

 無料電子ブック

学習

winforms

Free unaffiliated eBook created from  
**Stack Overflow contributors.**

#winforms

.....	1
<b>1: winforms</b> .....	<b>2</b>
.....	2
.....	2
Examples.....	2
Visual StudioWinForms.....	2
Windows.....	2
.....	3
.....	4
.....	5
CWinForms.....	5
VB.NET WinForms.....	6
<b>2:</b> .....	<b>9</b>
.....	9
.....	9
Examples.....	9
.....	9
<b>3:</b> .....	<b>10</b>
Examples.....	10
.....	10
.....	10
.....	10
.....	11
<b>4:</b> .....	<b>12</b>
.....	12
Examples.....	12
.....	12
.....	12
.....	13
<b>5:</b> .....	<b>14</b>
.....	14

HelpProvider.....	14
.....	14
HelpRequested.....	14
.....	14
MessageBoxCommonDialogs.....	14
.....	15
Examples.....	15
.....	15
MessageBox.....	15
CHM.....	16
CHM.....	16
CHM.....	16
URL.....	16
F1.....	16
CommonDialogs.....	17
HelpRequested.....	18
.....	19
.....	19
CHM.....	19
.....	19
.....	19
.....	20
URL.....	20
.....	20
.....	20
HelpButtonClicked.....	21
<b>6:</b> .....	<b>22</b>
Examples.....	22
.....	22
.....	22
.....	23
CheckBox.....	25

.....	25
NumericUpDown.....	29
.....	31
<b>7:</b> .....	<b>33</b>
.....	33
Examples.....	33
.....	33
NumberBox.....	36
.....	<b>44</b>

---

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [winforms](#)

It is an unofficial and free winforms ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official winforms.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to [info@zzzprojects.com](mailto:info@zzzprojects.com)

# 1: winformsをいめる

**Windows** フォーム「WinForms」とは、.NET FrameworkにするGUIクラスライブラリです。これはWin32 APIをとしたされたオブジェクトのラッパーで、.NET FrameworkをターゲットとするWindowsデスクトップおよびモバイルアプリケーションのをにします。

WinFormsはにイベントドリブンです。アプリケーションは、ユーザーがやりとりするコントロール ラベル、ボタン、テキストボックス、リストなどをむのフォームにウィンドウとしてでされています。これらのコントロールは、ユーザーのにして、プログラムができるイベントをさせてタスクをします。

Windowsと、WinFormsのすべてはコントロールであり、それがウィンドウのです。Controlクラスは、テキスト、サイズ、およびをするプロパティ、およびできるのイベントセットをむなをします。すべてのコントロールはControlクラスからし、をしています。のコントロールでは、Form、 UserControl またはレイアウト TableLayoutPanel、 FlowLayoutPanel のいずれかののコントロールをホストできます。

WinFormsは.NET Frameworkv1.0のオリジナルサポートされており、v4.5でもききできます。しかし、それはもはやにされておらず、しいはされていません。Build 2014カンファレンスの9のMicrosoftによると、

Windowsフォームはききサポートされていますが、メンテナンスモードです。らはされたバグをしますが、しいはテーブルかられています。

クロスプラットフォームのオープンソースのMonoライブラリは、Windows Formsのなをし、.NET 2.0のMicrosoftのがたしたすべてのをサポートしています。しかし、WinFormsはMonoでにされておらず、ネイティブWindows APIのプラットフォームではできませんとフレームワークがいかくにリンクしているかをえると、なはとみなされます。

- MSDNのWindowsフォームドキュメント

## Examples

Visual StudioをしてなWinFormsアプリケーションをする

このでは、Visual StudioでWindowsフォームアプリケーションプロジェクトをするをします。

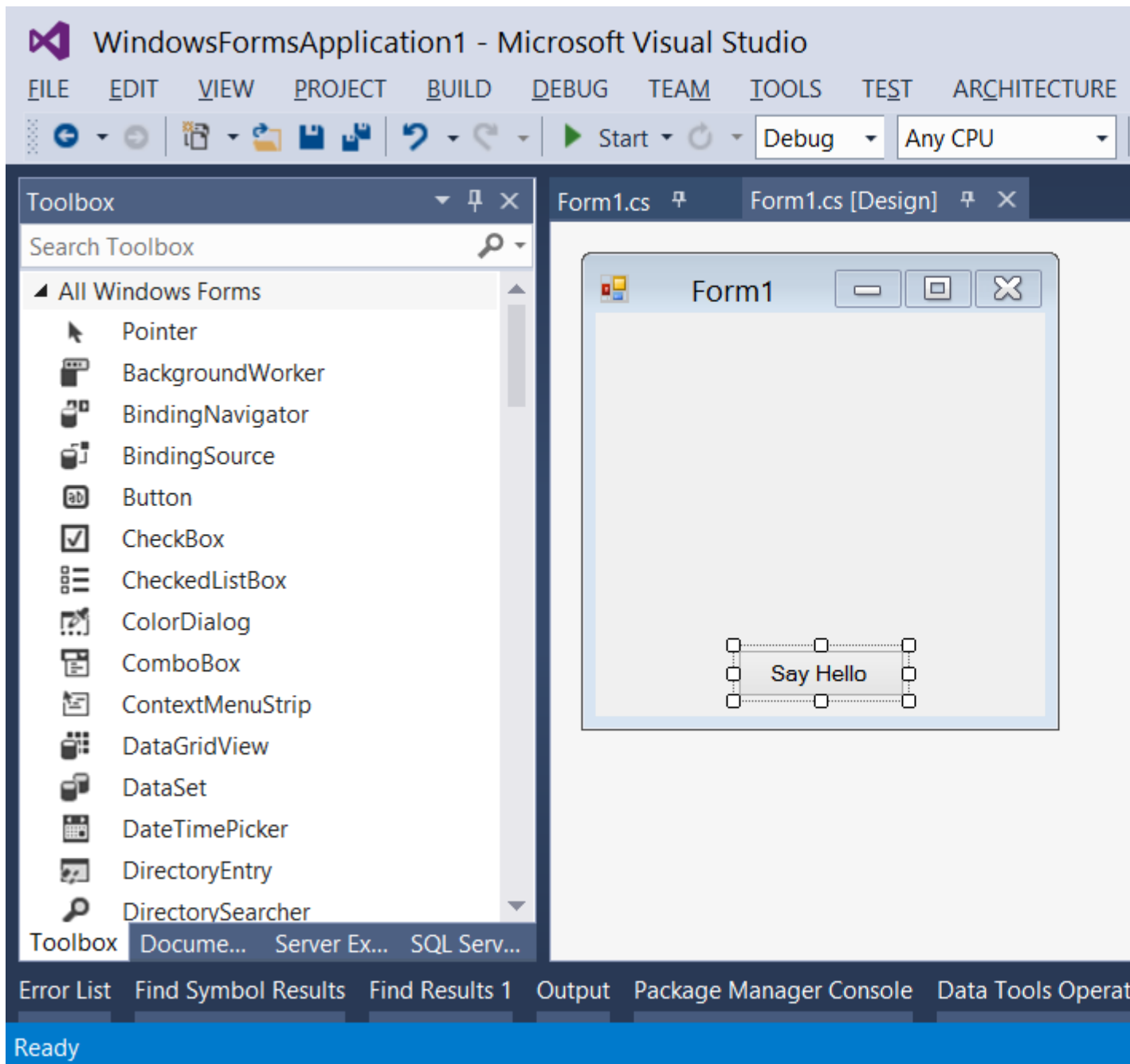
## Windows フォームプロジェクトの

1. Visual Studioをします。
2. [ファイル]メニューの[ ]をポイントし、[プロジェクト]をクリックします。[プロジェクト]ダイアログボックスがされます。

3. [インストールされているテンプレート]ウィンドウで、[Visual C]または[Visual Basic]をします。
4. ののには、ドロップダウンリストからターゲットフレームワークをできます。
5. のウィンドウで、 **Windows** フォームアプリケーションテンプレートをしします。
6. [ ]テキストボックスに、プロジェクトのをしします。
7. [ ]テキストボックスで、プロジェクトをするフォルダをしします。
8. [**OK**]をクリックしします。
9. Windows フォームデザイナーがき、プロジェクトの**Form1**がされます。

## フォームにコントロールをする

1. ツールボックスパレットから、 **Button** コントロールをフォームにドラッグしします。
2. ボタンをクリックしてしします。プロパティウィンドウで、 `Text` プロパティを **Say Hello** にしします。



## コードをく

1. ボタンをダブルクリックして `Click` イベントのイベントハンドラを `Click` ます。コードエディタがき、イベントハンドラにポイントがされます。
2. のコードをします。

### C

```
MessageBox.Show("Hello, World!");
```

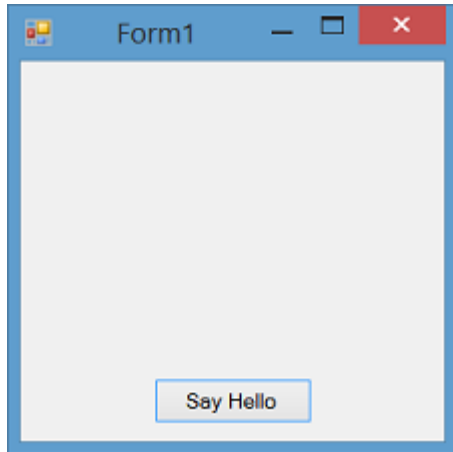
### VB.NET



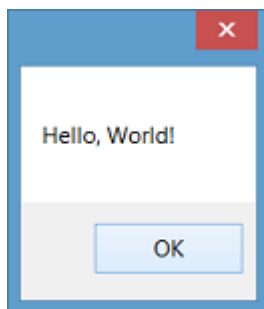
```
MessageBox.Show("Hello, World!");
```

