

電卓DIIの使い方

目次

リンクテキストはCtrl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	
• 表示が切り替わるタイミング	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
• 分割 • 削除	6
• 電卓Dllの組み込み方	7
• ツールボックスへの組み込み	8
• 組み込みサンプル	9
• CS_Calculator.dll のパラメータと戻り値	11
• 共通の指定 TextBoxの場合	12
• GrapeCityのGcSpreadGridの場合	13
• InfragisticsのXamDataGridの場合	14
• WPF既定のDataGridの場合	15
• 変更履歴	16

概要

CS_Calculator.dll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

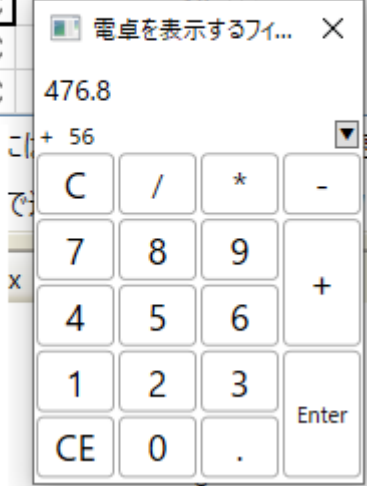
例えば演算子と数値を入力して=（Enter）キーの押下を繰り返すと繰り返し演算になりますが、電卓DLLはマウスとキーボードの持ち替えの煩雑さを減らす観点から=（Enter）キーの2回目の押下でDLLを割り付けたフィールドに値を戻す

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なっています。

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。

	Name	Price	Tax
1	みるきーくん	2,080	8
2	徳陽ほうじ茶	298	
3	バスサイズ石鯊	140	
4	5袋ラーメン	298	



結果の戻り方

WPF既定のTextBoxやDataGridのCellはTextBlockで扱われているデータはstringsなので電卓からの戻り値は文字列として扱われ、計算結果を転記すれば小数点以下も表示されていたまま表示されます。

GrapeCityのGcSpreadGridや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

画面

電卓の使い方を説明します。

A) 呼び出し元（最も基本的なTextBoxに紐づけた場合）

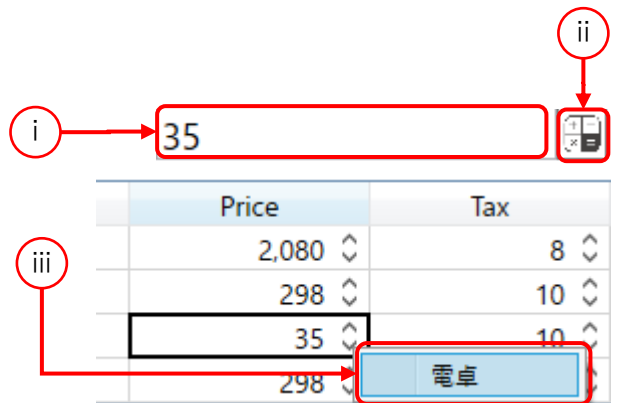
i. 電卓に紐づけられたフィールド

ii. 電卓を表示するボタン

もしくは

iii. データグリッドのコンテキストメニューから

i. 選択されているセルに紐付け。



B) 呼び出された電卓

① その時点の計算結果

② 入力値

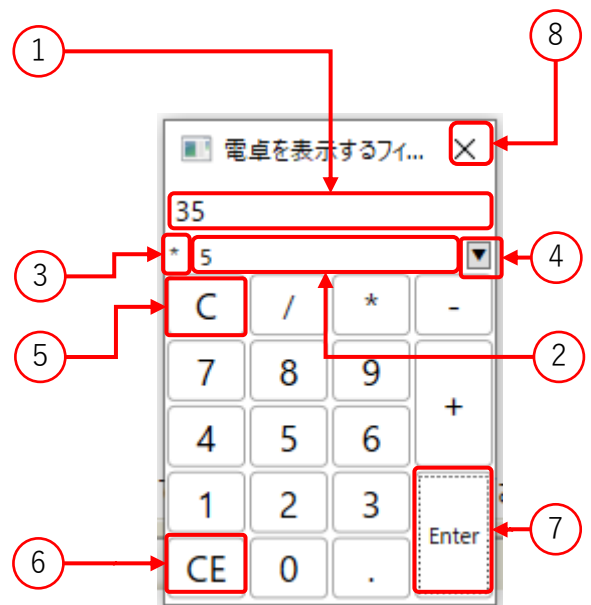
③ 演算子

④ 経過表示ボタン

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ています。



⑤ C：最後の入力から1文字ずつ消去はBackSpace

⑥ CE：全消去はDeleteキーに割り付けています

⑦ Enterは=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。

表示が切り替わるタイミング

1. 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。
2. 一回の入力ごとに経過の配列に入力値と演算子を格納すると②入力値のフィールドは空になります。
3. この状態でEnterキー押下もしくは⑧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 もしくは iii の電卓を紐づけたフィールドに転記します。

