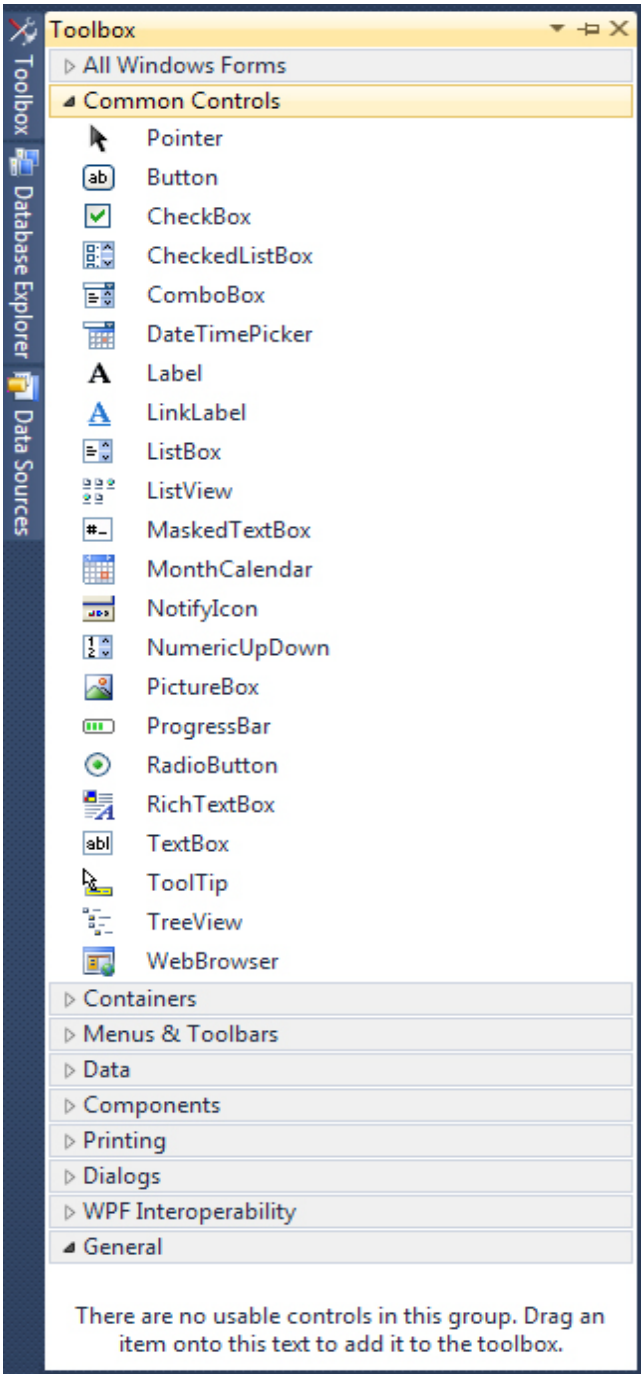
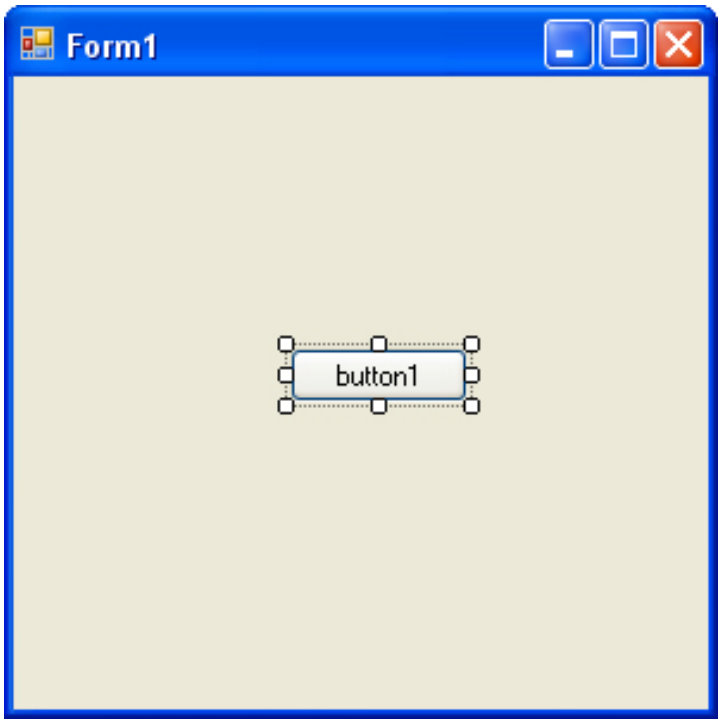
Adding Controls to a Blank Form