## とテスト

1. F5キーをしてアプリケーションをします。



2. アプリケーションがしているときは、ボタンをクリックして「Hello、World」をします。メッセージ。



3. フォームをじてVisual Studioにります。

## テキストエディタをしたなCWinFormsアプリケーションの

1. テキストエディタメモなどをき、のコードをします。

```
using System;
using System.ComponentModel;
using System.Drawing;
using System.Windows.Forms;

namespace SampleApp
{
    public class MainForm : Form
    {
        private Button btnHello;

        // The form's constructor: this initializes the form and its controls.
        public MainForm()
        {
```

```

// Set the form's caption, which will appear in the title bar.
this.Text = "MainForm";

// Create a button control and set its properties.
btnHello = new Button();
btnHello.Location = new Point(89, 12);
btnHello.Name = "btnHello";
btnHello.Size = new Size(105, 30);
btnHello.Text = "Say Hello";

// Wire up an event handler to the button's "Click" event
// (see the code in the btnHello_Click function below).
btnHello.Click += new EventHandler(btnHello_Click);

// Add the button to the form's control collection,
// so that it will appear on the form.
this.Controls.Add(btnHello);
}

// When the button is clicked, display a message.
private void btnHello_Click(object sender, EventArgs e)
{
    MessageBox.Show("Hello, World!");
}

// This is the main entry point for the application.
// All C# applications have one and only one of these methods.
[STAThread]
static void Main()
{
    Application.EnableVisualStyles();
    Application.Run(new MainForm());
}
}
}

```

2. みきなパスにファイルをします。、 x:\MainForm.cs など、そのファイルにまれるクラスのにファイルをけます。
3. コマンドラインからCコンパイラをし、コードファイルへのパスをとしてします。

```
%WINDIR%\Microsoft.NET\Framework64\v4.0.30319\csc.exe /target:winexe "X:\MainForm.cs"
```

の.NETフレームワークバージョンにCコンパイラのバージョンをするには、パス %WINDIR%\Microsoft.NET をて、それにじてのをしてください。Cアプリケーションのコンパイルについては、[のCプログラムをコンパイルしてするを](#)してください。

4. コンパイルがすると、 MainForm.exe というのアプリケーションがコードファイルとじディレクトリにされます。このアプリケーションは、コマンドラインからすることも、エクスプローラでダブルクリックすることもできます。

テキストエディタをしたな**VB.NET WinForms**アプリケーションの

1. テキストエディタメモなどをき、のコードをします。

```

Imports System.ComponentModel
Imports System.Drawing
Imports System.Windows.Forms

Namespace SampleApp
    Public Class MainForm : Inherits Form
        Private btnHello As Button

        ' The form's constructor: this initializes the form and its controls.
        Public Sub New()
            ' Set the form's caption, which will appear in the title bar.
            Me.Text = "MainForm"

            ' Create a button control and set its properties.
            btnHello = New Button()
            btnHello.Location = New Point(89, 12)
            btnHello.Name = "btnHello"
            btnHello.Size = New Size(105, 30)
            btnHello.Text = "Say Hello"

            ' Wire up an event handler to the button's "Click" event
            ' (see the code in the btnHello_Click function below).
            AddHandler btnHello.Click, New EventHandler(AddressOf btnHello_Click)

            ' Add the button to the form's control collection,
            ' so that it will appear on the form.
            Me.Controls.Add(btnHello)
        End Sub

        ' When the button is clicked, display a message.
        Private Sub btnHello_Click(sender As Object, e As EventArgs)
            MessageBox.Show("Hello, World!")
        End Sub

        ' This is the main entry point for the application.
        ' All VB.NET applications have one and only one of these methods.
        <SThread> _
        Public Shared Sub Main()
            Application.EnableVisualStyles()
            Application.Run(New MainForm())
        End Sub
    End Class
End Namespace

```

- みきなパスにファイルをします。、 X:\MainForm.vbように、ファイルにまれるクラスのにファイルをけるのがです。
- コマンドラインからVB.NETコンパイラをし、コードファイルへのパスをとします。

```
%WINDIR%\Microsoft.NET\Framework64\v4.0.30319\vbc.exe /target:winexe "X:\MainForm.vb"
```

の.NETフレームワークのバージョンにVB.NETコンパイラのバージョンをするには、パス %WINDIR%\Microsoft.NET をて、のをしてください。VB.NETアプリケーションのコンパイルについては、「[Hello World](#)」をしてください。

- コンパイルがすると、 MainForm.exe というのアプリケーションがコードファイルとじディレ

クトリにされます。このアプリケーションは、コマンドラインからすることも、エクスプローラでダブルクリックすることもできます。

オンラインでwinformsをいめるをむ <https://riptutorial.com/ja/winforms/topic/1018/winforms>をいめる

## 2: データバインディング

### パラメーター

プロパティ	バインドするコントロールのプロパティの。
	データソースをすObject
dataMember	バインドするプロパティまたはリスト。
formattingEnabled	されたデータをフォーマットするかどうかをします。
updateMode	データソースは、コントロールプロパティがされたときデフォルト、またはプロパティがされたときにされます
nullValue	データソースにこのがされている、バインドされたプロパティはDBNullにされます。
formatString	のをす1つの
formatInfo	IFormatProviderので、デフォルトのをオーバーライドします。

データバインドはプロパティでのみし、フィールドではできません。 <https://msdn.microsoft.com/en-us/library/ef2xyb33.aspx>

## Examples

### データオブジェクトへのコントロールのバインド

コントロールには、 `System.Windows.Forms.Binding` オブジェクトのリストであるプロパティ `DataBindings` があります。 `Add` - メソッドには、オブジェクトのプロパティににバインドできるいくつかのオーバーロードがあります。

```
textBox.DataBindings.Add( "Text", dataObj, "MyProperty" );
```

そのバインディングは、に、いのイベントをすることをします。のコードは `dataObj.MyProperty` の `changeevent` にし、されたときに `textBox.Text` をさせます。また、そののは、 `textBox.TextChanged` をサブスクライブし、に `dataObj.MyProperty` をさせます。

オンラインでデータバインディングをむ <https://riptutorial.com/ja/winforms/topic/7362/データバインディング>

## 3: テキストボックス

### Examples

のコレクションからの

```
var source = new AutoCompleteStringCollection();

// Add your collection of strings.
source.AddRange(new[] { "Guybrush Threepwood", "LeChuck" });

var textBox = new TextBox
{
    AutoCompleteCustomSource = source,
    AutoCompleteMode = AutoCompleteMode.SuggestAppend,
    AutoCompleteSource = AutoCompleteSource.CustomSource
};

form.Controls.Add(textBox);
```

ユーザーが**G**または**L**をしようとする、これがにします。

`AutoCompleteMode.SuggestAppend`はされたのリストをし、にするものをにします `Append`のみと `Suggest`のみがです。

テキストののみをする

```
textBox.KeyPress += (sender, e) => e.Handled = !char.IsControl(e.KeyChar) &&
!char.IsDigit(e.KeyChar);
```

これにより、`TextBox`のとののみがされます。テキストをブロックするには、`Handle`プロパティを `true`にするのとじをしてのみわせもです。

ユーザーはなをコピーしてりけることができるので、`TextChanged`にのチェックを `TextChanged`てを `TextChanged`するがあります。

```
textBox.TextChanged += (sender, e) => textBox.Text = Regex.Match(textBox.Text, @"\d+").Value
```

このでは、をしてテキストをフィルタリングしています。

なは**NumericUpDown**をにするがあります。

までスクロールする

```
textBox.SelectionStart = textBox.TextLength;
textBox.ScrollToCaret();
```

じをすると、 `SelectionStart` を 0 にするとにスクロールしたり、 のにしてのにすることができます。

テキストボックスにプレースホルダをする

このコードは、 ヒントテキストをフォームのみみにし、 のようにします。

## C

```
private void Form_load(object sender, EventArgs e)
{
    textBox.Text = "Place Holder text...";
}

private void textBox_Enter(object sender, EventArgs e)
{
    if(textBox.Text == "Place Holder text...")
    {
        textBox.Text = "";
    }
}

private void textBox_Leave(object sender, EventArgs e)
{
    if(textBox.Text.Trim() == "")
    {
        textBox.Text = "Place Holder text...";
    }
}
}
```

## VB.NET

```
Private Sub Form_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    textBox.Text = "Place Holder text..."
End Sub

Private Sub textBox_GotFocus(sender as Object,e as EventArgs) Handles textBox.GotFocus
    if Trim(textBox.Text) = "Place Holder text..." Then
        textBox.Text = ""
    End If
End Sub

Private Sub textBox_LostFocus(sender as Object,e as EventArgs) Handles textBox.LostFocus
    if Trim(textBox.Text) = "" Then
        textBox.Text = "Place Holder text..."
    End If
End Sub
```

オンラインでテキストボックスをむ <https://riptutorial.com/ja/winforms/topic/4674/テキストボックス>

## 4: フォームをする

き

このトピックでは、WinFormsエンジンがフォームをすると、フォームのライフタイムをするについてします。

### Examples

モードレスまたはモーダルフォームをする

WinFormsデザイナーでフォームのをした、2つのなるでコードをフォームにすることができます。

- メソッド - モデルレスフォーム

```
Form1 aForm1Instance = new Form1();
aForm1Instance.Show();
```

- メソッド - モーダルダイアログ

```
Form2 aForm2Instance = new Form2();
aForm2Instance.ShowDialog();
```

2つのはにないがあります。のモードレス1はフォームをし、にいたフォームのをたずにすぐになります。だからあなたのコードはショーのびしにくものになります。2のモーダル1では、フォームをき、じるボタンまたはフォームをじるようににされたボタンをしてフォームをじるまで、アプリケーションのアクティビティをブロックします

モードレスフォームをじる

アプリケーションのメインのにかをするがある、や、デバイスやMDIウィンドウからにデータストリームのビューをえる、モードレスフォームがされま。

しかし、あなたがそれをじたいときには、モードレスのフォームがユニークなになります。インスタンスをし、そのインスタンスでCloseメソッドをびすは

じたいインスタンスをするグローバルをすることができます。

```
theGlobalInstance.Close();
theGlobalInstance.Dispose();
theGlobalInstance = null;
```

