MessageBox. Stop what you are doing. Pay attention to me now. Dialog boxes interrupt users. They force users to respond before further action is taken.

Warnings, errors. MessageBox.Show is useful if a warning or error is important. This tutorial begins with eight different method calls. A complete example program is provided afterwards.

Example 1. We can call MessageBox.Show with just one argument. This is a string argument. An OK button is provided to dismiss the dialog.

C# program that uses MessageBox

```
//
// The simplest overload of MessageBox.Show. [1]
//
MessageBox.Show("Dot Net Perls is awesome.");
```

Example 2. Let's add a second argument to our Show method call. The second argument is also a string. Sorry about the silly dialog text.

C# program that uses two arguments

```
//
// Dialog box with text and a title. [2]
//
MessageBox.Show("Dot Net Perls is awesome.",
    "Important Message");
```

Example 3. Sometimes we want to ask a question. The question is answered with Yes or No. We can provide the MessageBoxButtons.YesNo enum argument for this.

DialogResult:
We assign a variable to the result of MessageBox.Show. We can later test result1 to find out which button was clicked.

C# program that uses three arguments

```
//
```

```
// Dialog box with two buttons: yes and no. [3]
//
DialogResult result1 = MessageBox.Show("Is Dot Net Perls awesome?",
    "Important Question",
    MessageBoxButtons.YesNo);
```

Example 4. This example adds another argument, the MessageBoxIcon.Question enum value. Other values (other than Question) can be used here.

C# program that uses four arguments

```
//
// Dialog box with question icon. [4]
//
DialogResult result2 = MessageBox.Show("Is Dot Net Perls awesome?",
    "Important Query",
    MessageBoxButtons.YesNoCancel,
    MessageBoxIcon.Question);
```

Example 5. We can specify a default button with the MessageBoxDefaultButton argument. Here we use Button2 to mean the second button should be default.

C# program that five arguments

```
//
// Dialog box with question icon and default button. [5]
//
DialogResult result3 = MessageBox.Show("Is Visual Basic awesome?",
    "The Question",
    MessageBoxButtons.YesNoCancel,
    MessageBoxIcon.Question,
    MessageBoxDefaultButton.Button2);
```

Example 6. Here we interrupt our logic to test the results of the previous button presses. We test for DialogResult.Yes and No. Then we should another message box.

Tip:

In most programs we would test just one DialogResult at a time. But here we test three in a single expression.

C# program that tests DialogResult

```
//
// Test the results of the previous three dialogs. [6]
//
if (result1 == DialogResult.Yes &&
    result2 == DialogResult.Yes &&
    result3 == DialogResult.No)
{
    MessageBox.Show("You answered yes, yes and no.");
}
```

Example 7. We can align buttons on the dialog. Here we align the buttons to the right. This is not a useful example, but some programs may need alignment.

C# that six arguments

```
//
// Dialog box that is right-aligned (not useful). [7]
//
MessageBox.Show("Dot Net Perls is the best.",
    "Critical Warning",
    MessageBoxButtons.OKCancel,
    MessageBoxIcon.Warning,
    MessageBoxDefaultButton.Button1,
    MessageBoxOptions.RightAlign,
    true);
```

Example 8. Do you like icons? I like icons. Here we use an exclamation on our dialog. We specify MessageBoxIcon.Exclamation in the fourth method call. Again sorry for the message text.

C# that uses MessageBox, eight arguments

```
//
// Dialog box with exclamation icon. [8]
//
MessageBox.Show("Dot Net Perls is super.",
    "Important Note",
    MessageBoxButtons.OK,
    MessageBoxIcon.Exclamation,
    MessageBoxDefaultButton.Button1);
```

Complete program. The MessageBox.Show method is a static method. This means you do not need to create a new MessageBox() anywhere in your code.

Instead:
You can simply type "MessageBox" and press the period, and then select Show.

Here:
In this example, the MessageBox.Show method is used in the Form1_Load event handler.

Tip:
To make the Form1_Load event handler, create a new Windows Forms application and double-click on the window in the designer.