The first thing we will do is to add a button to the blank form. We will then write a single line of code, so that you can see how things work. If you want to add a control to a form, you can use the Toolbox on the left of Visual Studio. Move your mouse over to the Toolbox, and click the arrow symbol (or plus symbol) next to **Common Controls**. You should see the following list of things that you can add to your form:



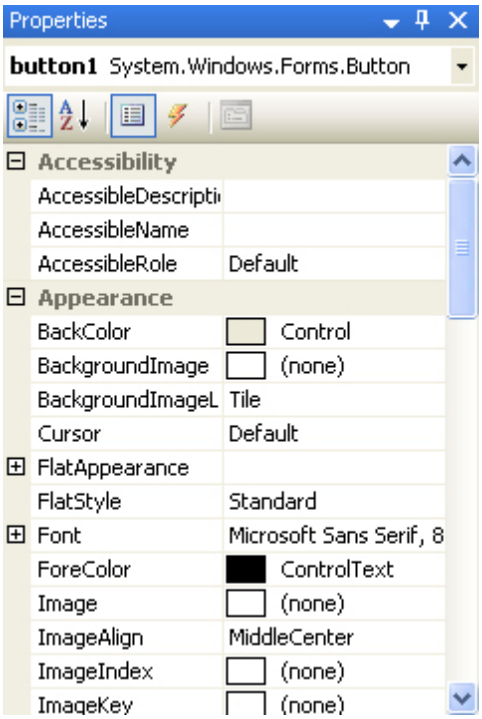
Click the **Button** item under the Common Controls heading. This will select it. Now click once anywhere on your form. A button will be drawn for you, and your Form will look like this:



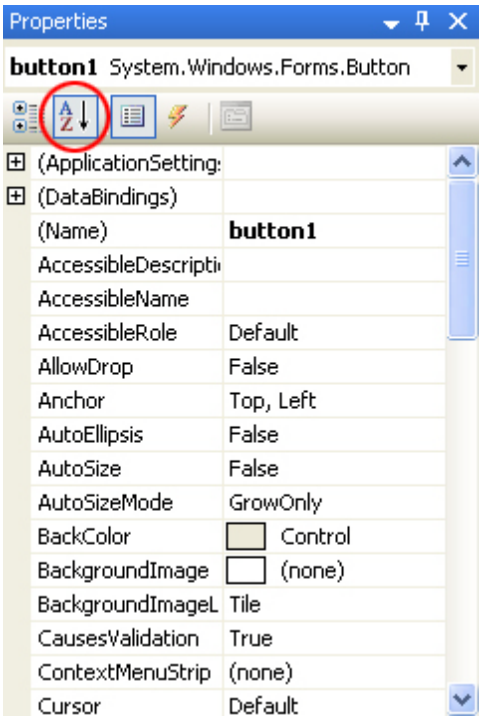
(You can also hold down your left mouse button and drag out a button to the size you want it). A button is something you want people to click on. When they do, the code you write gets executed. The text on the button, which defaults to "button1", can be changed. You can add anything you like here, but it should be something that is going to be useful for your users, such as "Open a text file", or "Calculate Now". We're going to display a simple message box when the button is clicked. Therefore, we need to add some text.

Properties of a Control

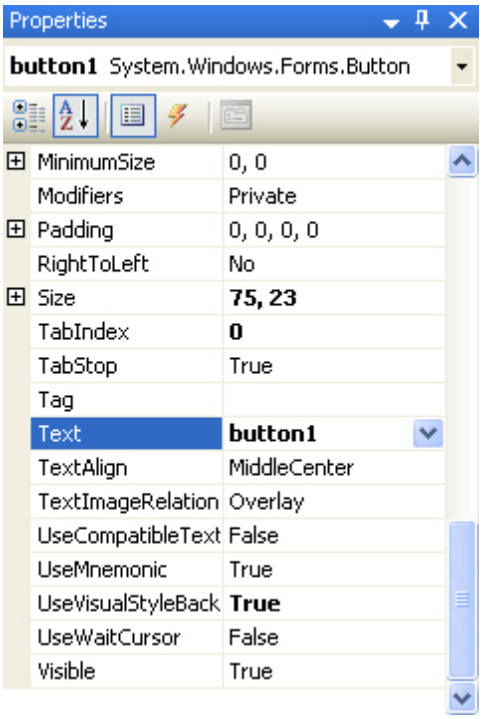
The controls you add to a form have something called **Properties**. A property of a control is things like its Height, its Width, its Name, its Text, and a whole lot more besides. To see what properties are available for a button, make sure the button is selected, as in the image above. If a control is selected, it will have white squares surrounding it. If your button is not selected, simply click it once. Now look in the bottom right of Visual C# Express, just below the Solution Explorer. You should see the Properties Window (if it is not there, select it from the **View** menu at the top. Again, 2010 users need to click **Tools > Settings > Expert Settings** to see the full list of menu items.):