しかし、フォームエンジンがされ、まだいているフォームインスタンスをすべてするApplication.OpenFormsコレクションをすることもできます。



このコレクションからそののインスタンスをし、Closeメソッドをびすことができます

```
Form2 toClose = Application.OpenForms.OfType<Form2>().FirstOrDefault();
if(toClose != null)
{
    toClose.Close();
    toClose.Dispose();
}
```

## モーダルフォームをじる

ShowDialogメソッドをしてフォームがされる、フォームのDialogResultプロパティをフォームにつけるようにするがあります。このプロパティは、DialogResultともばれるをしてできます。

フォームをじるには、あなただけのフォームのするがDialogResult によってのにプロパティを DialogResult.Noneいくつかのイベントハンドラで。コードがイベントハンドラをすると、WinFormエンジンはフォームをにし、のShowDialogメソッドびしにくコードはをします。

```
private cmdClose_Click(object sender, EventArgs e)
{
    this.DialogResult = DialogResult.Cancel;
}
```

びしコードは、ユーザーがフォームでクリックしたボタンをするために、ShowDialogからのりをできます。 ShowDialog()をしてすると、フォームはにはされませんにされていてじられていないためので、フォームをにするためにusingブロックをすることがです。

は、みみのOpenFileDialogをしたをチェックし、をチェックし、ダイアログからプロパティにアクセスするにそれにアクセスするです。

```
using (var form = new OpenFileDialog())
{
    DialogResult result = form.ShowDialog();
    if (result == DialogResult.OK)
    {
        MessageBox.Show("Selected file is: " + form.FileName);
    }
}
```

また、ボタンのDialogResultプロパティをすることもできます。このボタンをクリックすると、フォームのDialogResultプロパティがボタンにけられたにされます。これにより、コードに DialogResult をするイベントハンドラをせずにフォームをじることができます。

たとえば、フォームにOKボタンをし、そのプロパティをDialogResult.OKすると、そのボタンをすとフォームがにじられ、びしコードはShowDialog()メソッドびしからDialogResult.OKをけります。

オンラインでフォームをするをむ <https://riptutorial.com/ja/winforms/topic/8768/フォームをする>

## 5: ヘルプの

さまざまなWindowsフォームアプリケーションのフォームとコントロールのヘルプをすることができます。ポップアップヘルプをしたり、CHMファイルまたはURLをくことができます。フォーム、コントロール、ダイアログにのヘルプをすることができます。

### HelpProviderコンポーネント

`HelpProvider`コンポーネントをセットアップして、コンポーネントにするヘルプをすることができます。F1キーまたはフォームのヘルプボタンをすと、のようになにされます。

- コントロールのコンテキストヘルプのポップアップをする
- コンテキストに基づいたCHMファイルをくをする、キーワードまたはインデックスをする、トピックをする
- デフォルトブラウザをしてURLにする

### ヘルプクラス

コードで`Help`クラスをして、のような`Help`をできます。

- コントロールのヘルプポップアップをする
- コンテキストに基づいたCHMファイルをくをする、キーワードまたはインデックスをする、トピックをする
- デフォルトブラウザをしてURLにする

### HelpRequested イベント

ユーザーがF1キーをすか`Form`ヘルプボタンをクリックすると、`Control`オブジェクトまたは`Form``HelpRequested`イベントをしてカスタムアクションをできます。

### フォームのヘルプボタン

タイトルバーに[ヘルプ]ボタンをするように`Form`をできます。このようにして、ユーザーがヘルプボタンをクリックすると、カーソルは?わり?のをクリックすると、`HelpProvider`をしてコントロールにけられたヘルプがされます。

### MessageBoxとCommonDialogsのヘルプボタン

コンポーネントのヘルプボタンをして、`MessageBox`、`OpenFileDialog`、`SaveDialog`、および`ColorDialog`ヘルプをすることができます。

## ツールチップコンポーネント

ユーザーがコントロールをポイントすると、 `ToolTip`コンポーネントをしてヘルプテキストをでき `ToolTip`。 `ToolTip`は、のコントロールにけることができます。

`HelpProvider`と`Help`クラスのコンパイルされたヘルプファイル`.chm`またはHTMLファイルをHTMLヘルプでできます。コンパイルされたヘルプファイルは、、、およびページのキーワードリンクをします。ショートカットは、コンパイルされたヘルプファイルでのみします。HTMLヘルプワークショップをすると、HTMLヘルプ`1.x`ファイルができます。HTMLヘルプのについては、[Microsoft HTMLヘルプの「HTMLヘルプワークショップ」](#)およびそのHTMLヘルプトピックをしてください。

## Examples

### ヘルプファイルをする

`Help Class`は、HTML Help 1.0エンジンをカプセルします。ヘルプオブジェクトをして、コンパイルみヘルプファイル`.chm`またはHTMLファイルをHTMLヘルプですることができます。コンパイルされたヘルプファイルは、ページの、、、およびキーワードのリンクをします。ショートカットは、コンパイルされたヘルプファイルでのみします。HTML Help WorkshopとばれるMicrosoftのツールをして、HTMLヘルプ`1.x`ファイルをすることができます。

コンパイルされたヘルプファイルを2のウィンドウにするな

## C

```
Help.ShowHelp(this, helpProviderMain.HelpNamespace);
```

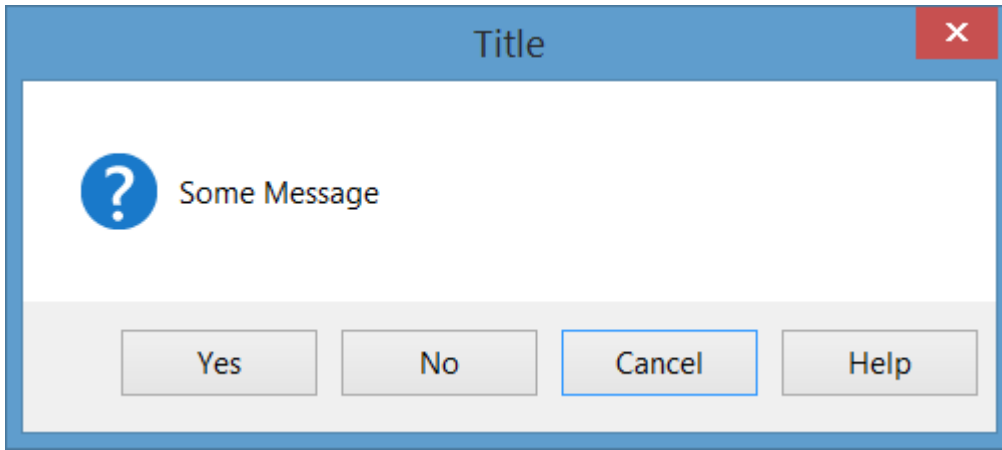
## VB.NET

```
Help.ShowHelp(Me, hlpProviderMain.HelpNamespace)
```

### MessageBoxのヘルプをする

メッセージボックスのヘルプは、さまざまなでできます。 `Help`ボタンをするように`MessageBox`をできます。また、ユーザーがヘルプボタンをクリックするか、`F1`キーをしてヘルプをしたときに、`CHM`ファイルをしたり、`URL`にしたり、カスタムアクションをしたりできるように`MessageBox`をできます。このトピックのいくつかのをします。

のすべてのでは、`MessageBox`はのようになります。



**CHM** ファイルをし、キーワードにします。

```
MessageBox.Show("Some Message", "Title", MessageBoxButtons.YesNoCancel,  
    MessageBoxIcon.Question, MessageBoxDefaultButton.Button3, 0,  
    "help.chm", HelpNavigator.KeywordIndex, "SomeKeyword");
```

**CHM** ファイルをしてトピックにする

```
MessageBox.Show("Some Message", "Title", MessageBoxButtons.YesNoCancel,  
    MessageBoxIcon.Question, MessageBoxDefaultButton.Button3, 0,  
    "help.chm", HelpNavigator.Topic, "/SomePath/SomePage.html");
```

**CHM** ファイルをし、ののヘルプページをナビゲートする

```
MessageBox.Show("Some Message", "Title", MessageBoxButtons.YesNoCancel,  
    MessageBoxIcon.Question, MessageBoxDefaultButton.Button3, 0,  
    "help.chm");
```

デフォルトブラウザをき、URLにナビゲートする

```
MessageBox.Show("Some Message", "Title", MessageBoxButtons.YesNoCancel,  
    MessageBoxIcon.Question, MessageBoxDefaultButton.Button3, 0,  
    "http://example.com");
```

ヘルプボタンまたは**F1**キーをしたときにカスタムアクションをする

この、`MessageBox`のparentの`HelpRequested`イベントをし、カスタムをするがあります。

```
private void Form1_HelpRequested(object sender, HelpEventArgs hlpevent)  
{  
    // Perform custom action, for example show a custom help form
```

```
var f = new Form();
f.ShowDialog();
}
```

に、 `MessageBox with Help` ボタンをすることができます。

```
MessageBox.Show("Some Message", "Title", MessageBoxButtons.YesNoCancel,
    MessageBoxIcon.Question, MessageBoxDefaultButton.Button3, 0, true);
```

または、ヘルプボタンをしないでください

```
MessageBox.Show("Some Message", "Title", MessageBoxButtons.YesNoCancel,
    MessageBoxIcon.Question, MessageBoxDefaultButton.Button3, 0, false);
```

