FinancialChart for WPF

2018.07.20 更新

グレープシティ株式会社

FinancialChart for WPF

目次

FinancialChart for WPF の概要

FinancialChart for WPF は、株価のトレンドチャートを作成できる、使いやすい視覚化コントロールです。このコントロールで事前定義されている株価インジケータと専用のチャートタイプを使用して、財務アプリケーションでテクニカル分析を実行できます。さらに、傾向線、移動平均、範囲セレクタなどの分析ツールを使用して、データを操作しながら傾向を分析できます。

以下のセクションでは、FinancialChart for WPF のすべての面について詳細に説明します。

主な特長

- **チャートタイプ**
  折れ線グラフを散布図などの他のチャートタイプに変更できます。それには、**ChartType** プロパティを設定するだけです。15 種類のチャートタイプから選択できます。
- **範囲セレクタ**
  FinancialChart の実行時のデータ表示範囲を調整します。
- **傾向線**
  データのトレンドを可視化し、予測の問題点を分析します。
- **ツールチップ**
  ツールチップを使用してチャートの値を表示できます。
- **ヘッダーとフッター**
  単純なプロパティを使用して、タイトルやフッターテキストを設定できます。
- **凡例**
  必要に応じて凡例の位置を変更できます。
- **新値足**
  ラインまたは縦長のボックスを使用して、資産や市場の価格変化を示します。
- **注釈**
  株価チャートの特定のデータポイントに重要なイベントやニュースをアタッチします。
- **移動平均**
  データセット全体の一部から一連の平均値を求めてデータポイントを分析します。
- **インジケータ**
  ATR（Average True Range）、RSI（相対力指数）、CCI（商品チャンネル指数）、ウィリアムズ %R などのテクニカルインジケータを使用して、商品の価格変化や取引量動向を分析および予測します。

クイックスタート

このクイックスタートでは、FinancialChart for WPF を使用してシンプルなアプリケーションを作成し、Visual Studio で実行するプロセスを説明します。

FinancialChart コントロールをすばやく体験するには、次の手順を実行します。

1. フォームへの FinancialChart の追加
2. データソースへの FinancialChart の連結
3. アプリケーションの実行

手順 1:フォームへの FinancialChart の追加

この手順では、新しいVisual Studioプロジェクトを作成し、FinancialChart コントロールを追加します。

1. Visual Studio で WPF アプリケーションを作成します。
    1. [**ファイル**]→[**新規作成**]→[**プロジェクト**]を選択します。[**新しいプロジェクト**]ダイアログボックスが表示されます。
    2. [**新しいプロジェクト**]ダイアログボックスの左側のペインで、言語を選択し、中央ペインのアプリケーションリストから[**WPF アプリケーション**]を選択します。
    3. アプリケーションに名前を付け、[**OK**]を選択します。
2. **MainWindow.xaml** ファイルを開きます。
3. 作成したアプリケーションのタイプに応じて、ウィンドウまたはユーザーコントロール内の**<Grid></Grid>** タグ間にカーソル置きます。
4. Visual Studio のツールボックスで、**C1FinancialChart** コントロールを見つけます。コントロールをダブルクリックしてアプリケーションに追加します。次の参照がプロジェクトに追加されます。
    - **C1.WPF.4.dll**
    - **C1.WPF.FinancialChart.4.dll**
    - **C1.WPF.FlexChart.4.dll**

    参照が追加されていない場合は、手動で追加する必要があります。**ソリューションエクスプローラー** では、参照フォルダを右クリックし、[**追加**]→[**参照の追加**]を選択します。

    XAML マークアップは次のようになります。
    - **XAML**

```
<Window
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
        x:Class="FinancialChart.MainWindow"
        Title="MainWindow" Height="387.285" Width="641.667">
    <Grid>

        <c1:C1FinancialChart x:Name="financialChart"
                             ChartType="HeikinAshi"
                             HorizontalAlignment="Left"
                             Height="325" VerticalAlignment="Top"
                             Width="523">
            <c1:FinancialSeries AxisX="{x:Null}" AxisY="{x:Null}"
                                Binding="High,Low,Open,Close,Volume"
                                BindingX="Date"
                                SeriesName="{x:Null}">
                <c1:FinancialSeries.ItemsSource>
                    <c1:QuoteCollection>
                        <c1:Quote Close="23.23" Date="01/23/15"
                                  High="24.73" Low="20.16"
                                  Open="20.2" Volume="42593223"/>
                        <c1:Quote Close="22.6" Date="01/26/15"
```

```
                                    High="24.39" Low="22.5"
                                    Open="23.67" Volume="8677164"/>
                            <c1:Quote Close="21.3" Date="01/27/15"
                                    High="22.47" Low="21.17"
                                    Open="22" Volume="3272512"/>
                            <c1:Quote Close="19.78" Date="01/28/15"
                                    High="21.84" Low="19.6"
                                    Open="21.62" Volume="5047364"/>
                            <c1:Quote Close="18.8" Date="01/29/15"
                                    High="19.95" Low="18.51"
                                    Open="19.9" Volume="3419482"/>
                        </c1:QuoteCollection>
                    </c1:FinancialSeries.ItemsSource>
                </c1:FinancialSeries>
            </c1:C1FinancialChart>

    </Grid>
</Window>
```

FinancialChart コントロールが正常にアプリケーションに追加されます。

手順 2：データソースへの FinancialChart の連結

このステップでは、FinancialChart コントロールを有効なデータソースに連結します。

1. 次のようにデータソースを作成します。
    1. プロジェクトを右クリックし、[**追加**]→[**クラス**]を選択します。
    2. テンプレートの一覧から[**クラス**]を選択し、「**DataService**」と名前を付け、[**追加**]をクリックします。
    3. **DataService** クラスに次のコードを追加して、データを生成します。
    - **Visual Basic**

```
Public Class DataService

    Public Shared Function CreateData() As List(Of DataItem)
        Dim data = New List(Of DataItem)()

        Dim dt As DateTime = DateTime.Today

        data.Add(New DataItem(dt.[Date], 79))
        data.Add(New DataItem(dt.[Date].AddDays(-7), 78))
        data.Add(New DataItem(dt.[Date].AddDays(-14), 73))
        data.Add(New DataItem(dt.[Date].AddDays(-21), 74))
        data.Add(New DataItem(dt.[Date].AddDays(-28), 76))
        data.Add(New DataItem(dt.[Date].AddDays(-35), 74))
        data.Add(New DataItem(dt.[Date].AddDays(-42), 75))
        data.Add(New DataItem(dt.[Date].AddDays(-49), 75))
        data.Add(New DataItem(dt.[Date].AddDays(-56), 80))
        Return data
    End Function

End Class

Public Class DataItem
    Public Sub New(date__1 As DateTime, sales__2 As Integer)
        [Date] = date__1
        Sales = sales__2
    End Sub

    Public Property [Date]() As DateTime
        Get
            Return m_Date
        End Get
```

```
            Set
                m_Date = Value
            End Set
        End Property
        Private m_Date As DateTime
        Public Property Sales() As Integer
            Get
                Return m_Sales
            End Get
            Set
                m_Sales = Value
            End Set
        End Property
        Private m_Sales As Integer
    End Class
```

