MSチャートとリフレクションの組み合わせ例

2019年08月05日
お仕事関係で調べた時のメモを自分用に貼っておく。

概要

あるクラスがあって、

- クラスの保持データをグラフで表示させる。
- クラスには随時手が入り、グラフ化したいデータを格納したフィールドは変化する。

な要件がある。

前者はMSチャートコンポーネントを使い、後者はリフレクションで対応する。

クラスの用意

MSチャートをいじるので、
using System.Windows.Forms.DataVisualization.Charting;
は必須。参照にも追加を忘れずに。

またリフレクションを使うので、
using System.Reflection;
も必要。

MSチャートを設定するメソッドSetLineBarChart()では□privateで型がDictionary<int,int>□名称が“ BarDatas”もしくは“LineDatas”で終わるフィールドをグラフ化対象としている。 名称が“ BarDatas”で終わるフィールドは棒グラフ□“LineDatas”で終わるフィールドは折れ線グラフ、で描画する。

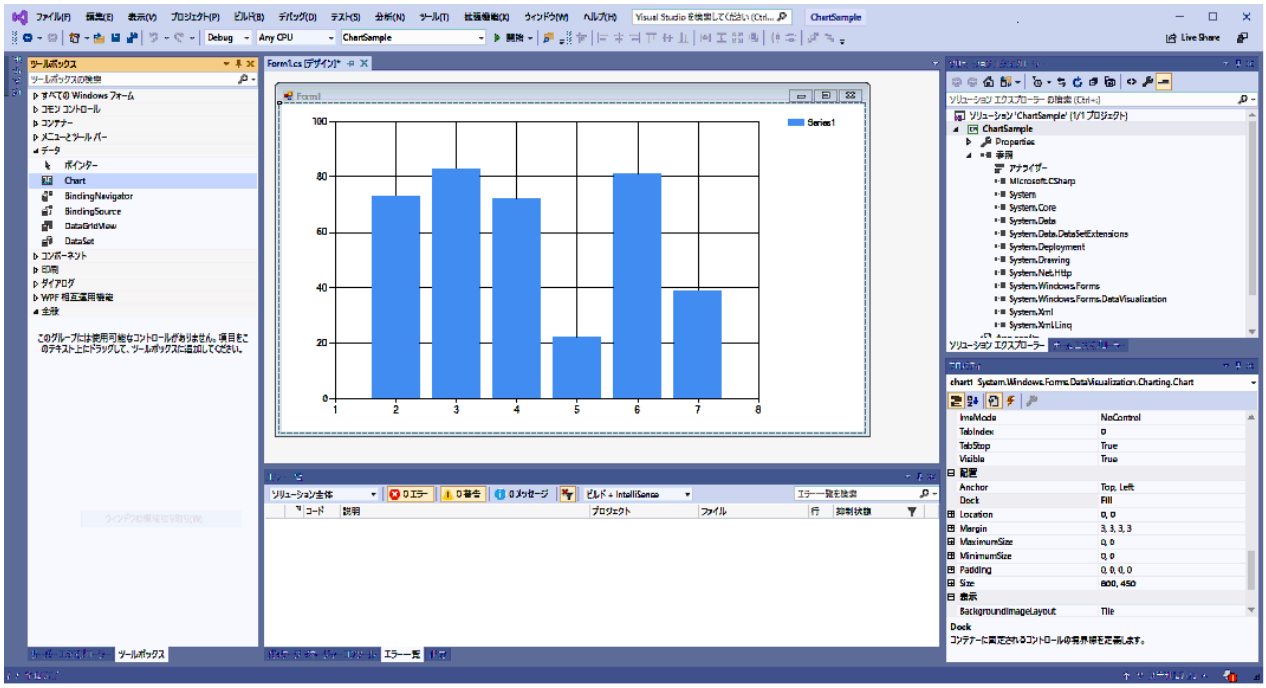
```
using System;
using System.Collections.Generic;
using System.Text.RegularExpressions;
using System.Reflection;
using System.Windows.Forms.DataVisualization.Charting;

namespace ChartSample
{
    public class DataSource
    {
        Dictionary<int, int> Sample1ChartBarDatas = new Dictionary<int,
int>();
        Dictionary<int, int> Sample2ChartLineDatas = new Dictionary<int,
int>();

        public DataSource()
        {
```

```
            Random r = new Random();

            // このサンプルではランダムな値を入れているが実際は他のメソッドで計算された値が入ることになる
            for (int x = 0; x < 10; x++)
            {
                Sample1ChartBarDatas.Add(x, Convert.ToInt32(r.NextDouble() * 61));
                Sample2ChartLineDatas.Add(x, Convert.ToInt32(r.NextDouble() * 51) + 40);
            }
        }
        public void SetLineBarChart(Chart chart)
        {
            // MSチャートの初期化
            chart.ChartAreas.Clear();
            chart.Series.Clear();
            chart.ChartAreas.Add(new ChartArea("LineBar")); //描画エリアの名称は"LineBar"

            //このクラス自身のPublicではないフィールド一覧を取得
