AttributeTargets.Class

AllowMultiple 一个bool值，是否允许为一个程序元素指定多个属性。默认值false

Inherited: 我们可以使用这个属性来控制定制特性的继承规则。它标记了我们的特性能否被继承。

# 17 集合类字典 枚举

所有集合类都实现了IEnumerable和IEnumerator接口

可枚举接口

public interface IEnumerable

{

IEnumerator GetEnumerator();

}

枚举器接口

public interface IEnumerator

{

object Current{get;}

bool MoveNext();

void Reset();

}

实现了IEnmerable<T>接口的集合，是强类型的。它为子对象的迭代提供类型更加安全的方式。

显式实现接口

显示实现继承非泛型IEnumerable接口的方法

IEnumerator IEnumerable.GetEnumerator()

{

}

关键字yield

# 18 属性

**自动属性3.0**

public class Test

{

public string Name

{

get;

set;

}

}

# 19 Object 比较

Equals

GetHashCode

实现了Equals必须实现GetHashCode

字典

键类型

如果两个对象的 Equals 比较结果相等，则每个对象的 GetHashCode 方法都必须返回同一个值。 但是，如果两个对象的比较结果不相等，则这两个对象的 GetHashCode 方法不一定返回不同的值。

GetHashCode()实现要求：

* 相同的对象应该总是返回相同的值。
* 不同的对象可以返回相同的值。

如果Equals 比较两个对象相等，则GetHashCode必须相等

如果不满足上述条件：某个类的实例放入字典后可能再也检索不到，或者会返回错误的项

A B 是相等的对象，但是HashCode不一致，经过散列后在一个桶里，这个时候可能A存在B的后面，那么性检索A，会得到B的值。

# 20 可访问性 accessibility

The protected internal accessibility means protected OR internal, not protected AND internal.

class Test

{

protected internal string name;

}

UNITY3D 脚本工程文件

* （1）Assembly-CSharp-firstpass

所有在Standard Assets，Pro Standard Assets或者 Plugins文件夹中的脚本会产生一个Assembly-CSharp-firstpass-vs.csproj文件，并且先编译

* （2）Assembly-CSharp-Editor-firstpass

所有在Standard Assets/Editor, Pro Standard Assets/Editor 或这Plugins/Editor文件夹中的脚本产生Assembly-CSharp-Editor-firstpass-vs.csproj工程，接着编译；

* （3）Assembly-CSharp

所有在Assets/Editor外面的, 并且不在(1),(2)中的脚本文件（一般这些脚本就是我们自己写的非编辑器扩展的脚本）会产生Assembly-CSharp-vs.csproj工程，被编译；

* （4）Assembly-CSharp-Editor

所以在Assets/Editor中的脚本产生一个Assembly-CSharp-Editor-vs.csproj工程，被编译。

之所有这样建立工程并按此顺序编译，也是因为DLL间存在的依赖关系所决定的。

UConsole