- **C#**

```
    class DataService
    {
        public static List<DataItem> CreateData()
        {
            var data = new List<DataItem>();

            DateTime dt = DateTime.Today;

            data.Add(new DataItem(dt.Date,79));
            data.Add(new DataItem(dt.Date.AddDays(-7), 78));
            data.Add(new DataItem(dt.Date.AddDays(-14), 73));
            data.Add(new DataItem(dt.Date.AddDays(-21), 74));
            data.Add(new DataItem(dt.Date.AddDays(-28), 76));
            data.Add(new DataItem(dt.Date.AddDays(-35), 74));
            data.Add(new DataItem(dt.Date.AddDays(-42), 75));
            data.Add(new DataItem(dt.Date.AddDays(-49), 75));
            data.Add(new DataItem(dt.Date.AddDays(-56), 80));
            return data;
        }
    }

    public class DataItem
    {
        public DataItem(DateTime date, int sales)
        {
            Date = date;
            Sales = sales;
        }

        public DateTime Date { get; set; }
        public int Sales { get; set; }
    }
```

2. 次のように、データを FinancialChartに連結します。
    1. **<Grid>** タグを編集して次のマークアップを作成し、FinancialChart にデータを提供します。
        - **MainWindow.xaml**

```
<Grid>
    <c1:C1FinancialChart x:Name="financialChart"
                         ChartType="LineSymbols"
                         ItemsSource="{Binding DataContext.Data}"
                         HorizontalAlignment="Left"
                         Height="321"
                         VerticalAlignment="Top"
                         Width="620"
                         Margin="81,94,0,0">
        <c1:FinancialSeries AxisX="{x:Null}"
                            AxisY="{x:Null}"
```

```
                                Binding="Sales"
                                BindingX="Date"
                                SeriesName="{x:Null}">
            </c1:FinancialSeries>
        </c1:C1FinancialChart>
    </Grid>
```

> 連結ソースを指定するには、**DataContext = "{Binding RelativeSource={RelativeSource Mode=Self}}"** マークアップを **MainWindow.xaml** ファイルの **<Window>** タグに追加する必要があります。

2. コードビューに切り替えます。MainWindow クラスに次のコードを追加して、チャートにデータをプロットします。
   - **MainWindow.xaml.vb**

```
Partial Public Class MainWindow
    Inherits Window
    Private _data As List(Of DataItem)

    Public Sub New()
        Me.InitializeComponent()

        financialChart.AxisX.Format = "M月d日"
    End Sub

    Public ReadOnly Property Data() As List(Of DataItem)
        Get
            If _data Is Nothing Then
                _data = DataService.CreateData()
            End If

            Return _data
        End Get
    End Property
End Class
```

   - **MainWindow.xaml.cs**

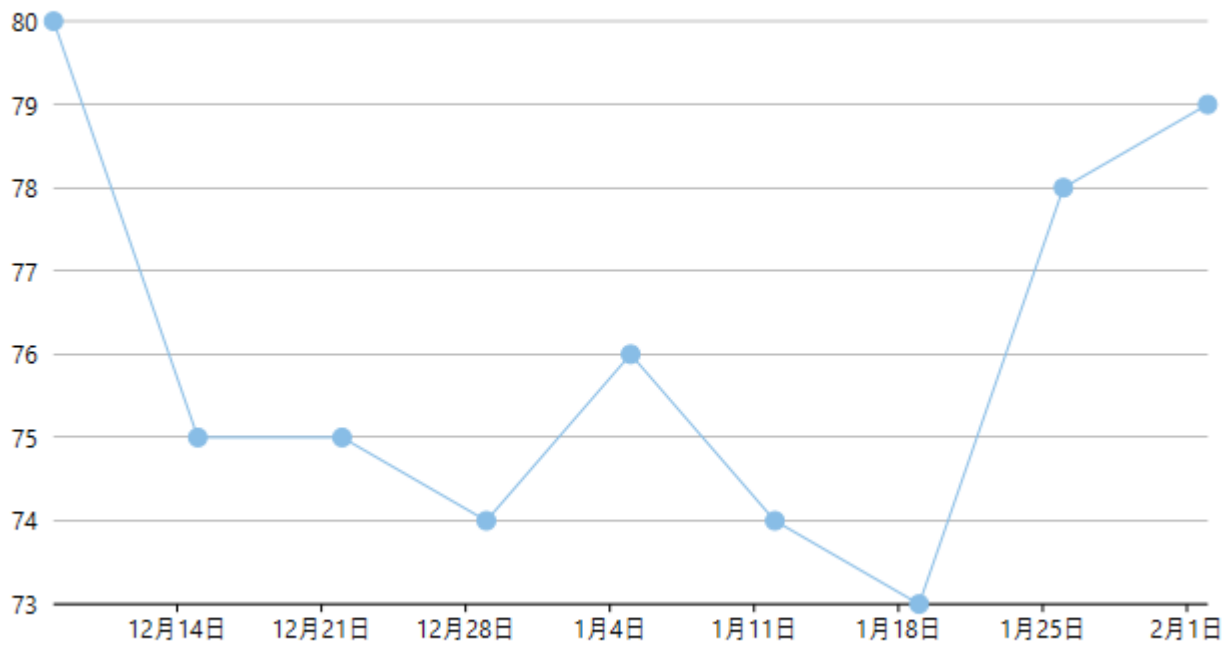
```
public partial class MainWindow : Window
{
    private List<DataItem> _data;

    public MainWindow()
    {
        this.InitializeComponent();

        financialChart.AxisX.Format = "M月d日";
    }

    public List<DataItem> Data
    {
        get
        {
            if (_data == null)
            {
                _data = DataService.CreateData();
            }

            return _data;
        }
    }
}
```