            FieldInfo[] fdinfo = GetType().GetFields(BindingFlags.NonPublic | BindingFlags.Instance );

            foreach (var item in fdinfo)
            {
                Series series = null;

                //目的の型でなかった場合はスキップ
                if (item.FieldType != typeof(Dictionary<int, int>)) continue;

                //棒グラフ対象のフィールド
                if (Regex.IsMatch(item.Name, @"BarDatas$"))
                {
                    series = new Series(item.Name);

                    series.ChartType = SeriesChartType.Column;
                    series.XValueType = ChartValueType.Int32;
                    series.YValueType = ChartValueType.Int32;
                    series.IsValueShownAsLabel = true;
                }

                //折れ線グラフ対象のフィールド
                if (Regex.IsMatch(item.Name, @"LineDatas$"))
                {
                    series = new Series(item.Name);

                    series.ChartType = SeriesChartType.Line;
                    series.XValueType = ChartValueType.Int32;
                    series.YValueType = ChartValueType.Int32;
```

```
                    series.IsValueShownAsLabel = true;
                    series.MarkerStyle = MarkerStyle.Circle;
                    series.MarkerSize = 12;
                }

                //描画対象じゃないようなのでスキップ
                if (series == null) continue;

                //グラフ化するデータを登録
                foreach (var dsitem in (Dictionary<int,
int>)item.GetValue(this))
                {
                    series.Points.Add(new DataPoint(dsitem.Key,
dsitem.Value));
                    series.ChartArea = "LineBar";
                }
                chart.Series.Add(series);
            }
        }
    }
}
```

