Three Methods of Integrating C# with MATLAB

Description

This demo shows three ways to integrate MATLAB code into a C# project. All three methods use the same example from MATLAB, but each interacts with it in a different fashion.

The first method shows how to use the MATLAB as an automation server from C# using the engine interface via com automation. This allows you to simultaneously debug your C# application from both the C# side and the MATLAB side, using debuggers on each side.

The second method uses MATLAB Builder for .NET to create a .NET assembly. This assembly exposes the MATLAB example as a method of a class included in the component. This method can be used directly in C#. It provides intellisense, automatic data marshalling and garbage collection. This .NET assembly can be deployed royalty-free to machines that do not have MATLAB.

The third method uses MATLAB Compiler to create a C shared library – unmanaged code. This library exposes the MATLAB example – but not in a manner that can be used directly in .NET. Instead a wrapper class was created to provide entry into the method included in the library, and to marshal the data from the managed framework into the unmanaged library. This C Shared library can also be deployed royalty-free to machines that do not have MATLAB.

It should be noted that third example is very fragile. It is not the recommended method of C# and MATLAB integration. It has only been provided to emphasize the additional work which becomes necessary to integrate C# with MATLAB when you do not use Builder for .NET. This is not a general solution to integrating a MATLAB produced C shared library with .NET but a single purpose solution designed only to work for this example. Upon further inspection it should be clear that the lack of typing when moving from managed to unmanaged code dramatically increases the risk of data handling errors. The lack of intellisense, automatic data marshalling, and garbage collection are additional detractors from this method.

The demo files

There should be two directories of files for this demo.

1. mcode contains all of the MATLAB source code
2. VSProj contains the visual studio 2005 project files

In addition, all of the output files required to run this application should be in the Output directory.

How to build the demo

1. Ensure that you have the MATLAB, MATLAB Compiler, and MATLAB Builder for .NET installed on your system. I am using version 7.9 (R2009b). If you do not have these products, you may wish to inspect the different approaches to coding in the Visual Studio project files, although you will not be able to run them.
2. Ensure that you have Microsoft Visual Studio installed on your system. This project was built using Professional Edition 2005.
3. In MATLAB, run build_mcode.m in the mcode folder. This will:
   a. Compile the m-file math_on_numbers.m into a .NET Assembly (dotnet.dll).
   b. Compile the m-file math_on_numbers.m into a C Shared Library (cshared.dll).
   c. Copy the appropriate files into the Output directory.
4. In Visual Studio, verify project references are correct
   a. MLApp is a com object that is part of core MATLAB:
      i. Name: MATLAB Application (Version 7.9) Type Library

        ii. Path: *matlabroot*/bin/win32/mlapp.tlb *
    b. MWArray is a .NET assembly that is part of Builder for .NET
        i. Name: MathWorks, .NET MWArray API
        ii. Path: *matlabroot*/bin/win32/MWArray.dll *

5. In the Visual Studio C# code “VSProj\Program.cs”, insure path reference to mcode is accurate (located at second bookmark, line# 148).
6. In Visual Studio, Build the Solution.

* *matlabroot* is the MATLAB installation directory

Note that the compilation process may not work correctly if there are spaces in the path for the demo.

Running the application:

Although the application was built as a console application – it is truly designed to be run from within Visual Studio for illustrative purposes only.

At the top of the file “VSProj\Program.cs” there is a runtime parameter variable Mode. The location of this variable has been bookmarked in Visual Studio (line #37). The value of mode will determine which method of C# MATLAB integration will be utilized.

The values are:

1. MATLAB Engine using via COM automation
2. .NET Assembly created in MATLAB using Builder for .NET
3. C Shared library created in MATLAB using Compiler

Each method has its own call function: UseEngine, UseDotNet, and UseClib respectively. Each function takes the same inputs and updates the say outputs.

Note on Precision: The calling method, input arguments, and output arguments are all displayed to the console. Only four decimal places are being displayed for cleanliness, the data itself is stored in double precision floating point values.