

Debugging MapInfo .Net Programs with Visual Studio 2008 or Visual Studio 2008 Express Edition

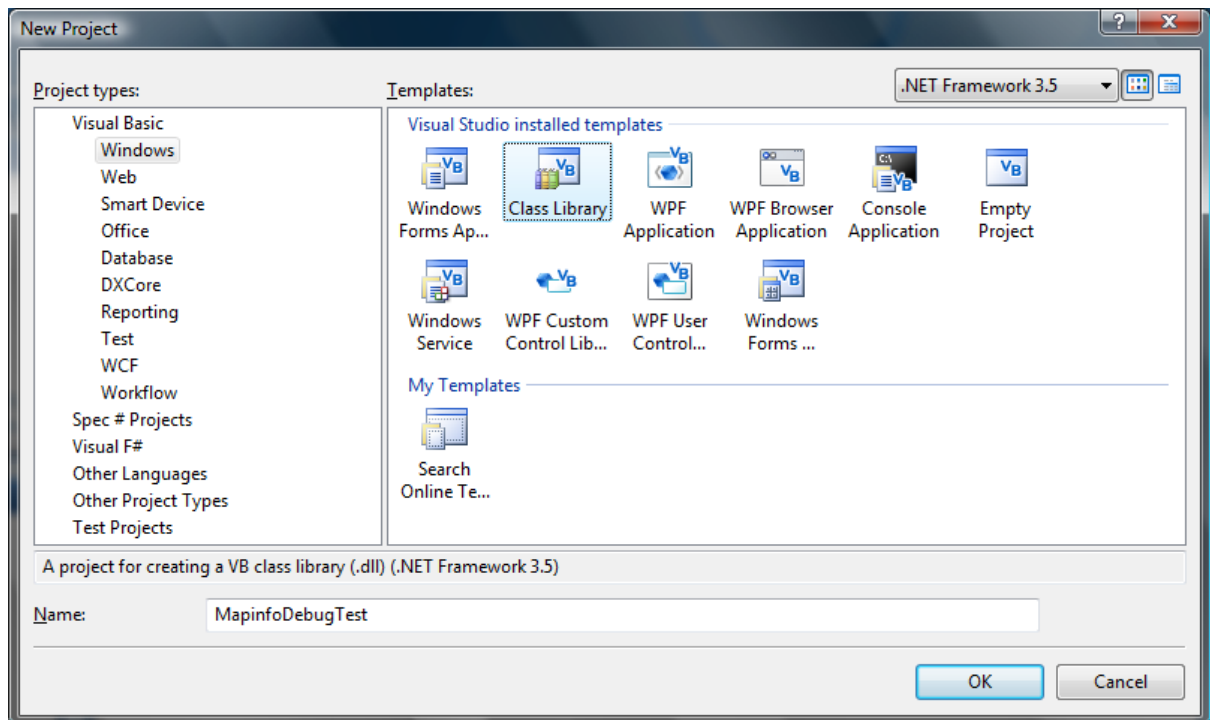
by Nathan Woodrow

This article will hopefully explain how to attach the Visual Studio debugger to an instance of MapInfo Professional 9.5 to allow you to debug a .NET application invoked from a MapBasic application.

NOTE! So far I have only tested the following steps with C# and VB .NET using Visual Studio 2008. The same basic principles should work in Visual Studio 2005 though it has not been tested


Step 1: Create Visual Studio Project

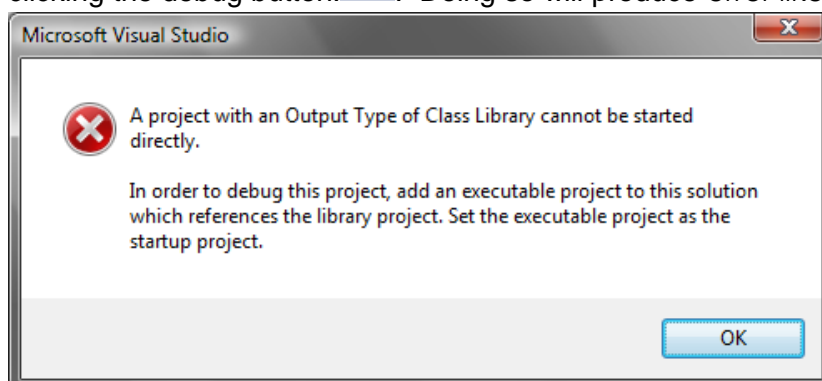
The first step is to create a new Visual Studio project. For the following examples we will be creating a Visual Basic Class Library as shown here:



The reason for choosing a Visual Basic class library (dll) is that we do not want our application to be run by itself, but only to be invoked by a MapBasic application.

As soon as the project has been created, it should be saved so that Visual Studio can create all of the necessary files and folders needed for the following steps.