チャートの用意

WinFormにMSチャートを1つ張り付ける。



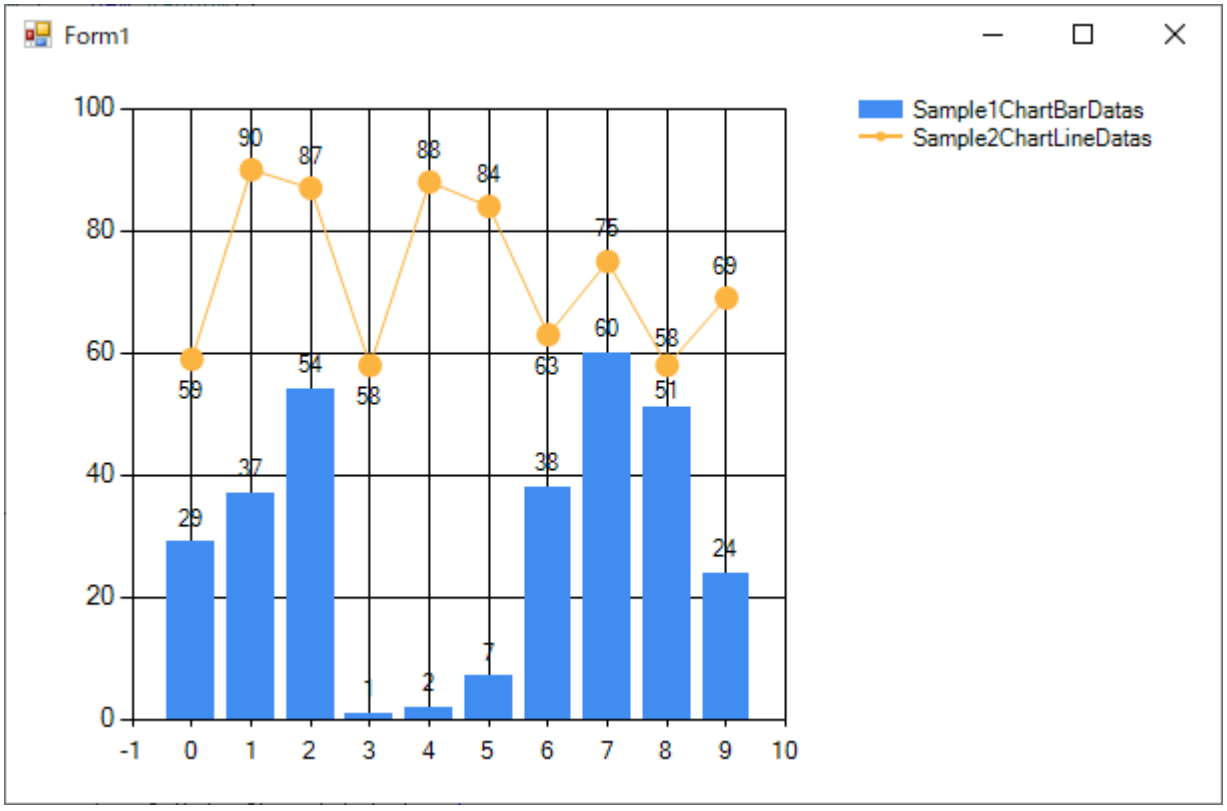
このチャートに対してデータを設定してグラフ描画させる。

```
using System;
```

```
using System.Windows.Forms;

namespace ChartSample
{
    public partial class Form1 : Form
    {
        DataSource ds = new DataSource();
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            ds.SetLineBarChart(this.chart1); // Form1に張り付けたチャートのインスタンスがchart1
        }
    }
}
```