FinancialChart コントロールが正常にデータソースに連結できました。

手順 3:アプリケーションの実行

この手順では、アプリケーションを実行し、出力を確認します。

アプリケーションを実行すると、次の出力が表示されます。



これで、単純な FinancialChart アプリケーションを作成できました。

株価チャートタイプ

FinancialChart には、金融データの視覚化要件に対応する 15 のチャートタイプがあります。**ChartType** プロパティを設定して、FinancialChart コントロールのチャートタイプを設定できます。このプロパティは、**FinancialChartType** 列挙に含まれる値を受け取ります。

次の表に、FinancialChart のすべてのチャートタイプを示します。

面グラフ	**アームズローソクボリュームチャート**	**ローソク足チャート**	**ローソクボリュームチャート**
縦棒グラフ	**カラムボリュームチャート**	**エクイボリュームチャート**	**平均足チャート**
HighLowOpenClose チャート	**カギ足チャート**	**折れ線グラフ**	**新値足チャート**
折れ線グラフ(シンボル付き)	**練行足チャート**	**散布図**	**ポイントアンドフィギュアチャート**

次のコードスニペットでは、**ChartType**プロパティを設定しています。

XAML

```
<c1:C1FinancialChart ChartType="ArmsCandleVolume"
                     x:Name="financialChart"
                     SelectionMode="Series"
                     BindingX="date"
                     Binding="high,low,open,close,volume"
                     Grid.Row="1">
    <c1:FinancialSeries SeriesName="Series" />
</c1:C1FinancialChart>
```

Code

Visual Basic copyCode

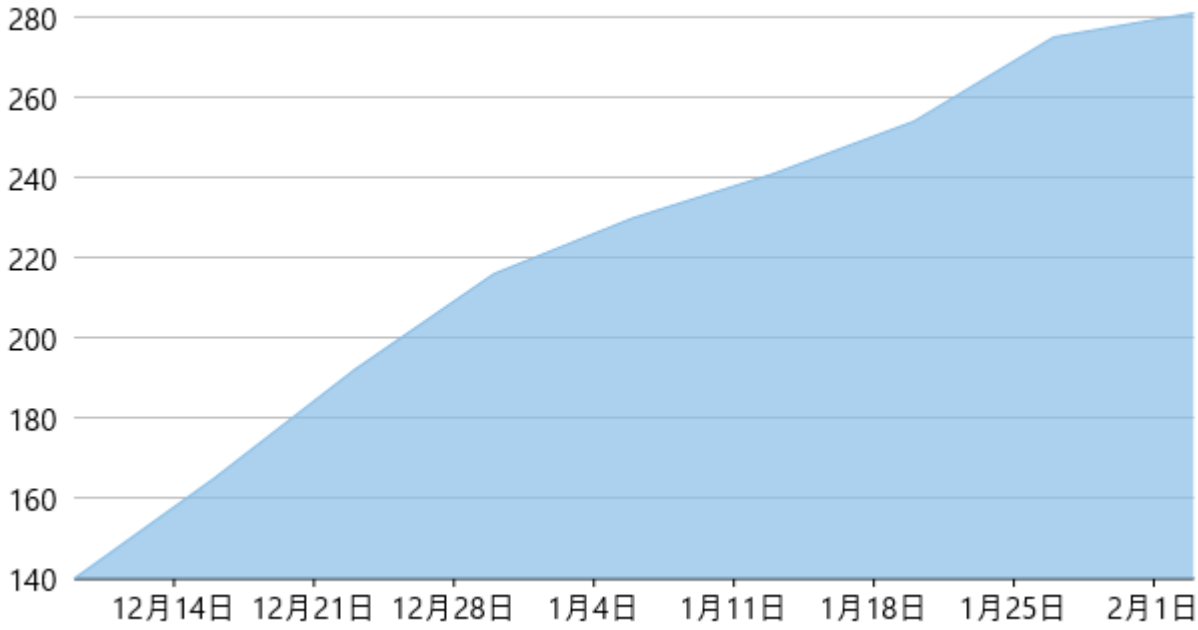
```
' 株価チャートタイプを設定します
financialChart.ChartType = FinancialChartType.ArmsCandleVolume
```

C# copyCode

```
// 株価チャートタイプを設定します
financialChart.ChartType = FinancialChartType.ArmsCandleVolume;
```

面グラフ

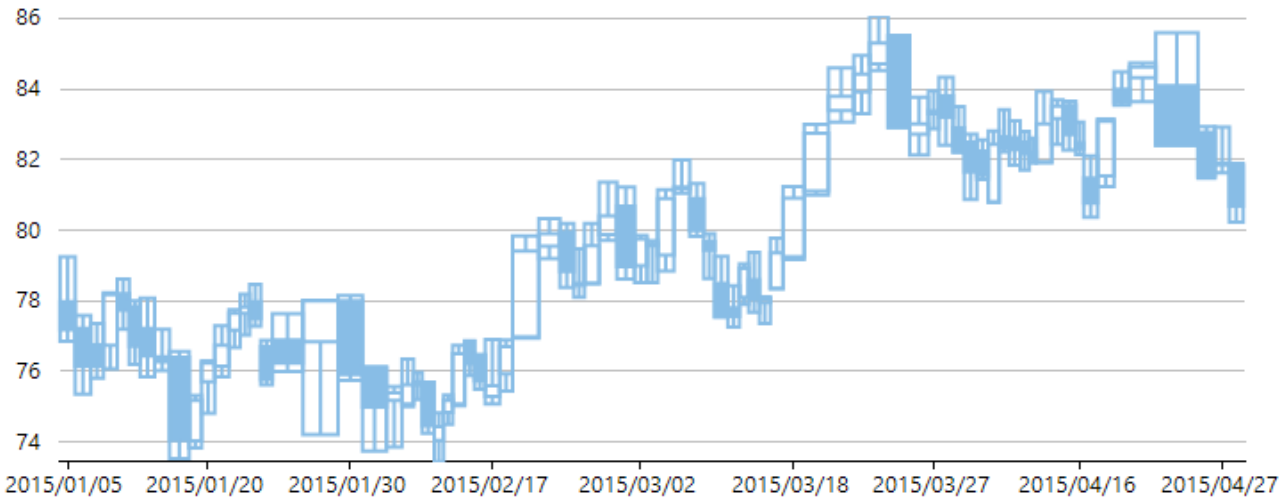
面グラフは、同じ系列のデータポイントどうしを線で結び、この線の下の領域を塗りつぶして量を示します。新しい系列は、それぞれ前の系列の上に表示されます。系列は、個別に描画することも、積層にして描画することもできます。これらのグラフは、関連する属性間の長期にわたるトレンドを示す場合によく使用されます。