使用例

入力と表示の様子

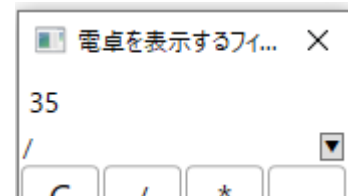
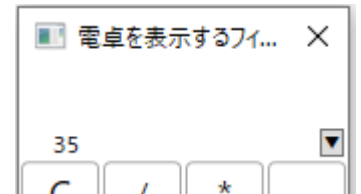
- i : 電卓に紐づけられたフィールドもしくはコンテキストメニューで電卓に紐づけたDataGridなどのセルの入力値を 3 5 にします。

Price	Tax
2,080	8
298	10
35	10
298	電卓

- 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。

※DataGridなどの場合はコンテキストメニューで電卓を呼び出し、四則演算子は電卓で入力します。

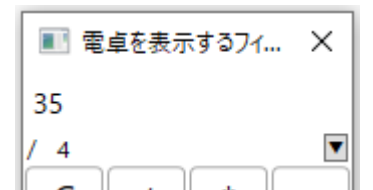


「①その時点の計算結果」にフィールドに入力されていた値、
「③演算子」には数値に続いて入力した演算子(/)が転記されます。

「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。



4. Enter (=) キーで演算した結果:8/75が計算結果に反映されます。

1. 次の演算子を入力した時点でもそこまでの計算結果が反映されます。

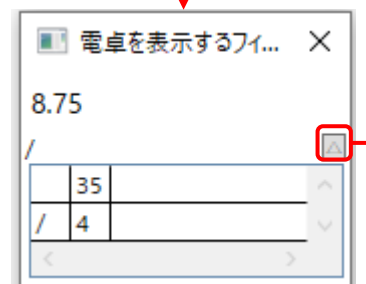
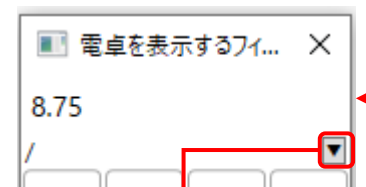
5. ④経過表示ボタンをクリックするとソフトキーボードから経過編集グリッドに表示が切り替わります。

- 経過編集グリッドには演算子と入力値がペアで記録され、入力されたどの段階でも修正でき様にします。

6. もう一度④経過表示ボタンをクリックするとソフトキーボードに戻ります。

(「経過 (確定入力) 編集」をご参照ください。)

ソフトキーボードが表示された状態で入力値の枠が空白（全ての演算が終わっている場合）ならばEnterキーでもウィンドウを閉じて計算結果を呼び出し元のフィールドに書き込みます。



C:クリア

1. 次の演算子「*」を入力（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。

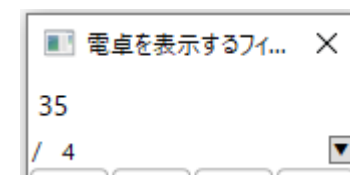
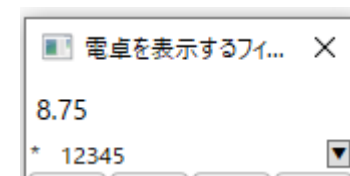
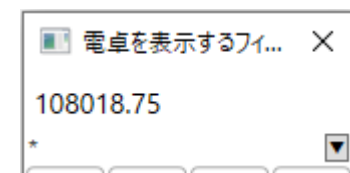
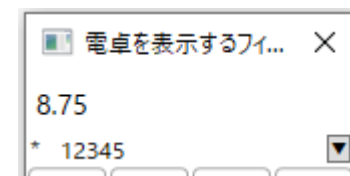
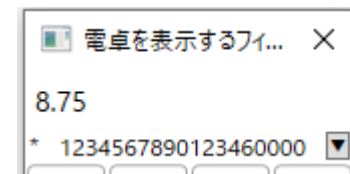
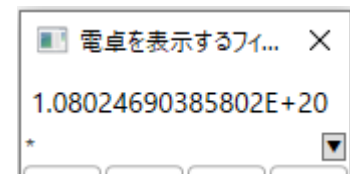
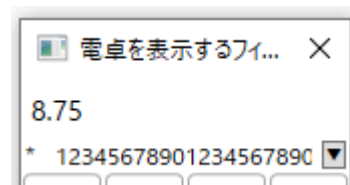
2. Cボタンクリック：BackSpaceキーの押下で直前の入力値を「④経過」リストから「②入力値」枠に戻します。