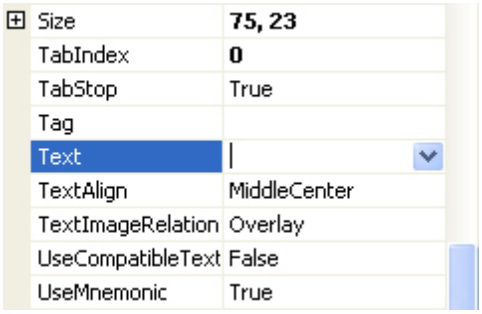
To view the list of Properties in alphabetical order, click the AZ symbol at the top, circled in red in the image below:



As you can see, there's a lot of Properties for a button. Scroll down to the bottom and locate the **Text** Property:



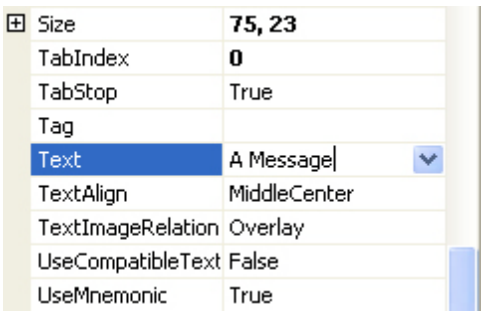
The Text Property, as its name suggests, is the Text you want to appear on the button. Now, it says button1. Click inside of the text area of button1. Delete the default text:



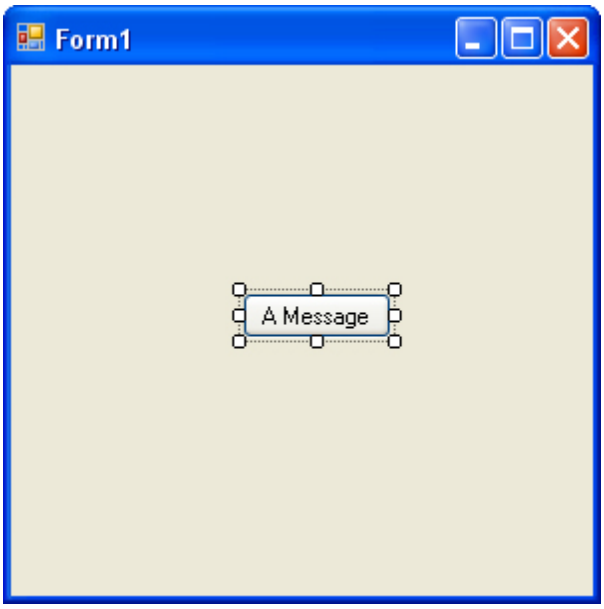
Now type the following:

A Message

The Text part of your Properties Window will then look like this:



Now press the enter key on your keyboard. Have a look at your Form, and the Text on the button should have changed:



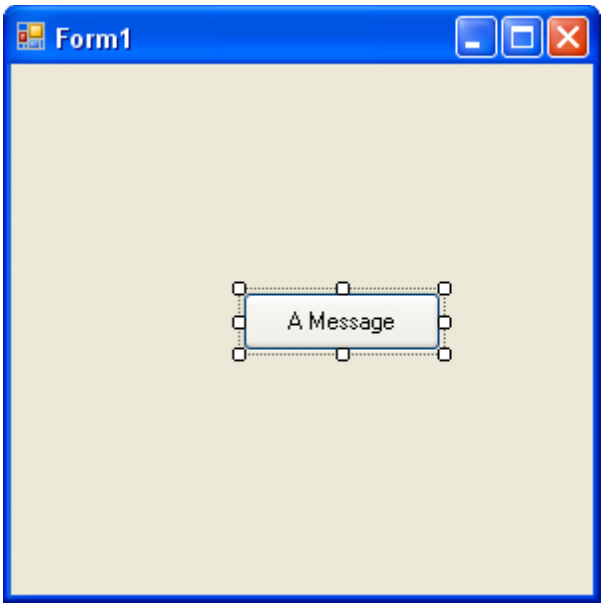
There are a few more Properties we can change before we get to the code. Locate **Size** in the Properties Window:

