電卓DLLの使い方

CORESOFT

2021年2月18日

目次

リンクテキストはCtrキーを押しながらマウスでクリックして下さい。

- 概要 1
- 画面 2
- 使用例
  - 入力と表示の様子 3
  - クリア 4
  - 経過（確定入力）編集 5
- 電卓Dllの組み込み方 6
  - ツールボックスへの組み込み 7
- 組み込みサンプル 8
  - 共通の指定 TextBoxの場合 9
  - GrapeCityのGcSpreadGridの場合 10
  - InfragisticsのXamDataGridの場合 12
  - WPF既定のDataGridの場合 13

- 変更履歴 14

概要

CS_CalculatorはWPFのViewで使用するDLLです

基本的な仕様は市販されている電卓（実機）やWindowsに付属する電卓アプリに合わせますが

例えば演算子と数値を入力して＝（Enter）キーの押下を繰り返すと繰り返し演算になりますが、電卓Dllはマウスとキーボードの持ち替えの煩雑さを減らす観点から＝（Enter）キーの2回目の押下でDllを割り付けたフィールドに値を戻す

など入力補助機構としての使いやすさを優先する仕様にしています。

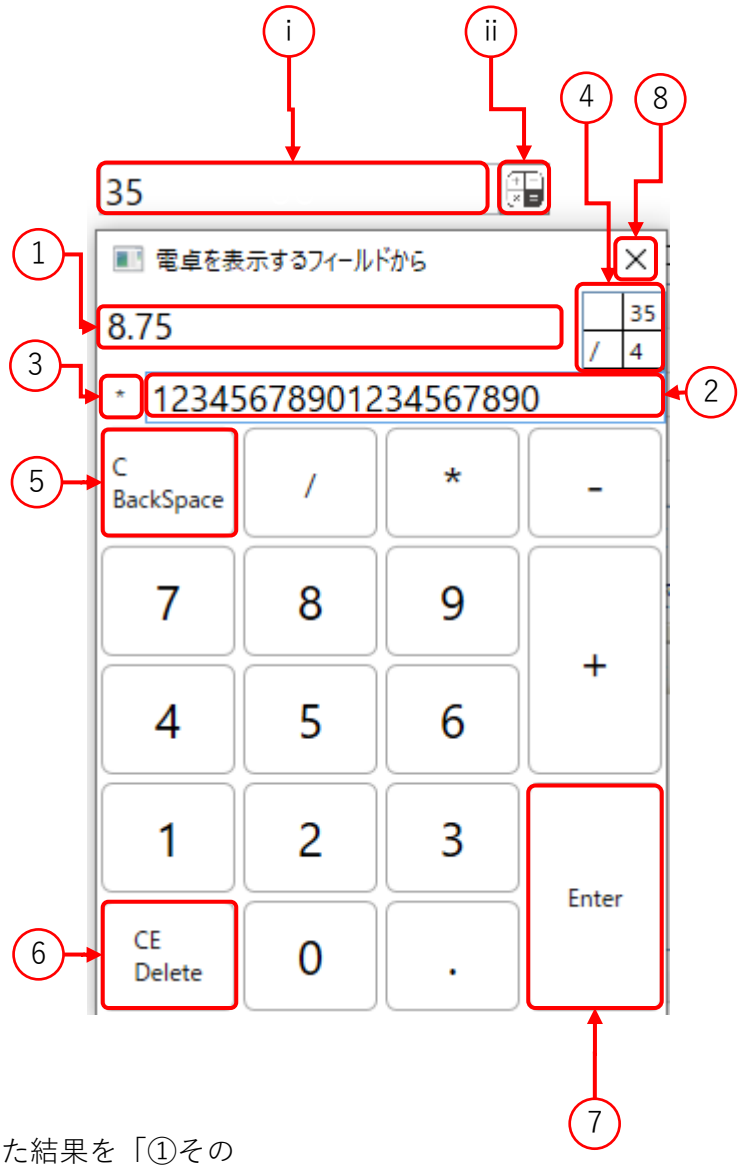
このDllは電卓ですので演算子の優先順位に沿った計算ではなく、それまでの確定値に対してEnterもしくは次の演算子を入力した時点での演算を行います。

例えば1+2*3は四則演算の場合、1に対して2＊3＝6の加算ですので答えは7になりますが、電卓処理の場合、1＋2＝3という演算結果に続けて＊3ですので答えは9になります。

画面

最も基本的なTextBoxに紐づけた場合で電卓の使い方を説明します。

A) 呼び出し元
 i.電卓に紐づけられたフィールド
 ii.電卓を表示するボタン



B)呼び出された電卓
 ① その時点の計算結果
 ② 入力値
 ③ 演算子
 ④ 経過
  次の演算子　もしくは
  Enterキーを押下したタイミングで
  その時点の入力値と演算子を配列に格納し、
  「①その時点の計算結果」を更新します。
  経過の配列に入力値と演算子を格納すると
  入力値のフィールドは空になり
  この状態でEnterキー押下
  もしくは⑤クローズボタンのクリックで
  ダイアログを閉じて①の演算結果を
  ｉの電卓を紐づけたフィールドに転記します。

