メニュー項目

メニュー項目を追加する

メニューバーのメニュー項目を選択すると、通常はポップアップが表示されます
このポップアップは、さらにその子メニュー項目が陳列されています

ポップアップを表示するには、メニュー項目に子メニュー項目を追加します
この方法はいくつか存在するので、簡単な方法から説明していきましょう
まず、もっとも単純なのは Menu.MenuItemCollection クラスの Add() メソッドで
メニュー項目のキャプションと、MenuItem の配列を指定する方法です

```
using System.Windows.Forms;

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

 public WinMain() {
  MainMenu mm = new MainMenu();
  MenuItem[] mi = {
   new MenuItem("Rena(&R)") ,
   new MenuItem("Yuki(&Y)") ,
   new MenuItem("Mimi(&M)")
  };
  mm.MenuItems.Add("Kitty on your lap(&K)" , mi);
  Menu = mm;
 }
}
```

このプログラムは、まずメニューバーにひとつのメニュー項目を追加しています
そして、同時にポップアップとなる MenuItem の配列を指定しています
これは MenuItem() コンストラクタでも似たようなことができます

```
using System.Windows.Forms;

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

 public WinMain() {
  MainMenu mm = new MainMenu();
  MenuItem[] mi = {
   new MenuItem("Rena(&R)") ,
   new MenuItem("Yuki(&Y)") ,
   new MenuItem("Mimi(&M)")
  };
  mm.MenuItems.Add(new MenuItem("Kitty(&K)" , mi));
  Menu = mm;
 }
}
```

上のプログラムを少しだけ改良したものです
Add() メソッドに、子メニューを含む MenuItem を渡しています

ただし、これらの方法はきわめて静的な追加方法といえます
より動的に行うには、MenuItem の MenuItemCollection に追加する方法が考えられます
MenuItem も Menu を継承しているので、これに MenuItem を追加できても不思議ではありません
メニューの中にある MenuItem は **MenuItemCollection.Item** で得られます

public virtual MenuItem this[int index] {get;}

メニューは、追加順にインデックスで管理されるためインデクサが制御の専門です
C# を使えば、予想通りメニュー項目へのアクセスはインデクサをもって行えます
しかし、メニュー項目の総数よりも大きい数字を指定しまえば例外が発生してしまいます
これを防ぐには **MenuItemCollection.Count** プロパティを使うとよいでしょう

public int Count {get;}

このプロパティは、MenuItem オブジェクトの総数を返します

```
using System.Windows.Forms;

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

 public WinMain() {
  MainMenu mm = new MainMenu();
```

```
  mm.MenuItems.Add("Kitty on your lap(&K)");

  MenuItem mi = mm.MenuItems[0];
  mi.MenuItems.Add("Rena(&R)");
  mi.MenuItems.Add("Yuki(&Y)");
  mi.MenuItems.Add("Mimi(&M)");

  Menu = mm;
 }
}
```

この方法は、比較的動的なメニューの作成手法です
文字列からメニュー項目を作り、その後インデクサを使って MenuItem にアクセスしています

MenuItem に追加した MenuItem もまた、MenuItemCollection を持っています
つまり、メニュー項目の子メニュー項目にも、メニュー項目を追加できることを意味しています
このように、メニュー項目に段階的に子ポップアップを含ませることができます

```
using System.Windows.Forms;

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

 public WinMain() {
  MainMenu mm = new MainMenu();
  mm.MenuItems.Add("Kitties(&K)");

  MenuItem mi = mm.MenuItems[0];
  MenuItem[] kitty = {
   new MenuItem("Rena") , new MenuItem("Yuki") ,
   new MenuItem("Mimi")
  };
  MenuItem[] maiden = {
   new MenuItem("Sakura") , new MenuItem("Forte") ,
   new MenuItem("Silva") , new MenuItem("Kaguya")
  };

  mi.MenuItems.Add("Kitty on your lap(&K)" , kitty);
  mi.MenuItems.Add("Maiden☆Breeder(&M)" , maiden);
  Menu = mm;
 }
}
```