⊞ Padding	0, 0, 0, 0
RightToLeft	No
⊞ Size	**75, 23**
TabIndex	**0**
TabStop	True

The first number, 75, is the width of the button. The second number, 23, is the height of the button. The two numbers are separated by a comma. Change the numbers to 100, 30:

Modifiers	Private
⊞ Padding	0, 0, 0, 0
RightToLeft	No
⊞ Size	**100, 30**
TabIndex	**0**
TabStop	True
Tag	
Text	**A Message**

Press the enter key again, and the size of your button will change:

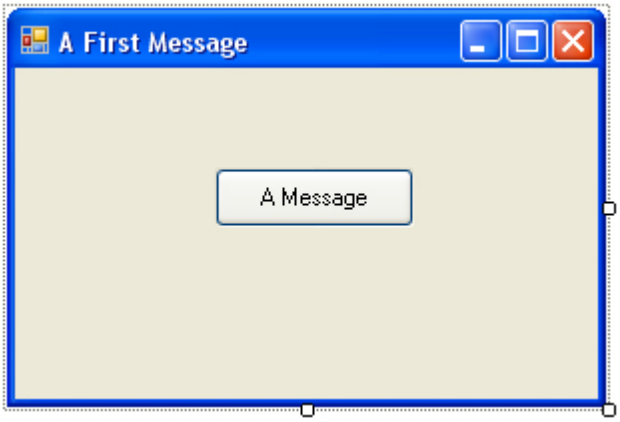


You can move your button around the Form by clicking it with the left mouse button to select it. Hold down you left mouse button and drag your button around the form. Let go of your left mouse button when you are happy with the new location.

Exercises:

1. You can also move a button by changing its **Location** property. Use the Properties Window to change the Location of your button to a position on the form of 100, 50. (100 means 100 units from the left edge of the form; 50 means 50 units down from the top of the form.)
2. A Form also has lots of Properties. Click away from the button and on to the Form itself. The Properties for the form will appear in the Properties Window. Change the **Text** Property of the Form to **A First Message**.
3. The Form, like the button, also has a Size Property. Change the Size of the Form to 300, 200.

After you complete the three exercises above, your Form should look like this:

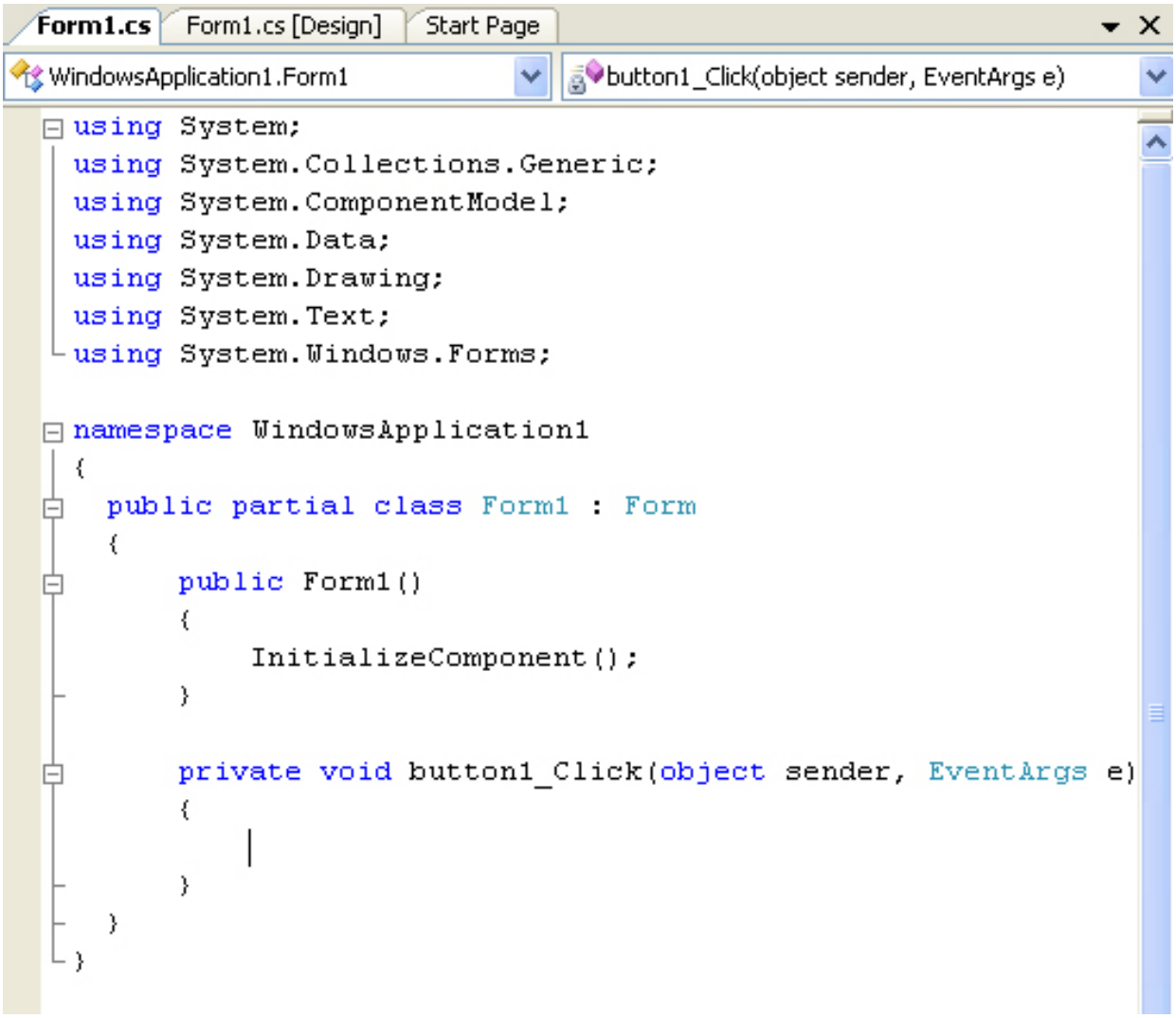


When you changed the Text property of the Form, you changed the text that runs across the blue bar at the top, the text in white. You can type anything you like here, but it should be something that describes what the form is all about or what it does. Often, you will see the name of the software here, like Microsoft Word, or Adobe Acrobat.

We can now move on to some code, and then run the Form to see what it all looks like?!!

Adding code to a Button

What we want to do is to display a **message box** whenever the button is clicked. Therefore, we need the coding window. To see the code for the button, double click the button you added to the Form. When you do, the coding window will open, and your cursor will be flashing inside of the button code. It will look like this:



The only thing different from the last time you saw this screen is the addition of the code for the button. This code:

```
private void button1_Click(object sender, EventArgs e)
  {
  }
```

This is just another Method, a piece of code that does something. The name of the Method is **button1_Click**. It is called button1 because that is currently the Name of the button. When you changed the Text, Location, and Size properties of the button, you could have also changed the **Name** property from button1 (the default Name) to something else. The _Click part after button1 is called an **Event**. Other events are MouseDown, LocationChanged, TextChanged, and lots more. You will learn more about Events later. After _Click, and in between a pair of round brackets, we have this:

```
object sender, EventArgs e
```

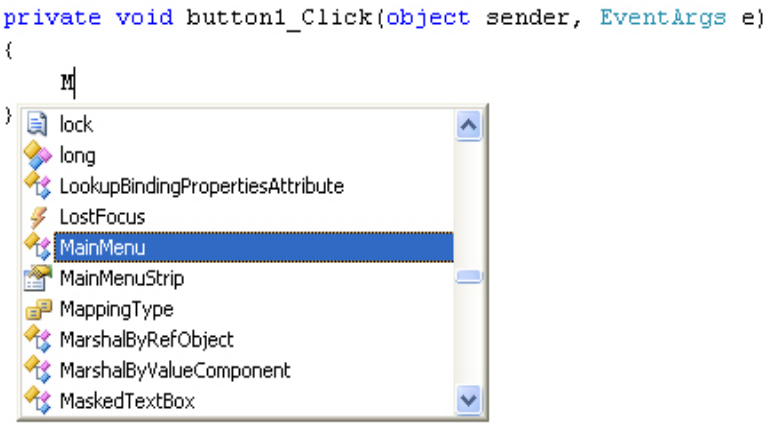
These two are known as arguments. One argument is called **sender**, and the other is called **e**. Again, you will learn more about arguments later, so do not worry about them for now. Notice that there is a pair of curly brackets for the button code:

```
private void button1_Click(object sender, EventArgs e)
  {
  }
```

