カーソル

カーソルの制御

カーソルは、マウスやタブレットなどのポインティングデバイスで操作する
GUI システムの要といえるユーザーインタフェイスのひとつです
このカーソルもまた、アイコン同様に特殊なビットマップのひとつと考えることができます

.NET では、カーソルは System.Windows.Forms.Cursor クラスであらわされます
このクラスは、カーソルオブジェクトだけではなく、静的にシステムにも関与します
最初に、Cursor クラスが持つ静的メンバについて詳しく見てみましょう

カーソルの制御は、主に OS が行っています
そのため、ユーザーが動かしているマウスカーソル自体の情報に直接アクセスすることはできませんが
Cursor クラスの静的メンバは、システムに近いカーソルの制御を行ってくれます

Corsor.Clip プロパティは、カーソルの動作範囲をクリップします
例えば、ゲームのような特殊なアプリケーションの場合
ゲームのプレイ中はマウスカーソルをウィンドウの外に移動できると不便な時があります
このような場合 Clip プロパティを設定することでカーソルの動きを制限できます

```
[Serializable]
public static Rectangle Clip {get; set;}
```

ただし、この設定が有効なのは、アプリケーションにフォーカスがある時だけです
それ以外の状態ではマウスキャプチャができないため、設定が開放されます

```
using System.Windows.Forms;

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

 override protected void OnMouseUp(MouseEventArgs e) {
  Cursor.Clip = Bounds;
 }
}
```

このプログラムを実行して、ウィンドウのクライアント領域をクリックすると
マウスカーソルがウィンドウの外に出れなくなるというものです
しかし、フォーカスを別のプログラムに当てれば設定は解除されます

また、Cursor クラスはマウスカーソルの表示/非標示を制御する機能も提供しています
カーソルを消すには Cursor.Hide() メソッドを
消したカーソルを再び表示するには Cursor.Show() メソッドを使います

```
[Serializable]
public static void Hide();

[Serializable]
public static void Show();
```

これらのメソッドの呼び出しは均等化しなければなりません
Hide() メソッドを呼び出した場合は、その後に Show() メソッドを呼び出してください
この理由は、Windows が内部でカウンタを用いてカーソルの表示を管理しているからです

```
using System.Windows.Forms;

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

 override protected void OnMouseEnter(System.EventArgs e) {
  Cursor.Hide();
 }
 override protected void OnMouseLeave(System.EventArgs e) {
  Cursor.Show();
 }
}
```

このプログラムのウィンドウのクライアント領域にカーソルを移動させると
クライアント領域に入った瞬間、唐突とカーソルが消えてしまいます
逆に、クライアント領域から外に出れば、再び表示されます

カーソルの位置は Cursor.Position プロパティで表されています
このプロパティから、カーソルの位置を取得、または設定することができます

```
[Serializable]
public static Point Position {get; set;}
```

このプロパティは、カーソルの位置を示す Point 型です
Position プロパティを設定すれば、プログラムから自由にカーソルの位置を変更できます

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

class WinMain : Form {
```

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

 override protected void OnMouseDown(MouseEventArgs e) {
  int x = Location.X + (Width / 2);
  int y = Location.Y + (Height - ClientRectangle.Height) / 2 ;
  Cursor.Position = new Point(x , y);
 }
}
```

このプログラムのウィンドウのクライアント領域をクリックすると
マウスカーソルがウィンドウのタイトルバーの中央に移動します
タイトルバーの中央は、ウィンドウの位置やサイズを計算して割り出しています

カーソルの変更

現在、ディスプレイに表示されているユーザーが動かしているカーソル
すなわちカレントカーソルは **Cursor.Current** プロパティで表されています

```
[Serializable]
public static Cursor Current {get; set;}
```

このプロパティから、現在のカーソルの Cursor オブジェクトを取得したり
あるいは、Cursor オブジェクトを渡すことによって、カーソルを変更できます

Windows システムが使う一般的なカーソルイメージは
System.Windows.Forms.Cursors クラスにまとめられています

public sealed class Cursors

このクラスは、一般的な Cursor オブジェクトのコレクションです
次のような、静的な読み込み専用公開プロパティを定義しています

メンバ	解説
AppStarting	アプリケーションの開始時に表示されるカーソルを取得します
Arrow	矢印カーソルを取得します
Cross	十字カーソルを取得します
Default	既定のカーソルを取得します。通常は矢印カーソルです
Hand	Web リンクの上にマウスを移動すると表示されるハンド カーソルを取得します
Help	矢印と疑問符が組み合わされたヘルプ カーソルを取得します
HSplit	マウスを水平方向の分割バーの上に置くと表示されるカーソルを取得します
IBeam	マウスをクリックしたときにテキスト カーソルの位置を示す I ビーム カーソルを取得します
No	現在の操作が無効な領域であることを示すカーソルを取得します
NoMove2D	マウスを動かさずに ウィンドウを水平および垂直の両方向にスクロールできるとき このホイール操作を表すカーソルを取得します
NoMoveHoriz	マウスを動かさずに 水平方向へのスクロールができるとき このホイール操作を表すカーソルを取得します
NoMoveVert	マウスを動かさずに ウィンドウを垂直方向にスクロールできるとき このホイール操作を表すカーソルを取得します
PanEast	マウスを動かしながら ウィンドウを水平方向に右スクロールできるとき このホイール操作を表すカーソルを取得します
PanNE	マウスを動かしながら ウィンドウを水平方向および垂直方向に右上へスクロールするとき このホイール操作を表すカーソルを取得します
PanNorth	マウスを動かしながら ウィンドウを垂直方向に上へスクロールするとき このホイール操作を表すカーソルを取得します
PanNW	マウスを動かしながら ウィンドウを水平方向および垂直方向に左上へスクロールするとき このホイール操作を表すカーソルを取得します
PanSE	マウスを動かしながら ウィンドウを水平方向および垂直方向に右下へスクロールするとき このホイール操作を表すカーソルを取得します
PanSouth	マウスを動かしながら ウィンドウを垂直方向に下へスクロールするとき このホイール操作を表すカーソルを取得します
PanSW	マウスを動かしながら ウィンドウを水平方向および垂直方向に左下へスクロールするとき このホイール操作を表すカーソルを取得します
PanWest	マウスを動かしながら ウィンドウを水平方向に左へスクロールするとき このホイール操作を表すカーソルを取得します