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。

3. 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。

4. 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。

- 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経過表示ボタンをクリックするとソフトキーボードから経過編集グリッドに表示が切り替わります。

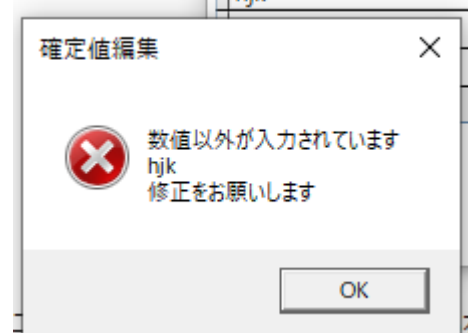
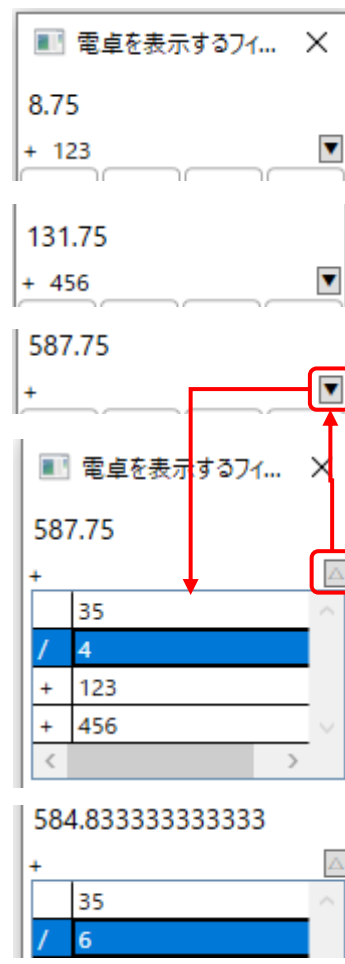
3. / 4 の列で 「4」 の枠をクリックして下さい。

4. 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。

- 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。

1. ④経過表示ボタンをクリックするとソフトキーボードから経過編集グリッドに表示が切り替わります。

6. 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計算結果を持ち帰ります。



	Name	Price	Tax
1	みるきーくん	2,080	
2	徳陽ほうじ茶	298	
3	バスサイズ石鹸	35	
4	5袋ラーメン	298	

	Name	Price
1	みるきーくん	2,080
2	徳陽ほうじ茶	298
3	バスサイズ石鹸	585

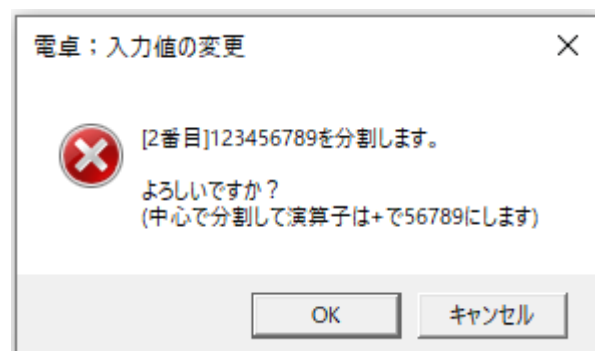
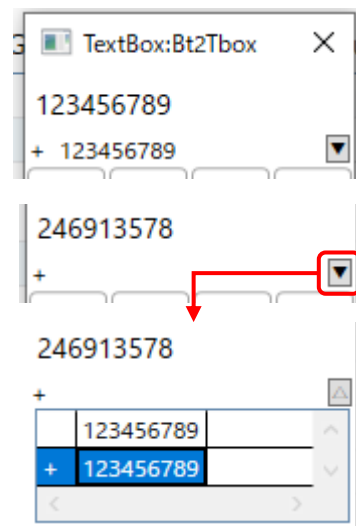
経過（確定入力）の分割