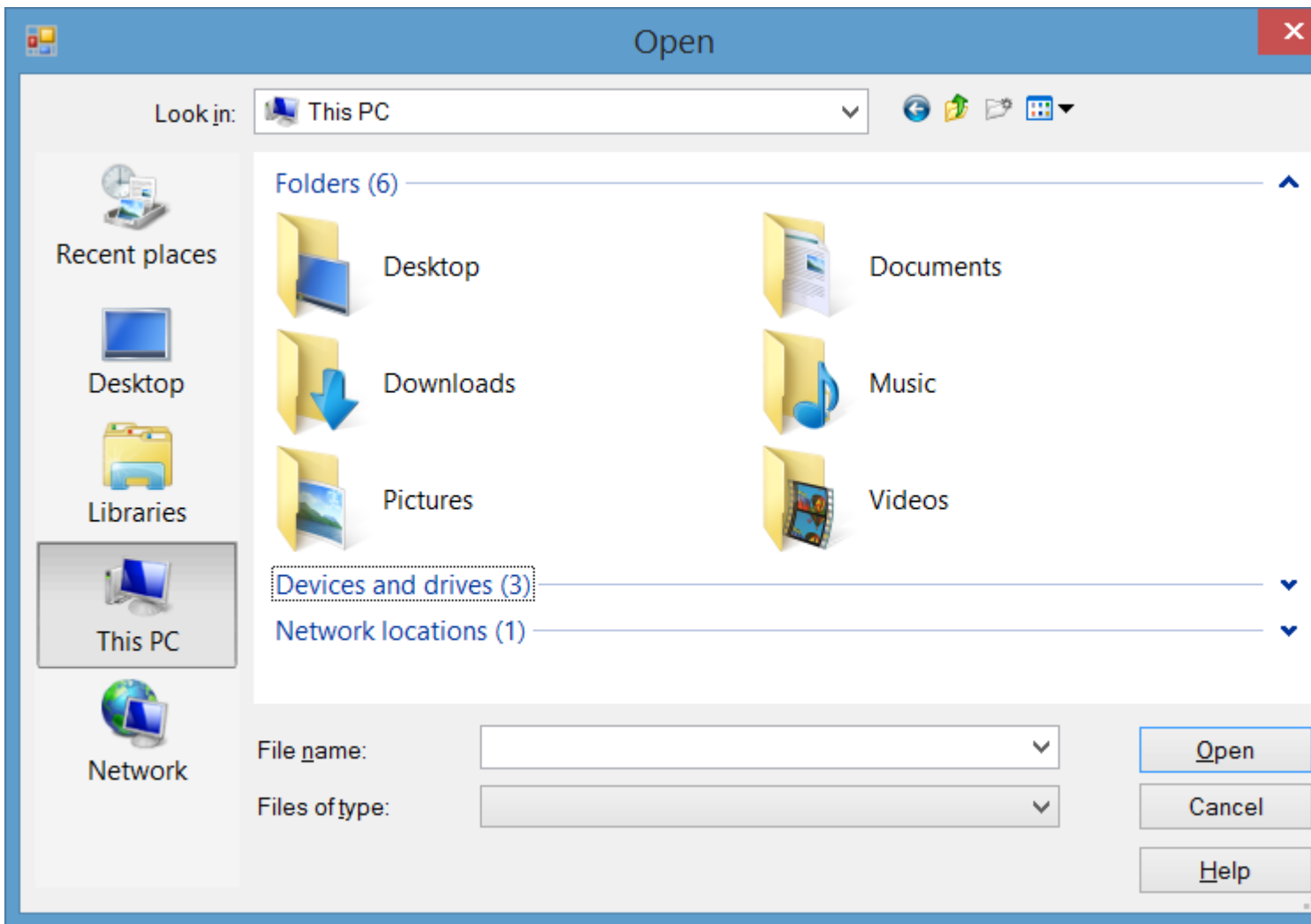
## CommonDialogs のヘルプをする

`OpenFileDialog`、`SaveFileDialog`、`ColorDialog` ヘルプをすることができます。これをうには、ダイアログの `ShowHelp` プロパティを `true` し、ダイアログの `HelpRequest` イベントをし `true`。

```
void openFileDialog1_HelpRequest(object sender, EventArgs e)
{
    //Perform custom action
    Help.ShowHelp(this, "Http://example.com");
}
```

- `ShowHelp` を `true` したにのみイベントが `ShowHelp` し `true`。
- このイベントは、[ `Help` ] ボタンをクリックするだけでし、F1 キーではしません。

のには、[ヘルプ] ボタンがある `OpenFileDialog` があります。



## コントロールとフォームの **HelpRequested** イベントの

ユーザーがコントロールで **F1** キーをすか、フォーム `□` のヘルプボタンをクリックしてコントロールをクリックすると、 **HelpRequested** イベントがします。

このイベントをして、ユーザーがコントロールまたはフォームのヘルプをしたときにカスタムアクションをできます。

**HelpRequested** は、バブルアップメカニズムをサポートしています。あなたのアクティブコントロールにしてします。イベントをせず、そのイベント `arg` の **Handled** プロパティを `true` にしないと、コントロールまでフォームがされます。

たとえば、のようなフォームの **HelpRequested** イベントを **HelpRequested**、 **F1** キーをすとアクティブなコントロールのがメッセージボックスにされますが、 `textBox1` はのメッセージがされます。

```
private void Form1_HelpRequested(object sender, HelpEventArgs hlpevent)
{
    var c = this.ActiveControl;
    if (c != null)
        MessageBox.Show(c.Name);
}
private void textBox1_HelpRequested(object sender, HelpEventArgs hlpevent)
```

```
{
    hlpevent.Handled = true;
    MessageBox.Show("Help request handled and will not bubble up");
}
```

URLへのナビゲートや`Help`クラスをしたCHMファイルのなど、のカスタムアクションをできます。

### ヘルプクラスをしてヘルプをする

コードで`Help`クラスをして、のような`Help`をできます。

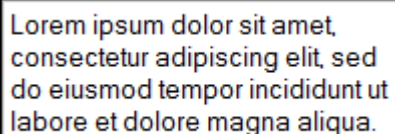
- コントロールのヘルプポップアップをする
- コンテキストに基づいたCHMファイルをくをする、キーワードまたはインデックスをする、トピックをする
- デフォルトブラウザをしてURLにする

### ヘルプをするポップアップウィンドウ

`Help.ShowPopup`をすると、ヘルプポップアップウィンドウをできます。

```
private void control_MouseClick(object sender, MouseEventArgs e)
{
    var c = (Control)sender;
    var help = "Lorem ipsum dolor sit amet, consectetur adipiscing elit, " +
               "sed do eiusmod tempor incididunt ut labore et dolore magna aliqua."
    if (c != null)
        Help.ShowPopup(c, "Lorem ipsum dolor sit amet.", c.PointToScreen(e.Location));
}
```

マウスポインタのにヘルプのポップアップがされます



### CHMヘルプファイルをする

`Help.ShowHelp`メソッドのさまざまなオーバーロードをしてCHMファイルをし、キーワード、トピック、インデックスまたはコンテンツテーブルにできます。

### ヘルプのをする

```
Help.ShowHelp(this, "Help.chm");
```

### のキーワードのヘルプをする

```
Help.ShowHelp(this, "Help.chm", HelpNavigator.Index, "SomeKeyword");
```

のトピックのヘルプをする

```
Help.ShowHelp(this, "Help.chm", HelpNavigator.Topic, "/SomePath/SomePage.html");
```

## URLを

ShowHelpメソッドをして、デフォルトのブラウザにのURLをできます。

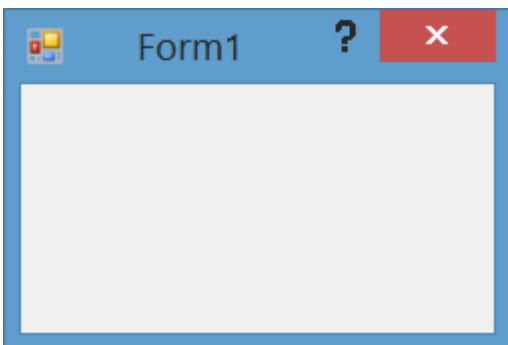
```
Help.ShowHelp(this, "Http://example.com");
```

フォームのタイトルバーにヘルプボタンをする

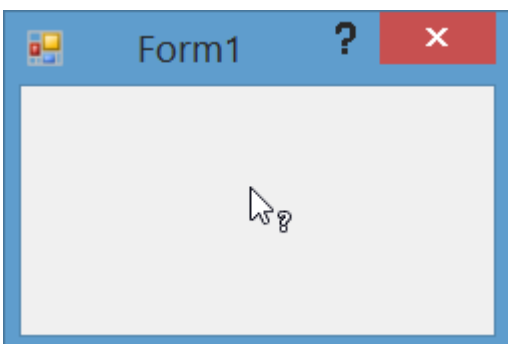
FormタイトルバーにヘルプボタンをすることができForm。これをうには、をうがあります。

1. フォームのHelpButtonプロパティをtrueしtrue。
2. MinimizeBoxとMaximizeBoxをfalseしfalse。

FormタイトルバーにヘルプボタンがされForm



また、ヘルプボタンをクリックすると、カーソルが?わり?カーソル



あなたはをクリックした、そのControlやForm、 HelpRequestedイベントがします、あなたがしているもHelpProvider、コントロールのヘルプをしてされます HelpProvider。

のフォームヘルプボタンのようにするカスタムヘルプボタンをする



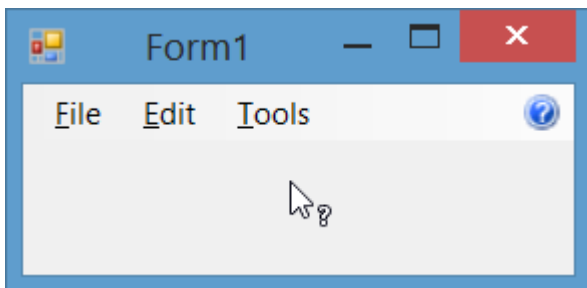
あなたがっているはFormでMinimizeBoxとMaximizeBoxにしtrue、あなたはのタイトルバーの[ヘルプ]ボタンをすることはできませんFormとカーソルがにコントロールをクリックできるようにするために、それをするために、ヘルプボタンをクリックしのをうことになりませヘルプをします。

MenuStripメニューをのヘルプボタンのようにさせることができます。これをうには、フォームにMenuStripをし、ToolStripMenuItemをして、アイテムのClickイベントをClickます。

```
private const int WM_SYSCOMMAND = 0x0112;
private const int SC_CONTEXTHELP = 0xF180;
[System.Runtime.InteropServices.DllImport("user32.dll")]
static extern IntPtr SendMessage(IntPtr hWnd, int Msg, int wParam, int lParam);
private void helpToolStripMenuItem_Click(object sender, EventArgs e)
{
    SendMessage(this.Handle, WM_SYSCOMMAND, SC_CONTEXTHELP, 0);
}
```

