因为在博客中给出的代码大多数都使用了C#6.0的新特性，如果各位对C#6.0还不了解，可以简单的看一下这篇随笔。o(￣▽￣)d

先来看一个Point类

[IMG_256](http://www.cnblogs.com/zhuimengdev/p/javascript:void(0);" \o "复制代码)

public class Point

{

public int X { get; set; }

public int Y { get; set; }

public Point(int x, int y)

{

X = x;

Y = y;

}

public double Dist

{

get { return Math.Sqrt(X \* X + Y \* Y); }

}

public override string ToString()

{

return String.Format("({0}, {1})", X, Y);

}

}

[IMG_257](http://www.cnblogs.com/zhuimengdev/p/javascript:void(0);" \o "复制代码)

现在我们一步步来看在C#6.0中的改进

1->在以前版本的C#代码中所有的自动属性都必须有Setter，但是现在可以没有了。注意，不能只有Setter

[IMG_258](http://www.cnblogs.com/zhuimengdev/p/javascript:void(0);" \o "复制代码)

public class Point

{

public int X { get; }

public int Y { get; }  
  
　　　　 public double Dist

{

get { return Math.Sqrt(X \* X + Y \* Y); }

}

public override string ToString()

{

return String.Format("({0}, {1})", X, Y);

}

}

[IMG_259](http://www.cnblogs.com/zhuimengdev/p/javascript:void(0);" \o "复制代码)

2->同时现在也可以为自动属性直接设置初值，无需放到构造函数中

[IMG_260](http://www.cnblogs.com/zhuimengdev/p/javascript:void(0);" \o "复制代码)

public class Point

{

public int X { get; } = 2;

public int Y { get; set; } = 1;  
  
　　　　 public double Dist

{

get { return Math.Sqrt(X \* X + Y \* Y); }

}

public override string ToString()

{

return String.Format("({0}, {1})", X, Y);

}

}

[IMG_261](http://www.cnblogs.com/zhuimengdev/p/javascript:void(0);" \o "复制代码)

3->使用静态成员

　　3.1在上面的代码中看一看到调用Sqrt函数需要使用Math开头，写一次两次还好，如果一个类中大规模使用Sqrt每次都要先写一个Math会不会太麻烦了！

现在我们来看看如何改进。

首先在命名空间中添加下面这一行代码

using static System.Math;

规则是 using + static + 静态类命名空间

于是Dist属性就可改成下面这样

public double Dist

{

get { return Sqrt(X \* X + Y \* Y); }

}

　　3.2上面的规则也适用于枚举类型

[IMG_262](http://www.cnblogs.com/zhuimengdev/p/javascript:void(0);" \o "复制代码)

using static Desktop.Color;

namespace Desktop

{

enum Color

{

Yellow,

Red

}

class Program

{

static void Main(string[] args)

{

Console.WriteLine(Yellow);

Console.ReadKey();

}

}

}

[IMG_263](http://www.cnblogs.com/zhuimengdev/p/javascript:void(0);" \o "复制代码)

4->关于String.Format()方法的改进

这是经典写法

return String.Format("({0}, {1})", X, Y);

接下来一步步简化（先将String.Format用一个$代替）

return $"({0}, {1})", X, Y);

然后将0,1两个占位符直接换成X,Y

return $"({X}, {Y})";

好的化简完成。

5->对于Lambda表达式的改进

以前写匿名函数的时候可以写成一行，现在一般的函数也可以了

ToString()函数可以改写成如下形式

public override string ToString() => $"({X}, {Y})";

类似的属性可以改成这样

public double Dist => Sqrt(X \* X + Y \* Y);

注意属性是没有()的

 简化后的Point类是这样的

[IMG_264](http://www.cnblogs.com/zhuimengdev/p/javascript:void(0);" \o "复制代码)

public class Point

{

public int X { get; } = 2;

public int Y { get; set; } = 1;  
  
 public double Dist => Sqrt(X \* X + Y \* Y);

public override string ToString() => $"({X}, {Y})";

}

[IMG_265](http://www.cnblogs.com/zhuimengdev/p/javascript:void(0);" \o "复制代码)

6->索引初始化

先来看一段Json.Net的代码

[IMG_266](http://www.cnblogs.com/zhuimengdev/p/javascript:void(0);" \o "复制代码)

public JObject ToJson()

{

var result = new JObject();

result["X"] = X;

result["Y"] = Y;

return result;

}

[IMG_267](http://www.cnblogs.com/zhuimengdev/p/javascript:void(0);" \o "复制代码)

改进后的代码可以这么写

[IMG_268](http://www.cnblogs.com/zhuimengdev/p/javascript:void(0);" \o "复制代码)

public JObject ToJson()

{

var result = new JObject()

{

["X"] = X,

["Y"] = Y

};

return result;

}

[IMG_269](http://www.cnblogs.com/zhuimengdev/p/javascript:void(0);" \o "复制代码)

最终可以化简成一行代码

public JObject ToJson() => new JObject() { ["X"] = X, ["Y"] = Y };

7-> ?.运算符

?.运算符其实很简单，主要就是检查变量是否为null，如果不为null那就执行.

先来看一个函数，这个判断语句中大部分的检查都是在

[IMG_270](http://www.cnblogs.com/zhuimengdev/p/javascript:void(0);" \o "复制代码)

public static Point FromJson(JObject json)

{

if (json != null &&

json["X"] != null &&

json["X"].Type == JTokenType.Integer &&

json["Y"] != null &&

json["Y"].Type == JTokenType.Integer

)

{

return new Point((int)json["X"], (int)json["Y"]);

}

return null;

}

[IMG_271](http://www.cnblogs.com/zhuimengdev/p/javascript:void(0);" \o "复制代码)

这个函数可以用?.运算符化简成

[IMG_272](http://www.cnblogs.com/zhuimengdev/p/javascript:void(0);" \o "复制代码)

public static Point FromJson(JObject json)

{

if (json != null &&

json["X"]?.Type == JTokenType.Integer &&

json["Y"]?.Type == JTokenType.Integer

)

{

return new Point((int)json["X"], (int)json["Y"]);

}

return null;

}

[IMG_273](http://www.cnblogs.com/zhuimengdev/p/javascript:void(0);" \o "复制代码)

如果json["x"]为null,那么就不执行. 如果json["x"]不为null，那么就会执行.然后判断类型是否为int

所以代码可以被再次化简

[IMG_274](http://www.cnblogs.com/zhuimengdev/p/javascript:void(0);" \o "复制代码)

public static Point FromJson(JObject json)

{

if (json?["X"]?.Type == JTokenType.Integer &&

json?["Y"]?.Type == JTokenType.Integer

)

{

return new Point((int)json["X"], (int)json["Y"]);

}

return null;

}

[IMG_275](http://www.cnblogs.com/zhuimengdev/p/javascript:void(0);" \o "复制代码)

?.还有一个比较大的用处在触发事件的时候

OnChanged(this, args);

如果此时OnChanged为空显然是不行的所以呢，可以改写成这样

if (OnChanged == null)

{

OnChanged(this, args);

}

如果很不幸的另外一个线程就在判断之后，触发事件之前，再次设置OnChanged变为null，同样会导致错误

为了保证线程安全

需要先Copy一份，但是这样写显然就。。。

var onChanged = OnChanged;if (onChanged != null)

{

onChanged(this, args);

}

现在可以改写成这样

OnChanged?.(this, args);

8-> 支持在catch和finally块中使用await

http://www.cnblogs.com/zhuimengdev/p/5072752.html