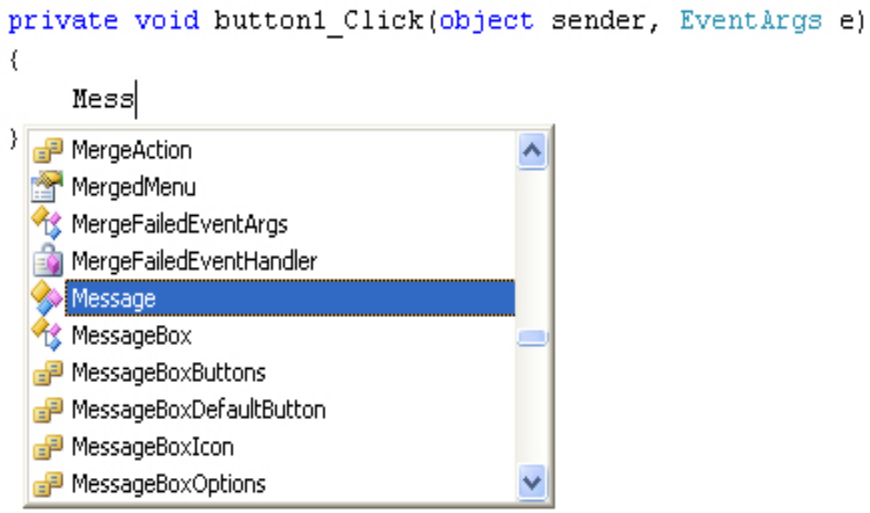
If you want to write code for a button, it needs to go between the two curly brackets. We will add a single line of code.

A MessageBox

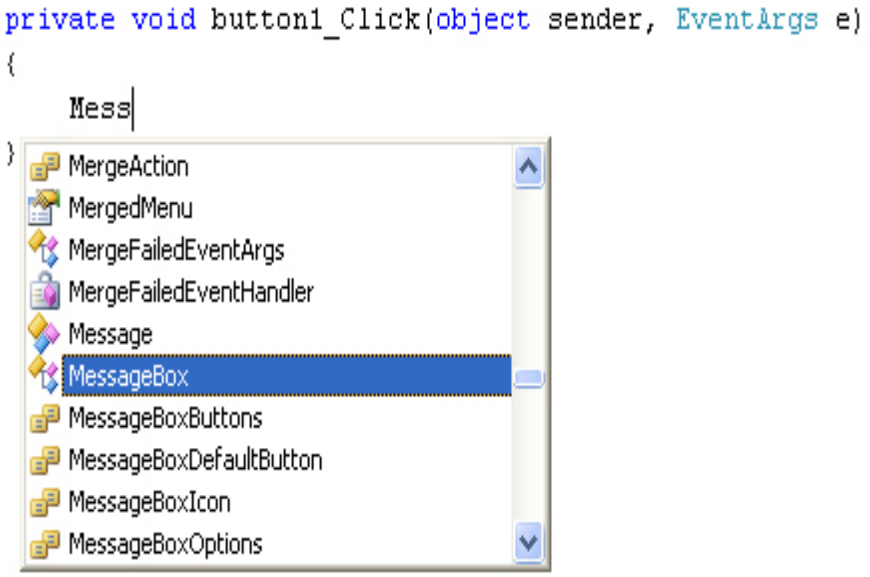
We want to display a message box, with some text on it. This is quite easy to do in C#. Position your cursor between the two curly brackets. Then type a capital letter "**M**". You will see the IntelliSense list appear:



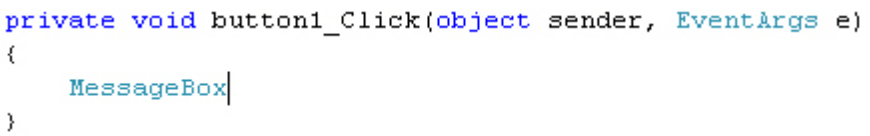
Now type “ess” after the “M”. IntelliSense will jump down:



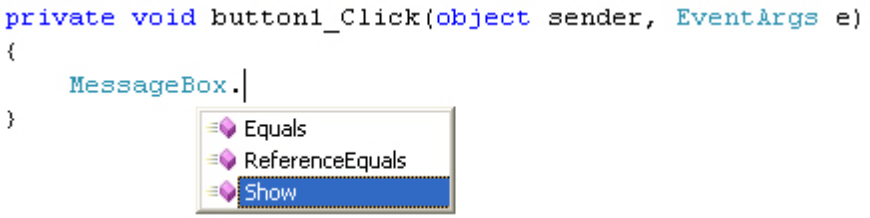
The only options that start with Mess are all Message ones. The one we want is MessageBox. You can either just type the rest, or even easier is to press the down arrow on your keyboard to move down to MessageBox:



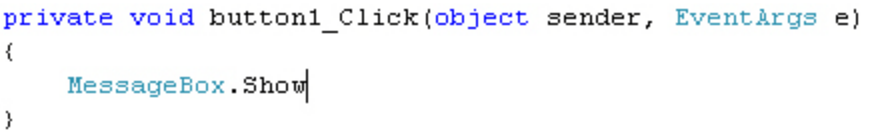
When you have MessageBox selected, hit the enter key on your keyboard (or double click the entry on the list, or press the tab key). The code will be added for you:



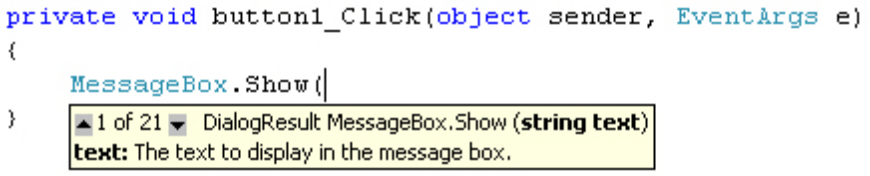
Now type a full stop (period) after the "**x**" of **MessageBox**. The IntelliSense list will appear again:



There are only three items on the list now, and all Methods (you can tell they are Methods because they have the purple block icon next to them) Double click on **Show**, and it will be added to your code:



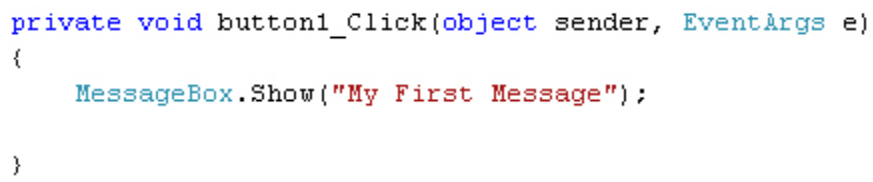
Because **Show** is a Method, we need some round brackets. The text for our message box will go between the round brackets. So type a left round bracket, just after the "w" of "Show":



As soon as you type the left round bracket after the "w", you'll see all the different ways that the Show method can be used. There are 21 different ways in total. Fortunately, you don't have to hunt through them all! Type the following, after the left round bracket: (Don't forget the double quotation marks. But don't copy and paste below, because the quote marks are a different style from the ones C# uses.)

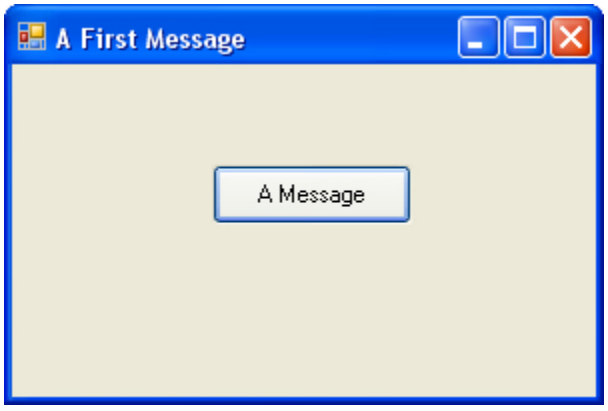
"My First Message"

After the final double quote mark, type a right round bracket. Then finish off the line by typing a semi-colon (**;**), Your coding window will then look like this:



The text in the dark reddish colour is what will be displayed in your message box. To try it out, save your work by clicking **File** from the menu bar at the top of Visual Studio. From the File menu,

click **Save All**. You will then see the same **Save** box. Save the project. Run your program by clicking **Debug > Start Debugging**. Or just press the F5 key on your keyboard. Your program will look like this:



Click your button to see your Message Box:

Congratulations! It is your first message!

Other things to do with the Message Box

If you look at the message box in the image above, you will notice there is no Title in the blue area to the left of the red X – it is blank. You can add a Title quite easily. Click OK on your Message Box. Then click the Red X on your program to exit it. This will return you to Visual Studio. Go back to the coding window (press F7 on your keyboard, if you cannot see it). Position your cursor after the final double quote of "My First Message", circled in red in the image below:

```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show("My First Message");

}
```

Now type a comma. As soon as you type a comma, you will see the list of **Show** options again:

```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show("My First Message",);
}
```

2 of 21 DialogResult MessageBox.Show (IWin32Window owner, **string text**)
Displays a message box in front of the specified object and with the specified text.