先頭に移動

アームズローソクボリュームチャート

アームズローソクボリュームチャートは、エクイボリュームおよびローソクボリュームチャートタイプの組み合わせとして、Richard Arms によって作成されました。このチャートタイプのデータは、FinancialChart または FinancialSeries 連結プロパティを使用して、「highProperty, lowProperty, openProperty, closeProperty, volumeProperty」形式のカンマ区切り値で定義できます。このチャートタイプは、FinancialChart レベルでのみ使用できます。FinancialSeries オブジェクトに適用することはできません。現在は、1 つの FinancialChart に 1 セットのボリュームデータだけがサポートされています。



先頭に移動

ローソク足チャート

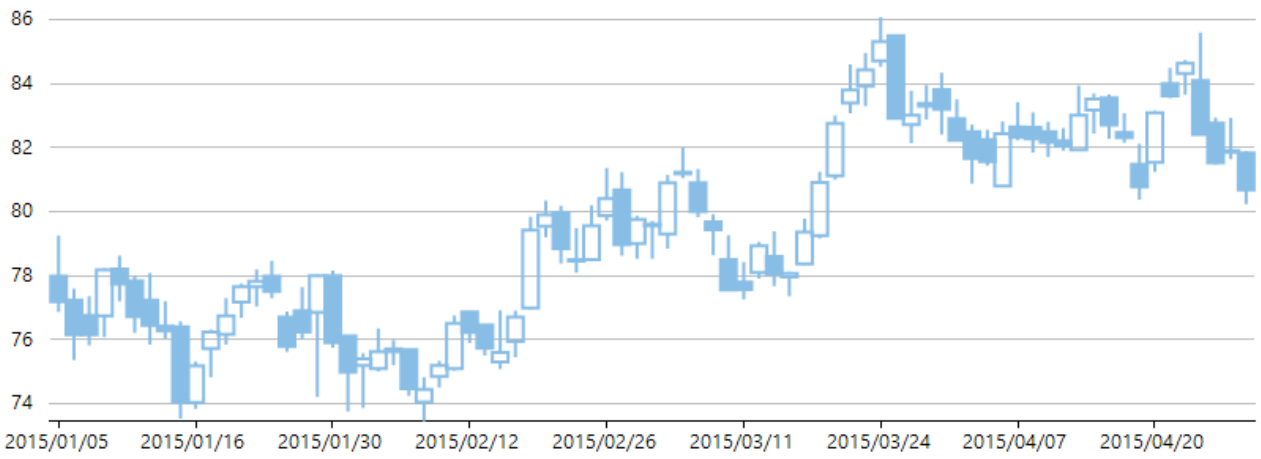
ローソク足チャートは、Hi-Lo-Open-Close チャートの特殊なタイプで、株価の始値、終値、高値、低値を示します。ローソク足チャートは、横棒グラフと折れ線グラフを合わせたような形式で経時的な値の範囲を表します。ローソクと呼ばれるビジュアル要素で構成され、ローソクは胴体、上ヒゲ、下ヒゲの 3 つの要素から成ります。

- **胴体**は始値と終値を表現し、上ヒゲと下ヒゲはそれぞれ高値と安値を表現します。
- **白抜きの胴体**は、株価が上昇したことを示します（終値が始値より高い）。
- **黒塗りの胴体**は、株価が下落したことを示します（始値が終値より高い）。

ヒゲ線のサイズは高値と安値で決定され、棒のサイズは、始値と終値で決定されます。棒は、終値が始値より高いか低いかに基づいて異なる色で表示されます。このチャートタイプのデータは、FinancialChart または FinancialSeries 連結プロパティを使用して、「highProperty, lowProperty, openProperty, closeProperty」形式のカンマ区切り値で定義できます。

ローソク足では、系列内のデータポイントごとに 5 つの値があります。

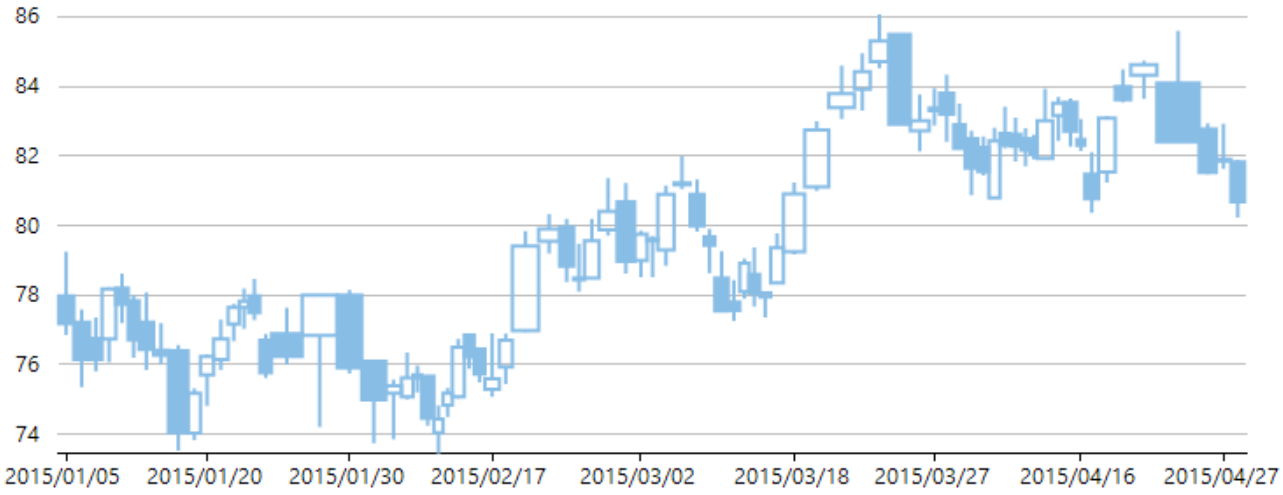
- **x**: X 軸方向の日付の位置を決定します。
- **high**: その日の最高値を決定し、それを Y 軸方向にローソクの上端としてプロットします。
- **low**: その日の最安値を決定し、それを Y 軸方向にローソクの下端としてプロットします。
- **open**: その日の始値を決定します。
- **close**: その日の終値を決定します。



