ウィンドウスタイル

スタイルを設定・取得する

Win32 API で Windows プログラミングを経験したことがある方は
ウィンドウの作成時に、CreateWindow() メソッドでウィンドウスタイルを指定したでしょう
.NET の Control クラスもまた、スタイルを設定するメソッドを提供しています

スタイルの設定は **Control.SetStyle()** メソッドを使います
このメソッドは、次のように定義されています

protected void SetStyle(ControlStyles flag , bool value);

flag にはコントロールスタイルを示すビットフラグを指定します
指定したフラグを設定するならば value には true を、そうでなければ false を指定します

スタイルビットは **System.Windows.Forms.ControlStyles** 列挙型で表されます

```
[Flags]
[Serializable]
public enum ControlStyles
```

この列挙型は、次のような意味を持つメンバを持っています

メンバ	解説
AllPaintingInWmPaint	コントロールはウィンドウ メッセージ WM_ERASEBKGND を無視することによって、ちらつきを抑えます このスタイルは、UserPaint が true に設定されている場合だけ適用されます
CacheText	コントロールは、必要に応じて毎回 Handle から テキストのコピーを取得するのではなく、テキストのコピーを保持しています このスタイルは、既定では false に設定されます この動作によってパフォーマンスは向上しますが テキストを同期させておくことが難しくなります
ContainerControl	コントロールは、コンテナのような機能を果たします
DoubleBuffer	描画はバッファで実行され、完了後に、結果が画面に出力されます ダブル バッファリングは、コントロールの描画によるちらつきを防ぎます ダブル バッファリングを完全に有効にするには UserPaint ビットと AllPaintingInWmPaint ビットを true に設定する必要もあります
EnableNotifyMessage	true の場合、コントロールの WndProc に送信されたすべてのメッセージに対して OnNotifyMessage メソッドが呼び出されます このスタイルは、既定では false に設定されます
FixedHeight	コントロールの高さは固定されています
FixedWidth	コントロールの幅は固定されています
Opaque	コントロールが不透明に描画され、背景は描画されません
ResizeRedraw	コントロールのサイズが変更されると、そのコントロールが再描画されます
Selectable	コントロールはフォーカスを取得できます
StandardClick	コントロールは、標準の Click 動作を実装します
StandardDoubleClick	コントロールは、標準の DoubleClick 動作を実装します このスタイルは、StandardClick が設定されていない場合は無視されます
SupportsTransparentBackColor	コントロールはアルファ値が 255 未満の BackColor を受け入れ 透明度をシミュレートします この透明度は、UserPaint ビットが true に設定され 親コントロールが Control から派生している場合だけシミュレートされます
UserMouse	コントロールがマウスの操作を独自に処理し OSではマウス イベントが処理されません
UserPaint	コントロールは、OSによってではなく、独自に描画されます このスタイルは、Control から派生したクラスだけに適用されます

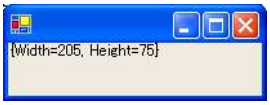
例えば、通常のアプリケーションはコントロールのサイズが変更されれば
新しいサイズで全体を描画しなおす必要があるため、サイズ変更と同時に再描画します
これを実現するには ResizeRedraw フラグを指定すればよいのです

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

class WinMain : Form {
 public static void Main(string[] args) {
  Application.Run(new WinMain());
 }

 public WinMain() {
  SetStyle(ControlStyles.ResizeRedraw , true);
 }

 override protected void OnPaint(PaintEventArgs e) {
  Graphics g = e.Graphics;
  g.DrawString(Size.ToString() , Font , Brushes.Black , 0 , 0);
 }
}
```



通常、デフォルトではウィンドウのサイズを変更しても全体は再描画されません
サイズを拡大したときのみ、拡大した分の無効領域が描画されるだけです
しかし、このプログラムでは ResizeRedraw を設定したため
ウィンドウのサイズを変更すると、クライアント領域全体が再描画されます

前のページへ　戻る　次のページへ