Normally when creating a class library project, Visual Studio will not allow it to be run when clicking the debug button: . Doing so will produce error like this:



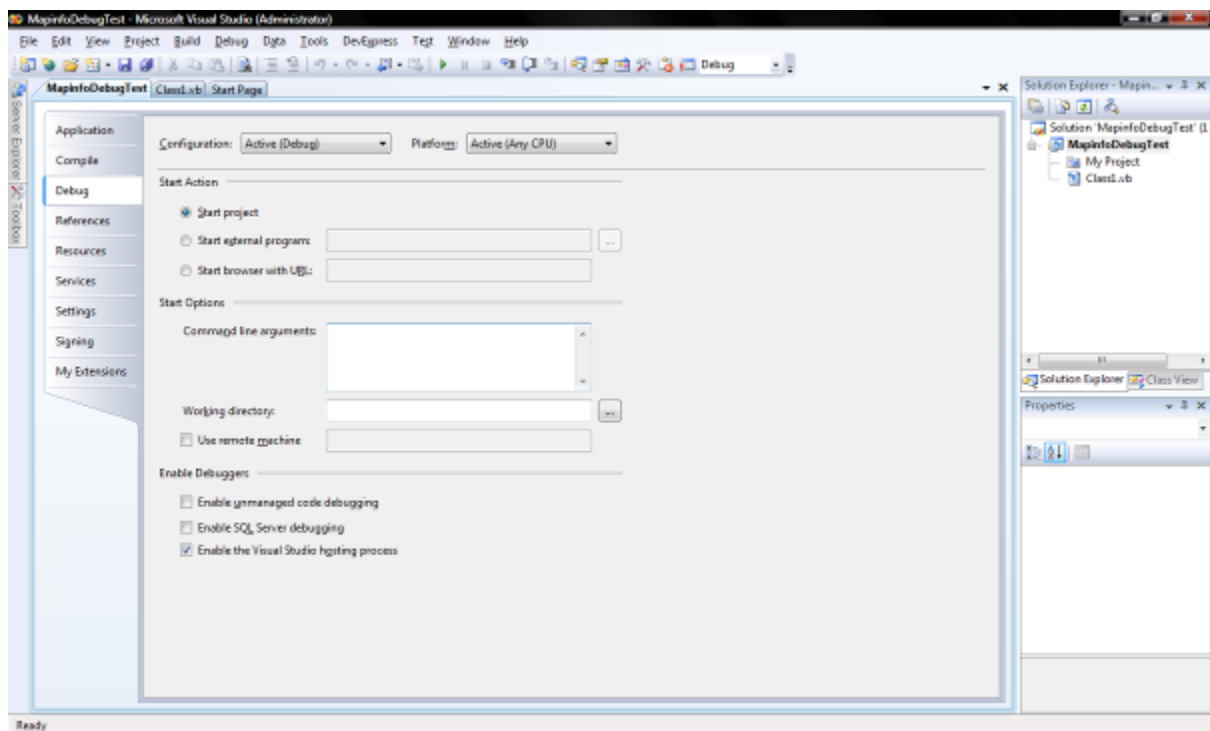
In order to get around this problem, we would normally create or include an application project in Visual Studio, create a reference to our class library and then set the application project as the start-up project but because we are using MapInfo, we are unable to add a copy of it to Visual Studio.

In order for us to attach the Visual Studio debugger to our project and MapInfo, we need to make a few changes in the project settings page of our class library.

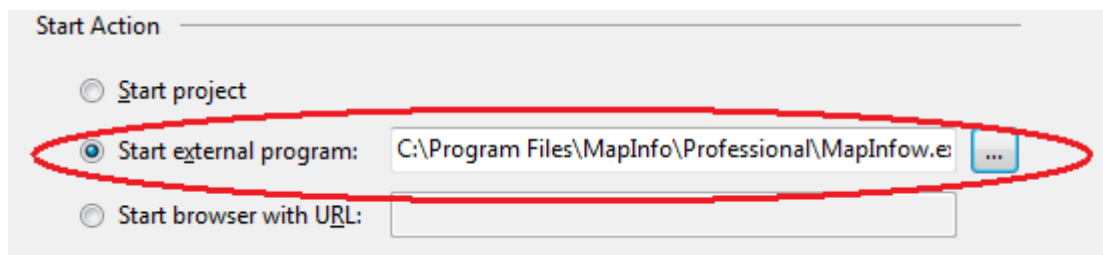
NOTE! This next part of this current step will be split into two different sections. One for changing the settings in the full Visual Studio and the other for the Express editions (VB Express and C# Express etc). You will only need to do one of the following steps.

Step 1.1 Editing Settings for the Full Visual Studio

First we need to open the project settings page by going to Project->MapinfoDebugTest Properties or you can use the keyboard shortcut Alt+P+P. Once the project properties screen is up, click on the “Debug” tab. The screen should now look something like this.



