コンボボックス

編集とリスト

コンボボックスは、テキストボックスとリストボックスを合わせたようなコントロールで
編集やリストの展開及び選択というような機能をまとめて提供します
コンボボックスの作成には System.Windows.Forms.ComboBox クラスを用います

```
System.Object
   System.MarshalByRefObject
      System.ComponentModel.Component
         System.Windows.Forms.Control
            System.Windows.Forms.ListControl
               System.Windows.Forms.ComboBox

public class ComboBox : ListControl
```

コンストラクタはデフォルトコンストラクタしか定義されていません
コンボボックスも、リストに表示する項目を管理するための内部クラスを持ちます
これにアクセスするには ComboBox.Items プロパティを用います

public ComboBox.ObjectCollection Items {get;}

ComboBox.ObjectCollection クラスは次のように定義されています

```
public class ComboBox.ObjectCollection :
 IList , ICollection , IEnumerable
```

このクラスの基本的なメソッドは ListBox.ObjectCollection と同じです
項目の追加は ComboBox.ObjectCollection.Add() メソッド
または ComboBox.ObjectCollection.AddRange() メソッドを用います

public int Add(object item);
public void AddRange(object[] items);

item には追加する項目のオブジェクトを
items には追加する項目の配列を指定します
Add() メソッドはコレクション内の項目の 0 から始まるインデックスを返します

項目数を得る ComboBox.ObjectCollection.Count プロパティや
項目を示すオブジェクトの ComboBox.ObjectCollection.Item インデクサもあります

public int Count {get;}
public virtual object this[int index] {get; set;}

また、項目を削除するには ComboBox.ObjectCollection.Remove()
インデックスを指定して削除するには ComboBox.ObjectCollection.RemoveAt()
全て削除するには ComboBox.ObjectCollection.Clear() メソッドを使います

public void Remove(object value);
public void RemoveAt(int index);
public void Clear();

value には削除する項目を示すオブジェクトを
index には削除するインデックスを指定します

これらは、基本的にリストボックスの項目操作とほとんど同じです
そのため、リストボックスを理解していれば項目の制御は簡単でしょう

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

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

 public WinMain() {
  ComboBox comboBox1 = new ComboBox();
  comboBox1.Size = new Size(200 , 20);
  comboBox1.Items.AddRange(new object[] {
   "Kitty on your lap" , "The forever that you wish"
  });
  Controls.Add(comboBox1);
 }
}
```

このプログラムを実行すると、上の図のようなコンボボックスが表示されます
テキストボックスの右端にあるボタンを押すと、リストが表示されます

デフォルトで、テキストボックスの文字列は編集可能となっています

しかし、多くの場合、コンボボックスはスペースを有効利用したリストとして利用します
この場合、テキストボックスの編集はむしろ不都合になってしまうでしょう

そこで、コンボボックスの基本的なスタイルを変更することで、基本的な挙動を設定できます
これは **ComboBox.DropDownStyle** プロパティを用います

public ComboBoxStyle DropDownStyle {get; set;}

System.Windows.Forms.ComboBoxStyle 列挙型は次のように定義されています

public enum ComboBoxStyle

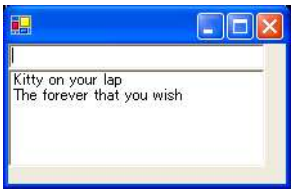
この列挙型には、次のような意味を持つメンバが定義されています

メンバ	解説
DropDown	テキスト部分は編集できます リスト部分を表示するには、矢印ボタンをクリックします
DropDownList	ユーザーはテキスト部分を直接編集できません リスト部分を表示するには、矢印ボタンをクリックします
Simple	テキスト部分は編集できます リスト部分は常に表示されます

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

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

 public WinMain() {
  ComboBox comboBox1 = new ComboBox();
  comboBox1.DropDownStyle = ComboBoxStyle.Simple;
  comboBox1.Size = new Size(200 , 100);
  comboBox1.Items.AddRange(new object[] {
   "Kitty on your lap" , "The forever that you wish"
  });
  Controls.Add(comboBox1);
 }
}
```



このプログラムは、上の図のような単純なコンボボックスを表示します
これは、純粋にテキストボックスとリストボックスだと思ってよいでしょう

ドロップダウン部分が表示されているかどうかを調べたり、制御するには
ComboBox.DroppedDown プロパティを用います
このプロパティを使えば、プログラムからドロップダウンを展開することもできます

public bool DroppedDown {get; set;}

コンボボックスのリストから、現在何が選択されているのかを取得したり設定するには
インデックスを用いて **ComboBox.SelectedIndex** プロパティで行うか
オブジェクトを用いて **ComboBox.SelectedItem** プロパティで行います

public override int SelectedIndex {get; set;}
public object SelectedItem {get; set;}

これらのプロパティを受ける、もっとも最良のタイミングは
ユーザーがリストで別の項目を選択するか、ドロップダウンリストを閉じた時でしょう
SelectedIndex プロパティの値が変わると **ComboBox.SelectedIndexChanged** イベントが
変更内容がコミットされると **ComboBox.SelectionChangeCommitted** イベントが発生します

public event EventHandler SelectedIndexChanged;
public event EventHandler SelectionChangeCommitted;

SelectionChangeCommitted などは、選択内容が変更されると発生するため
これを用いて変更のタイミングにあわせて、何らかのアクションをおこすことができます

前のページへ　　戻る　　次のページへ