子コントロール

コントロールを追加する

ウィンドウシステムの最大の魅力は、コントロールの中に別のコントロールを挿入し
それぞれのコントロールに専門的な処理や役割を与えることができることです

.NET では Control クラスに子コントロールをコレクションする管理機能があり
すなわち、これを継承する全てのクラスにその機能があると考えることができます

子コントロールは、独自に Control クラスを拡張したクラスでもかまいませんし
.NET クラスライブラリが用意する汎用的で高度なコントロール群でもよいでしょう
もちろん、ボタンや入力ボックスなどの基本的なコントロールが用意されています

子コントロールの管理は
System.Windows.Forms.Control.ControlCollection が行います
このクラスは、Control クラスの内部で定義される内部クラスです
Menu クラスにおける MenuItemCollection クラスと同じような存在です

```
public class Control.ControlCollection :
 IList , ICollection , IEnumerable , ICloneable
```

このクラスは、インデクサで子コントロールのコレクションを管理したり
コントロールの追加や削除などを包括的に行ってくれます
このクラスのコンストラクタは次のように定義されています

public Control.ControlCollection(Control owner);

owner には、このクラスを所有している Control クラスを指定します
しかし、通常外部からこのクラスのインスタンスを作成するようなことはありません
ControlCollection にアクセスするには **Control.Controls** プロパティを使います

public Control.ControlCollection Controls {get;}

このプロパティは、コントロールが所有する ControlCollection オブジェクトを返します

コントロールを追加するには **ControlCollection.Add()** メソッド
または **ControlCollection.AddRange()** メソッドを用います

public virtual void Add(Control value);
public virtual void AddRange(Control[] controls);

value には追加する子コントロールを
controls には、追加する子コントロールの配列を指定します
複数のコントロールを同時に追加するには AddRange() を使うと便利でしょう

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

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

 public WinMain() {
  Control ctrl = new Control();
  ctrl.Bounds = new Rectangle(0 , 0 , 100 , 50);
  ctrl.BackColor = Color.Red;
  Controls.Add(ctrl);
 }
}
```

このプログラムを実行すると、図のようなウィンドウが表示されます
しかし、あの赤い長方形は FillRectangle() などで塗りつぶしているわけではありません
Control クラスのインスタンスを生成し、背景色を赤にし、
横 100 ピクセル、縦 50 ピクセルでフォームに追加したのです

ControlCollection は、さらに追加されたコントロールのコレクションを高度に管理します
現在コントロールが保有する子コントロールの数やインデックス番号
指定の子コントロールの削除なども包括的に行ってくれるので、レイアウトが極めて簡単です

コントロールを削除するには **ControlCollection.Remove()** メソッド
または **ControlCollection.RemoveAt()** メソッドを使います
前者はコントロールオブジェクトで、後者はインデックス番号でコントロールを指定します

public virtual void Remove(Control value);
public void RemoveAt(int index);

value には、削除対象のコントロールを
index には、追加された順番で 0 から数えられるインデックス番号を指定します
もし RemoveAt() メソッドで 2 を指定すれば、3番目の位置にあるコントロールを示します
コントロールが削除されれば、それ以降のコントロールのインデックス番号が詰められます

ControlCollection.Count プロパティを用いれば
コントロールに追加されている子コントロールの数を知ることができます
インデックスを指定する時に、番号の最大値を知る時などに役に立つでしょう

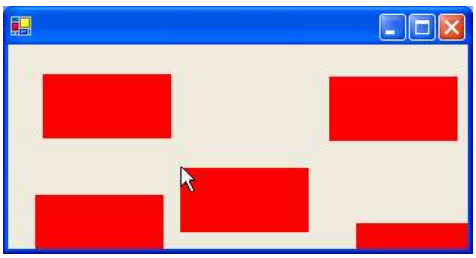
public int Count {get;}

このプロパティが返す値 - 1 が、指定可能なインデックスの最大値と考えられます

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

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

 override protected void OnMouseUp(MouseEventArgs e) {
  if (e.Button == MouseButtons.Left) {
   Control ctrl = new Control();
   ctrl.Bounds = new Rectangle(e.X , e.Y , 100 , 50);
   ctrl.BackColor = Color.Red;
   Controls.Add(ctrl);
  }
  else {
   if (Controls.Count != 0) Controls.RemoveAt(0);
  }
 }
}
```



このプログラムは、ウィンドウのクライアント領域を左クリックするとコントロールが追加され
右クリックすると、古いコントロールの順番にクライアント領域から削除していきます
また、Count プロパティを使って、子コントロールの数を監視しています
因みに ControlCollection.Clear() メソッドで全てを削除することもできます

public virtual void Clear();

このメソッドは ControlCollection に登録されている全てのコントロールを解除します

アンカー

多くの場合、フォームのサイズは実行時にユーザーが変更することができます
そして、サイズが変更されれば子コントロールの位置も変更する必要があります

手段のひとつとしては、OnResize() や OnLayout() メソッドをオーバーライドし
子コントロールの位置は、常に親に対して相対的に決定されるように仕組みます
しかし、.NET では、これらの座標の計算を自動化してくれる手段を提供しています