表示されているボタンは主にNumPadと

キーボード上の数値や演算子に紐づけています。

それ以外はWindowsに付属する電卓に合わせ

 ⑤ C：最後の入力から1文字づつ消去はBackSpace

 ⑥ CE：全消去はDeleteキーに割り付けています

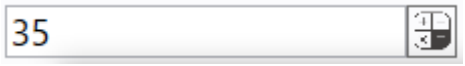
 ⑦ Enter;＝に該当し、その時点までの入力で計算した結果を「①その時点の計算結果」に表示します。
  もう一度Enterを押下する事で電卓ダイアログを閉じ、計算結果を呼び出し元のフフィールドに書き込みます。

 ⑧ クローズボックス　ダイアログを閉じ、「①その時点の計算結果」を呼び出し元のフィールドに転記します。

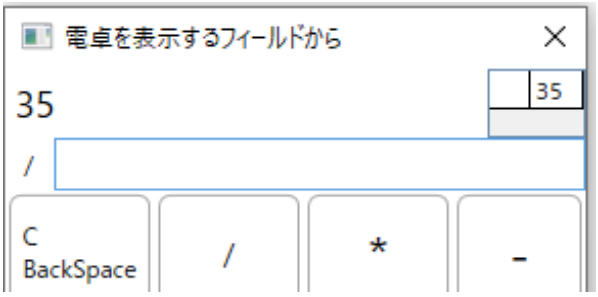
使用例

入力と表示の様子

1. ｉ：電卓に紐づけられたフィールドの入力値を３５にします



2. ii：電卓を表示するボタンをクリック
   もしくは数値に続いて四則演算子を入力すると電卓ダイアログを表示します。
   (表示例では / を入力しました)
   キーボードからマウスへの持ち替えの煩わしさを削減する工夫です。



「①その時点の計算結果」にフィールドに入力されていた値、
「③演算子」には数値に続いて入力した演算子(/)が転記されます。
「④経過」には元のフィールド値35が演算子無しで計算の開始値として記録されます。
※フィールドの入力無しでもiiのボタンをクリックすれば初期化された状態で電卓ダイアログが表示されます。

3. 続いてキーボードから数字の「4」を入力（もしくは4ボタンを押下）すると②入力値に「４」が入力されます。

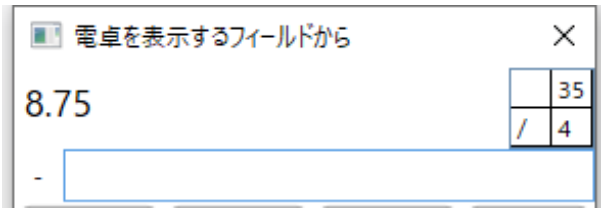
クリア

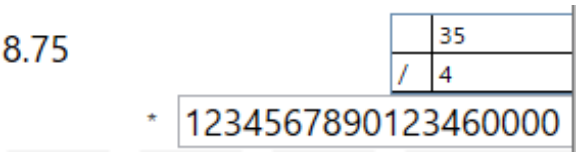
1. 次の演算子「*」を入力した時点で「④経過」に　演算子/ と入力値4　をペアで記憶します。

   続いて35/4＝8.75を算出し「①その時点の計算結果」を更新。

   次の演算子「*」をセットし「②入力値」を空にして数値の入力を待ちます。

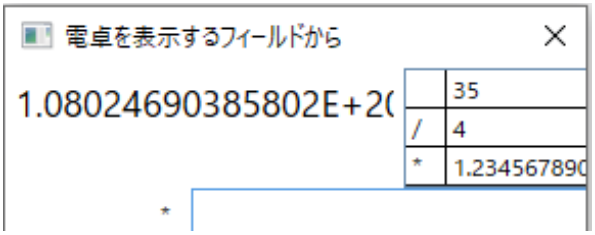
   演算子「-」と確定値「4」をペアで記憶します。





2. （わざと桁あふれさせるサンプルですが）
   12345678901234567890　と入力します。

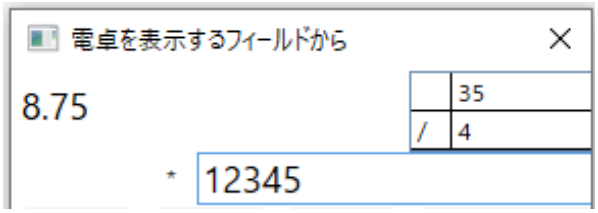
   16桁以上の数値は指数表示になります。



3. Cボタンクリック：BackSpaceキーの押下で直前の入力値を「④経過」リストから「②入力値」枠に戻します。

   16桁以上は桁あふれが発生し、最後の7890が消えてしまいますが計算の経過を記録してますので、その直前までは正常な計算値に復元できます。

4. 更にCボタンクリック：BackSpaceキーの押下を続けると「②入力値」枠にある数値の消去を続けます（12345になるまで消去）Enter押下で桁あふれが無い値に再計算されます。