電卓の入力時によくある失敗の一つは演算子を入力し忘れて次の値入力してしまう事です。

1. そんな時は④経過表示ボタンをクリックして経過編集グリッドに表示。

1. 失敗したセルを選択して右クリックで表示されるメニューから「分割」を選択してください。

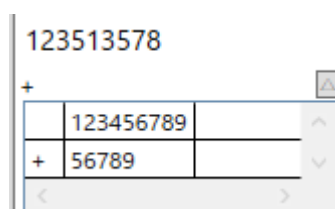
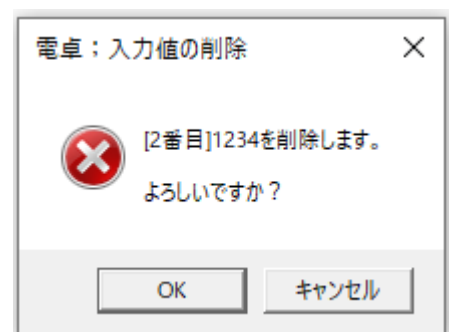
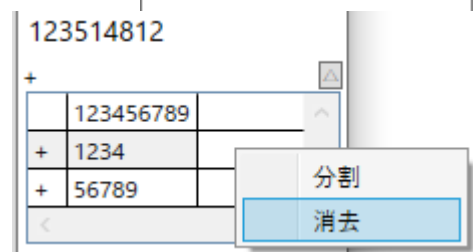
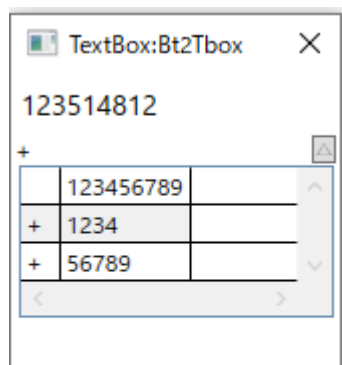
2. 表示されたダイアログでOKボタンをクリック。



※特定セルの編集ではメニューを出せないで入力されている数値を半分の位置で区切って分割し次の行へ追加します。

経過（確定入力）の分割

C(BackSpaceキー)で一文字ずつ消さなくてもコンテキストメニューの「消去」で演算子と値をペアにした一回分の入力値を消去できます。



電卓DIIの組み込み方

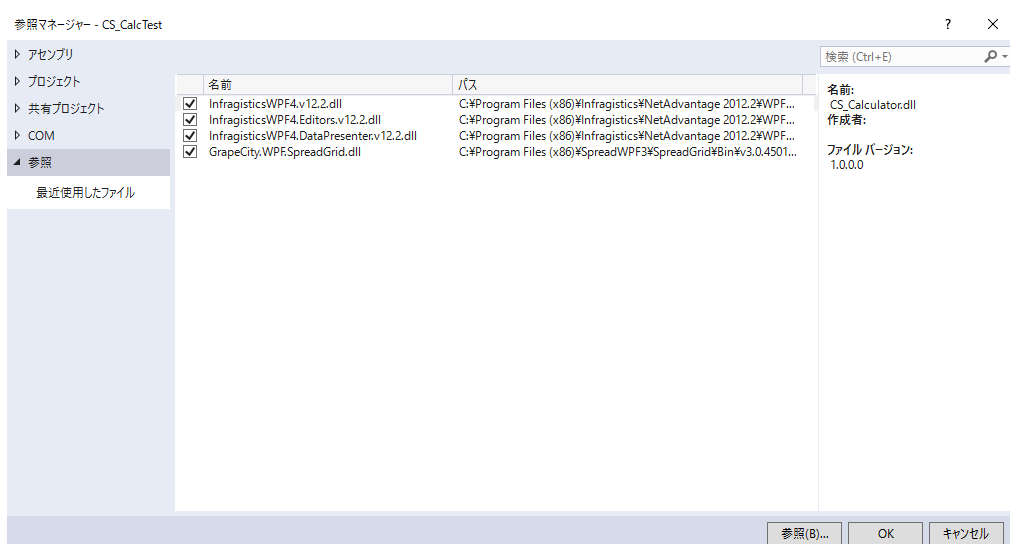
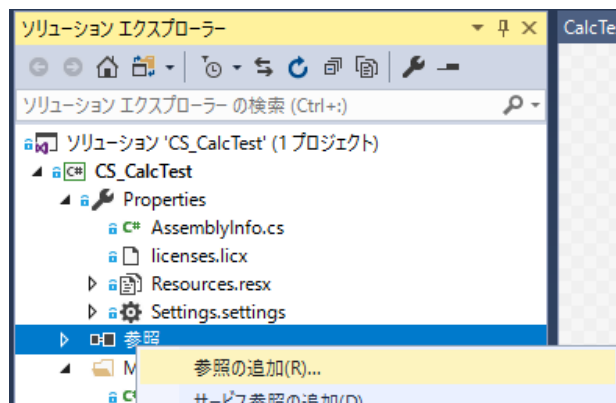
1. CS_Calculator.zipを解凍。