Buttonをしてするは、button1.Capture = false;をするもありbutton1.Capture = false;メッセージをするにただし、ToolStripMenuItemではありません。

その、ヘルプメニューをクリックすると、カーソルは?にわり?ヘルプボタンをクリックしたときとじようにします



フォームのHelpButtonClicked イベントの

ユーザーがクリックしたときには、することができますHelpButtonりうことにより、フォームのタイトルバーのHelpButtonClicked。イベントargsのCancelプロパティをtrueすることにより、イベントのまたはキャンセルをすることができtrue。

```
private void Form1_HelpButtonClicked(object sender, CancelEventArgs e)
{
    e.Cancel = true;
    //Perform some custom action
    MessageBox.Show("Some Custom Help");
}
```

オンラインでヘルプのをむ <https://riptutorial.com/ja/winforms/topic/3285/ヘルプの>

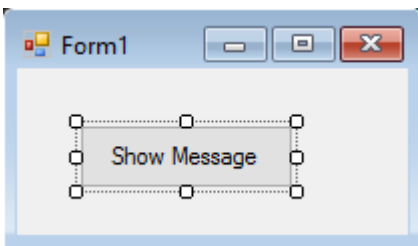
## 6: なコントロール

### Examples

#### ボタン

ボタンはもなコントロールの1つであり、にユーザーがむときにコードをするためにされます。

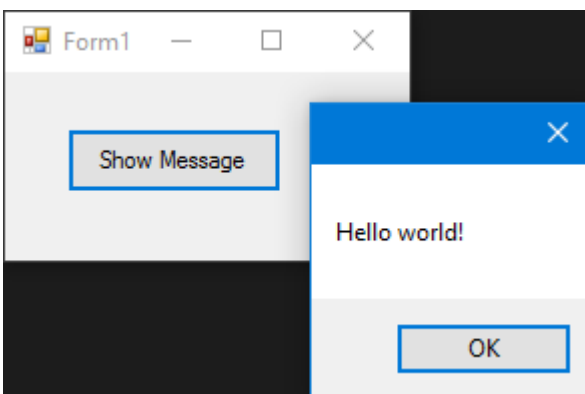
ここでは、ボタンをクリックしたときにメッセージボックスをする、とてもシンプルなケースがあります。フォームにボタンをし、コードで `cmdShowMessage` というをけてオブジェクトをし、ボタンのテキストを「メッセージを」にします。



ビジュアルデザイナーのボタンをダブルクリックするだけで、Visual Studioはclickイベントのコードをします。ここでMessageBoxのコードをすだけです。

```
private void cmdShowMessage_Click(object sender, EventArgs e)
{
    MessageBox.Show("Hello world!");
}
```

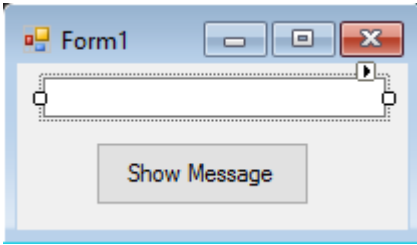
すぐプログラムをしてボタンをクリックすると、のメッセージがされます。



#### テキストボックス

テキストボックスをすと、ユーザーはプログラムにデータをできます。

フォームをし、テキストボックスをして、メッセージボックスにユーザーがむメッセージをするようにします。たちのフォームはのようになります

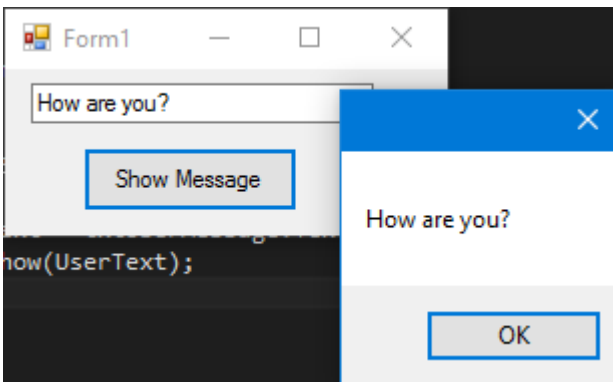


ボタンのクリックイベントをして、テキストボックスのテキストをします。

```
private void cmdShowMessage_Click(object sender, EventArgs e)
{
    string UserText = txtUserMessage.Text;
    MessageBox.Show(UserText);
}
```

このとおり、Textboxの.Text プロパティをしています。これは、textboxにまれるテキストです。

プログラムをすると、テキストボックスにきむことができます。ボタンをクリックすると、たちがいたテキストがMessageBoxにされます

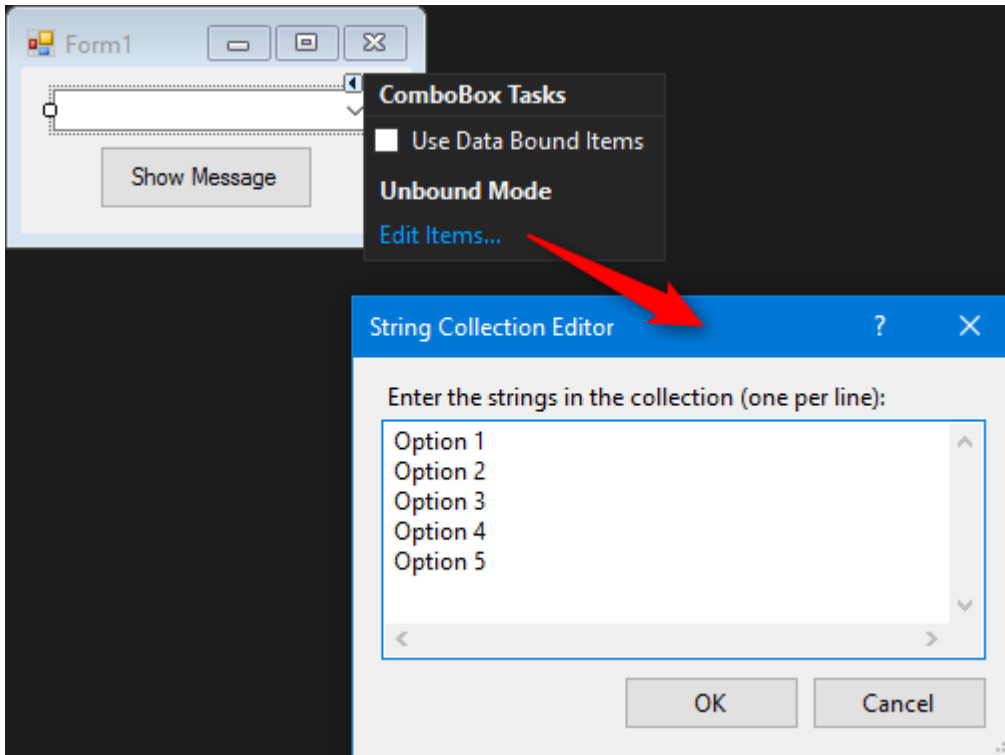


## コンボボックス

ComboBoxをすると、がするさまざまなオプションの1つをできます。

フォームをしてコンボボックスをして、メッセージボックスにたちがするリストからユーザーがむメッセージをするようにします。

フォームにコンボをした、コンボにオプションのリストをします。これをうには、Items プロパティをするがあります

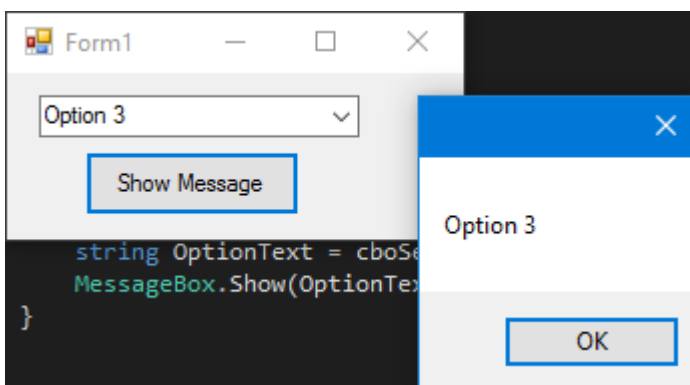


これでclickイベントのコードをするがあります

```
private void cmdShowMessage_Click(object sender, EventArgs e)
{
    string OptionText = cboSelectOption.SelectedItem.ToString();
    MessageBox.Show(OptionText);
}
```

このとおり、`SelectedItem`プロパティをすると、したオプションのオブジェクトがまれます。するがであり、コンパイラはオブジェクトがかどうかわからないので、`ToString()`メソッドをするがあります。

プログラムをすると、きなオプションをできるようにになります。ボタンをクリックすると、メッセージボックスにそのオプションがされま



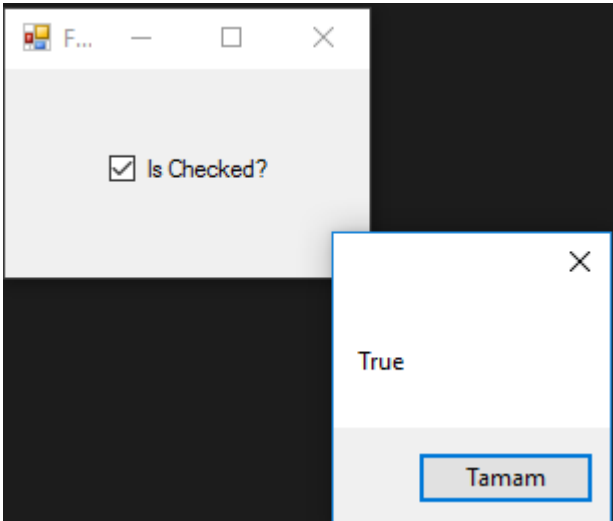
ユーザーがコンボボックスからアイテムをしたときにをけるには、`SelectionChangeCommitted`イベントをします。`SelectedIndexChanged`イベントをすることもできますが、これはコンボボックスのをプログラムでするときにもします。