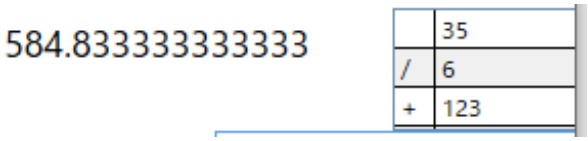
5. もう一度Cボタンクリック：BackSpaceキーの押下すると「②入力値」枠にある数値をすべて消去します。
   1. 経過に記憶された1演算分の入力を消去すると、その前の入力値と演算子が入力枠に戻されます。計算の開始値を消去しつくすまでクリアの操作を続けることができます

経過（確定入力）編集

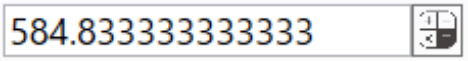
1. 一旦　/　4　のまま続けて　＋123　、+456　と入力してEnter.。

2. / 4 の列で　「4」の枠をクリックして下さい。
「④経過」は演算子も値も直接入力可能なリストです。
※演算子以外の文字、数値と小数点以外の入力が有った場合はエラーメッセージを表示して元の確定値に戻します。

3. 値を 6 に変更して Enterを押下すると、それ以降の値も含め再計算が行われます。



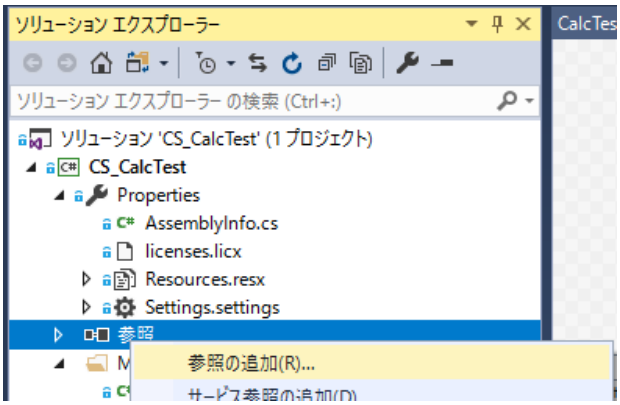
4. 再度Enter押下で計算結果を電卓が紐づけられたフィールドに持ち帰ります。



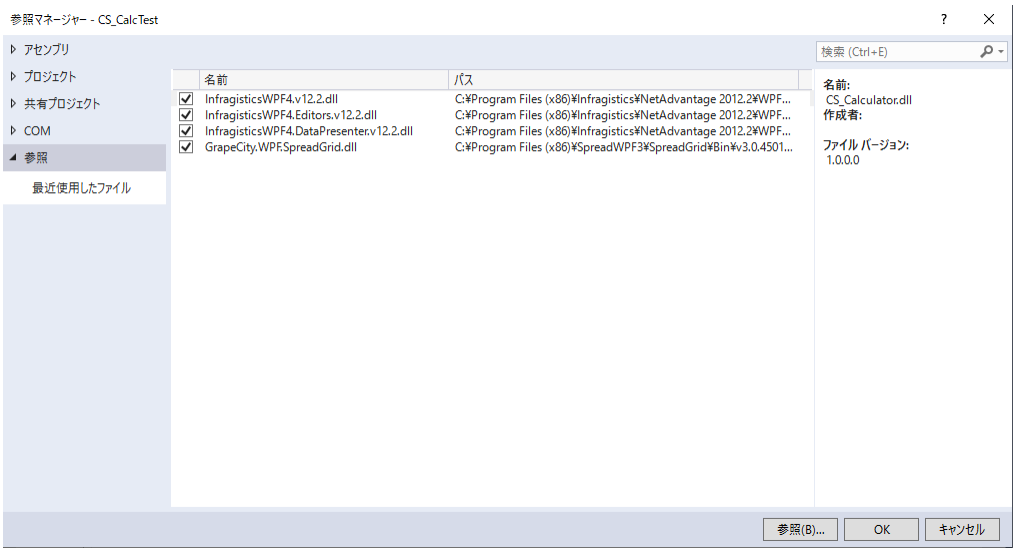
電卓Dllの組み込み方

1. CS_Calculator.zipを解凍。

   まずプロジェクト内の.csソースからusingでDll内の機能や設定を参照できるようにします。



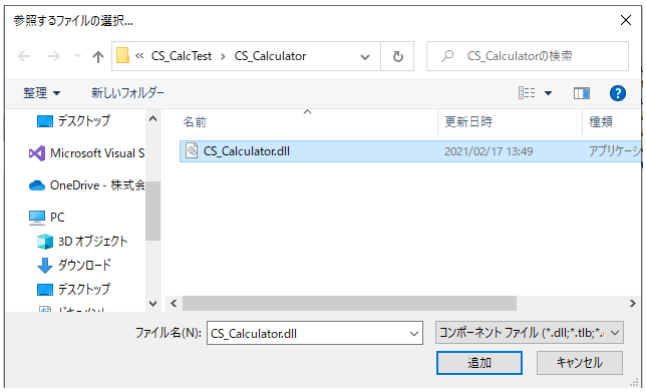
2. ソリューションエクスプローラで
   参照を右クリックし、参照の追加を追加。
   参照マネージャーの左ペインで「参照」を選択し、「参照(B)...」ボタンをクリック。