先頭に移動

ローソクボリュームチャート

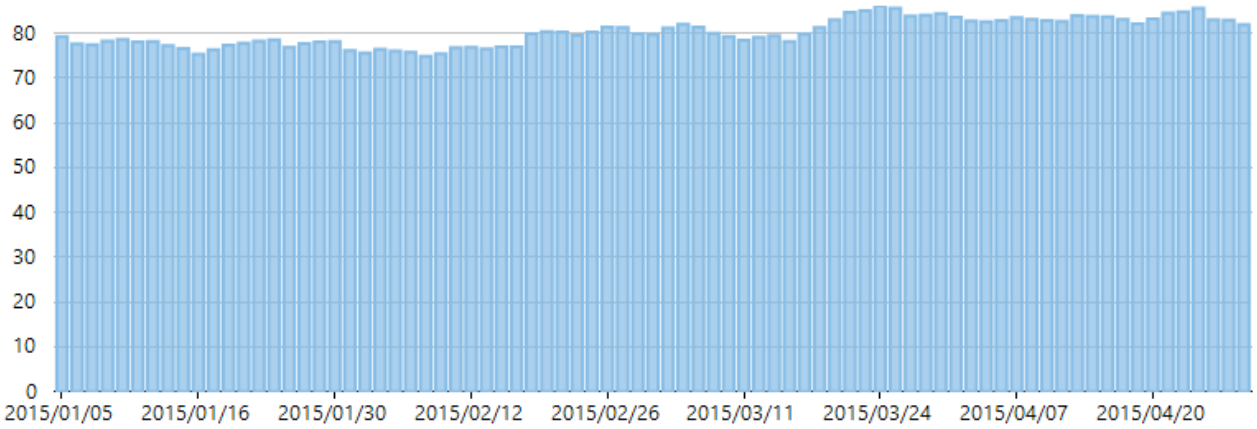
ローソクボリュームチャートは、各棒の幅が Volume 値で決まること以外は、標準のローソク足チャートと同じです。このチャートタイプのデータは、FinancialChart または FinancialSeries 連結プロパティを使用して、「highProperty, lowProperty, openProperty, closeProperty, volumeProperty」形式のカンマ区切り値で定義できます。



先頭に移動

縦棒グラフ

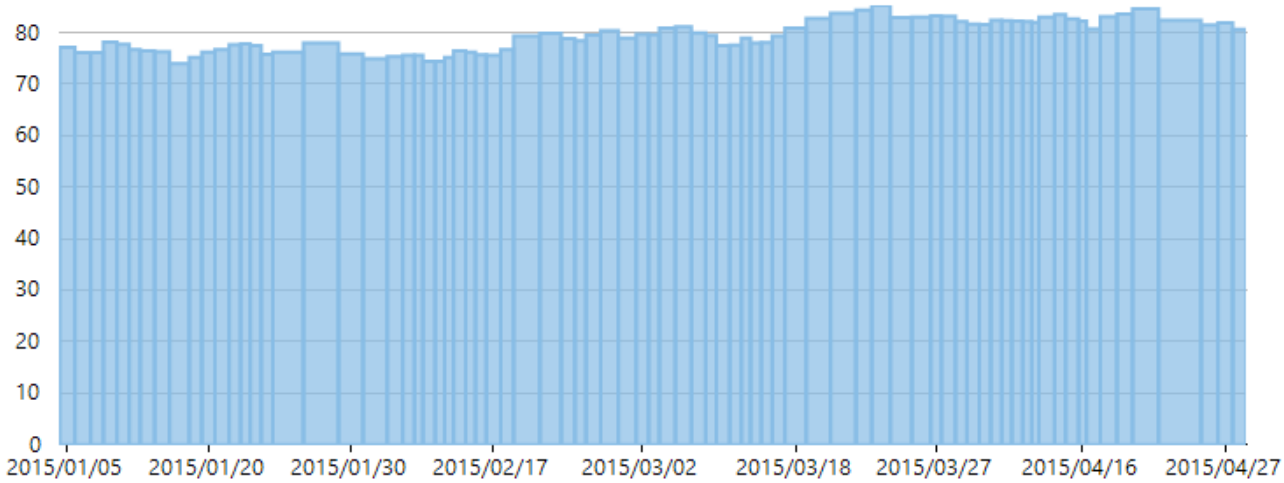
縦棒グラフは、各系列を棒の形式で表し、さまざまなカテゴリの項目の値を比較できます。1 つ以上の項目の値を縦棒として Y 軸に表示し、項目やカテゴリを X 軸に配置します。



先頭に移動

カラムボリュームチャート

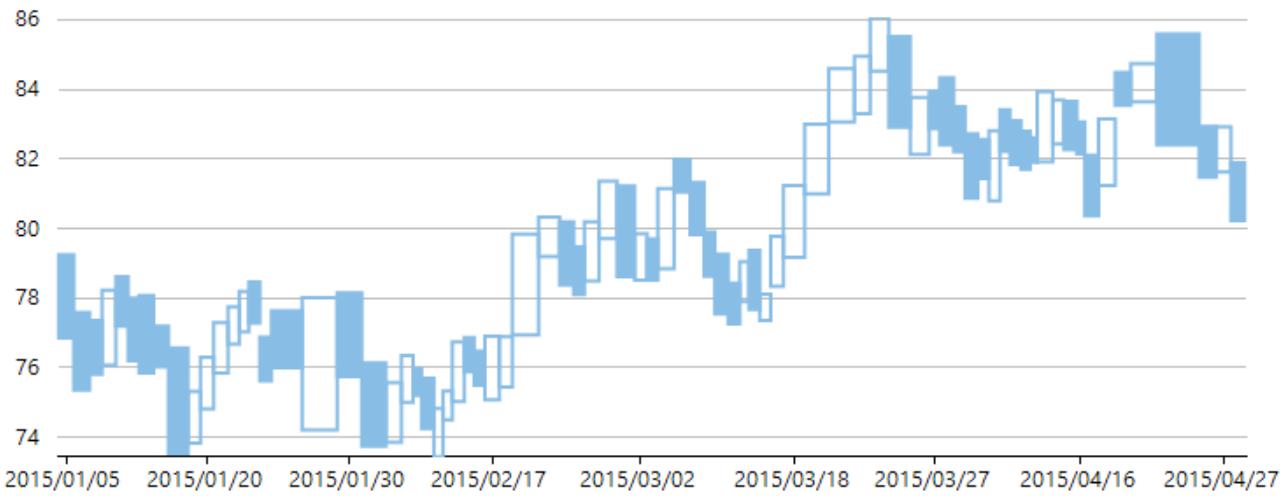
カラムボリュームチャートは、各棒の幅が Volume 値で決まること以外は、標準の縦棒グラフと同じです。このチャートタイプのデータは、FinancialChart または FinancialSeries 連結プロパティを使用して、「yProperty, volumeProperty」形式のカンマ区切り値で定義できます。