## CheckBox

チェックボックスは、ユーザーが「あなたはですか」というようなしに、ユーザーから `boolean` をできるようにするコントロールです。

`check` プロパティがされるたびに `CheckedChanged` というイベントがあります。

CheckBoxには「Is Checked」というがあります。



この `MessageBox` を `CheckedChanged` イベントから `CheckedChanged`、

```
private void checkBox1_CheckedChanged(object sender, EventArgs e)
{
    bool IsChecked = checkBox1.Checked;
    MessageBox.Show(IsChecked.ToString());
}
```

チェックボックスがチェックされている -> `IsChecked` になります `true`。

**CheckBox** がチェックされていない、 -> `IsChecked` は `false` になり `false`。

## リストボックス

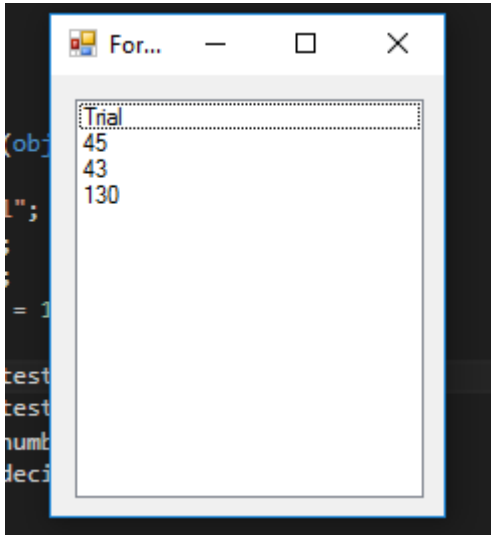
`Listbox` は、オブジェクトのコレクションをむことができるコントロールです。 `Listbox` は `Listbox` にて `Combobox` が、 `Combobox` ます。したのみがユーザーにされます。 `Listbox` すべてのがユーザーにされます。

**Listbox** にアイテムをするには

```
private void Form3_Load(object sender, EventArgs e)
{
    string test = "Trial";
    string test2 = "45";
    int numberTest = 43;
    decimal decimalTest = 130;
```

```
listBox1.Items.Add(test);  
listBox1.Items.Add(test2);  
listBox1.Items.Add(numberTest);  
listBox1.Items.Add(decimalTest);  
}
```

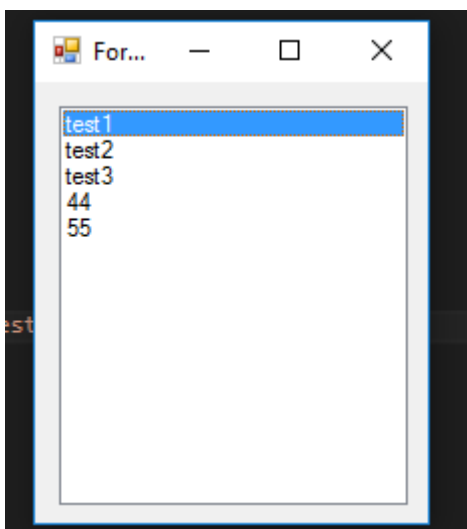
;



また、datasources をえることもできdatasources が、

```
private void Form3_Load(object sender, EventArgs e)  
{  
    List<string> TestList = new List<string> { "test1", "test2", "test3", "44", "55"  
};  
    listBox1.DataSource = TestList;  
}
```

;

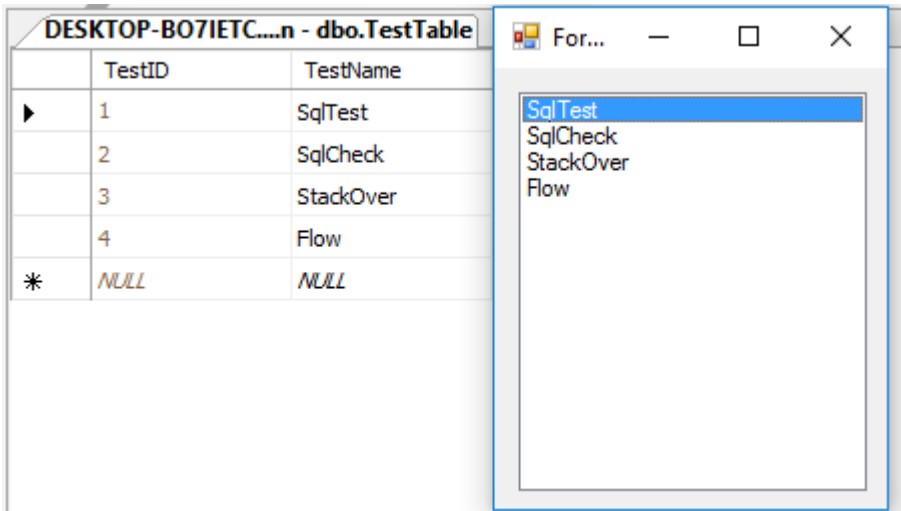


```
private void Form3_Load(object sender, EventArgs e)  
{  
    SqlConnection Connection = new  
    SqlConnection("Server=serverName;Database=db;Trusted_Connection=True;"); //Connetion to MS-
```

SQL (RDBMS)

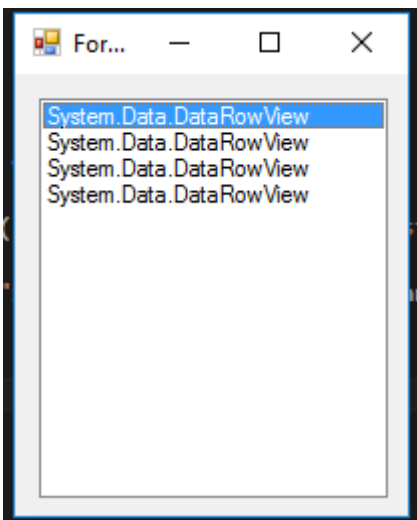
```
Connection.Open(); //Connection open
SqlDataAdapter Adapter = new SqlDataAdapter("Select * From TestTable",
Connection); // Get all records from TestTable.
DataTable DT = new DataTable();
Adapter.Fill(DT); // Fill records to DataTable.
listBox1.DataSource = DT; // DataTable is the datasource.
listBox1.ValueMember = "TestID";
listBox1.DisplayMember= "TestName";
}
```

な。



SQLデータソースをリストボックスにえるには、ValueMemberとDisplayMemberがです。

そうでないには、このようになります、



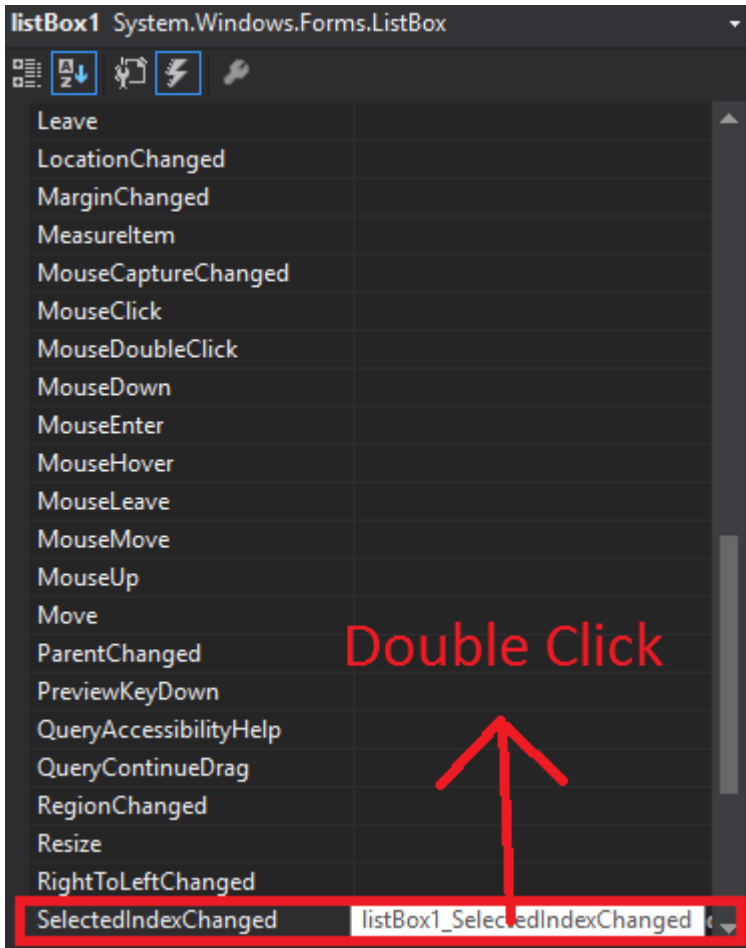
につイベント。

**SelectedIndex\_Changed;**

データソースをえるためのリストをする

```
private void Form3_Load(object sender, EventArgs e)
{
    List<string> DataList = new List<string> {"test1" , "test2" , "test3" , "44" ,
"45" };
    listBox1.DataSource = TestList;
}
```

フォームのデザインで `listBox` をし、`F4`キーをすか、ライトニングアイコンをクリックします。

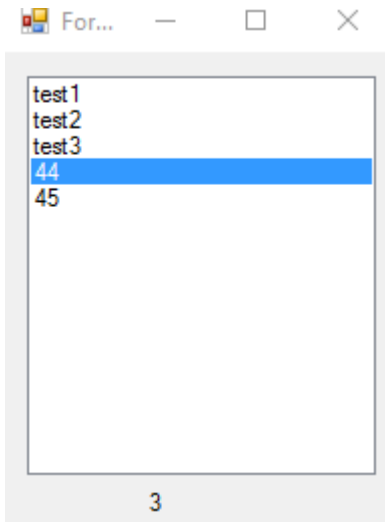


Visual Studioは、コードビハインドに `listBox1_SelectedIndexChanged` をします。

```
private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    int Index = listBox1.SelectedIndex;
    label1.Text = Index.ToString();
}
```