まず、コントロールにはアンカーと呼ばれるプロパティがあります
アンカーは、親のサイズ変更に対してどの位置を対象に固定するかを決めます
これは、コンテナの長方形の特定の部分と子を関連付けるために用います

アンカーは Control.Anchor プロパティで設定することができます

public virtual AnchorStyles Anchor {get; set;}

このプロパティ型の AnchorStyles は次のような列挙型です
FlagsAttribute 属性が指定されているので、ビット演算が可能です

[Flags]
[Serializable]
public enum AnchorStyles

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	意味
Bottom	コントロールがそのコンテナの下端に固定されています
Left	コントロールがそのコンテナの左端に固定されています
None	コントロールがそのコンテナの端に固定されていません

Right	コントロールがそのコンテナの右端に固定されています
Top	コントロールがそのコンテナの上端に固定されています

デフォルトで、コントロールはコンテナのサイズに対して静的です
これは、デフォルトで AnchorStyles.Left | AnchorStyles.Top が設定されているためです
そのため、コンテナのサイズが変更してもコンテナから見て絶対座標が変化しません
なぜならば、コントロールが常にコンテナの左上に対して固定されているからです
この場合、コントロールの位置が変化するのはコンテナの左上が移動した場合だけです

これとは逆に、コンテナの右下のコントロールを固定すれば
例えば、ダイアログボックスの OK ボタンなどを作成する時に便利でしょう

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

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

 public WinMain() {
  ClientSize = new Size(400 , 300);

  Control ctrl = new Control();
  ctrl.BackColor = Color.Red;
  ctrl.Bounds = new Rectangle(190 , 250 , 200 , 40);
  ctrl.Anchor = AnchorStyles.Right | AnchorStyles.Bottom;
  Controls.Add(ctrl);
 }
}
```

このプログラムは、追加したコントロールが常にフォームの右下に固定されています
そのため、サイズを変更しても、常にフォームの右下に対して
縦横 10 ピクセルのマージンをとった位置にコントロールが固定されます

では、ここでアンカーのちょっとした Tips を紹介したいと思います
次のようなプログラムは、どのように動作するか考えてください

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

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

 public WinMain() {
  ClientSize = new Size(400 , 300);

  Control ctrl = new Control();
  ctrl.BackColor = Color.Red;
  ctrl.Bounds = new Rectangle(100 , 75 , 200 , 150);
  ctrl.Anchor =  AnchorStyles.Top | AnchorStyles.Bottom |
    AnchorStyles.Left | AnchorStyles.Right;
  Controls.Add(ctrl);
 }
}
```

このプログラムは、子コントロールがコンテナの全ての辺に固定されています
フォームのサイズを変更すると、はたして子コントロールはどのようになるでしょうか？

ドッキング

例えば、ステータスバーやメニューバー、ツールバーのようなコントロールを想像して下さい
常にコンテナの端に配置されるようなこのようなコントロールを設計する場合
コントロールの座標の計算は、ドッキングに委託するべきです

ドッキングはコンテナの辺に関連付ける点でアンカーと性質が似ていますが
アンカーが関連付けた辺に対して固定させるものなのに対し
ドッキングは**関連付けた辺と隣接する２辺に接触**させます
この機能は、ステータスバーのようなコントロールに最適です

ドッキングの設定は **Control.Dock** プロパティで行います

public virtual DockStyle Dock {get; set;}

DockStyle 列挙型は、基本的に AnchorStyles に似ていますがビット演算は行えません
そのため、アンカーのように複数の辺に関連付けることはできません

[Serializable]
public enum DockStyle

この列挙型のメンバは次のようなものが定義されています

メンバ	意味
Bottom	格納されるコントロールの下端は、格納する側のコントロールの下端にドッキングされます

Fill	格納されるコントロールの四辺は 格納する側のコントロールの四辺にドッキングされ、適切なサイズに調節されます
Left	格納されるコントロールの左端は、格納する側のコントロールの左端にドッキングされます
None	コントロールはドッキングされていません
Right	格納されるコントロールの右端は、格納する側のコントロールの右端にドッキングされます
Top	格納されるコントロールの上端は、格納する側のコントロールの上端にドッキングされます

デフォルトで、コントロールのドッキングは DockStyle.None です

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

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

 public WinMain() {
  ClientSize = new Size(400 , 300);

  Control ctrl = new Control();
  ctrl.BackColor = Color.Red;
  ctrl.Height = 30;
  ctrl.Dock = DockStyle.Bottom;
  Controls.Add(ctrl);
 }
}
```

ドッキングを設定した場合、関連付けられた辺と隣接する2辺に接触する位置とサイズは
Bounds プロパティなどで絶対座標を指定しても無視されます
このプログラムは、子コントロールが常にフォームの下部にドッキングされています
いかに、ステータスバーやツールバーのようなコントロールに適しているかを確認できるでしょう

前のページへ　戻る　次のページへ