テキストボックス

文字入力のサポート

GUI アプリケーションがユーザーに文字列の入力を求める場合
テキスト入力、及び表示専用のテキストボックス・コントロールを用います

.NET では、まず System.Windows.Forms.TextBoxBase 抽象クラスがあります
このクラスは、テキスト入力コントロールが提供するべき基本的な機能を提供しています

```
System.Object
   System.MarshalByRefObject
      System.ComponentModel.Component
         System.Windows.Forms.Control
            System.Windows.Forms.TextBoxBase

public abstract class TextBoxBase : Control
```

このクラスを継承した System.Windows.Forms.TextBox を使えば
ユーザーに文字を入力させたり、あるいは文字を表示させるコントロールとして使うことができます
TextBox クラスの重要な機能のほとんどは TextBoxBase クラスで提供されているメンバです

```
System.Object
   System.MarshalByRefObject
      System.ComponentModel.Component
         System.Windows.Forms.Control
            System.Windows.Forms.TextBoxBase
               System.Windows.Forms.TextBox

public class TextBox : TextBoxBase
```

このクラスのコンストラクタは、デフォルトコンストラクタのみです
コントロールに表示される文字列は、Contorl.Text プロパティです
Text プロパティを使うことで、テキストの取得や設定ができます

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

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

 public WinMain() {
  TextBox txt = new TextBox();
  txt.Size = new Size(200 , 20);
  txt.KeyUp+= new KeyEventHandler(_KeyUp);

  Controls.Add(txt);
 }

 public void _KeyUp(object sender , KeyEventArgs e) {
  Text = ((Control)sender).Text;
 }
}
```

このプログラムは、1行テキストボックスを表示します
テキストボックスに文字列を入力すると、イベントによってタイトルバーと同期します
ちなみに、このコントロールはデフォルトで標準のポップアップメニューをサポートしています

テキストボックスは、デフォルトでは1行入力の状態です
複数行入力可能なマルチラインテキストボックスを表示したい場合は
TextBoxBase.Multiline プロパティを設定する必要があります

public virtual bool Multiline {get; set;}

コントロールが複数行テキスト ボックス コントロールである場合は true
それ以外の場合は false を指定します

このとき、コントロールの幅以上に文字列が入力された時に
自動的に改行するかどうかを TextBoxBase.WordWrap プロパティで選択できます

public bool WordWrap {get; set;}

折り返される場合は true、そうでなければ false を示します
デフォルトでは treu と定義されているので、自動的に改行されます
折り返しがない場合や、一定行以上の入力があるとテキストボックスのサイズを超える場合があります
これに備えて TextBox.ScrollBars プロパティでスクロールバーを指定できます

public ScrollBars ScrollBars {get; set;}

このプロパティは、スクロールバーの状態を示す
System.Windows.Forms.ScrollBars 列挙型を用います

```
[Serializable]
```

```
public enum ScrollBars
```

この列挙型は、次のような意味を持つメンバを定義しています

メンバ	解説
Both	水平スクロール バーと垂直スクロール バーの両方が表示されます
Horizontal	水平スクロール バーだけが表示されます
None	スクロール バーは表示されません
Vertical	垂直スクロール バーだけが表示されます

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

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

 public WinMain() {
  TextBox txt = new TextBox();
  txt.Multiline = true;
  txt.Size = new Size(200 , 100);
  txt.WordWrap = false;
  txt.ScrollBars = ScrollBars.Both;

  Controls.Add(txt);
 }
}
```

このプログラムは、図のようなマルチラインテキストボックスを表示します
スクロールバーを表示することによって、テキストボックスの幅や高さ以上の入力があっても
ユーザーはスクロールバーを利用してテキストを自由に編集したり参照できるでしょう

テキストボックスに入力できる最大文字数を指定したい場合は
TextBoxBase.MaxLength プロパティを使います

public virtual int MaxLength {get; set;}

このプロパティは、入力可能な最大文字数を指定します
ただし 0 の場合は、使用可能なメモリを使い果たすまで入力できることを示します

テキストボックスは、特定の範囲の文字列を選択することができますが
選択されてる文字列は **TextBoxBase.SelectedText** プロパティで
文字数は **TextBoxBase.SelectionLength** プロパティで
選択開始位置は **TextBoxBase.SelectionStart** プロパティで得られます

public virtual string SelectedText {get; set;}
public virtual int SelectionLength {get; set;}
public int SelectionStart {get; set;}