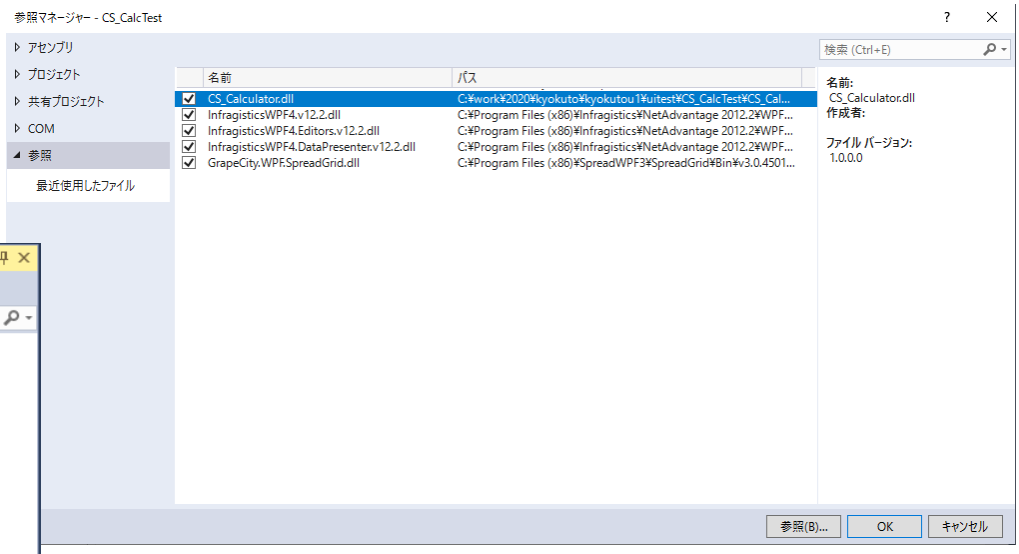


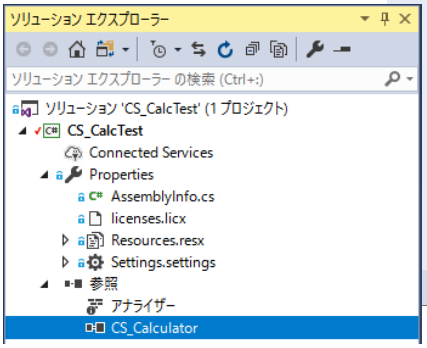
   ※この時点で目的の機能に該当するDllが有れば
   チェックを入れるだけでプロジェクトへの参照設定は完了です。

3. 表示されたファイル選択ダイアログで解凍したフォルダの中に有る「 CS_Calculator.dll 」を選択し、「追加」ボタンをクリック。

4. これで参照に選択したダイアログへのリンクが作成され、 usingでDllの機能が使える様になります。



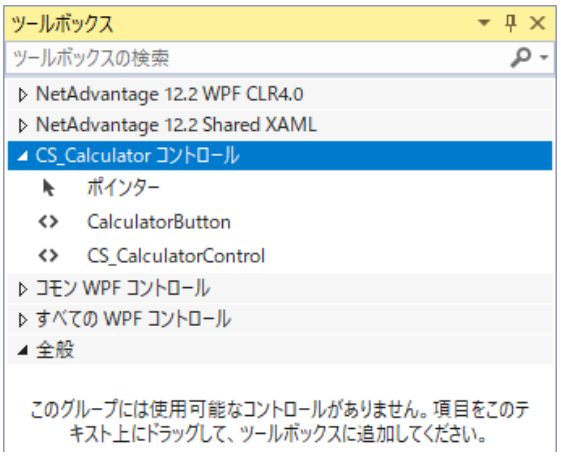




ツールボックスへの組み込み

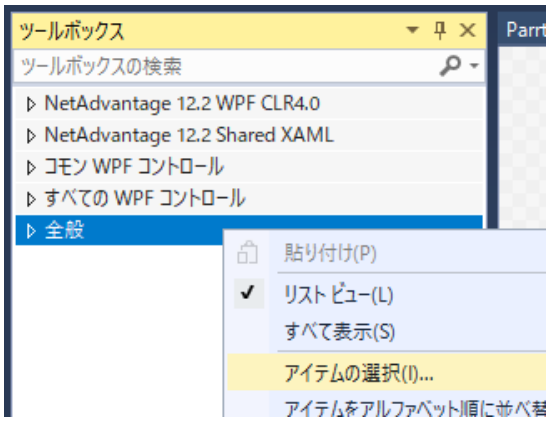
前ページのプロジェクトからの参照で目的のDllが指定できていればこのページの操作は不要です。