まずプロジェクト内の.csソースからusingでDII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。

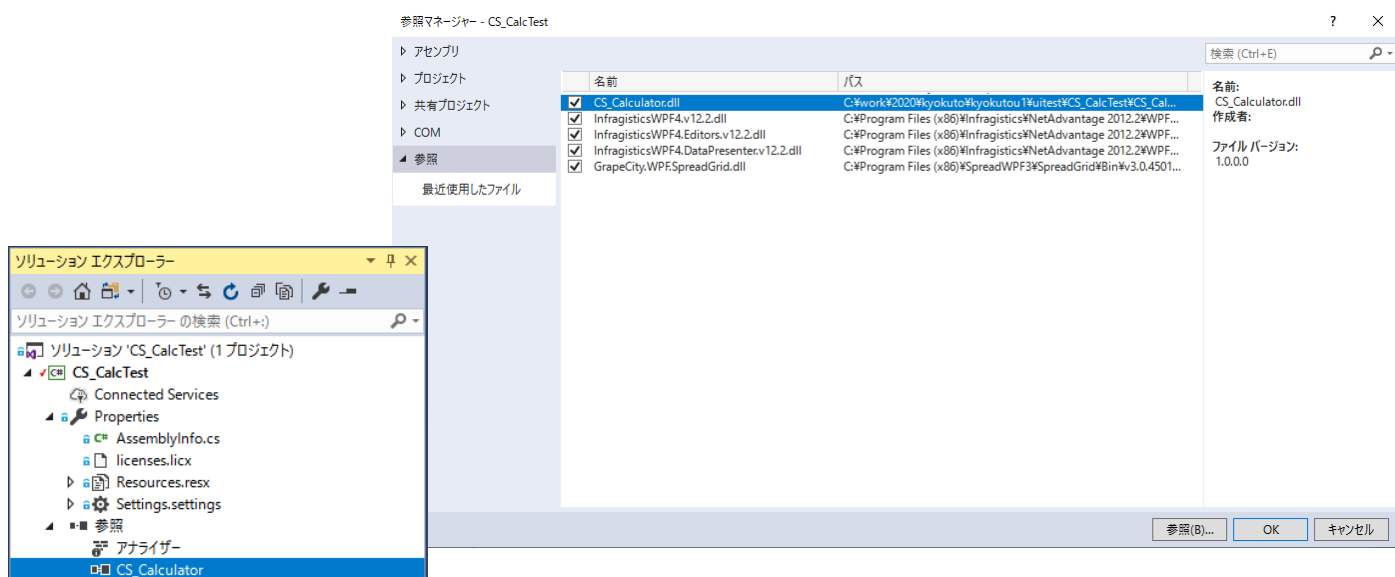
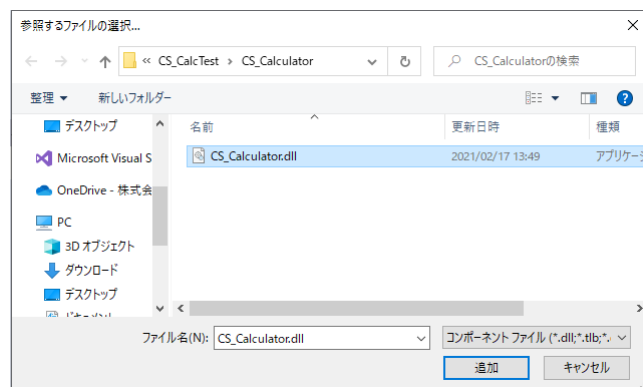
参照マネージャーの左ペインで「参照」を選択し、「参照(B)...」ボタンをクリック。



※この時点で目的の機能に該当するDII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DIIの機能が使えるようになります。



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み込みます。

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」タブで目的のDLLが有ればチェックを入れて読み込み終了。

