Atitit .c#的未来新特性计划草案

[1. C#的未来：追踪空引用 1](#_Toc31602)

[1.1. 2. 变量命名空间 1](#_Toc2113)

[1.2. 10. 项目引用Native dll 2](#_Toc22877)

[1.3. 10. 项目引用Native dll 2](#_Toc24096)

[2. 三、设计面向 面向变量的汇编语言，高级语言出现后，从Pascal/C语言面向过程和变量，到C++以后的面向对象 2](#_Toc2942)

[2.1. 1. 面向集合 3](#_Toc9376)

[2.2. 2. 面向并发 3](#_Toc14588)

[2.3. 3. 面向任务（异步 3](#_Toc21074)

[3. 参考 3](#_Toc25087)

# **C#的未来：追踪空引用**

olang 都体现了新一代编程语言的设计哲学。  
简约的关键字,  
非侵入式的面向对象设计(没有了继承)

## ****2. 变量命名空间****

class ProductUtil{

namespace Price

{

float old;

float New;

float standard;

}

float GetPrice(){

float Price.discount; //声明折扣

//计算价格...

return Price.New;

}

}

命名空间可以在方法内，也可以在类型内。声明命空间后，可以单独声明一个变量。

作者:: 绰号:老哇的爪子 （ 全名：：Attilax Akbar Al Rapanui 阿提拉克斯 阿克巴 阿尔 拉帕努伊 ） 汉字名：艾龙，  EMAIL:1466519819@qq.com

转载请注明来源： http://blog.csdn.net/attilax

## 10. 项目引用Native dll

如今要调用C/C++的程序集，必须查好参数，转化类型，再像这种方式声明：

[DllImport("User32.dll", CharSet = CharSet.Unicode)]public static extern int MessageBox(int h, string m, string c, int type);

 为什么不能像引用.NET dll那样引用C/C++写的dll呢，或者像Service Reference那样，增加一个Native Reference的目录，自动生成代理类。

## 10. 项目引用Native dll

如今要调用C/C++的程序集，必须查好参数，转化类型，再像这种方式声明：

[DllImport("User32.dll", CharSet = CharSet.Unicode)]public static extern int MessageBox(int h, string m, string c, int type);

 为什么不能像引用.NET dll那样引用C/C++写的dll呢，或者像Service Reference那样，增加一个Native Reference的目录，自动生成代理类。

　使用元组的目标是以一种轻量级的方式从一个函数中返回多个值。对元组的良好支持能够消除对out参数的使用，这种参数通常被认为是一种笨重的方案。此外，out参数无法兼容 async/await，因此在许多场景中out参数将变得毫无作用

# 三、设计面向 面向变量的汇编语言，高级语言出现后，从Pascal/C语言面向过程和变量，到C++以后的面向对象

编程语言发展已经超过半个世纪了，先是面向变量的汇编语言，高级语言出现后，从Pascal/C语言面向过程和变量，到C++以后的面向对象。 C#和Java只是语言特性上有大幅改进，设计思想并没有飞跃。

C#作为最为先进的编程语言，反映了当前语言发展的瓶颈。要有所突破，必须要有新的设计思想，把面向粒度提高到新的层次。

个人分析后认为，未来C#或C#的后继者，会向三个方向发展：

## ****1. 面向集合****

未来编程语言遇到的业务逻辑将更复杂，对集合处理是业务逻辑的核心内容。LINQ使C#走在业界的前列，然而还有许多问题。

由于历史原因，集合类型太多太乱。支持泛型是必须，我们需要根据可变性、排序性、Hash特性、并发要求等，使用一致的高性能集合类型。这些集合类能够灵活转化，智能地处理扩容、复制等底层操作，且没有LINQ那样无法跨程序域传递的限制。这需要框架和CLR双重支持。

## ****2. 面向并发****

这个是很自然的方向，除非出现光、生物、量子计算技术的飞跃，不然半导体电路处理器单核极限愈来愈难突破，多核趋势愈演愈烈。未来的编程语言，并发支持必须融入其底层。

还是拿Node.js来说，已经初具此特性，其对IO的访问全部非阻塞的，是从底层支持的异步操作。

对于C#来说，就不只是框架上修修补补，而是CLR的全面支持。async和await出现是个很好的苗头，期待看到更多这样的发展。

## ****3. 面向任务（异步****

Node.js它通过事件轮询（event loop）来实现并行操作，这只能处理最简单地多任务同步。要实现真正的并发语言，并满足日益复杂的业务逻辑处理，必须对并发的单元－任务进行有力的支持。

.NET对任务有了System.Threading.Tasks下一系列类的支持，但这只是开始。我们需要动态地创建、分解、修改、取消任务，需要方便地获取和控制任务的状态，管理超时和资源，统计任务效率，处理异常。

# 参考

论C#未来发展 - 小城故事 - 博客园.htm

C#的未来：元组及匿名结构体-IT168 技术开发专区.htm

C#的未来：扩展属性及更多 - 推酷.htm