組み込めていない場合は以下の操作でDLLをツールボックスに読込みます。



1. CS_Calculator.zipを解凍。

2. WPFのプロジェクトでコントロールを組み込むXAMLを開く。

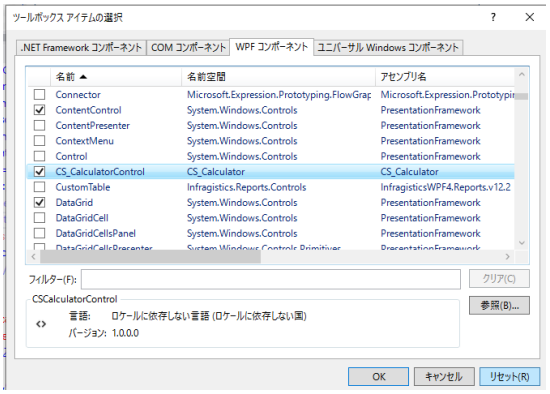


3. ツールボックスのペイン内を選択して右クリックして「アイテムの選択」ダイアログを開く。

   特定のタブを選択するとそこに組み込まれます。

4. 「ツールボックスアイテムの選択」ダイアログで「参照」ボタンをクリック。

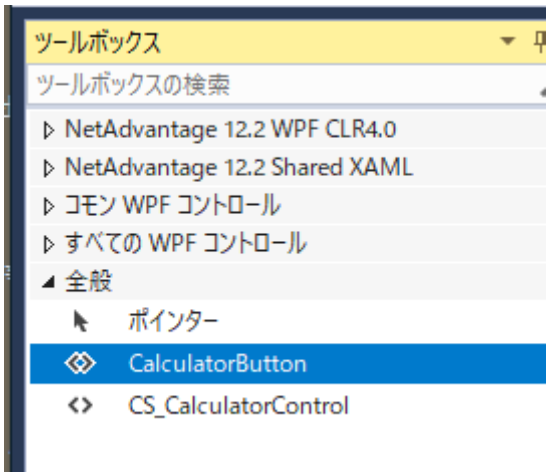
5. 「WPFコンポーネント」タグで目的のDLLが有ればチェックを入れて読み込み終了。



6. 無ければ「参照(B)...」ボタンをクリック。

7. 解凍した「CS_Calculator」フォルダの中から CS_Calculator.dll」を選択し、「開く」ボタンをクリック。

8. ツールボックスにCS_CalculatorControlとCalculatorButtonが追加されます。



① CS_CalculatorControlが電卓の本体でダイアログなどのViewに読み込ませて使用します。

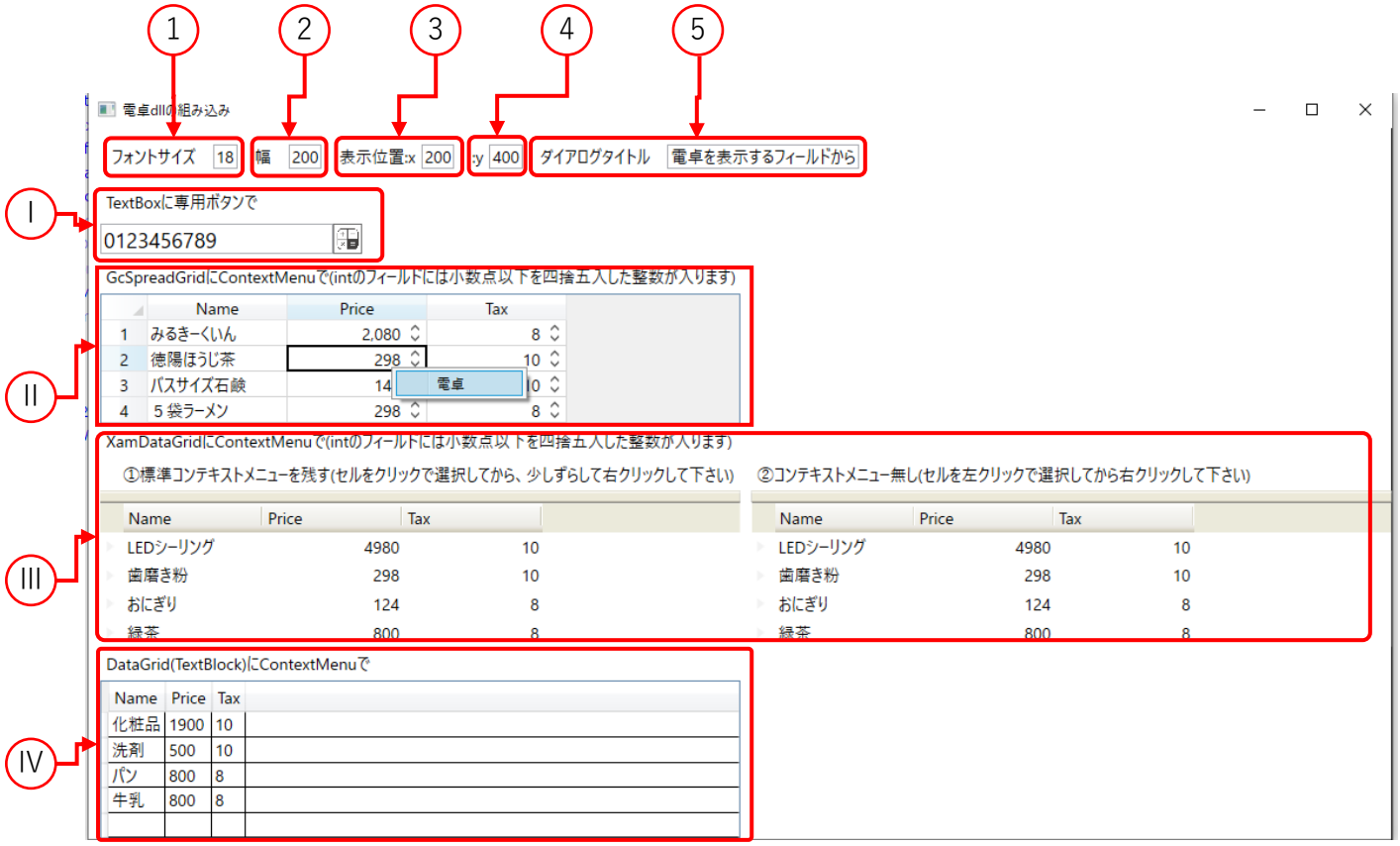
② CalculatorButtonはCS_CalculatorControlをダイアログに組み込んで表示し、このボタンのクラス変数に電卓の計算結果を指定します。