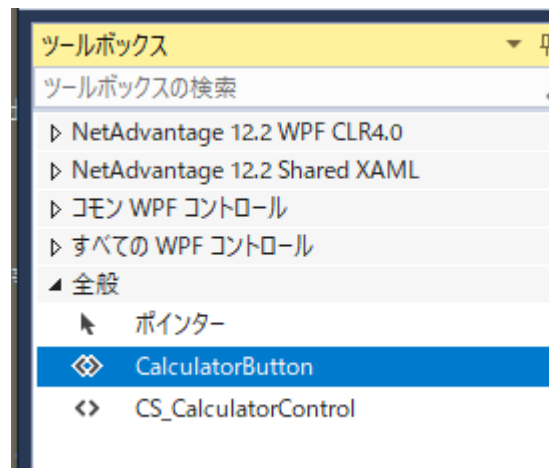
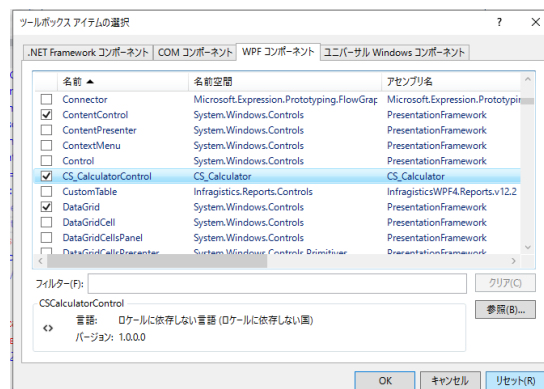
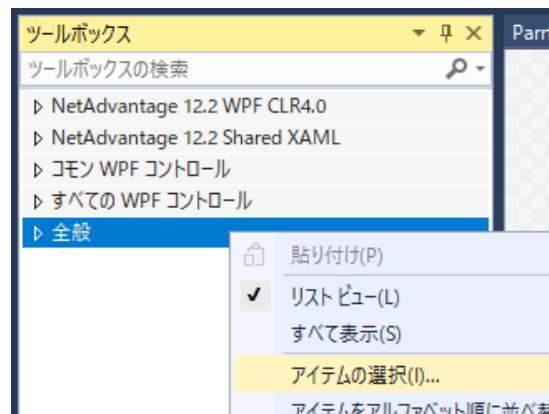
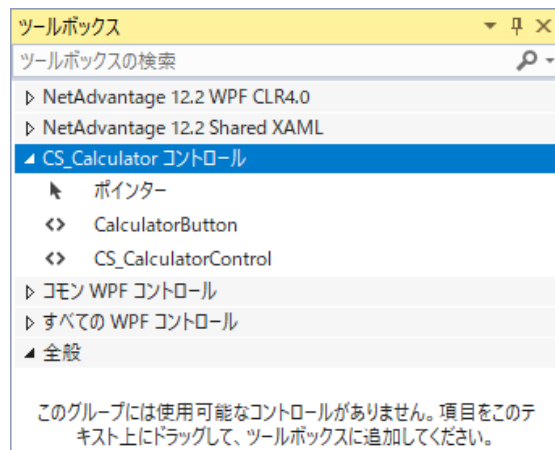
6. 無ければ「参照(B)...」ボタンをクリック。