先頭に移動

エクイボリュームチャート

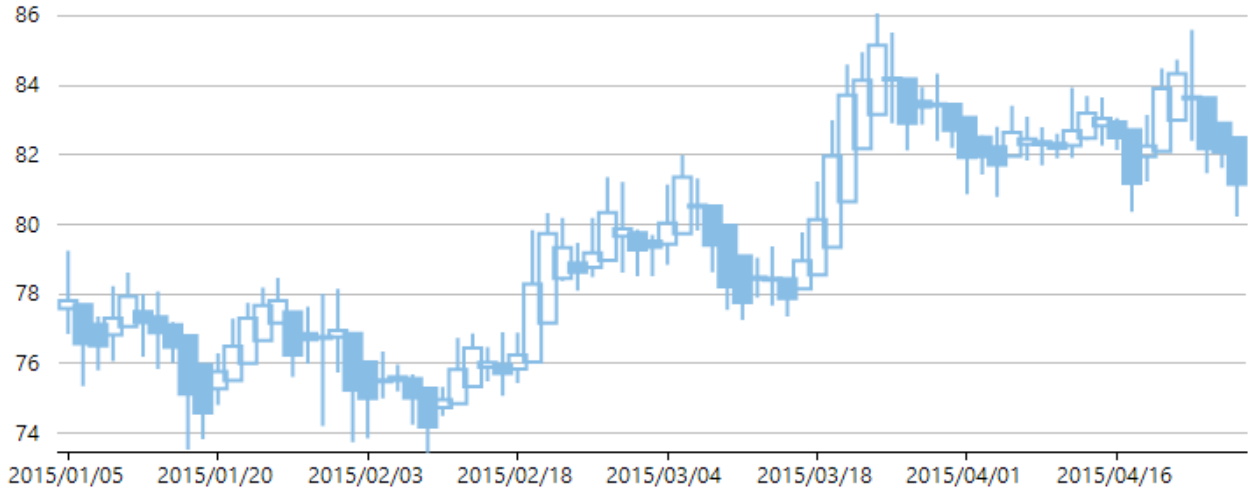
EquiVolume チャートはローソク足チャートに似ていますが、これらのチャートではローソク足がさまざまな幅の四角形のボックス（上ヒゲなし）に置き換えられます。EquiVolume ボックスには、高値および安値の成分と、3 つ目の次元として各ボックスの幅を決定する Volume が含まれます。色は、終値の数字が直前のボックスの終値よりも上か下かを表します。



先頭に移動

平均足チャート

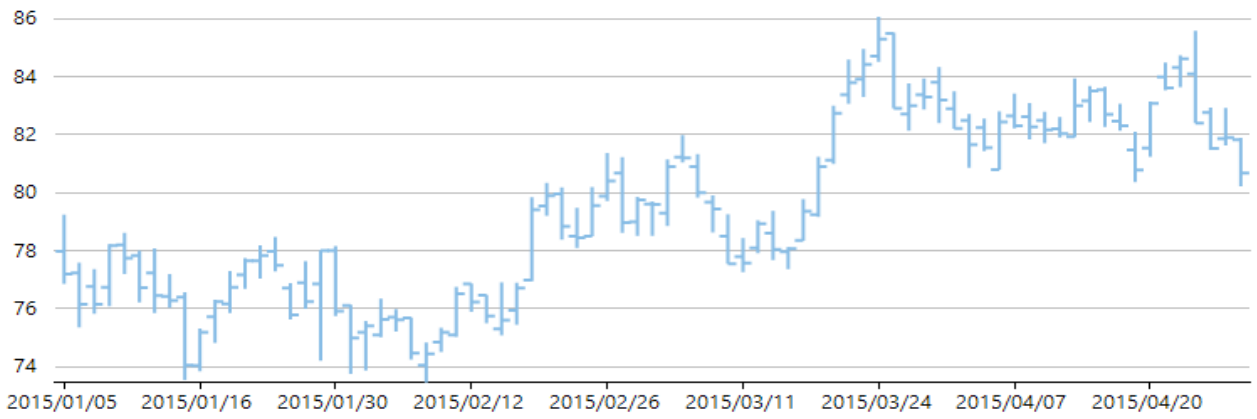
平均足チャートは日本のローソク足チャートのバリエーションで、ローソク足チャートからノイズを取り除き、移動平均によく似た動作をするように設計されています。このチャートを使用すると、トレンド、潜在的反転ポイントなどのテクニカル分析パターンを見極めることができます。



先頭に移動

HighLowOpenClose チャート

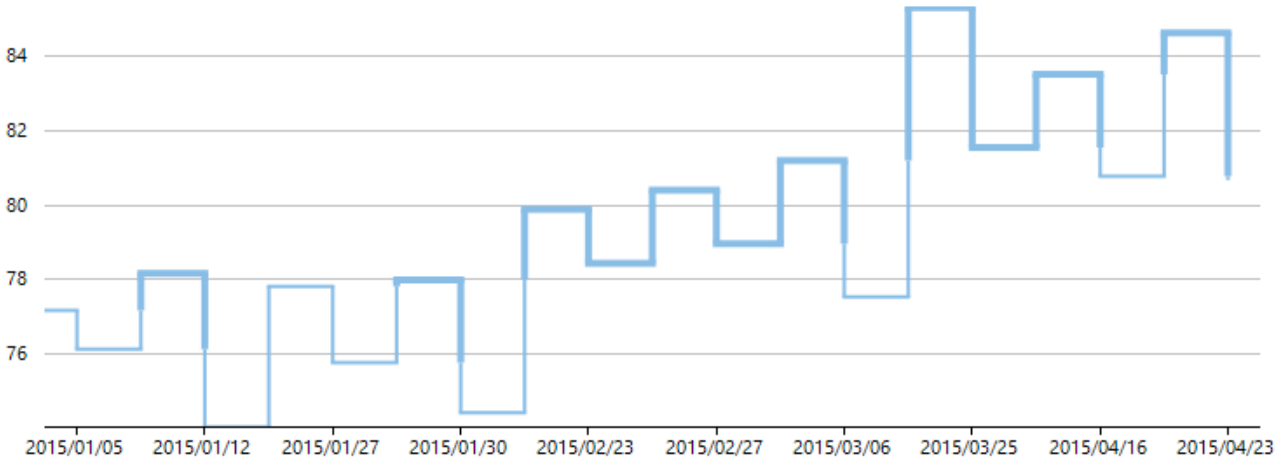
HighLowOpenClose チャートは 4 つの独立した値を組み合わせて、高値、安値、始値、終値のデータ値を系列内のデータポイントごとに提供します。ローソク足チャートと同じ情報を表示しますが、始値が左向きの線で、終値が右向きの線で表されます。



