Articles » General Programming » Programming Tips » General

Creating your own library in Visual Studio

Chrene91, 26 Jan 2012 CPOL

★★★★☆ 4.20 (5 votes)

A tip to fellow programmers developing in the .NET environment

It is common when you are new to Visual Studio that you will at some point be writing some good algorithms you want to keep, so you can use them in other projects. Therefore you should always have a library / *.dll* file you can import to your projects so you don't have to rewrite them over and over for each project you will use them in.

So my hint is: go open up Visual Studio and create a new project. Select class library and name it something catchy like "`MyFreakingAwesomeLibrary`".

You will now see an auto generated class called "`Class1`". Go delete that one.

When you have a library, it is good practice to have some namespaces (folders and subfolders) which will hold classes according to their category. Say for instance, you have written some good math algorithms about pythagoras. Then you should add those classes holding those classes to a namespace called `trigonometry` *under another namespace called* `math`.

Well let's say you want to write some pythagoras algorithms like above. Go to your solution explorer and right click your project and select new -> folder. Name that folder "*math*". Then right click your new folder and select new -> folder. Name that folder "*trigonometry*". Now right click that folder and select new -> class and name that "pythagoras". You will now see a generated class called pyhagoras. Now go make that public.

You should now have a class looking like this:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace MyFreakingAwesomeLibrary.math.trigonometry
{
    public class Pythagoras
    {
    }
}
```

If you do, you have done a great job. Look again at this line:

```
namespace MyFreakingAwesomeLibrary.math.trigonometry
```

As you can see, you have made your very own library with namespaces now.

When you are coding your library, it is also a good idea to create a test project. So go to your Solution Explorer and right click your solution and add -> new -> project. Select Windows Forms application and name it "`testApp`".

Now go back to your Pythagoras class and add this method:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ClassLibrary1.math.trigonometry
{
    public class Pythagoras
    {
        public static double GetLengthOfHypotenuse(double a, double
b)
        {
            return Math.Sqrt(Math.Pow(a, 2) + Math.Pow(b, 2));
        }
    }
}
```

Now go back to your solution explorer. Right click your test app and click "set as startup project. Under your test app, right click "references" and click add reference. Under the tab "Projects", select your library and click ok.

Now double click "*Form1.cs*" and add a button. Double click a random spot within the form and scroll to the top where we have our "`using`" statements.
Now type in "`using <your class="" library="">.math.trigonometry;`"

Go back to you `form1` design and double click the button.

Insert this line:

```
MessageBox.Show(Pythagoras.GetLengthOfHypotenuse(2, 3).ToString());
```

Your code for `form1` should look like this:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using MyFreakingAwesomeLibrary.math.trigonometry;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            MessageBox.Show(Pythagoras.GetLengthOfHypotenuse(2,
3).ToString());
        }
    }
}
```

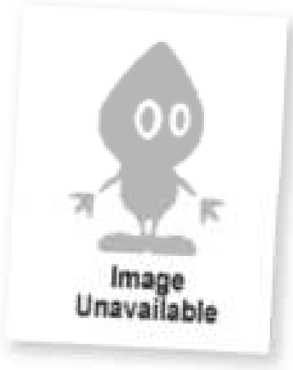
Now go run your test app and voila. You have now created your first lib.

If you want to use it later on, just add it as a reference to the project and do as we did in this little tip :)

License

Share

About the Author



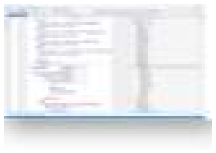
Chrene91

United States

No Biography provided

You may also be interested in...

Demystifying Cloud Latency



10 Ways to Boost COBOL Application Development

Building Boost libraries for Visual Studio

SAPrefs - Netscape-like Preferences Dialog

Visual Studio Add-in Library

Generate and add keyword variations using AdWords API

Comments and Discussions

8 messages have been posted for this article Visit **https://www.codeproject.com/Tips/319950/Creating-your-own-library-in-visual-studio** to post and view comments on this article, or click **here** to get a print view with messages.