Chapter7_Windowsアプリケーションの作成

冒頭の記述

7-1 消費税計算機～アプリケーションで計算処理～

コントロール

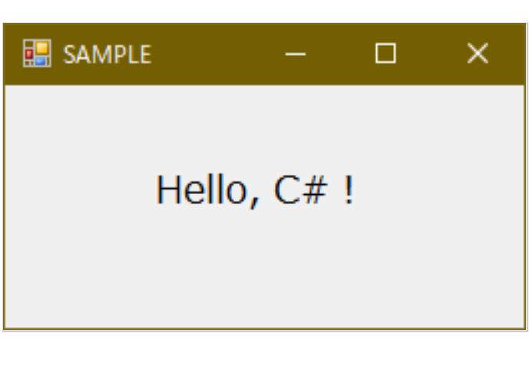
◆Label

```
public class Label : System.Windows.Forms.Control
```

一般的なラベル。プロパティ「Text」の値が表示される。

Label クラス (System.Windows.Forms)

図7-1 Label



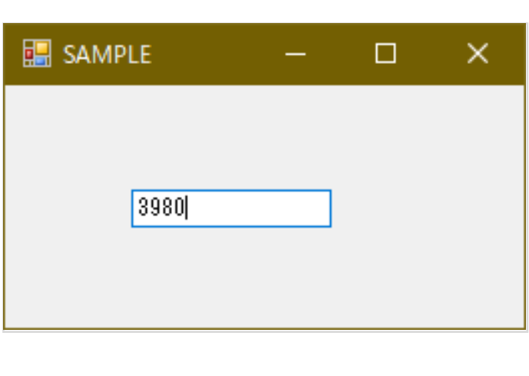
◆TextBox

```
public class TextBox : System.Windows.Forms.TextBoxBase
```

テキストを入力することができる。プロパティ「Enabled」の値を変更することで入力可否を変えることができる（True：入力可能　False：入力不可能）。

TextBox クラス (System.Windows.Forms)

図7-2 TextBox



イベント

◆Control.Click イベント

```
public event EventHandler Click
```

コントロールがクリックされたときに発生するイベント。

Control.Click イベント (System.Windows.Forms)

処理内容

◆文字列を数値に変換するメソッド（Int32.TryParseメソッド）

```
public static bool TryParse (string s, out int result)
```

文字列sを数値に変換することを試みる。変換に成功した場合は変換した値をresultに代入する。変換に失敗した場合は0をresultに代入する。戻り値は変換に成功したかどうかの真偽値（true：成功　false：失敗）。変換に失敗しても例外をスローしない。

※Int32のエイリアス（別名）がint。Int32イコールintと考えてよい。

Int32.TryParse メソッド (System)

Types - C# language specification

例7-1 Int32.TryParseメソッドの例

```
static void Main(string[] args)
{
      string str1 = "198";
      string str2 = "一九八";

      if (int.TryParse(str1, out int result1))
      {
            Console.WriteLine(str1 + "の数値変換に成功しました。結果：" + result1);
      }
      else
      {
            Console.WriteLine(str1 + "の数値変換に失敗しました。結果：" + result1);
      }

      if (int.TryParse(str2, out int result2))
      {
            Console.WriteLine(str2 + "の数値変換に成功しました。結果：" + result2);
      }
      else
      {
            Console.WriteLine(str2 + "の数値変換に失敗しました。結果：" + result2);
      }
}
```

例7-1の実行結果

```
198の数値変換に成功しました。結果：198
一九八の数値変換に失敗しました。結果：0
```

◆数値を文字列に変換するメソッド（Int32.ToString メソッド）

```
public override string ToString ()
```

数値を文字列に変換する。戻り値は変換結果の文字列。

Int32.ToString メソッド (System)

例7-2 Int32.ToStringメソッドの例

```
static void Main(string[] args)
{
      int price = 3980;
      Console.WriteLine("私の靴は" + price.ToString() + "円です。");
}
```