先頭に移動

カギ足チャート

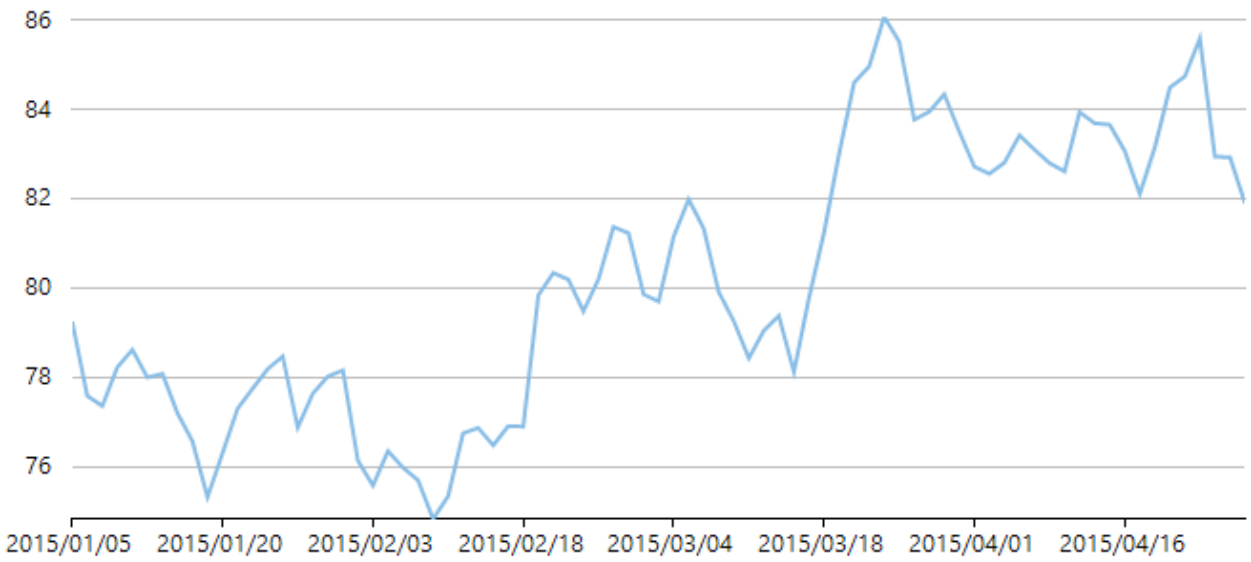
カギ足チャートは、一連の垂直線を連結して、需要と供給のトレンドを示します。線の太さと方向は、株価の動きによって決まります。終値が直前の終値と同じ方向に進む場合は、そのカギ足ラインが延長されます。しかし、終値があらかじめ設定した反転幅以上に反転した場合は、次の列に、新しいカギ足ラインが反対方向に引かれます。細い線は、価格が直前の底値を下回ったこと(売り)を、太い線は、価格が直前の高値を上回ったこと(買い)を示します。



先頭に移動

折れ線グラフ

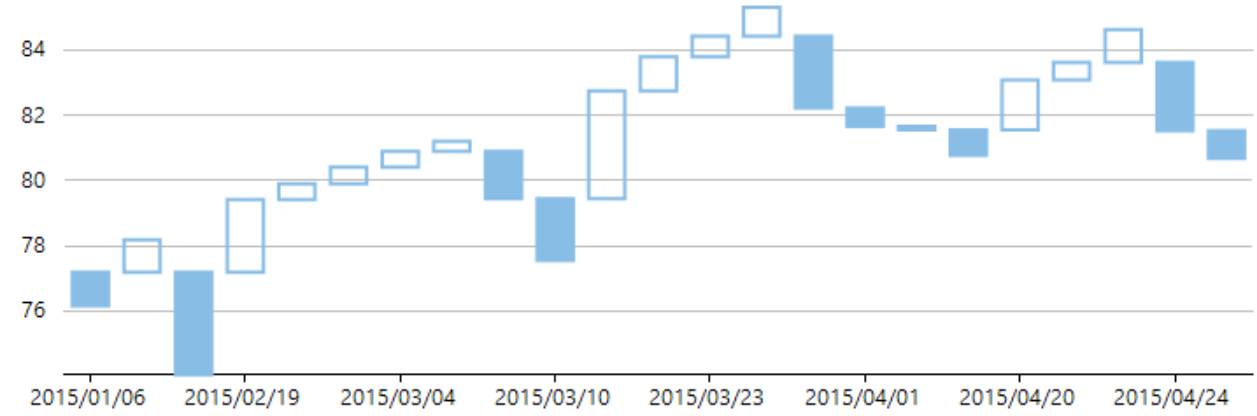
折れ線グラフは、系列内の異なるデータポイントを直線で接続することで、一定期間の傾向を表示します。入力を X 軸に沿って等間隔に並ぶカテゴリ情報として取り扱います。



先頭に移動

新値足チャート

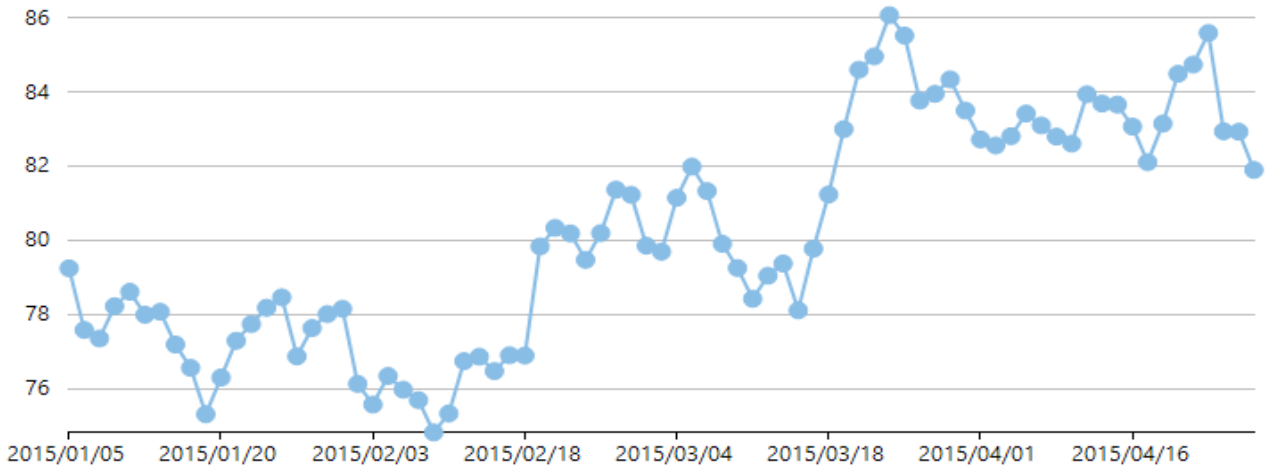
新値足チャートは、垂直のボックスまたはラインを使用して、資産や市場の価格変動を示します。値動きは、ボックスの色とスタイルで表されます。直前のボックスのトレンドが続く値動きは同じ色で表され、反対方向のトレンドを示す値動きは異なる色/スタイルで示されます。反対方向のトレンドは、値が直前 n 個のボックスまたはラインの最大/最小値を超えた場合にのみ描画されます。この個数は、newLineBreaks オプションで決定されます。