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。

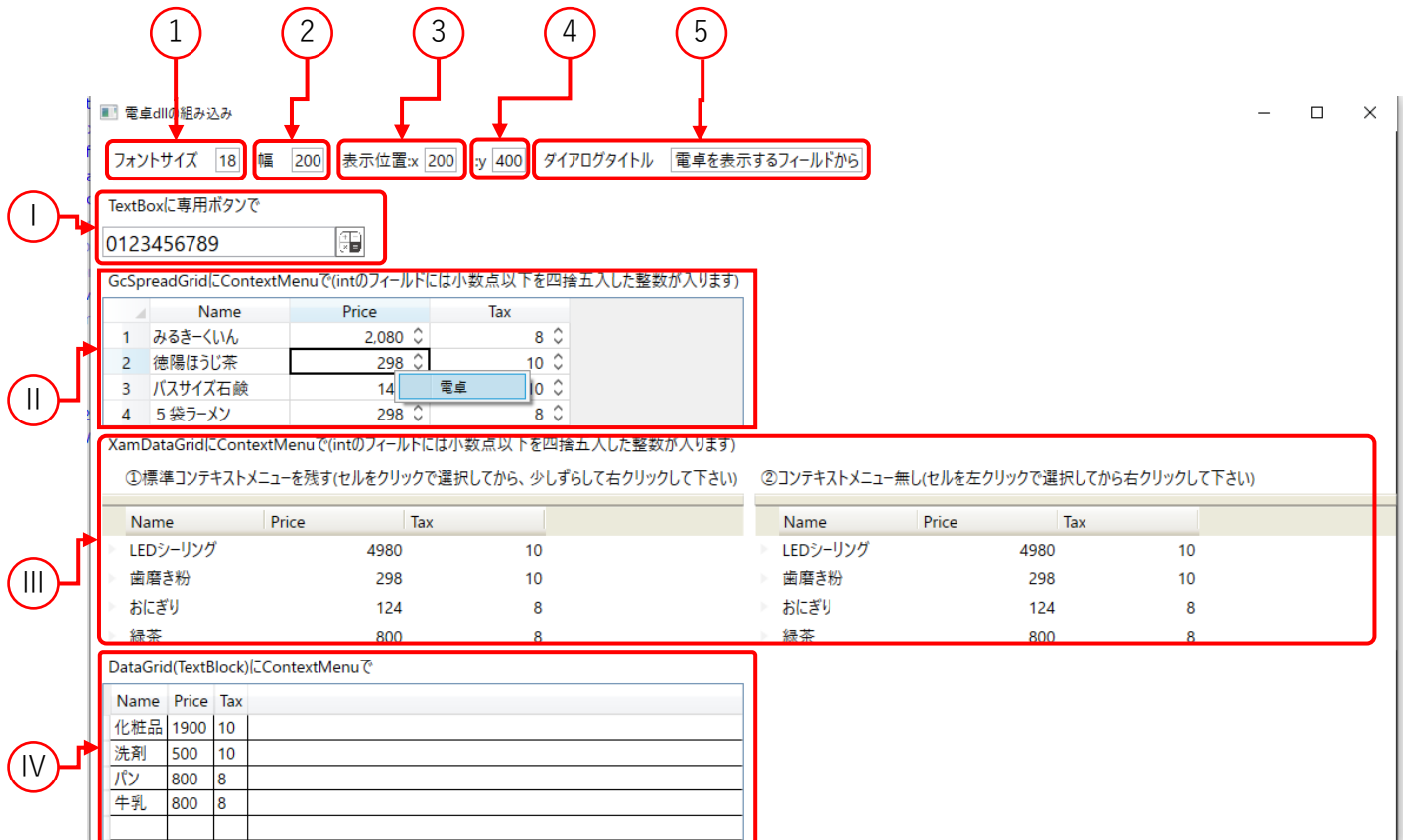


組み込みサンプル

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DII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 : CalcTextDLogTitotl
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み込んで起動画面を変更する場合は
App.xamlの<Application タグで StartupUri="Views/ParrrtsTestView.xaml" などに変更

CS_Calculator.dllのパラメータと戻り値

XAMLから="{Binding ElementName=〇〇〇}"などで指定できるもの

指定名称	内容
InputStr	初期値として渡す値。 WPFの規定エレメント(TextBox、TargetTextBlock)の場合はTargetTextBoxなどでエレメント指定すれば起動時に内容を確認して取り込みます。
ResultStr	CS_Calculator.dllを組み込んだウィンドウを閉じた時点の計算結果。 WPFの規定エレメントの場合はウィンドウを閉じた時点で指定しエレメントにCS_Calculator.dllが値を書き込みます。
CalcWindow	CS_Calculator.dllを組み込んだウィンドウ。 Close処理に使用します。
TargetTextBox	単項目フィールドなどに使用するWPF規定のTextBox
TargetTextBlock	DataGridのCellなどに使用されるWPF規定のTextBlock

ダイアログのウィンドウ生成時での指定

指定名称	内容	省略時
Title	表示するダイアログのタイトル ソース例ではViewTitleから作成した文字列を代入しています。	“電卓”
Width	表示するダイアログの幅 ソース例ではCalcWindowWidthからでBindingしたdouble値。	180
Height	表示するダイアログの高さ ソース例ではCalcWindowHeightからでBindingしたdouble値。	250
Left	表示するダイアログのX座標：プライマリーディスプレイの左上基準 ソース例ではShowXからでBindingしたdouble値。	20
Top	表示するダイアログのY座標：プライマリーディスプレイの左上基準 ソース例ではCalcWindowHeightからでBindingしたdouble値。	20