例7-2の実行結果

```
私の靴は3980円です。
```

7-2 電話帳アプリ～ファイルからデータを取得する～

コントロール

◆ListBox

```
public class ListBox : System.Windows.Forms.ListControl
```

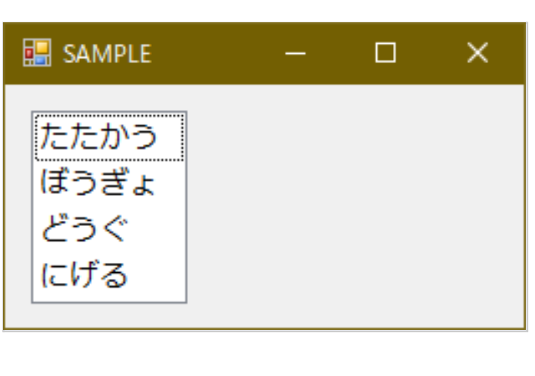
複数の項目を一覧表示するリスト。プロパティ「Items」はコレクションになっており、文字列をAddメソッドで追加すると項目が増える。プロパティ「Text」は現在選択している項目のテキストを表す。

ListBox クラス (System.Windows.Forms)

ListBox.Items プロパティ (System.Windows.Forms)

ListBox.Text プロパティ (System.Windows.Forms)

図7-3 ListBox



イベント

◆ListBox.SelectedIndexChanged

```
public event EventHandler SelectedIndexChanged
```

選択されている項目が変わった時に発生するイベント。

ListBox.SelectedIndexChanged イベント

ListBox.SelectedIndex プロパティ (System.Windows.Forms)

処理

◆キーと値をペアで格納できる型（KeyValuePair<TKey,TValue> 構造体）

```
public struct KeyValuePair<TKey,TValue>
```

KeyValuePairはキーと値をペアで格納できる構造体。foreach文を使ってDictionary型のキーと値を取り出す受け口として使われることが多い。プロパティ「Key」でキーを取得でき、プロパティ「Value」で値を取得できる。

KeyValuePair<TKey,TValue> 構造体

KeyValuePair<TKey,TValue>.Key プロパティ (System.Collections.Generic)

KeyValuePair<TKey,TValue>.Value プロパティ (System.Collections.Generic)

◆ファイルのデータを読み込む（StreamReaderクラス）

ファイルのデータを読み込むにはStreamReaderクラスを使う。StreamReaderクラスのコンストラクタは例外をスローする可能性があるので、try-catchするべき。StreamReaderをインスタンス生成する場合はusingステートメントを使う。usingステートメントを使うことで、usingステートメントのブロックの処理が終わった後、自動的にファイルを閉じてくれる。プロパティ「EndOfStream」でファイルを最後まで読み終えたかを取得できる（true: 読み終えた、false: 読み終えていない）。ファイルパスを指定する場合は「@」を文字列の先頭につけて、エスケープシーケンスを無効にする方法が定石になっている。

StreamReader クラス (System.IO)

StreamReader.EndOfStream プロパティ(System.IO)

例外処理 - C# によるプログラミング入門

例7-3 StreamReaderクラスの例

```
using System;

namespace Example
{
      class Program
      {
            static void Main(string[] args)
            {
                  int row = 1;
                  try
                  {
                        using (System.IO.StreamReader file =
                              new System.IO.StreamReader(@"..\..\..\sample.txt"))
                        {
                              while (!file.EndOfStream)
                              {
                                    Console.WriteLine(row + "行目：" + file.ReadLine());
                                    row++;
                              }
                        }
                  }
                  catch (Exception)
                  {
                        // 例外のインスタンスを使わないときはcatch文でインスタンス記述を省略できる
                        // 使うのなら Exception eと書く。
                        Console.WriteLine("ファイル読み込みでエラー発生");
                  }
            }
      }
}
```

※ファイルパスは環境によって異なる

例7-3の実行結果

```
1行目：たかしのこうげき！
2行目：たかしのこうげきは当たらなかった。
3行目：たかしのこうげき！
4行目：たかしはこうげきの時に転んだ。
```

例7-3で利用したsample.txt

```
たかしのこうげき！
たかしのこうげきは当たらなかった。
たかしのこうげき！
たかしはこうげきの時に転んだ。
```

◆文字列を分割する（String.Split メソッド）

```
public string[] Split (params char[] separator)
```

指定された区切り文字に基づいて文字列を部分文字列に分割するメソッド。戻り値はstring型の配列。

String.Split メソッド (System)