独自に、テキストボックス内の文字列を編集する能力をつける場合に必要となるでしょう
または、選択範囲をプログラムから制御する時にも使うことができます

単純にテキストを表示するコントロールとしてテキストボックスを使いたい場合
TextBoxBase.ReadOnly プロパティを設定することで、編集を不能にします

public bool ReadOnly {get; set;}

テキストが読み取り専用の場合は true 。それ以外の場合は false を指定します
デフォルトは false になっています
テキストの位置を **TextBox.TextAlign** プロパティで設定することもできます

public HorizontalAlignment TextAlign {get; set;}

このプロパティは **System.Windows.Forms.HorizontalAlignment** 列挙型です

```
[Serializable]
[ComVisible(true)]
public enum HorizontalAlignment
```

この列挙型は、次のような意味を持つメンバを定義しています

メンバ	解説
Center	コントロール要素の中央に配置されます
Left	コントロール要素の左側に配置されます

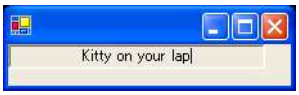
Right	コントロール要素の右側に配置されます

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

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

 public WinMain() {
  TextBox txt = new TextBox();
  txt.Size = new Size(200 , 20);
  txt.ReadOnly = true;
  txt.TextAlign = HorizontalAlignment.Center;
  txt.Text = "Kitty on your lap";

  Controls.Add(txt);
 }
}
```



このプログラムは、読み取り専用のテキストボックスを表示します
このテキストボックスのテキストの配置位置は、中央に設定されているため
図のように、コントロールの中央に文字列が配置されます

編集

Windows 標準の「メモ帳」アプリケーションのような
テキストデータの編集を目的としたソフトウェアを作る場合は
入力以外にも、コピーや貼り付けといったクリップボードを介した編集処理が必要です

クリップボードについては後ほど詳しく解説しますが
そのような処理を記述する場合、ひとつの方法としては SelectedText プロパティで
選択中のテキストを得て、これをクリップボードに転送する手段があります
もちろん、これがもっとも正当な方法ですが、実は TextBoxBase クラスが
すでにコピーや、削除、ペーストというような基本機能をすでに実装しています

選択中のテキストをクリップボードに送るには **TextBoxBase.Copy()** メソッド
コピーと同時に選択中のテキストを消すには **TextBoxBase.Cut()**、
クリップボードのテキストを貼り付けるには **TextBoxBase.Paste()**、
テキストボックスのテキストを削除するには **TextBoxBase.Clear()** を使います

public void Copy();
public void Cut();
public void Paste();
public void Clear();

因みに、TextBox クラスにおいて Clear() は Text = ""と実装上同じです
直前の操作に状態を戻すには **TextBoxBase.Undo()** メソッドを
アンドゥ可能かどうかを調べるには **TextBoxBase.CanUndo** プロパティを使います

public void Undo();
public bool CanUndo {get;}

CanUndo は、アンドゥ可能であれば true を返します
ただし、CanUndo が false でも、Undo() を呼び出して例外が発生することはありません
これらのメソッドを用いれば、単純なテキストエディタを作成することができます

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

public class WinMain : Form {
 MenuItem[] menuItem;
 TextBox textBox1 = new TextBox();

 public static void Main() {
  Application.Run(new WinMain());
 }

 public WinMain() {
  menuItem = new MenuItem[] {
   new MenuItem("切り取り(&T)" , new EventHandler(menuItemClick)) ,
   new MenuItem("コピー(&C)" , new EventHandler(menuItemClick)) ,
   new MenuItem("貼り付け(&P)" , new EventHandler(menuItemClick)) ,
   new MenuItem("削除(&L)" , new EventHandler(menuItemClick)) ,
   new MenuItem("-") ,
   new MenuItem("元に戻す(&U)" , new EventHandler(menuItemClick)) ,
  };
  textBox1.Dock = DockStyle.Fill;
  textBox1.Multiline = true;
  textBox1.ContextMenu = new ContextMenu(menuItem);
  Controls.Add(textBox1);
 }
```

```
 private void menuItemClick(object sender , EventArgs e) {
  if(sender == menuItem[0]) textBox1.Cut();
  else if(sender == menuItem[1]) textBox1.Copy();
  else if(sender == menuItem[2]) textBox1.Paste();
  else if(sender == menuItem[3]) textBox1.Clear();

  else if(sender == menuItem[5]) textBox1.Undo();
 }
}
```

このプログラムは、極めて単純なテキストエディタです
後はこれに保存や読み込み、検索などの機能をつければメモ帳になるでしょう

前のページへ　戻る　次のページへ