実行結果

実行結果はこんな感じ。

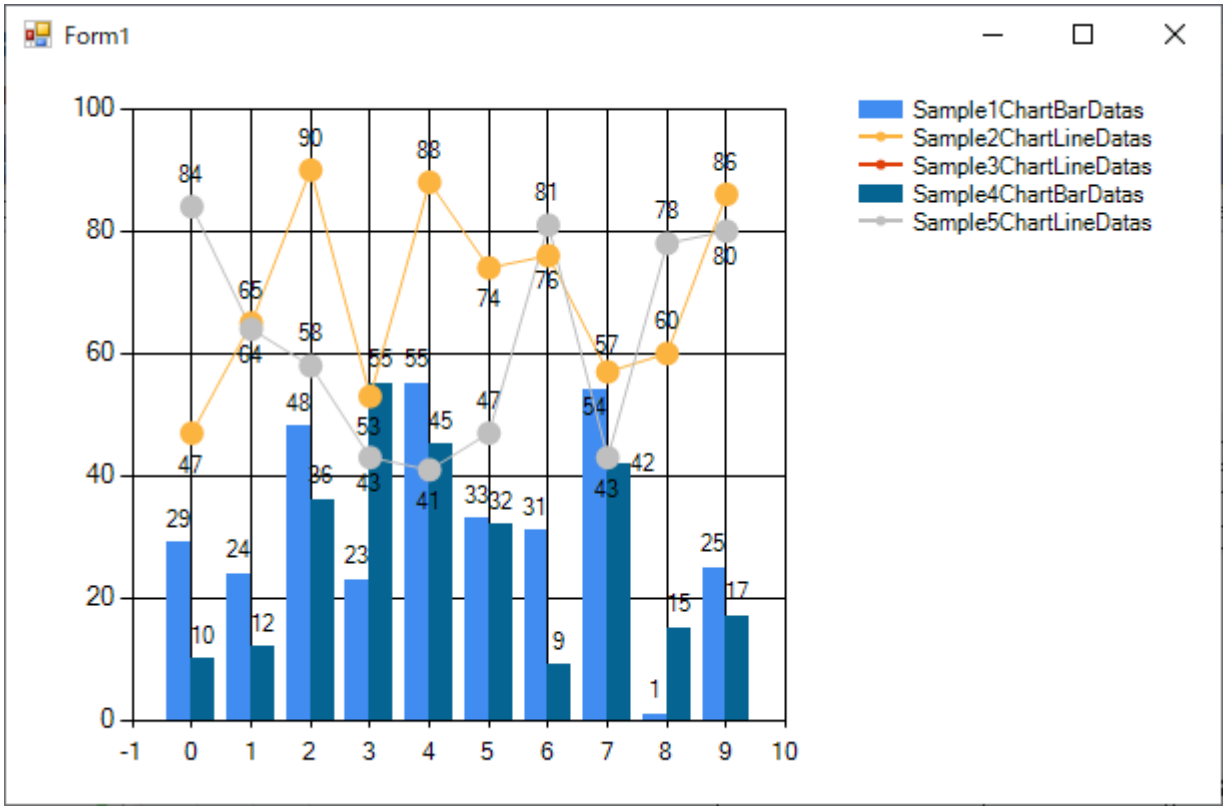


フィールド増加に対応

こんな感じにフィールドを増やしてみる。

```
        Dictionary<int, int> Sample1ChartBarDatas = new Dictionary<int,
int>();
        Dictionary<int, int> Sample2ChartLineDatas = new Dictionary<int,
int>();
        Dictionary<int, int> Sample3ChartLineDatas = new Dictionary<int,
int>();
        Dictionary<int, int> Sample4ChartBarDatas = new Dictionary<int,
int>();
        Dictionary<int, int> Sample5ChartLineDatas = new Dictionary<int,
int>();

        public DataSource()
        {
            Random r = new Random();

            for (int x = 0; x < 10; x++)
            {
                Sample1ChartBarDatas.Add(x, Convert.ToInt32(r.NextDouble() *
61));
                Sample2ChartLineDatas.Add(x, Convert.ToInt32(r.NextDouble()
* 51) + 40);
                Sample4ChartBarDatas.Add(x, Convert.ToInt32(r.NextDouble() *
61));
                Sample5ChartLineDatas.Add(x, Convert.ToInt32(r.NextDouble()
* 51) + 40);
            }
        }
```