組み込みサンプル

組み込みサンプル「CS_CalcTest」で各コントロールへの組み込みを紹介します。
CS_CalcTestをビルドすると左の様な画面が表示されます。

一番上の行はプログラム中で電卓Dllに渡すパラメータ：サンプル中でのElementNameです。

① フォントサイズ　　：CalcTextFontSize
　このViewで計算結果を戻すフィールドのフォントサイズ
② 幅　　：CalcTexWidth
　このViewで計算結果を戻すフィールドの幅
③ 表示位置：X　　：CalcTextShowX
　Display上の表示X座標
④ (表示位置）:Y　　：CalcTextShowY
　Display上の表示Y座標
⑤ ダイアログタイトル　　：CalcTextDLogTitol
　表示されるダイアログのタイトル



組み込みサンプル

I. TextBox
II. GrapeCityのGcSpreadGrid
III. InfragisticsのXamDataGrid
IV. WPF既定のDataGrid

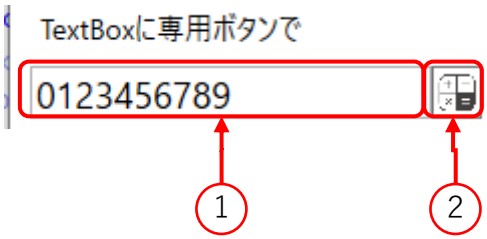
※サンプルソースを他のプロジェクトに読込んで起動画面を変更する場合は
App.xamlの<Application　タグで StartupUri="Views/ParrtsTestView.xaml" などに変更

共通の指定

① Viewの基底クラスへassemblyを指定します。
　具体的には<Window や<UserControl　タグに
　xmlns:cs_calculator="clr-namespace:CS_Calculator;assembly=CS_Calculator“
を追記してください。

```
<Window x:Class="CS_CalcTest.Views.CalcTestView"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:behaviors="http://schemas.microsoft.com/xaml/behaviors"
    xmlns:igWPF="http://schemas.infragistics.com/xaml/wpf"
    xmlns:sg="http://schemas.grapecity.com/windows/spreadgrid/2012"
    xmlns:cs_calculator="clr-namespace:CS_Calculator;assembly=CS_Calculator"
    mc:Ignorable="d"
    FontSize="14"
    WindowStartupLocation="CenterScreen"
    Topmost="true" Height="800" Width="1200"
    Title="電卓dllの組み込み" SizeToContent="WidthAndHeight">
```

TextBoxの場合



① 電卓で計算した結果のフィールドElement

```
<TextBox x:Name="Bt2Tbox“
        Width="{Binding ElementName=CalcTexWidth}“
        FontSize="{Binding ElementName=CalcTextFontSize}“
        Text="{Binding CalcResult ,Mode=TwoWay}"/>
```

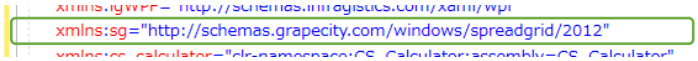
② 電卓を表示するボタンElement

```
<CS_Calculator:CalculatorButton
        x:Name="CalcCallBt"
        TargetTextBox="{Binding ElementName=Bt2Tbox}“
        ViewTitle="{Binding ElementName=CalcTextDLogTitol}"/>
```