VeiwModelでのソース例

```
CS_CalculatorControl calculatorControl = new CS_CalculatorControl(); //電卓クラスを生成
Window CalcWindow = new Window{ //Windowを生成
    Title = ViewTitle,
    Width = CalcWindowWidth,
    Height = CalcWindowHeight,
    Left = ShowX,
    Top = ShowY,
    Content = calculatorControl,
    ResizeMode = ResizeMode.NoResize,
    Topmost=true //ShowDialogの場合は省略可能
};
calculatorControl.CalcWindow = CalcWindow; //dllからクローズなどのwindow制御を行う為の指定
calculatorControl.InputStr = result.ToString(); //dllに元の書き込み値を渡す
CalcWindow.ShowDialog(); //ダイアログ表示
string resultStr = calculatorControl.ResultStr; //ダイアログを閉じると戻り値が返る
activeCell.Value = double.Parse(resultStr); //計算結果を連携元へ書き込む
```

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"
```

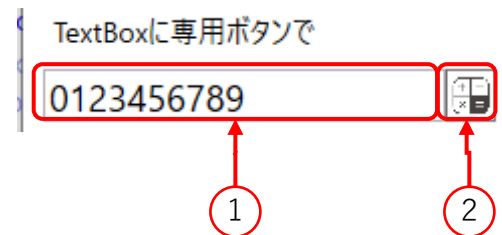
を追記してください。

```
<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はダイア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。

```
xmlns:igwpf="http://schemas.imagisus.com/igwpf"
xmlns:sg="http://schemas.grapecity.com/windows/spreadgrid/2012"
xmlns:calculator="dxamomono:CS_Calculator;assembly=CS_Calculator"
```

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

ソリューション エクスプローラー の検索 (Ctrl+:)

ソリューション 'CS_CalcTest' (1 プロジェクト)

- CS_CalcTest
 - Connected Services
 - Properties
 - AssemblyInfo.cs
 - licenses.licx

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	みるきーいん	2,080	8
2	徳陽ほうじ茶	298	10
3	バスサイズ石鹸	14	10
4	5袋ラーメン	298	8

③ コンテキストメニューからViewModelへのBind

```
private ViewModelCommand _MyGsGridContextMenuClick;
```

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

③ コンテキ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(); // 元の書き込み値を取得
        var result = 0;
        if (int.TryParse(orgVal, out result)) {
            ViewTitle = CalcTextDLogTitel;
            ShowX = Double.Parse(CalcTextShowX);
            ShowY = Double.Parse(CalcTextShowY);
            CS_CalculatorControl calculatorControl = new CS_CalculatorControl(); //電卓クラス生成
            Window CalcWindow = new Window{ //Windowを生成
                Title = ViewTitle,
                Width = CalcWindowWidth,
                Height = CalcWindowHeight,
                Left = ShowX,
                Top = ShowY,
                Content = calculatorControl,
                ResizeMode = ResizeMode.NoResize,
                Topmost=true
            };
            calculatorControl.CalcWindow = CalcWindow;
            //dllからクローズなどのwindow制御を行う為の指定
            calculatorControl.InputStr = result.ToString(); //dllに元の書き込み値を渡す
            CalcWindow.ShowDialog(); //ダイアログ表示
            string resultStr = calculatorControl.ResultStr; //ダイアログを閉じると戻り値が返る
            activeCell.Value = double.Parse(resultStr);
        } else { // msgStr += “電卓は数値を入力するセルをご利用ください”;
            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を参照します

※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 XDGCCell = 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/wpff

を追記してください。

```
xmlns:igWPF="http://schemas.infragistics.com/xaml/wpff"
xmlns:ig="http://schemas.infragistics.com/windows/igdatagrid/2012"
```

- ② Element 1 ; Elementのコンテキ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 XDGCEIRIGHTClick}" />
                    </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 XDGCEIRIGHTClick}" />
        </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 コンテキストメニューから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 XDGCEIRIGHTClick 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

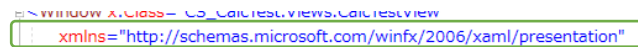
WPF既定のDataGridの場合

- 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。



xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

- 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		電卓
牛乳	800	8	

- ③ コンテキストメニューから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。

ページ番号は修正時点のページ番号です



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