`SelectedIndex_Changed` です。 のラベルには、したのインデックスがされます

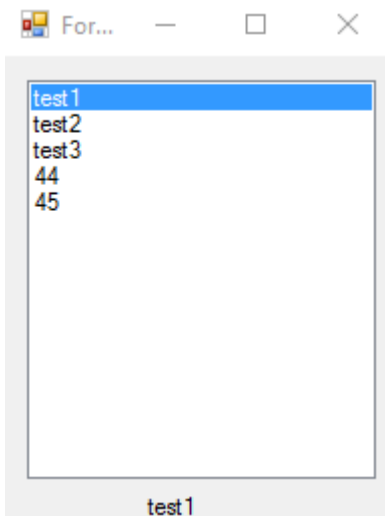




**SelectedValue\_Changed;** データソースはとじて、SelectedIndex\_Changedのようにこのイベントをできます

```
private void listBox1_SelectedValueChanged(object sender, EventArgs e)
{
    label1.Text = listBox1.SelectedValue.ToString();
}
```

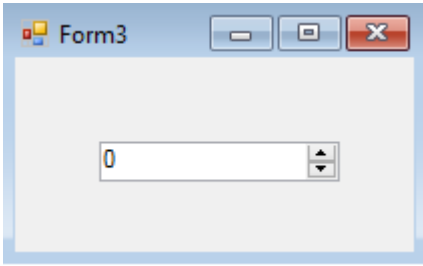
;



## NumericUpDown

NumericUpDownは、TextBoxのようにえるコントロールです。このコントロールでは、からを/することができます。とはテキストボックスのをしています。

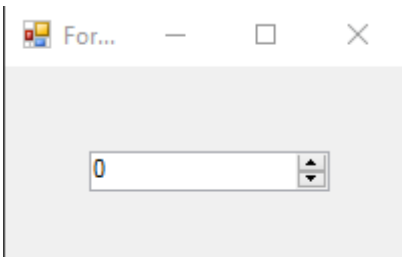
コントロールはのように入ります。



Form\_Loadですることができます。

```
private void Form3_Load(object sender, EventArgs e)
{
    numericUpDown1.Maximum = 10;
    numericUpDown1.Minimum = -10;
}
```

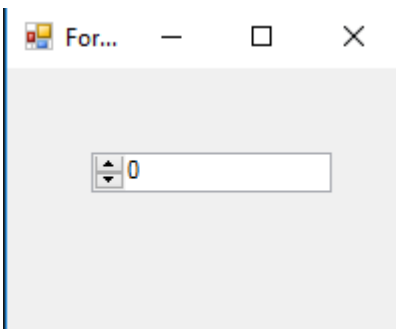
;



UpDownAlignはのをします。

```
private void Form3_Load(object sender, EventArgs e)
{
    numericUpDown1.UpDownAlign = LeftRightAlignment.Left;
}
```

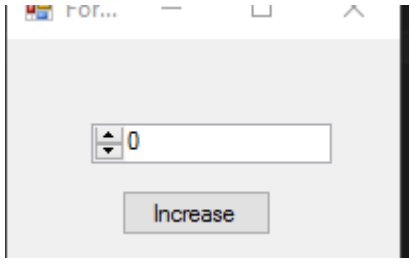
;



UpButton()メソッドは、コントロールのをやします。どこからでもびすことができます。はそれをびすためにbuttonをいました。

```
private void button1_Click(object sender, EventArgs e)
{
    numericUpDown1.UpButton();
}
```

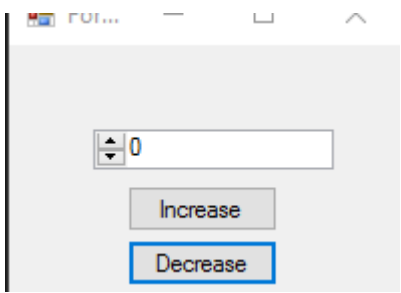
\*\*



DownButton() メソッドは、コントロールのをらします。 どこからでもびすことができます。 はも  
うびすためにbuttonをいました。

```
private void button2_Click(object sender, EventArgs e)
{
    numericUpDown1.DownButton();
}
```

;



## につイベント

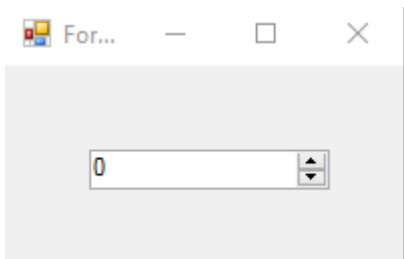
### ValueChanged;

このイベントは、またはがクリックされたときにします。

```
private void numericUpDown1_ValueChanged(object sender, EventArgs e)
{
    decimal result = numericUpDown1.Value; // it will get the current value
    if (result == numericUpDown1.Maximum) // if value equals Maximum value that we set
in Form_Load.
    {
        label1.Text = result.ToString() + " MAX!"; // it will add "MAX" at the end of
the label
    }
    else if (result == numericUpDown1.Minimum) // if value equals Minimum value that
we set in Form_Load.
    {
        label1.Text = result.ToString() + " MIN!"; // it will add "MIN" at the end of
the label
    }
    else
    {
        label1.Text = result.ToString(); // If Not Max or Min, it will show only the
```

```
number.  
    }  
}
```

;



オンラインでなコントロールをむ <https://riptutorial.com/ja/winforms/topic/5816/なコントロール>

## 7: コントロール

コントロールはのクラスとまったく同じです。しなければならないのは、イベントをオーバーライドすることです。は、のイベントハンドラをびしたに、ずベースイベントハンドラをびすことをおめします。のがあるは、イベントをびします。

### Examples

アプリケーションの

ほとんどのサイトをすばやくんでみると、WinFormsがWPFよりも持っているとわかります。もにけられるの1つは、アプリケーションの「ルック・アンド・フィール」にするアプリケーションのをうことがしいとえられることです。

には、コントロールのをけ、のコントロールをさせるだけであれば、とのでになWinFormsでアプリケーションをすることはくほどです。

TextBoxをにる。らかのでTextBoxをすることをしないWindowsアプリケーションをするのはしいです。したがって、のTextBoxをつことはににかなっています。のをえてみましょう。

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

namespace StackOverflowDocumentation
{
    public class SOTextBox : TextBox
    {
        public SOTextBox() : base()
        {
            base.BackColor = SOUserPreferences.BackColor;
            base.ForeColor = SOUserPreferences.ForeColor;
        }
        protected override void OnEnter(EventArgs e)
        {
            base.BackColor = SOUserPreferences.FocusColor;
            base.OnEnter(e);
        }
        protected override void OnLeave(EventArgs e)
        {
            base.BackColor = SOUserPreferences.BackColor;
            base.OnLeave(e);
        }
    }
}
```

ユーザーがくのボックスをつデータフォームでもつもの1つは、フォーカスがされたボックスのをつことです。にると、のするカーソルよりもやすい。のコードは、それをにうTextBoxをしま

す。

このプロセスでは、クラスのプロパティをします。はからをにえる

```
using System;
using System.Threading;
using Microsoft.Win32;
using System.Globalization;
using System.Data;
using System.Drawing;

namespace StackOverflowDocumentation
{
    public class SOUserPreferences
    {
        private static string language;
        private static string logPath;
        private static int formBackCol;
        private static int formForeCol;
        private static int backCol;
        private static int foreCol;
        private static int focusCol;

        static SOUserPreferences()
        {
            try
            {
                RegistryKey HKCU = Registry.CurrentUser;
                RegistryKey kSOPrefs = HKCU.OpenSubKey("SOPrefs");
                if (kSOPrefs != null)
                {
                    language = kSOPrefs.GetValue("Language", "EN").ToString();
                    logPath = kSOPrefs.GetValue("LogPath", "c:\\windows\\logs\\").ToString();
                    formForeCol = int.Parse(kSOPrefs.GetValue("FormForeColor", "-2147483630").ToString());
                    formBackCol = int.Parse(kSOPrefs.GetValue("FormBackColor", "-2147483633").ToString());
                    foreCol = int.Parse(kSOPrefs.GetValue("ForeColor", "-2147483640").ToString());
                    backCol = int.Parse(kSOPrefs.GetValue("BackColor", "-2147483643").ToString());
                    focusCol = int.Parse(kSOPrefs.GetValue("FocusColor", "-2147483643").ToString());
                }
                else
                {
                    language = "EN";
                    logPath = "c:\\windows\\logs\\";
                    formForeCol = -2147483630;
                    formBackCol = -2147483633;
                    foreCol = -2147483640;
                    backCol = -2147483643;
                    focusCol = -2147483643;
                }
            }
            catch (Exception ex)
            {
                //handle exception here;
            }
        }
    }
}
```

```

public static string Language
{
    get
    {
        return language;
    }
    set
    {
        language = value;
    }
}

public static string LogPath
{
    get
    {
        return logPath;
    }
    set
    {
        logPath = value;
    }
}

public static Color FormBackColor
{
    get
    {
        return ColorTranslator.FromOle(formBackCol);
    }
    set
    {
        formBackCol = ColorTranslator.ToOle(value);
    }
}

public static Color FormForeColor
{
    get
    {
        return ColorTranslator.FromOle(formForeCol);
    }
    set
    {
        formForeCol = ColorTranslator.ToOle(value);
    }
}

public static Color BackColor
{
    get
    {
        return ColorTranslator.FromOle(backCol);
    }
    set
    {
        backCol = ColorTranslator.ToOle(value);
    }
}

```

```

public static Color ForeColor
{
    get
    {
        return ColorTranslator.FromOle(foreCol);
    }
    set
    {
        foreCol = ColorTranslator.ToOle(value);
    }
}

public static Color FocusColor
{
    get
    {
        return ColorTranslator.FromOle(focusCol);
    }
    set
    {
        focusCol = ColorTranslator.ToOle(value);
    }
}
}
}

```

このクラスは、Windowsレジストリをしてプロパティをしますが、にじてデータベースまたはファイルができます。このようにクラスをするは、アプリケーションのをだけでなく、にユーザーができることです。はいつものアプリケーションにフォームをみんで、ユーザーがみのをできるようにします。saveはレジストリまたはデータベースなどにするだけでなく、にクラスのプロパティをします。クラスのプロパティはではないことにしてください。このでは、アプリケーションのとなすことができます。つまり、されたにいていたフォームは、されたによってすぐにをけます。

あなたは、じでであることをむのアプリケーションのいプロパティをにえることができます。フォントはのにいです。

## NumberBox

くの、だけをするボックスがになります。やはりコントロールからきすことによって、これはにされます。