例7-4 String.Splitメソッドの例

```
static void Main(string[] args)
{
      // charsは「,」と「、」で初期化された配列
      char[] chars = { ',', '、'};
      string str1 = "Hello, World!";
      string str2 = "こんにちは、世界。Hello, World!";

      foreach(string str in str1.Split(chars))
      {
            Console.Write(str + "\t");
      }
      Console.WriteLine();

      foreach (string str in str2.Split(chars))
      {
            Console.Write(str + "\t");
      }
      Console.WriteLine();
}
```

例7-4の実行結果

```
Hello    World!
こんにちは       世界。Hello     World!
```

操作

◆テキストファイルを追加する

プロジェクトにテキストファイルを追加するには以下の操作を行う。

プロジェクトを右クリック ＞ 追加 ＞ 新しい項目 ＞ ウィンドウ左側メニューのVisual C#アイテム内の「全般」を選択 ＞ ウィンドウ中心にある一覧内のテキストファイルを選択 ＞ 名前を入力 ＞ 追加

用語

◆CSV

テキストをカンマ区切りで並べたデータ、またはそのファイル自体を指す。Comma-Separated Valuesの略。1行ずつをデータ区切りとする場合が多い。カンマではなくタブでデータを区切った形式のTSV（Tab-Separated Values）があり、TSVを含めてCSVと呼ぶ場合がある。

7-3 天気予報アプリ～ウェブから情報を取得する～

コントロール

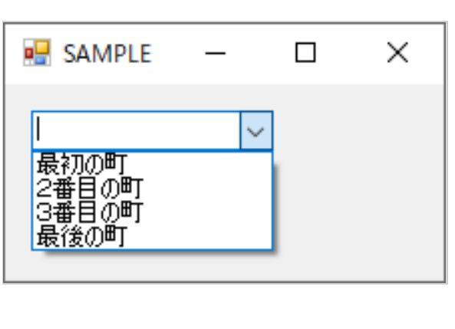
◆ComboBox

```
public class ComboBox : System.Windows.Forms.ListControl
```

プルダウン形式で複数の項目を表示できるコンボボックス。プロパティ「Items」はコレクションになっており、文字列をAddメソッドで追加すると項目が増える。

ComboBox クラス(System.Windows.Forms)

図7-4 ComboBox



◆PictureBox

```
public class PictureBox :
System.Windows.Forms.Control,System.ComponentModel.ISupportInitialize
```

イメージを表示するためのコントロール。プロパティ「SizeMode」を変更すると画像のサイズの合わせ方を変更することができる。プロパティ「ImageLocation」はPictureBoxに表示するイメージのパスまたはURLを表す。

PictureBox クラス(System.Windows.Forms)

PictureBox.SizeMode プロパティ (System.Windows.Forms)

PictureBoxSizeMode 列挙型 (System.Windows.Forms)

PictureBox.ImageLocation プロパティ (System.Windows.Forms)

図7-5 PictureBox

イベント

◆ComboBox.SelectedIndexChanged

```
public event EventHandler SelectedIndexChanged
```

選択されている項目が変わった時に発生するイベント。

ComboBox.SelectedIndexChanged イベント (System.Windows.Forms)

処理

◆リクエスト・レスポンスを送受信する（HttpClient クラス）

```
public class HttpClient : System.Net.Http.HttpMessageInvoker
```

リクエストやレスポンスを送受信するにはHttpClientクラスを利用する。

```
public System.Threading.Tasks.Task<string> GetStringAsync (string requestUri)
```

HttpClientクラスのGetStringAsyncメソッドを使うことで、引数に指定したURIにGETリクエストを送信することができる。戻り値の型はTask<string>となり、本来非同期処理を扱うものだがTask<TResult>クラスはプロパティ「Result」を併用することで同期処理の様に扱うことができる。詳しくは非同期処理について学ぶと良い。

HttpClient クラス (System.Net.Http)

HttpClient.GetStringAsync メソッド

Task<TResult> クラス (System.Threading.Tasks)

Task<TResult>.Result プロパティ (System.Threading.Tasks)

用語

◆JSON

もともとはJavaScriptのオブジェクトをデータでやり取りするための記述方法。現在では幅広い言語で扱うことができる。JSON自体はテキストデータなので、慣れている人が中身をみれば専用のソフトなどを使わなくてもある程度理解できる。