メッセージボックス

メッセージボックスを表示する

単純な情報や警告をユーザーに知らせる単純なダイアログウィンドウを表示するには
通常は、Windows がサポートするメッセージボックスを利用します

これは、すでに作成されているモーダルダイアログウィンドウです
モーダルダイアログボックスとは、表示されている間プログラムが進行しないウィンドウで
ユーザーが OK ボタンを押すなどしない限り
ダイアログ発生元のオーナーウィンドウをアクティブにすることはできません

メッセージボックスは **System.Windows.Forms.MessageBox** クラスを使います
このクラスはシステムに問い合わせる手段であり、インスタンスを生成することはできません

public class MessageBox

このクラスは、**MessageBox.Show()** 静的メソッドが定義されています
このメソッドこそ、システムに問い合わせてメッセージボックスを表示させるもので
このメソッド以外は、object クラスから特に変更はありません

```
public static DialogResult Show(string text);
public static DialogResult Show(IWin32Window owner , string text);
public static DialogResult Show(string text , string caption);

public static DialogResult Show(
 IWin32Window owner ,
 string text , string caption
);

public static DialogResult Show(
 string text , string caption ,
 MessageBoxButtons buttons
);

public static DialogResult Show(
 IWin32Window owner ,
 string text , string caption , MessageBoxButtons buttons
);

public static DialogResult Show(
 string text , string caption ,
 MessageBoxButtons buttons , MessageBoxIcon icon
);

public static DialogResult Show(
 IWin32Window owner ,
 string text , string caption ,
 MessageBoxButtons buttons , MessageBoxIcon icon
);

public static DialogResult Show(
 string text , string caption,
 MessageBoxButtons buttons , MessageBoxIcon icon ,
 MessageBoxDefaultButton defaultButton
);

public static DialogResult Show(
 IWin32Window owner , string text , string caption,
 MessageBoxButtons buttons , MessageBoxIcon icon ,
 MessageBoxDefaultButton defaultButton
);

public static DialogResult Show(
 string text , string caption ,
 MessageBoxButtons buttons , MessageBoxIcon icon ,
 MessageBoxDefaultButton defaultButton ,
 MessageBoxOptions options
);

public static DialogResult Show(
 IWin32Window owner , string text , string caption ,
 MessageBoxButtons buttons , MessageBoxIcon icon ,
 MessageBoxDefaultButton defaultButton ,
 MessageBoxOptions options
);
```

ずいぶんさまざまな機能があって、難しそうに感じますが
見慣れない型のほとんどは単純な列挙型なので問題はありません

text にはメッセージボックスでユーザーに伝える文字列を指定します
owner は、このメッセージボックスのオーナーオブジェクトを指定します
caption には、メッセージボックスのタイトルバーに表示する文字列を指定します

buttons はメッセージボックスに表示されるボタンを表す列挙型メンバを
icon にはメッセージボックスに表示されるアイコンを示す列挙型メンバを
defaultButton には、既定のボタンを示す列挙型メンバを
options には、表示などに関連したオプションを指定する列挙型メンバを指定します

このメソッドの戻り値は、ダイアログの戻り値を示す列挙型です
これは、ユーザーがダイアログに対してどのようなボタンを押したかを表しています
この列挙型について詳しくはすぐ後に説明します

たくさんあるので、少しずつ順を追って機能を見ていきましょう
まず、最も簡単なのは文字列だけを指定する方法です

```
using System.Windows.Forms;

class WinMain : Form {
 public static void Main(string[] args) {
  MessageBox.Show("Do you like a cat?" , "Kitty on your lap");
 }
}
```

このプログラムを実行すると、上の図のようなメッセージボックスを表示します
メッセージボックスが表示されている間、アプリケーションは処理を中断しています
OK ボタンを押せば、メッセージボックスは閉じられ次のステップに処理が移動するため
アプリケーションはそのままプログラムを終了します

owner パラメータは **IWin32Window** という型です
通常、Microsoft のネーミング規則で頭文字が大文字の I で、その後に名前が続くと
これはインターフェイスであるという意味で、すなわちこれはインターフェイス型です
この型は **System.Windows.Forms.IWin32Window** インターフェイスです

```
[ComVisible(true)]
[Guid("")]
[InterfaceType(ComInterfaceType.InterfaceIsIUnknown)]
public interface IWin32Window
```