増えたフィールドの描画も行われる□Sample3ChartLineDatasには値が設定されていないのでグラフは描かれないが、凡例は描画されている。



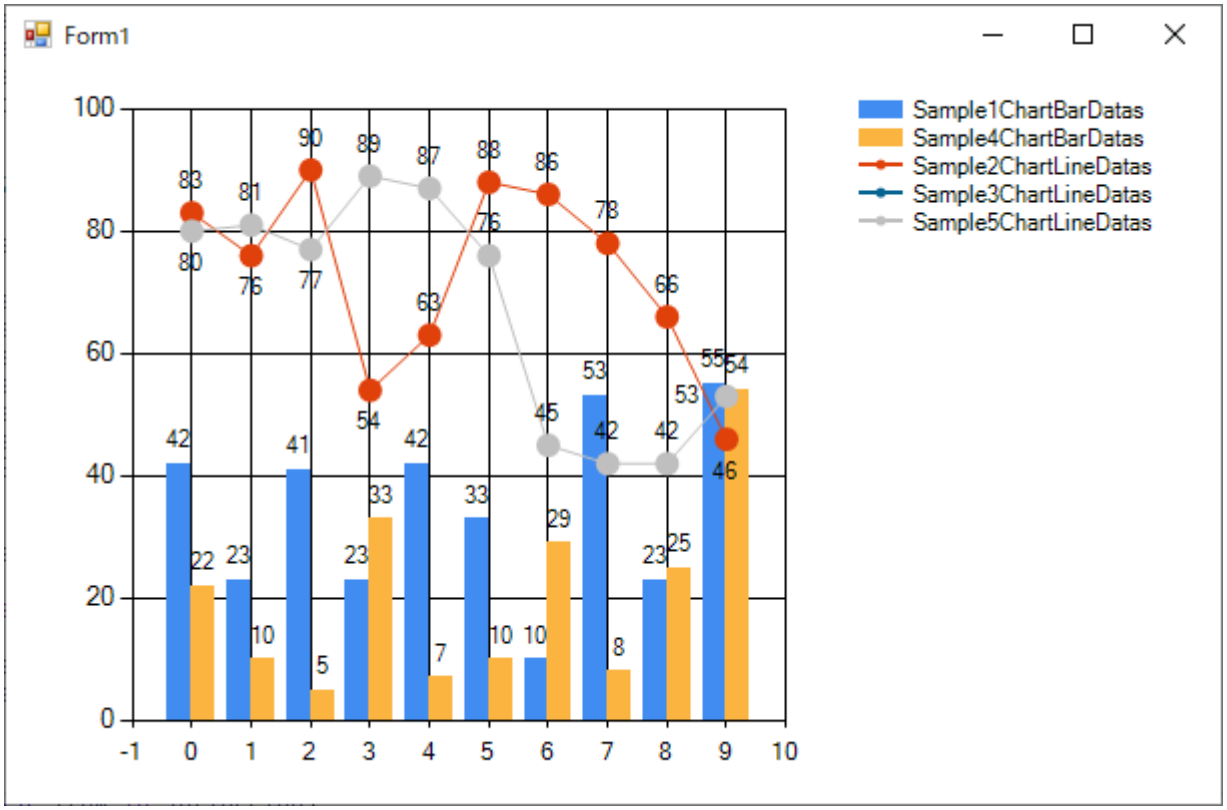
凡例の並びを変える

凡例の並びを変えるなら、chart.Series への追加順を制御するのが楽。

```
        public void SetLineBarChart(Chart chart)
        {
            FieldInfo[] fdinfo = GetType().GetFields(BindingFlags.NonPublic
| BindingFlags.Instance);

            chart.ChartAreas.Clear();
            chart.Series.Clear();
            chart.ChartAreas.Add(new ChartArea("LineBar"));

            List<Series> SeriesLine = new List<Series>();

            foreach (var item in fdinfo)
            {
                Series series = null;

                if (item.FieldType != typeof(Dictionary<int, int>))
continue;

                if (Regex.IsMatch(item.Name, @"BarDatas$"))
                {
                    series = new Series(item.Name);

                    series.ChartType = SeriesChartType.Column;
                    series.XValueType = ChartValueType.Int32;
```

```
                    series.YValueType = ChartValueType.Int32;
                    series.IsValueShownAsLabel = true;

                    //棒グラフを先に登録
                    chart.Series.Add(series);
                }

                if (Regex.IsMatch(item.Name, @"LineDatas$"))
                {
                    series = new Series(item.Name);

                    series.ChartType = SeriesChartType.Line;
                    series.XValueType = ChartValueType.Int32;
                    series.YValueType = ChartValueType.Int32;
                    series.IsValueShownAsLabel = true;
                    series.MarkerStyle = MarkerStyle.Circle;
                    series.MarkerSize = 12;
                    SeriesLine.Add(series);
                }

                if (series == null) continue;

                foreach (var dsitem in (Dictionary<int,
int>)item.GetValue(this))
                {
                    series.Points.Add(new DataPoint(dsitem.Key,
dsitem.Value));
                    series.ChartArea = "LineBar";
                }
            }

            //折れ線グラフを後に登録
            foreach(var item in SeriesLine)
            {
                chart.Series.Add(item);
            }
        }
```

上記例だと、棒グラフの後に折れ線グラフの凡例が表示される。



Windows, MSチャート, ms-chart, リフレクション, refrection