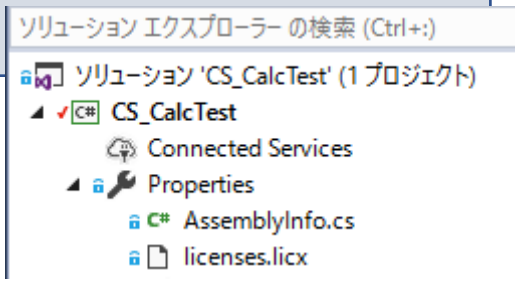
- 名称は「CalcCallBt」とします。
- TargetTextBoxに計算結果を書き写すフィールドを指定
- ViewTitleはダイヤログタイトルで指定しなければフィールド名を表示

GrapeCityのGcSpreadGridの場合

① Viewの基底クラスへassemblyを指定します。
具体的には<Window や<UserControl　タグに
xmlns:sg=http://schemas.grapecity.com/windows/spreadgrid/2012
を追記してください。



GcSpreadGridをビルドできない場合は
1. https://docs.grapecity.com/help/spread-wpf-gcspreadgrid-3/#sg-startaddcontrol.htmlのライセンスの組み込みを参照してlicenses.licxを手動作成し、
2. インストールフォルダの…¥Program Files (x86)¥SpreadWPF3¥SpreadGrid¥Bin¥v3.0.4501.2015からビルド番号を取得し
GrapeCity.Windows.SpreadGrid.GcSpreadGrid, GrapeCity.WPF.SpreadGrid, Version=3.0.4501.2015, Culture=neutral, PublicKeyToken=5c5c8ff7a6858ccc
をlicenses.licxに記入



② Element

```
<sg:GcSpreadGrid  Name="MyGsGrid“
          ItemsSource="{Binding GsGlist, Mode=OneWay}" >
   <sg:GcSpreadGrid.ContextMenu>
          <ContextMenu>
                <MenuItem Header="電卓"
                Command="{Binding MyGsGridContextMenuClick}"/>
          </ContextMenu>
   </sg:GcSpreadGrid.ContextMenu>
</sg:GcSpreadGrid>
```

GcSpreadGridにContextMenuで(intのフィールドには小数点以下を四捨五入した整数が入ります)

	Name	Price	Tax
1	みるきーくいん	2,080	8
2	徳陽ほうじ茶	298	10
3	バスサイズ石鹸	14	0
4	5袋ラーメン	298	8

電卓

③ コンテクストメニューからViewModelへのBind

```
      private ViewModelCommand _MyGsGridContextMenuClick;

      public ViewModelCommand MyGsGridContextMenuClick {
          get {
              if (_MyGsGridContextMenuClick == null)
              {
                  _MyGsGridContextMenuClick = new ViewModelCommand(CalcDlogGsG);
              }
              return _MyGsGridContextMenuClick;
          }
      }
```

③ コンテクストメニューから呼ばれたメソッドの主な処理

```
public void CalcDlogGsG() {
    GcSpreadGrid DG = MyView.MyGsGrid;                                          ※1
    GsSGCell activeCell = DG.ActiveCell;                                        ※2
    if (activeCell != null) {
        int rowIndex = activeCell.Position.Row;                    // 行番号(0起算)
        int columnIndex = activeCell.Position.Column;              // 列番号(0起算)
        string orgVal = activeCell.Value.ToString();               // 元の書込み値を取得
        var result = 0;
        if (int.TryParse(orgVal, out result)) {
            CS_CalculatorControl calculatorControl = new CS_CalculatorControl(); //電卓クラス生成
            calculatorControl.TargetGsCell = activeCell;                        //書込先の参照を渡す
                    //Windowを生成；タイトルの初期値は書き戻し先のフィールド名
            Window CalcWindow = new Window {;                                   //Windowを生成
                        Content = calculatorControl,
                        ResizeMode = ResizeMode.NoResize
                    };
            Double ShowX = Double.Parse(CalcTextShowX);                         //表示位置
            Double ShowY = Double.Parse(CalcTextShowY);
            CalcWindow.Left = ShowX; );                                         //指定された位置に表示
            CalcWindow.Top = ShowY;
            CalcWindow.Topmost = true;
            string ViewTitle = DG.Name + "[" + (rowIndex + 1) + "," + (columnIndex + 1) + "]";
            ViewTitle += ":" + activeCell.Position.ColumnName;
            CalcWindow.Title = ViewTitle;
            calculatorControl.CalcWindow = CalcWindow;
            calculatorControl.InputStr = result.ToString();
            Nullable<bool> dialogResult = CalcWindow.ShowDialog();      //ダイアログとして表示
                    dbMsg += ",dialogResult=" + dialogResult;
    } else {//   msgStr += “電卓は数値を入力するセルでご利用ください”;
                    など元の値が数値でない場合の処理
    }
}
```

※1：コードビハインドから　public Views.CalcTestView MyView { get; set; }　でViewを参照

※2;クラス名「Cell」はInfragisticsとコンフリクトするので
using GsSGCell = GrapeCity.Windows.SpreadGrid.Cell;
using XDGCell = Infragistics.Windows.DataPresenter.Cell;
で区分