このクラスは、コントロールの参照を数値で識別するための値を取得するための
IWin32Window.Handle プロパティを宣言しています

```
[ComVisible(true)]
[Guid("")]
[InterfaceType(ComInterfaceType.InterfaceIsIUnknown)]
IntPtr Handle {get;}
```

このインターフェイスを実装したクラスは、コントロールを識別するための
Windows システムが管理する数値を返す Handle プロパティを実装しなければなりません
実は、このインターフェイスは **Control クラスが実装**しています
オーナーウィンドウを指定したい場合は、Control クラスの継承クラスを指定するとよいでしょう

buttons パラメータを指定すれば、メッセージボックスに表示するボタンを指定できます
デフォルトでは OK ボタンしか表示されませんが、これを用いることで
YES/NO やキャンセルなどのボタンを表示することができるようになります
この指定には **System.Windows.Forms.MessageBoxButtons** 列挙型を使います

```
[Serializable]
public enum MessageBoxButtons
```

この列挙型は、次のようなメンバを定義しています

メンバ	解説
AbortRetryIgnore	[中止] ボタン、[再試行] ボタン、および [無視] ボタンを表示します
OK	[OK] ボタンを表示します
OKCancel	[OK] ボタンと [キャンセル] ボタンを表示します
RetryCancel	[再試行] ボタンと [キャンセル] ボタンを表示します
YesNo	[はい] ボタンと [いいえ] ボタンを表示します
YesNoCancel	[はい] ボタン、[いいえ] ボタン、および [キャンセル] ボタンを表示します

```
using System.Windows.Forms;

class WinMain : Form {
 public static void Main(string[] args) {
  MessageBox.Show("Do you know a Hizano Ueno Partner?" ,
   "Kitty on your lap" , MessageBoxButtons.YesNo);
 }
}
```

図を見てわかるように、このプログラムではメッセージボックスのボタンを指定しています
ところで、ボタンが複数個表示されている場合、どのボタンが押されたのかを知りたいですよね
そのために、このメソッドは DialogResult 型の戻り値を返します

これは **System.Windows.Forms.DialogResult** 列挙型です

```
[Serializable]
```

```
[ComVisible(true)]
public enum DialogResult
```

この列挙型は、次のようなメンバを持っています
それぞれのメンバは、どのボタンが押されたのかを示します

メンバ	解説
Abort	ダイアログ ボックスの戻り値は Abort です (通常は "中止"というラベルが指定されたボタンから送られます)
Cancel	ダイアログ ボックスの戻り値は Cancel です (通常は "キャンセル"というラベルが指定されたボタンから送られます)
Ignore	ダイアログ ボックスの戻り値は Ignore です (通常は "無視"というラベルが指定されたボタンから送られます)
No	ダイアログ ボックスの戻り値は No です (通常は "いいえ"というラベルが指定されたボタンから送られます)
None	ダイアログ ボックスからの戻り値はありません つまり、モーダル ダイアログ ボックスの実行が継続します
OK	ダイアログ ボックスの戻り値は OK です (通常は OK というラベルが指定されたボタンから送られます)
Retry	ダイアログ ボックスの戻り値は Retry です (通常は "再試行"というラベルが指定されたボタンから送られます)
Yes	ダイアログ ボックスの戻り値は Yes です (通常は "はい"というラベルが指定されたボタンから送られます)

これを利用すれば、ユーザーが押したボタンによって異なる処理を行うことができます

```
using System.Windows.Forms;

class WinMain : Form {
 public static void Main(string[] args) {
  switch (MessageBox.Show("Do you know a Hizano Ueno Partner?" ,
   "Kitty on your lap" , MessageBoxButtons.YesNo)) {
  case DialogResult.Yes:
   MessageBox.Show("It has been an interesting game.");
   break;
  case DialogResult.No:
   MessageBox.Show("This is a game that Nekomimi girls appear.");
   break;
  }
 }
}
```

