# How to combine Unmanaged C++ code in C# Application

In this article I’ll show how to import your old C++ dll into a C# application.

First of all for the ease of reading and writing: C++ ==unmanaged c++, MC++ == managed c++)

There are two possible ways for doing so:

1. The first is using the dllimport attribute to import C++ functions into C#. I will not discuss this way in this article.
2. The second way, and the subject of this article, is importing through a MC++ wrapper class.

This type of importing allows you to wrap your C++ code or dll with a MC++ dll.

Let’s get to the code without further due.

Assume we have this code written in C++:

Note: I discarded the implementation but without the implementation in a .cpp or a .c file the dll won’t export any symbol.

#pragma once

#define MAX\_STRING\_LENGTH 250

class StringClass

{

public:

StringClass();

~StringClass();

void PutString( const char\* \_str );

void PrintString();

private:

int mLength;

char mCharacters[MAX\_STRING\_LENGTH];

};

int convertToHe(int \_chrInput);

long convertFromHe(long \_chrInput);

void strncpy(char\* \_dest, const char\* \_source, int \_length);

We want to use this code in a managed environment. So we need to wrap these code with MC++ code. The following code is an example:

#pragma once

#using <mscorlib.dll>

#include "Unmanaged.h"

namespace MCClassLibrary

{

public \_\_gc class ManagedStringClass

{

public:

ManagedStringClass(void);

virtual ~ManagedStringClass(void);

void PutString( const char\* \_str );

void PrintString();

StringClass\* mStringClass;

};

public \_\_gc class ManagedClass

{

public:

ManagedClass(void);

virtual ~ManagedClass(void);

static int mConvertToHe(int \_chrInput);

static long mConvertFromHe(long \_chrInput);

static void mStrncpy(char\* \_dest, const char\* \_source, int \_length);

};

}

Notes:

* Setting the project to use managed extensions is necessary. In Project Properies-> General -> Use managed extensions set the option to yes.
* The class you export, MUST be declared in a namespace and MUST be public.

The C# application code that uses our MC++ dll will look like this:

using MCClassLibrary;

namespace CSharpApp

{

/// <summary>

/// Summary description for Class1.

/// </summary>

class Class1

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main(string[] args)

{

ManagedStringClass x = new ManagedStringClass();

x.PutString (“string”);

x.PrintString ();

//ManagedClass managedClass = new ManagedClass();

int number = ManagedClass.mConvertToHe(34);

}

}

}

Note:

* Add reference to our MC++ dll in the project setting.

In this example I showed how to import your C++ code into C#.

Hope you enjoyed.

You can use a very similar technic if you want to import a “closed” C++ dll into C#. I leave this subject for you.