InfragisticsのXamDataGridの場合

① Viewの基底クラスへassemblyを指定します。
具体的には<Window や<UserControl　タグに
xmlns:igWPF=http://schemas.infragistics.com/xaml/wpf
を追記してください。

② Element 1 ; Elementのコンテクストメニューに設定する例

```
<igWPF:XamDataGrid Name="MyXDG"
                   DataSource="{Binding XDGDatas, Mode=TwoWay}" >
    <igWPF:XamDataGrid.FieldLayoutSettings>
        <igWPF:FieldLayoutSettings
            SelectionTypeRecord="None"
            SelectionTypeField="None"
            SelectionTypeCell="Single"
            />
    </igWPF:XamDataGrid.FieldLayoutSettings>
    <igWPF:XamDataGrid.ContextMenu>
        <ContextMenu>
            <MenuItem Header="電卓" Command="{Binding XDGContextMenuClick}"/>
        </ContextMenu>
    </igWPF:XamDataGrid.ContextMenu>
</igWPF:XamDataGrid>
```

XamDataGridにContextMenuで(intのフィールドには小数点以下を四捨五入した整数が入ります)

①標準コンテキストメニューを残す(セルをクリックで選択してから、少しずらして右クリックして下さい)

Name	Price	Tax
LEDシーリング	4980	10
歯磨き粉	298	10
おにぎり	124	8
緑茶	800	8

② Element2; Cell選択後、右クリックで電卓を呼び出す例

```
<igWPF:XamDataGrid Name="MyXDG2"
                   DataSource="{Binding XDGDatas, Mode=TwoWay}" >
    <igWPF:XamDataGrid.Resources>
        <Style TargetType="DataGridCell">
            <Setter Property="ContextMenu">
                <Setter.Value>
                    <ContextMenu>
                        <MenuItem Header="電卓"
                Command="{Binding XDGCEllRIGHTClick}"/>
                    </ContextMenu>
                </Setter.Value>
            </Setter>
        </Style>
    </igWPF:XamDataGrid.Resources>
    <igWPF:XamDataGrid.FieldLayoutSettings>
        <igWPF:FieldLayoutSettings
            SelectionTypeRecord=“None” SelectionTypeField="None"  SelectionTypeCell="Single" />
    </igWPF:XamDataGrid.FieldLayoutSettings>
    <behaviors:Interaction.Triggers>
        <behaviors:EventTrigger EventName="MouseRightButtonUp">
            <behaviors:InvokeCommandAction Command="{Binding XDGCEllRIGHTClick}" />
        </behaviors:EventTrigger>
    </behaviors:Interaction.Triggers>
</igWPF:XamDataGrid>
```

②コンテキストメニュー無し(セルを左クリックで選択してから右クリックして下さい)

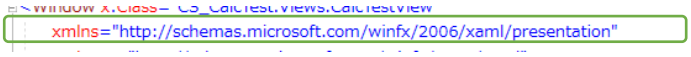
Name	Price	Tax
LEDシーリング	4980	10
歯磨き粉	298	10
おにぎり	124	8
緑茶	800	8

③ コンテクストメニューからViewModelへのBind
④ メソッド
はCalcTestViewModelの
1. #region XDGContextMenuClick　XamDataGridのコンテキストメニューから電卓を呼び出す
2. #region XDGCEllRIGHTClick　XamDataGridのCellを右クリックで電卓を呼び出す

を参照してください

WPF既定のDataGridの場合

① Viewの基底クラスへassemblyを指定します。
具体的には<Window や<UserControl　タグに
xmlns=http://schemas.microsoft.com/winfx/2006/xaml/presentation
を追記してください。



② Element例

```
<DataGrid Name="MyDG"
        ItemsSource="{Binding DGDatas, Mode=OneWay}"
        SelectedItem="{Binding SelectedDGData,Mode=TwoWay}"
    >
      <DataGrid.ContextMenu>
        <ContextMenu>
          <MenuItem Header="電卓" Command="{Binding DGContextMenuClick}"/>
        </ContextMenu>
      </DataGrid.ContextMenu>
</DataGrid>
```

DataGrid(TextBlock)にContextMenuで

Name	Price	Tax
化粧品	1776.67840375587	10
洗剤	500	10
パン	800	
牛乳	800	8

電卓

③ コンテクストメニューからViewModelへのBind
④ メソッド
はCalcTestViewModelの
#region DGContextMenuClick　規定DataGridのコンテキストメニューから電卓を呼び出す
を参照してください。

GrapeCityのGcSpreadGridとInfragisticsのXamDataGridはそれぞれのCellクラスを使用し、データはModelで設定した型が反映されます。
例えばintのフィールドは電卓から戻り値に小数を含んでも、小数点以下を四捨五入された整数が書き込まれます

WPF既定のDataGridのCellはTextBlockなので電卓からの戻り値は文字列として扱われ、計算結果の小数点以下をそのまま表示します。

変更履歴

変更日	修正概要
2021年2月18日	初版
ページ番号は修正時点のページ番号です	

https://www.coresoft-net.co.jp/