ウィンドウの制御

ウィンドウ文字列

全てのコントロールには、コントロールに関連付けられる文字列のプロパティがあります
この文字列がどのように利用されるかは、基本的にコントロールの役割で変化しますが
エディットウィンドウならば、入力されている文字列になるでしょうし
ボタンならば、ボタンに表示される文字列という形で、文字列が利用されます

この、コントロールに関連付けられる文字列は
Control.Text プロパティで設定したり取得したりすることができます

public virtual string Text {get; set;}

Form クラスを継承した、タイトルバーを保有するウィンドウの場合
この文字列はタイトルバーやタスクバーのアイコンに表示される文字列として利用されます

```
using System.Windows.Forms;

class WinMain : Form {
 public static void Main(string[] args) {
  WinMain win = new WinMain();
  win.Text = "Kitty on your lap";
  Application.Run(win);
 }
}
```

このプログラムは、ウィンドウのタイトルバーに文字列を表示させています
図を見てわかるように、ウィンドウのタイトルバーに指定した文字列が表示されています

背景と前景

Control クラスは、プロパティで背景色や前景色を設定することができます
具体的にこれがどのように反映されるかは、そのコントロールによって異なりますが
カスタムコントロールも含め、コントロールはこれにしたがって描画を行うべきです

プログラムは、コントロールの外部からこういった公開プロパティを制御することで
コントロールのフォント、色、背景色などを要求することができるべきであると考えられます

背景色は Control.BackColor プロパティで定めることができます
また、背景イメージを Control.BackgroundImage プロパティで指定して
コントロールに対し、背景に静止画を使うように要求することもできます

public virtual Color BackColor {get; set;}
public virtual Image BackgroundImage {get; set;}

見てわかるように、このプロパティは仮想プロパティです
要求された背景色やイメージをどう使うかは、実装されるコントロールしだいです

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

class WinMain : Form {
 public static void Main(string[] args) {
  WinMain win = new WinMain();
  win.BackColor = Color.White;
  Application.Run(win);
 }
}
```

このプログラムは、Form クラスの BackColor を白色に設定しています
実行すると、予想通り背景が白いウィンドウが生成されます

動揺の考え方で、前景色やフォントを設定、または取得することもできます
前景色は Control.ForeColor プロパティで
フォントは Control.Font プロパティでアクセスできます

public virtual Color ForeColor {get; set;}
public virtual Font Font {get; set;}

これを用いれば、コントロールに対して文字列や図形を描画する時に
特定のフォントや色を使ってほしいという要求を外部から知らせることができます

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

class WinMain : Form {
 public static void Main(string[] args) {
  WinMain win = new WinMain();
  win.Font = new Font(args[0] , 30);
  Application.Run(win);
 }
```

```
 override protected void OnPaint(PaintEventArgs e) {
  Graphics g = e.Graphics;
  Brush brush = new SolidBrush(ForeColor);
  g.DrawString("Kitty on your lap" , Font , brush , 0 , 0);
 }
}
```

このプログラムは、コマンドライン引数で指定したフォント名のフォントで
ウィンドウに、設定されている前景色のブラシを用いて文字列を描画します

ウィンドウの表示

コントロールを破棄せずに、単純にコントロールを隠したい場合などは
コントロールの表示/非表示を制御するプロパティやメソッドを使います
これは Control.Visible プロパティで表現されます

public bool Visible {get; set;}

true ならば表示されることを意味し、false ならば隠れることを意味します
また、これを間接的に false に設定する Control.Hide() メソッドと
true に設定する Control.Show() メソッドというものも存在します

public void Hide();
public void Show();

筆者はこういった作業はプロパティで行うべきであると考えます

```
using System.Windows.Forms;

class WinMain : Form {
 public static void Main(string[] args) {
  WinMain mainWin = new WinMain();
  WinMain subWin = new WinMain();

  mainWin.Text = "This is Main Window";
  subWin.Text = "This is Sub Window";
  subWin.Visible = true;

  Application.Run(mainWin);
 }
}
```

このプログラムは、2つのウィンドウを生成しています
Run() メソッドに渡して、終了動作に関連付けるメインウィンドウは
Run() メソッドが自動的にウィンドウを生成、表示してくれますが
これに渡していない subWin は、手動で Visible を true にして表示しています

前のページへ　　戻る　　次のページへ