There are two settings on this form that will need to be changed but for now we will only be changing one which is the one that looks like this:



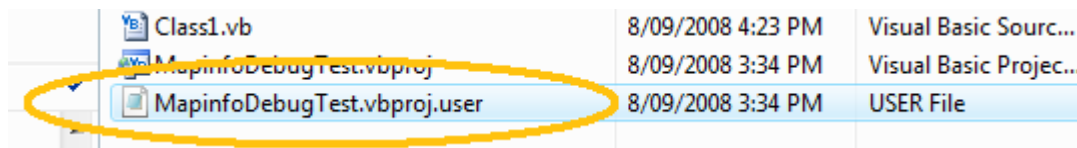
This setting will need to be changed to the path of Mapinfo.exe which on most computers is the same as above. Once you have edited this property, save.

Step 1.2 Edit Settings in Express Editions

In the Express Editions of Visual Studio, we are not able to just change a couple of options in the project properties form like in the full Visual Studio. However with a little bit of xml editing, we can achieve the same result.

In order to complete the next few steps, Visual Studio Express Edition must be closed. This is to ensure that any manual edits to the configuration files are not overridden by the program.

We will need to edit the project user settings xml file. This file is located in the same location as where the project was saved. It will be named something like this:



Class1.vb	8/09/2008 4:23 PM	Visual Basic Sourc...
MapinfoDebugTest.vbproj	8/09/2008 3:34 PM	Visual Basic Projec...
MapinfoDebugTest.vbproj.user	8/09/2008 3:34 PM	USER File

After locating the file, it needs to be opened in a text editing program like Notepad (or something like Notepad++ to get coloured syntax highlighting as shown below)

Once the file has been opened, it should contain a line similar to this:

```
<PropertyGroup Condition=" $(Configuration)|$(Platform)' == 'Debug|AnyCPU' />
```

← Note the forward slash needs to be removed in order to add to the property group

This is property group that holds all of the actions for the current type of build to be run for the project. As we can see, this group is for debug build.

We will need to expand the property group and add the following code inside:

```
<StartAction>Program</StartAction>  
<StartProgram>C:\Program Files\MapInfo\Professional\MapInfo.exe</StartProgram>
```

The “StartAction” tag just tells Visual Studio that a program will need to be started.


The “StartProgram” tag is the path to an executable that will run once the debugger is launched.

In the end, the entire configuration file should look something like this:

```
<Project xmlns="http://schemas.microsoft.com/developer/msbuild/2003">  
  <PropertyGroup Condition=" $(Configuration)|$(Platform)' == 'Debug|AnyCPU' ">  
    <StartAction>Program</StartAction>  
    <StartProgram>C:\Program Files\MapInfo\Professional\MapInfo.exe</StartProgram>  
  </PropertyGroup>  
</Project>
```

Once we have made the changes to the file, we need to save & exit and launch Visual Studio Express Edition once again.

Step 2: (Optional) Testing start application

Before going further, we can test the changes we have just made. If we close the settings form and return to the code window and press the  button, we should be presented with the MapInfo Pro splash screen. After MapInfo has loaded, if we take a look at the currently running processes in a program like Process Explorer you will notice that the instance of MapInfo is attached to Visual Studio:



devenv.exe	1592
fsi.exe	5388
MapInfow.exe	3424

When we stop the debugger by clicking the blue stop button in Visual Studio, the instance of MapInfo attached to Visual Studio will also close.

Now that we have everything set up correctly, we can begin creating a simple application.

Step 3: Writing the Code to be Called

Once we are back in Visual Studio, we need to add some code to our Visual Basic Class Library that our MapBasic application can call. We will add the following class:

Public Class DebugTest

```
Public Shared Sub ShowHelloWorld()  
    MsgBox("Hello World")  
End Sub
```

End Class

Note that the method to be called needs to be declared as a “Shared” sub in order for the MapBasic application to access it. For now we will save the class and move on to creating our MapBasic application.

Step 4: Creating the Mapbasic Application

Here is an example of a simple MapBasic application that will call our .NET method.

```
Include "MapBasic.def"
```

```
Declare Sub Main()
```

```
Declare Method ShowHelloWorld Class "MapInfoDebugTest.DebugTest" Lib "MapinfoDebugTest.dll"
```

```
Sub Main()
```

```
    Call ShowHelloWorld
```

```
End Sub
```

NOTE! The above code is explained in depth in the MapBasic user guide.

As you can see, it will just call the method without passing anything through to .NET.

We can now save & compile our MapBasic application.


Step 5: Debugging the Application

The final step in this article is debugging our application after we have called it from MapInfo.

Now that we have created and linked everything, we are ready to start debugging our application. In order to do so, we should head back into our Visual Studio project.

Before we run our application, we can set a breakpoint in our .NET code like so:

```
Public Shared Sub ShowHelloWorld()  
    MsgBox("Hello World")  
End Sub
```

Once we have set our breakpoints in our code, we can go ahead and click on the debug button:  If everything has been set up correctly, MapInfo should load and be attached to the debugger. After MapInfo has loaded, we can run the MapBasic application created earlier.

After the MapBasic application has been run, it should call the .NET method and break in the Visual Studio debugger. Now that we are attached to the Visual Studio debugger, we can do anything we would normally be able to do in the debugger.

Have fun!