このプログラムは、メッセージボックスの戻り値を監視することによって
「はい」ボタンと「いいえ」ボタンを押した時の処理を分岐させています

icon パラメータを指定すれば、さらにメッセージボックスにアイコンを付加することができます
注目してほしい警告などを表示する時は、特にアイコンをつけることが推奨されます
このアイコンは **System.Windows.Forms.MessageBoxIcon** 列挙型で表されます

```
[Serializable]
public enum MessageBoxIcon
```

この列挙型は、次のようなメンバを定義しています

Asterisk	円で囲まれた小文字の i から成る記号を表示します
Error	背景が赤の円で囲まれた白い X から成る記号を表示します
Exclamation	背景が黄色の三角形で囲まれた感嘆符から成る記号を表示します
Hand	背景が赤の円で囲まれた白い X から成る記号を表示します
Information	円で囲まれた小文字の i から成る記号を表示します
None	記号を表示しません
Question	円で囲まれた疑問符から成る記号を表示します
Stop	背景が赤の円で囲まれた白い X から成る記号を表示します
Warning	背景が黄色の三角形で囲まれた感嘆符から成る記号を表示します

オブジェクト指向の設計者ならば、なぜ Icon オブジェクトの静的メンバを持つ
メッセージボックス用のアイコンコレクションクラスを作らなかったのかと思うでしょう
これは、おそらく単純に Win32 API のMessageBox() 関数を呼び出しているからです
MessageBox() 関数は、アイコンなどの指定には全て論理的な意味を持つ数値で行います

```
using System.Windows.Forms;

class WinMain : Form {
 public static void Main(string[] args) {
  MessageBox.Show("Kitty on your lap!" , "I love cat" ,
   MessageBoxButtons.OK , MessageBoxIcon.Exclamation);
 }
}
```

このプログラムを実行すると、図のようにアイコン付きのメッセージボックスを表示します

defaultButton パラメータを指定すれば、既定ボタンを設定できます
既定ボタンとは、デフォルトで Enter キーを押して入力されるボタンです
フォーカスが当てられているこのボタンは、他のボタンよりもやや強調されているでしょう
通常は、最も左側のボタンが既定ボタンに設定されています

これを設定するには
System.Windows.Forms.MessageBoxDefaultButton 列挙型を使います

```
[Serializable]
public enum MessageBoxDefaultButton
```

この列挙型は、次のようなメンバを定義しています

メンバ	解説
Button1	1 番目のボタンが既定のボタンです
Button2	2 番目のボタンが既定のボタンです
Button3	3 番目のボタンが既定のボタンです

1番目というのは、通常左のボタンから数えます

```
using System.Windows.Forms;

class WinMain : Form {
 public static void Main(string[] args) {
  MessageBox.Show("Kitty on your lap!" , "I love cat" ,
   MessageBoxButtons.YesNo , MessageBoxIcon.Exclamation ,
   MessageBoxDefaultButton.Button2);
 }
}
```

このプログラムのメッセージボックスは、2番目のボタンを既定ボタンとしています

あまり使用することはないと思いますが、その他のオプションを指定することもできます
これには System.Windows.Forms.MessageBoxOptions 列挙型を使います

```
[Flags]
[Serializable]
public enum MessageBoxOptions
```

この列挙型は、次のようなメンバを定義しています

メンバ	解説
DefaultDesktopOnly	メッセージ ボックスをアクティブ デスクトップに表示します この定数は、システムが対話形式のウィンドウ ステーションの 既定のデスクトップだけにメッセージ ボックスを表示することを除いて ServiceNotification と同じです
RightAlign	メッセージ ボックスのテキストを右揃えで表示します
RtlReading	メッセージ ボックスのテキストを右から左へ読むように指定します
ServiceNotification	メッセージ ボックスをアクティブ デスクトップに表示します 呼び出し元は、ユーザーにイベントを通知するサービスです この関数は、コンピュータにログオンしているユーザーがいない場合でも 現在のアクティブ デスクトップにメッセージ ボックスを表示します

通常、これらのオプションは設定することはないと思われますが
例えば、アラビア語のような右から左へ読み進める言語を表示する場合などは使うことになるでしょう

前のページへ　戻る　次のページへ