C# サンプルコード

[C#][言語まとめ C#]

・C# Document http://msdn.microsoft.com/ja-jp/library/kx37x362.aspx
・.Net Class Library Reference http://msdn.microsoft.com/ja-jp/library/ms229335.aspx

ファイル

■ファイル名を抽出

```
var filename = Path.GetFileName(file);
```

■ファイルを書く

```
using (var writer = new StreamWriter(Path.Combine(outPath, "Hoge.csv")))
{
    writer.WriteLine("Something to write.");
}
```

■エンコーディングを指定してファイルを読む

```
using System;
using System.IO;
namespace ConsoleApplication1
{
     class Program
    {
        static void Main(string[] args)
        {
            string path = @"c:\work\test.csv";
                                    using (var reader = new StreamReader(path,
System.Text.Encoding.GetEncoding("shift_jis")))
            {
                string line = null;
                while ((line = reader.ReadLine()) != null)
                {
                    Console.WriteLine(line);
                }
            }
        }
    }
}
```

■ CSV ファイルを解析 (TextFieldParser)

・http://msdn.microsoft.com/ja-jp/library/x710fk43.aspx
・便利だが、どんな挙動をするのかよくわからない (どこにドキュメントがあるの？)
    ・二重引用符で囲まれたフィールドでは、カンマはセパレータとしてではなく、値として判定してくれる (Excel 仕様 ?)
    ・なぜ VisualBasic 名前空間にある？

```
using System;
using Microsoft.VisualBasic.FileIO;

namespace ConsoleApplication1
{
     class Program
    {
        static void Main(string[] args)
        {
            string path = @"c:\work\test.csv";
                                        using (TextFieldParser parser = new
TextFieldParser(path,System.Text.Encoding.GetEncoding("shift_jis")))
            {
```

```
                parser.TextFieldType = FieldType.Delimited;
                parser.SetDelimiters(",");
                try
                {
                    while (!parser.EndOfData)
                    {
                        string[] fields = parser.ReadFields();
                        foreach (string field in fields)
                        {
                            Console.Write("[{0}]", field);
                        }
                        Console.WriteLine();
                    }
                }
                catch (MalformedLineException e)
                {
                    Console.WriteLine(e);
                    System.Environment.Exit(1);
                }
            }
        }
    }
}
```

■ UTF-8 エンコード BOM ありなし

・UTF-8 を GetEncode すると、BOM が付く。new System.Text.UTF8Encoding(false) で BOM なし

```
Encoding enc = null;
if(String.Equals(toEnc,"utf-8", StringComparison.OrdinalIgnoreCase)) {
    // BOM なし
    enc = new System.Text.UTF8Encoding(false);
}
else
{
    enc = Encoding.GetEncoding(toEnc);
}
```

■ファイルエンコード一括変換

```
public class FileEncodingConverter
{
    public FileEncodingConverter() {
        System.Text.Encoding.RegisterProvider(System.Text.CodePagesEncodingProvider.Instance);
    }

    /// <summary>
    /// 指定されたディレクトリのすべてのファイルのエンコードを変換して別のディレクトリに書き出す
    /// </summary>
    /// <param name="srcDir">変換元ディレクトリ</param>
    /// <param name="dstDir">変換先ディレクトリ</param>
    /// <param name="fromEnc">変換元エンコーディング</param>
    /// <param name="toEnc">変換後エンコーディング</param>
    public void convertDir(string srcDir, string dstDir, string fromEnc, string toEnc)
    {
        if (!Directory.Exists(dstDir))
        {
            Directory.CreateDirectory(dstDir);
        }

        var files = Directory.GetFiles(srcDir);
        foreach(var file in files)
        {
            var filename = Path.GetFileName(file);
            convert(file, Path.Combine(dstDir, filename), fromEnc, toEnc);
        }
    }

    /// <summary>
    /// 指定されたファイルのエンコードを変換して別ファイルに書き出す
    /// </summary>
    /// <param name="srcFile">変換元ファイル</param>
    /// <param name="dstFile">変換後ファイル</param>
    /// <param name="fromEnc">変換元エンコーディング</param>
```

```
        /// <param name="toEnc">変換後エンコーディング</param>
        public void convert(string srcFile, string dstFile, string fromEnc, string toEnc)
        {
            using(var writer = new StreamWriter(File.OpenWrite(dstFile),Encoding.GetEncoding(toEnc)))
            {
                foreach(var line in File.ReadLines(srcFile, System.Text.Encoding.GetEncoding(fromEnc)))
                {
                    writer.WriteLine(line);
                }
             }
        }
    }
```

設定

■ C# 設定情報を保存する

・C# 設定情報を保存する

LINQ

■ C# LINQ 使用例

・C# LINQ 使用例

非同期処理

■ C# 非同期処理から UI スレッドにアクセスし画面を更新する

・C# 非同期処理から UI スレッドにアクセスし画面を更新する

■ C# async と await の動作確認

・C# async と await の動作確認

WPF

・http://typea.info/blg/glob/wpf/

HttpClient

■同期通信

```
    public void GetLoginToken()
    {
        var param = new Dictionary<string, string>()
        {
            { "action", "query" },
            {"meta","tokens" },
            {"type","login" },
            {"format","json" },
        };

                                var          response          =          client.GetAsync($"{API_URL}?{new
FormUrlEncodedContent(param).ReadAsStringAsync().Result}").Result;
        if (response.StatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine("RES:" + response.ToString());
        }
    }
```