<http://duckii.egloos.com/1293394>

평소에 .Net Framework은 거들떠 보지도 않다가 이렇게 꺼내게 된 건 Visual Studio .Net의 AddIn 개발의 필요성을 느껴서이다. Dandelion은 C++ Native 코드로 만들어지고 있다. 거기에 첫 번째 Requirement Spec이 Unit Test 지원이기 때문에 [TUT](http://tut-framework.sourceforge.net/)를 Integration하는 도중

“아! Visual Studio에서 편하게 Unit Test를 돌릴 수 있도록 AddIn을 만드는 것도 좋겠구나~”

라는 생각이 들었다. 아직 AddIn 개발을 시작하진 않았지만 C++ Native 코딩 한다는 게 만만치 않게 보이는 거라… 그래서 .Net Framework로 눈을 돌리게 되었다. 빨리 만들 수 있으니깐…   
Visual Basic.Net은 왠지 정이 안가고, VS.Net 2008에서는 J#이 없어졌고.. C#밖에 안 남았다. Managed C++도 있긴 한데, Native나 Managed나 시간은 비슷할 듯 하여 C#으로 강행하려던 차에 걸림돌이 생겼다. AddIn 만드는 방법은 일단 모른다 쳐도 문제는…

* Native DLL 혹은 EXE 모듈 (Unmanaged 코드)에 있는 Unit Test실행 함수를 C#에서 어떻게 호출하느냐?

대충 어림 짐작되는 키워드로 구글링 해보니 정확히 나와버렸다.

|  |
| --- |
| using System.Runtime.InteropServices;  public class Win32 {       [DllImport("user32.dll", CharSet=CharSet.Auto)]       public static extern IntPtr MessageBox(int hWnd, String text,                       String caption, uint type);  }  public class HelloWorld {      public static void Main() {         Win32.MessageBox(0, "Hello World", "Platform Invoke Sample", 0);      }  } |

.Net Framework에서 상호 운용성 이라는 이름으로 위와 같은 기능을 제공한다. 위의 코드는 Win32 API의 MessageBox함수를 .Net Framework에서 호출하는 예제이다. 뭐 Platform Invoke (짧게 PInvoke)라고 불리는 방법이라고 한다.   
자~ 이제 기쁜 마음에 시작하려고 하니 한 가지 걸림돌이 또 생겼다. Unit Test의 결과를 AddIn측에서 알아야 어떻게든 리포트를 해줄 수 있다. 결과를 알아내는 방법 하면 콜백 함수가 떠오르는데… 그렇다면 Managed Code에서 함수 포인터를 Unmanaged Code에 넘겨주고 Unmanaged Code에서 Managed Code의 함수를 호출할 수 있는 방법이 필요하다. 역시 구글님께서 또 답을 알려주셨다.

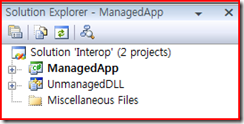
<http://www.informit.com/guides/content.aspx?g=dotnet&seqNum=473>

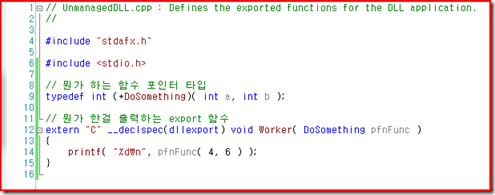
C#에서 delegation을 함수 포인터로 넘겨준다고 한다. 잠깐 읽어보니 예전에는 어셈블리를 까서 직접 수정해야 했다고 하는데, .Net Framework 2.0부터는 알아서 해준단다.

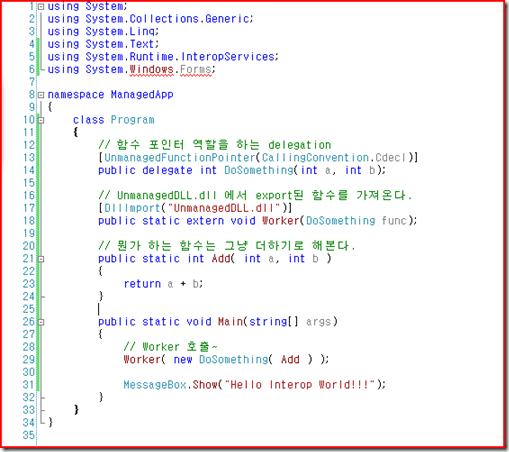
|  |
| --- |
| [UnmanagedFunctionPointer(CallingConvention.Cdecl)]  public delegate IntPtr FdiMemAllocDelegate(int numBytes); |

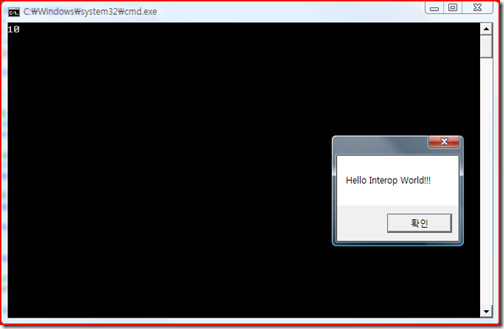
요약하면 이게 핵심이다. Delegation을 선언하고 이 타입으로 넘겨주면 된다고 한다. 위에 Attribute에서는 여러 Calling Convention으로 marshaling할 수 있게 해준다고 한다. Cdecl, StdCall, ThisCall 등이 있다고 한다.

**그래서 해봤다.**

[](http://pds11.egloos.com/pds/200901/06/22/f0060522_4963672495f2a.png)    
먼저 Interop이라는 솔루션 안에 UnmanagedDLL 이라는 C++ DLL 프로젝트와 ManagedApp 라는 C# Console 프로젝트를 만들었다.

[](http://pds11.egloos.com/pds/200901/06/22/f0060522_496368a88d03b.png)    
UnmanagedDLL 구현부.   
Worker라는 함수는 DLL에서 Export되는 함수이며 파라메터로 int 두 개 받아 뭔가 한 결과 int를 콘솔에 출력하는 함수이다. 그 뭔가 하는 함수는 managed code로 구현될 것이다.

[](http://pds13.egloos.com/pds/200901/06/22/f0060522_496368a91c6f6.png)    
ManagedApp 구현부   
위에서 검색했던 방법을 합쳤다. 뭔가 하는 함수는 그냥 더한 결과를 리턴하도록 했다.

[](http://pds11.egloos.com/pds/200901/06/22/f0060522_496368a9d248c.png)    
결과…

P.S. C# Language Specification문서가 VisualStudio 설치된 디렉토리\C#\Specifications 에 있다. 단. 2005부터 있는거 같다.

P.S. 처음에 DLL 함수를 호출할 때 자꾸 System.BadImageFormatException이 발생했다. 몇 시간 고생하다가 프로젝트 구성이 잘못 되어있음을 발견했다. Unmanaged DLL이 Win32로 빌드 되었는데 ManagedApp가 Any CPU로 되어 있었다. Managed App를 x86으로 변경하니 정상 동작 했다.