Windows Forms program that uses MessageBox: C#

```
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;

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

        private void Form1_Load(object sender, EventArgs e)
        {
            //
            // The simplest overload of MessageBox.Show. [1]
            //
            MessageBox.Show("Dot Net Perls is awesome.");
            //
            // Dialog box with text and a title. [2]
            //
            MessageBox.Show("Dot Net Perls is awesome.",
                "Important Message");
            //
            // Dialog box with two buttons: yes and no. [3]
            //
            DialogResult result1 = MessageBox.Show("Is Dot Net Perls awesome?",
                "Important Question",
                MessageBoxButtons.YesNo);
            //
            // Dialog box with question icon. [4]
            //
            DialogResult result2 = MessageBox.Show("Is Dot Net Perls awesome?",
                "Important Query",
                MessageBoxButtons.YesNoCancel,
                MessageBoxIcon.Question);
            //
            // Dialog box with question icon and default button. [5]
            //
            DialogResult result3 = MessageBox.Show("Is Visual Basic awesome?",
                "The Question",
                MessageBoxButtons.YesNoCancel,
                MessageBoxIcon.Question,
                MessageBoxDefaultButton.Button2);
            //
            // Test the results of the previous three dialogs. [6]
            //
            if (result1 == DialogResult.Yes &&
                result2 == DialogResult.Yes &&
                result3 == DialogResult.No)
            {
                MessageBox.Show("You answered yes, yes and no.");
            }
            //
            // Dialog box that is right-aligned (not useful). [7]
            //
            MessageBox.Show("Dot Net Perls is the best.",
                "Critical Warning",
                MessageBoxButtons.OKCancel,
                MessageBoxIcon.Warning,
                MessageBoxDefaultButton.Button1,
                MessageBoxOptions.RightAlign,
                true);
            //
            // Dialog box with exclamation icon. [8]
            //
            MessageBox.Show("Dot Net Perls is super.",
                "Important Note",
                MessageBoxButtons.OK,
                MessageBoxIcon.Exclamation,
                MessageBoxDefaultButton.Button1);
        }
    }
}
```

In Form1_Load, there are eight calls to MessageBox.Show. The Form1_Load method is executed immediately after the program starts. When run, the program shows all the dialogs in order.

> **And:**
> The MessageBox.Show calls above call into different, overloaded implementations of the function based on the parameter lists.
> **Overload**

Visual Studio. The easiest way to use MessageBox.Show is to type in "MessageBox", and then press period, and then select Show. Next, Visual Studio shows a popup with the overload list.

> **Tip:**
> You can scroll through the overload lists. This is pretty handle and makes tutorial sites a bit less useful.

> **Tip 2:**
> For a parameter such as "MessageBoxButtons", type in "MessageBoxButtons" and press period to see all the options.

>> **Also:**
>> You do not need to create a new MessageBoxButtons() object. This is an enum type, not a class.

Parameter order. The order of the parameters in the MessageBox.Show method calls is important. The compiler applies overload resolution to call the best method in the method group.

> **Images:**
> The image at the top of this document shows eight dialog boxes. These correspond to MessageBox.Show calls.

> **Note:**
> Dialog box [6] only is shown when you specify certain options on the previous three dialogs. It tests the DialogResult enumeration.

DialogResult. This is an enum. This means you cannot create a new DialogResult with the "new" operator.

First assign your variable to the result of MessageBox.Show.

Next:
Type in "==" and Visual Studio will suggest options from the DialogResult enumeration.

Tip:
You can compare DialogResult like you would compare an integral type such as int. You can even use it in a switch.

Overloads. There are several more overloads of MessageBox.Show that are not shown in this document. They allow you to specify owner windows, which you do not need to do in simple cases.

Interface:
The IWin32Window owner parameter is an interface type. Interfaces treat object instances in a more general way.
Interface

HelpNavigator parameter. The MessageBox.Show method also has overloads that allow you to specify Help options. In my experience, these options are not usually needed.

User experience. When designing programs for the end user, it is usually best to make non-critical errors as unobtrusive as possible.

Here:
The Microsoft User Experience Guidelines provide many tips on dialog boxes.

Well-written, helpful error messages are crucial to a quality user experience. Poorly written error messages result in low product satisfaction, and are a leading cause of avoidable technical support costs. Unnecessary error messages break users' flow.
Error Messages: MSDN

A summary. MessageBox.Show is an effective approach to dialog boxes in Windows Forms. We looked at screenshots of the results of the MessageBox.Show method.

Many options. We can choose between many parameters to the static method. The MessageBox.Show method is ideal for many simpler Windows Forms programs.