Type the following:

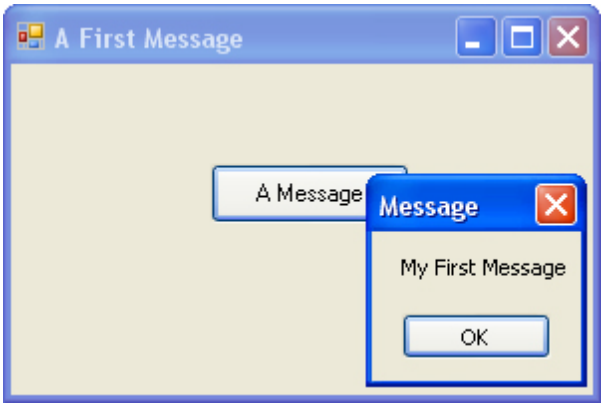
"Message"

Again, you need the double quotes. But your line of code should look like this:

```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show("My First Message", "Message");

}
```

When your line of code looks like the one above, Run your program again. Click your button and you should see a Title on your Message Box:

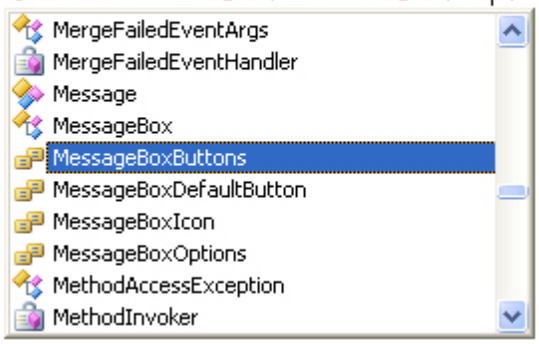


Other Button Options

Rather than having just an OK button, you can add buttons like Yes, No, and Cancel. We will add a **Yes** and a **No** button. Return to your coding window. After the second double quote of the Title you have just added, type another comma. Hit the spacebar on your keyboard once, and you will see the IntelliSense list appear. (If it does not appear, just type a capital letter "M").

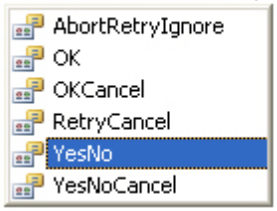
```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show("My First Message", "Message", M );

}
```



The one that adds buttons to a message box is, you will not be surprised to hear, **MessageBoxButtons**. Press the enter key on your keyboard when this option is highlighted. It will be added to your code. Now type a full stop (period) after the final "s" of MessageBoxButtons. You will see the button options:

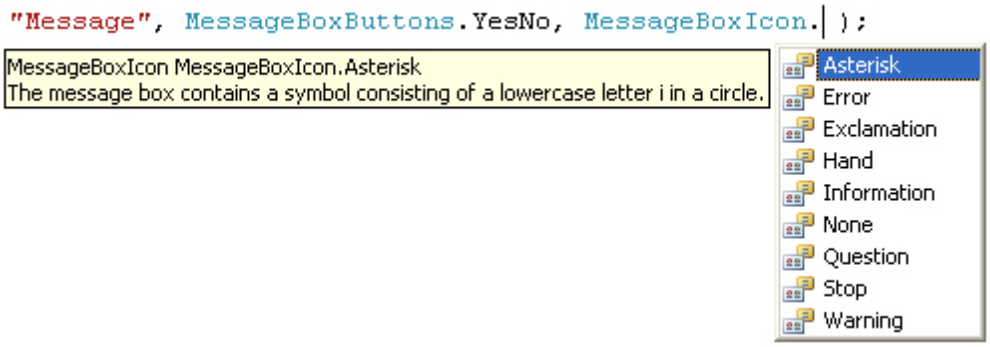
```
"Message", MessageBoxButtons.| );
```



Double click the one for **YesNo**, and it will be added to your code. Run your program again, and click your button. Your Message Box will then look like this:

Adding Icons to a Message Box

Another thing you can add to brighten up your Message Box is an Icon. It is easier to see what these are than to explain! Type another comma after **MessageBoxButtons.YesNo**. After the comma, type a capital letter "M" again. From the IntelliSense list that appears, double click **MessageBoxIcon**. After MessageBoxIcon, type a full stop to see the available icons:



We have gone for **Asterisk**. Double click this to add it to your code. Run your program again to see what the icon looks like on your Message Box:

Looks pretty impressive, hey! And all that with one line of code!