```

using System;
using System.Windows.Forms;
using System.Globalization;

namespace StackOverflowDocumentation
{
    public class SONumberBox : SOTextBox
    {
        private int decPlaces;
        private int extraDecPlaces;
        private bool perCent;
        private bool useThouSep = true;
        private string decSep = ".";
    }
}

```



```

private string thouSep = ",";
private double numVal;

public SONumberBox() : base()

{
}

public bool PerCent
{
    get
    {
        return perCent;
    }
    set
    {
        perCent = value;
    }
}

public double Value
{
    get
    {
        return numVal;
    }
    set
    {
        numVal = value;
        if (perCent)
        {
            double test = numVal * 100.0;
            this.Text = FormatNumber(test) + "%";
        }
        else
        {
            this.Text = FormatNumber(value);
        }
    }
}

public bool UseThousandSeparator
{
    get
    {
        return useThouSep;
    }
    set
    {
        useThouSep = value;
    }
}

public int DecimalPlaces
{
    get
    {
        return decPlaces;
    }
    set
    {
        decPlaces = value;
    }
}

```

```

}
public int ExtraDecimalPlaces
{
    get
    {
        return extraDecPlaces;
    }
    set
    {
        extraDecPlaces = value;
    }
}
protected override void OnTextChanged(EventArgs e)
{
    string newVal = this.Text;
    int len = newVal.Length;
    if (len == 0)
    {
        return;
    }
    bool neg = false;
    if (len > 1)
    {
        if (newVal.Substring(0, 1) == "-")
        {
            newVal = newVal.Substring(1, len - 1);
            len = newVal.Length;
            neg = true;
        }
    }
    double val = 1.0;
    string endChar = newVal.Substring(newVal.Length - 1);
    switch (endChar)
    {
        case "M":
        case "m":
            if (len > 1)
            {
                val = double.Parse(newVal.Substring(0, len - 1)) * 1000000.0;
            }
            else
            {
                val *= 1000000.0;
            }
            if (neg)
            {
                val = -val;
            }
            this.Text = FormatNumber(val);
            break;
        case "T":
        case "t":
            if (len > 1)
            {
                val = double.Parse(newVal.Substring(0, len - 1)) * 1000.0;
            }
            else
            {
                val *= 1000.0;
            }
            if (neg)

```

```

        {
            val = -val;
        }
        this.Text = FormatNumber(val);
        break;
    }

    base.OnTextChanged(e);
}
protected override void OnKeyPress(KeyPressEventArgs e)
{
    bool handled = false;
    switch (e.KeyChar)
    {
        case '-':
            if (this.Text.Length == 0)
            {
                break;
            }
            else if (this.SelectionStart == 0)
            {
                //negative being inserted first
                break;
            }
            else
            {
                handled = true;
                break;
            }
        case '1':
        case '2':
        case '3':
        case '4':
        case '5':
        case '6':
        case '7':
        case '8':
        case '9':
        case '0':
        case (char)Keys.Back:
            break;
        case 'M':
        case 'm':
        case 'T':
        case 't':
        case '%':
            //check last pos
            int l = this.Text.Length;
            int sT = this.SelectionStart;
            int sL = this.SelectionLength;
            if ((sT + sL) != l)
            {
                handled = true;
            }
            break;
        default:
            string thisChar = e.KeyChar.ToString();
            if (thisChar == decSep)
            {
                char[] txt = this.Text.ToCharArray();
                for (int i = 0; i < txt.Length; i++)

```

```

        {
            if (txt[i].ToString() == decSep)
            {
                handled = true;
                break;
            }
        }
        break;
    }
    else if (thisChar != thouSep)
    {
        handled = true;
    }
    break;
}

if (!handled)
{
    base.OnKeyPress(e);
}
else
{
    e.Handled = true;
}
}

protected override void OnLeave(EventArgs e)
{
    string tmp = this.Text;
    if (tmp == "")
    {
        tmp = "0";
        numVal = NumberLostFocus(ref tmp);
        this.Text = tmp;
    }
    if (tmp.Substring(tmp.Length - 1) == "%")
    {
        tmp = tmp.Substring(0, tmp.Length - 1);
        numVal = 0.0;
        numVal = NumberLostFocus(ref tmp) / 100.0;
        double test = numVal * 100.0;
        this.Text = FormatNumber(test) + "%";
    }
    else if (perCent)
    {
        numVal = NumberLostFocus(ref tmp);
        double test = numVal * 100.0;
        this.Text = FormatNumber(test) + "%";
    }
    else
    {
        numVal = NumberLostFocus(ref tmp);
        this.Text = tmp;
    }
    base.OnLeave(e);
}

private string FormatNumber(double amount)
{
    NumberFormatInfo nF = new NumberFormatInfo();
    nF.NumberDecimalSeparator = decSep;
    nF.NumberGroupSeparator = thouSep;
}

```

```

string decFormat;
if (useThouSep)
{
    decFormat = "#,##0";
}
else
{
    decFormat = "#0";
}
if (decPlaces > 0)
{
    decFormat += ".";
    for (int i = 0; i < decPlaces; i++)
    {
        decFormat += "0";
    }
    if (extraDecPlaces > 0)
    {
        for (int i = 0; i < extraDecPlaces; i++)
        {
            decFormat += "#";
        }
    }
}
else if (extraDecPlaces > 0)
{
    decFormat += ".";
    for (int i = 0; i < extraDecPlaces; i++)
    {
        decFormat += "#";
    }
}
return (amount.ToString(decFormat, nF));
}
private double NumberLostFocus(ref string amountBox)
{
    if (amountBox.Substring(0, 1) == decSep)
        amountBox = "0" + amountBox;
    NumberFormatInfo nF = new NumberFormatInfo();
    nF.NumberDecimalSeparator = decSep;
    nF.NumberGroupSeparator = thouSep;

    try
    {
        double d = 0.0;
        int l = amountBox.Length;
        if (l > 0)
        {
            char[] c = amountBox.ToCharArray();
            char endChar = c[l - 1];

            switch (endChar)
            {
                case '0':
                case '1':
                case '2':
                case '3':
                case '4':
                case '5':

```

```

        case '6':
        case '7':
        case '8':
        case '9':
            {
                stripNonNumerics(ref amountBox);
                d = Double.Parse(amountBox, nF);
                break;
            }
        case 'm':
        case 'M':
            {
                if (amountBox.Length == 1)
                    d = 1000000.0;
                else
                {
                    string s = amountBox.Substring(0, l - 1);
                    stripNonNumerics(ref s);
                    d = Double.Parse(s, nF) * 1000000.0;
                }
                break;
            }
        case 't':
        case 'T':
            {
                if (amountBox.Length == 1)
                    d = 1000.0;
                else
                {
                    string s = amountBox.Substring(0, l - 1);
                    stripNonNumerics(ref s);
                    d = Double.Parse(s, nF) * 1000.0;
                }
                break;
            }
        default:
            {
                //remove offending char
                string s = amountBox.Substring(0, l - 1);
                if (s.Length > 0)
                {
                    stripNonNumerics(ref s);
                    d = Double.Parse(s, nF);
                }
                else
                    d = 0.0;
                break;
            }
    }
    amountBox = FormatNumber(d);
    return (d);
}
catch (Exception e)
{
    //handle exception here;
    return 0.0;
}
}
private void stripNonNumerics(ref string amountBox)

```

```

{
    bool dSFound = false;
    char[] tmp = decSep.ToCharArray();
    char dS = tmp[0];
    string cleanNum = "";
    int l = amountBox.Length;
    if (l > 0)
    {
        char[] c = amountBox.ToCharArray();
        for (int i = 0; i < l; i++)
        {
            char b = c[i];
            switch (b)
            {
                case '0':
                case '1':
                case '2':
                case '3':
                case '4':
                case '5':
                case '6':
                case '7':
                case '8':
                case '9':
                    cleanNum += b;
                    break;
                case '-':
                    if (i == 0)
                        cleanNum += b;
                    break;
                default:
                    if ((b == dS) && (!dSFound))
                    {
                        dSFound = true;
                        cleanNum += b;
                    }
                    break;
            }
        }
        amountBox = cleanNum;
    }
}

```

をにするだけでなく、このクラスにはいくつかのなががあります。これは、の2のをすプロパティをし、オプションで1000のりをしてテキストのをし、のいをしします10Mが10,000,000.00へののでされますのはプロパティ。にするために、と1000のりはハードコードされています。システムでは、これらもユーザーのみです。

オンラインでコントロールをむ <https://riptutorial.com/ja/winforms/topic/6476/コントロール>

## クレジット

S. No		Contributors
1	winformsをいめる	<a href="#">4444</a> , <a href="#">Bjørn-Roger Kringsjå</a> , <a href="#">Chris Shao</a> , <a href="#">Cody Gray</a> , <a href="#">Community</a> , <a href="#">Reza Aghaei</a>
2	データバインディング	<a href="#">Kai Thoma</a>
3	テキストボックス	<a href="#">gplumb</a> , <a href="#">Jones Joseph</a> , <a href="#">Stefano d'Antonio</a>
4	フォームをする	<a href="#">Cody Gray</a> , <a href="#">Jeff Bridgman</a> , <a href="#">Steve</a>
5	ヘルプの	<a href="#">help-info.de</a> , <a href="#">Reza Aghaei</a>
6	なコントロール	<a href="#">Aimnox</a> , <a href="#">Berkay</a> , <a href="#">help-info.de</a> , <a href="#">Jeff Bridgman</a>
7	コントロール	<a href="#">Balagurunathan Marimuthu</a>