SizeAll	十字型の方向を指す矢印が結合して構成されている 4 方向のサイズ変更用カーソルを取得します
SizeNESW	2 方向の対角線 (右斜めと左斜め) で構成されている サイズ変更用カーソルを取得します
SizeNS	垂直の 2 方向 (上と下) で構成されているサイズ変更用カーソルを取得します
SizeNWSE	2 方向の対角線 (左斜めと右斜め) で構成されている サイズ変更用カーソルを取得します
SizeWE	水平の 2 方向 (左と右) で構成されているサイズ変更用カーソルを取得します
UpArrow	通常はカーソル位置の識別に使用する上向きの矢印カーソルを取得します
VSplit	マウスを垂直方向の分割バーの上に置くと表示されるカーソルを取得します
WaitCursor	待機カーソルを取得します。通常は砂時計の形です

これらのメンバは、全てが静的な読み取り専用プロパティとして定義されています

```
using System.Windows.Forms;

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

 override protected void OnMouseDown(MouseEventArgs e) {
  Cursor.Current = Cursors.WaitCursor;
 }
 override protected void OnMouseUp(MouseEventArgs e) {
  Cursor.Current = Cursors.Default;
 }
}
```

このプログラムのウィンドウのクライアント領域でマウスボタンを押すと
カーソルが砂時計に変化し、ボタンを離すと元に戻ります

また、カーソルは Cursor() コンストラクタを用いて
独自に作成した *.cur ファイルを使用 することもできます
Cursor クラスのコンストラクタは次のものが定義されています

```
[Serializable]
public Cursor(IntPtr handle);

[Serializable]
public Cursor(Stream stream);

[Serializable]
public Cursor(string fileName);

[Serializable]
public Cursor(Type type , string resource);
```

handle には作成するカーソルの Windows ハンドルを指定します
stream には読み込みもとのデータストリームを、fileName にはファイル名を指定します
type にはリソースタイプを、resource にはリソース名を指定します

```
using System.Windows.Forms;

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

 override protected void OnMouseDown(MouseEventArgs e) {
  Cursor.Current = new Cursor("test.cur");
 }
 override protected void OnMouseUp(MouseEventArgs e) {
  Cursor.Current = Cursors.Default;
 }
}
```

図を見てわかるように、このプログラムは独自のカーソルを表示します
あの、猫の手のようなマークが、今回使った test.cur というカーソルです

因みに、Control クラスにはカーソルを設定する Control.Cursor プロパティがあり
これを設定すれば、カーソルがコントロールの上にあるとき
このプロパティが指す Cursor オブジェクトを表示するように動作させることができます

public virtual Cursor Cursor {get; set;}

見てわかるように、このプロパティはあくまで仮想メンバです
このカーソルは利用者の要求であり、どのように利用するかはコントロール設計者に委ねられます

```
using System.Windows.Forms;

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

 public WinMain() {
  Cursor = Cursors.WaitCursor;
 }
}
```

このプログラムは特にイベント処理を施していませんが
カーソルがウィンドウのクライアント領域に移動すると、カーソルが砂時計に変わります

カーソルの描画

さらに、.NET ではカーソルを描画する手段も提供しています
ただし、アイコンの時とは異なりカーソルの描画は普段とは逆の手順を踏みます

通常は、オブジェクトを Graphics クラスの Draw***() に渡していましたが
残念ながら Graphics.DrawCursor() というメソッドは存在しません
カーソルの描画は、逆に **Cursor.Draw()** メソッドを使って描画します

```
[Serializable]
public void Draw(Graphics g , Rectangle targetRect);
```

g には描画対象であるデバイスコンテキストを表す Graphics オブジェクトを
targetRect にはカーソルを描画する範囲を示す長方形を指定します
ただし、このメソッドは常にカーソルのデフォルトサイズで描画します
カーソルを拡大縮小して描画したい場合は **Cursor.DrawStretched()** を使います

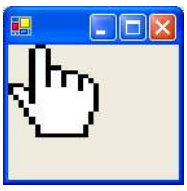
```
[Serializable]
public void DrawStretched(Graphics g , Rectangle targetRect);
```

パラメータの意味は Cursor.Draw() メソッドと同じですが
このメソッドの場合は、指定した長方形のサイズにカーソルをスケーリングします

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

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

 override protected void OnResize(System.EventArgs e) {
  Invalidate();
 }
 override protected void OnPaint(PaintEventArgs e) {
  Graphics g = e.Graphics;
  Cursors.Hand.DrawStretched(g , ClientRectangle);
 }
}
```



このプログラムは、Cursors.Hand カーソルをクライアント領域に描画しています
ちなみに、カーソルのサイズを取得したければ **Cursor.Size** プロパティで得られます

```
[Serializable]
public Size Size {get;}
```

描画処理などで、カーソルの正確なサイズが知りたければ使うとよいでしょう

前のページへ　　戻る　　次のページへ