**参考文献**

[关于托管与非托管](http://blog.csdn.net/atlashr/article/details/4154387)

[到底什么是托管，什么是非托管的研究](http://www.cnblogs.com/mqcan/archive/2008/05/04/1181284.html)

**前言**

最近在看《ASP.NET MVC 4框架揭秘》，里面有很多微软.net的东西，其中就很多次提到了托管与非托管，搞得我云里雾里的，今天特地抽空来整理一下。大部分内容都是参考别人的。

**托管代码**

托管代码就是Visual Basic .NET和C#编译器编译出来的代码。编译器把代码编译成中间语言(IL)，而不是能直接在你的电脑上运行的机器码。中间语言被封装在一个叫程序集(assembly)的文件中，程序集中包含了描述你所创建的类，方法和属性(例如安全需求)的所有元数据。你可以拷贝这个程序集到另一台服务器上部署它。通常来说，这个拷贝的动作就是部署流程中唯一的一个操作。

托管代码在**公共语言运行库(CLR)**中运行。这个运行库给你的运行代码提供各种各样的服务，通常来说，他会加载和验证程序集，以此来保证中间语言的正确性。当某些方法被调用的时候，运行库把具体的方法编译成适合本地计算机运行的机械码，然后会把编译好的机械码缓存起来，以备下次调用(这就是即时编译)。随着程序集的运行，运行库会持续地提供各种服务，例如安全，内存管理，线程管理等等。这个程序被“托管”在运行库中。Visual Basic .NET和C#只能产生托管代码。如果你用这类语言写程序，那么所产生的代码就是托管代码。如果你愿意，Visual C++ .NET可以生成托管代码。当你创建一个项目的时候，选择名字是以.Managed开头的项目类型。例如.Managed C++ application。

**非托管代码**

非托管代码就是在Visual Studio .NET 2002发布之前所创建的代码，例如Visual Basic 6, Visual C++ 6。 最糟糕的是，连那些依然残存在你的硬盘中、拥有超过15年历史的陈旧C编译器所产生的代码都是非托管代码。非托管代码直接编译成目标计算机的**机械码**，这些代码只能运行在编译出它们的计算机上，或者是其它相同处理器或者几乎一样处理器的计算机上。非托管代码不能享受一些运行库所提供的服务，例如安全和内存管理等。如果非托管代码需要进行内存管理等服务，就必须显式地调用操作系统的接口，通常来说，它们会调用Windows SDK所提供的API来实现。就最近的情况来看，非托管程序会通过COM接口来获取操作系统服务。跟Visual Studio平台的其他编程语言不一样，Visual C++可以创建非托管程序。当你创建一个项目，并且选择名字以MFC，ATL或者Win32开头的项目类型，那么这个项目所产生的就是非托管程序。

总而言之，非托管代码是运行在**公共语言运行库环境(CLR)**的外部，由操作系统直接执行的代码。非托管代码必须提供自己的垃圾回收、类型检查、安全支持等服务；它与**托管代码**不同，后者从公共语言运行库中获得这些服务。

**两者区别**

对于Visual Basic和C#来说，生活是简单的，因为你没有其它选择。当你在那些语言里面声明一个类，那么这个类的实例会在托管堆中被创建，垃圾收集器(GC)会帮我们管理这些对象的回收。但是在Visual C++中，你有另一个选择。即使你正创建一个托管程序，你可以决定哪些类是托管类型，哪些类是非托管类型的。

**非托管类型：**

[复制代码](javascript:void(0);)

class Foo

{

private:

int x;

public:

Foo(): x(0){}

Foo(int xx): x(xx) {}

};

[复制代码](javascript:void(0);)

**托管类型**

[复制代码](javascript:void(0);)

\_gc class Bar

{

private:

int x;

public:

Bar(): x(0){}

Bar(int xx): x(xx) {}

};

[复制代码](javascript:void(0);)

他们唯一的区别就是类Bar的定义中有\_gc关键字。这个关键字会给代码带来巨大的区别。

托管类型是可以被垃圾回收器所回收的。他们必须要用关键字new来创建，永远都不会在栈中出现。所以下面这行代码是合法的：

Foo f;//非托管类型

但是这一行代码就是非法的：

Bar b;//托管类型必须用new来创建

如果我在堆中创建一个Foo对象，那么我必须要负责清理这个对象：

Foo\* pf = new Foo(2);

// . . .

delete pf;//手动清理

C++编译器实际上会用两个堆，一个托管堆和一个非托管堆，然后通过对new操作符的重载来实现对创建不同类型类的实例，分配不同的内存。如果我在堆里面创建一个Bar实例，那么我可以忽略它。当没有其他代码在使用它的时候，垃圾回收器会自动清理这个类，释放其占用的资源。对于托管类型会有一些约束：它们不能实现多重继承，或者继承与非托管类型；它们不能用friend关键字来实现私有访问，它们不能实现拷贝构造函数。所以，你有可能不想把你的类声明为托管类型。但是这并不意味着你不想让你的代码成为托管代码。在Visual C++中，你可以选择。

**托管代码的执行过程**

1. 选择编译器：为获得公共语言运行库提供的优点，必须使用一个或多个针对运行库的语言编译器，如 Visual Basic、C#、Visual C++、JScript 或许多第三方编译器（如 Eiffel、Perl 或 COBOL 编译器）中的某一个。由于运行库是一个多语言执行环境，因此它支持各种数据类型和语言功能。您所用的语言编译器首先确定可用的运行库功能，然后使用这些功能设计代码。编译器（而不是运行库）建立代码必须使用的语法。如果您的组件必须完全能够被用其他语言编写的组件使用，您的组件的导出类型必须只公开公共语言规范 (CLS) 中包括的语言功能。
2. 编译，将源代码翻译为microsoft中间语言（MSIL）并生成所需的元数据。
3. 在执行时，实时 (JIT) 编译器将 MSIL 翻译为本机代码。在此编译过程中，代码必须通过验证过程，该过程检查 MSIL 和元数据以查看是否可以将代码确定为类型安全。
4. 运行代码：公共语言运行库提供使执行能够发生以及可在执行期间使用的各种服务的结构。