図を見てわかるように、メニューポップアップが入れ子になっていますね
子メニュー項目をもつポップアップは、図のようにさらに展開されていきます
ブラウザのブックマークのような機能をメニューに実装する時や
ディレクトリ構造をメニューを用いて表現する場合などには、この機能が使われるでしょう

あまり見かけられませんが、必要に応じて複数行/複数列のメニューを作れます
メニューを改行する場合 **MenuItem.BarBreak** を使います

public bool BarBreak {get; set;}

このプロパティを true にすると、項目は改行されることを意味します
メニューバーにこの項目を追加すると新しい行に
ポップアップにこの項目を追加すると、新しい列にメニュー項目が追加されます

```
using System.Windows.Forms;

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

 public WinMain() {
  MainMenu mm = new MainMenu();
  mm.MenuItems.Add("Kitty on your lap");
  mm.MenuItems.Add("Maiden☆Breeder");

  MenuItem mi = new MenuItem("Tokyo mew mew");
  mi.BarBreak = true;
  mm.MenuItems.Add(mi);

  Menu = mm;
 }
}
```

このプログラムは、メニュー項目 "Tokyo mew mew" をメニューバーに追加する前に
BarBreak プロパティを true にすることによって改行しています

また、きわめて似た機能を持っている MenuItem.Break プロパティもあります
Break プロパティと BarBreak プロパティの機能は基本的に同じです

public bool Break {get; set;}

メニュー項目で新しい列を追加した時、Break はその間に仕切りを作りませんが
BarBreak は項目の左側に仕切りを作るという点で、これらのプロパティは異なっています

```
using System.Windows.Forms;

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

 public WinMain() {
  MainMenu mm = new MainMenu();
  MenuItem[] mi = {
   new MenuItem("Rena") , new MenuItem("Yuki") ,
   new MenuItem("Mimi")
  };
  mi[1].Break = true;
  mi[2].BarBreak = true;

  mm.MenuItems.Add("Kitty on your lap" , mi);

  Menu = mm;
 }
}
```

図を見ると、Rena と Yuki の間には何もありませんが
Yuki と Mimi の間には、縦線で仕切りが作られています
これが、BarBreak と Break プロパティの違いで、これ以外に大きな違いはありません

メニュー項目の装飾

メニュー項目は、単純に選択するだけではなく
チェックボタンとしての役割を果たしたり、状況に応じて選択できなくする必要があります
これらの機能も、MenuItem クラスはプロパティとして提供しています

メニューにチェックを入れるには MenuItem.Checked プロパティを
メニューの有効/無効の選択には MenuItem.Enabled プロパティを
表示/非表示には MenuItem.Visible プロパティを使って制御します

public bool Checked {get; set;}
public bool Enabled {get; set;}
public bool Visible {get; set;}

これらのプロパティは、bool 値で機能の有効/無効を選択することができます
次章で説明するイベントと組み合わせれば、様々な使い方ができるようになるでしょう
また、MenuItem.RadioCheck プロパティを使えば
チェックマークの代わりにオプションボタンを表示することができます

public bool RadioCheck {get; set;}

オプションボタンを表示するなら true を、そうでなければ false を指定します

```
using System.Windows.Forms;

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

 public WinMain() {
  MainMenu mm = new MainMenu();
  MenuItem[] mi = {
   new MenuItem("Rena") , new MenuItem("Yuki") ,
   new MenuItem("Mimi")
  };
  mi[0].Checked = true;
  mi[1].Enabled = false;

  mi[2].RadioCheck = true;
```

```
   mi[2].Checked = true;

   mm.MenuItems.Add("Kitty on your lap" , mi);

   Menu = mm;
  }
 }
```

図のメニューポップアップは、上からチェック、無効、オプションボタンチェックの項目です

前のページへ　戻る　次のページへ