先頭に移動

折れ線シンボルグラフ

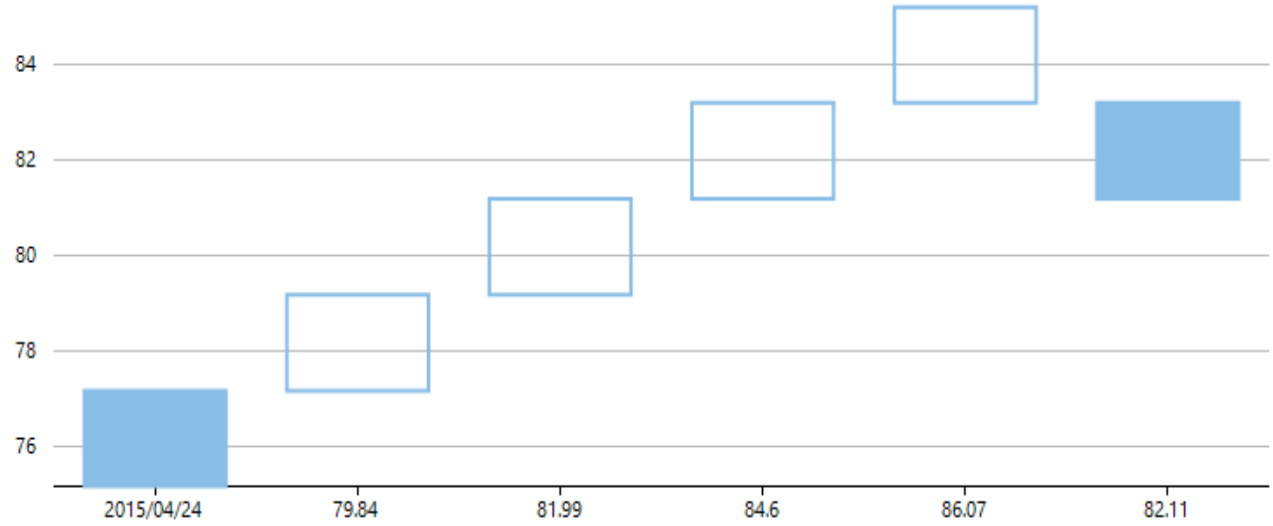
折れ線シンボルグラフは、折れ線グラフと散布図グラフを組み合わせたグラフです。等間隔に並ぶデータの傾向を表示し、同じイベントに関連付けられた 2 つの変数の関係を視覚化します。シンボルを使用してデータポイントをプロットし、データポイント間を直線で接続します。



先頭に移動

練行足チャート

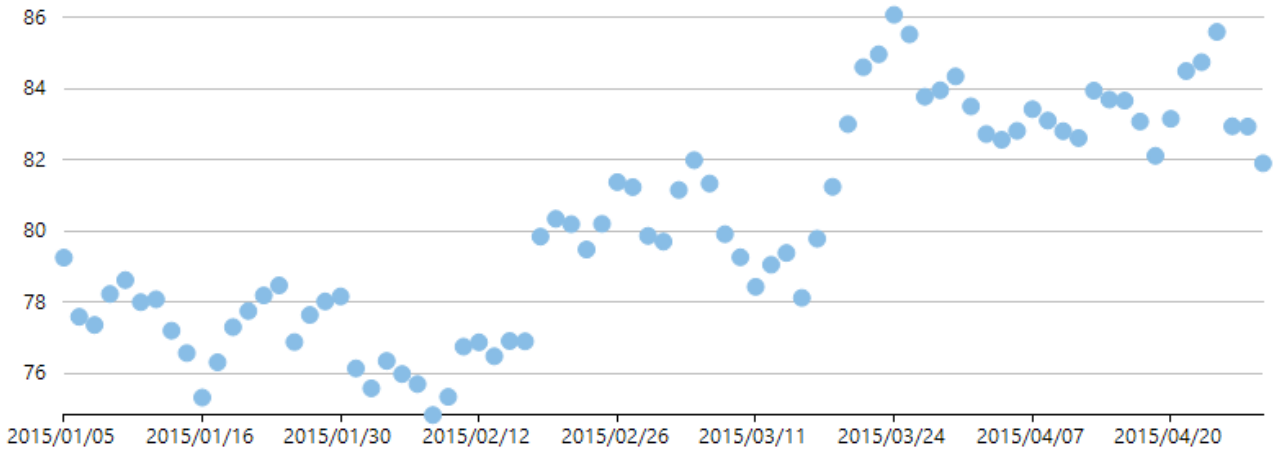
練行足チャートは時間を無視し、指定された金額に一致する価格の変化に注目します。一様なサイズのブロックを使用して株価の動きを示します。価格が、新しいブロックの描画に必要なあらかじめ設定された boxSize オプションより大きい値または小さい値に変化すると、次の列に新しいブロックが描画されます。ボックスの色と方向の変化は、トレンドの反転を示します。



先頭に移動

散布図

散布図グラフは別名 XY グラフと呼ばれ、複数のデータ系列の項目間の関係を表します。簡単に言えば、X 値と Y 値を 2 つの軸にプロットしたものです。データポイントは接続されず、異なるシンボルを使用してカスタマイズできます。通常、このチャートタイプは科学的データを表現するために使用され、予測データや結果データに含まれる集中データのばらつきを強調できます。



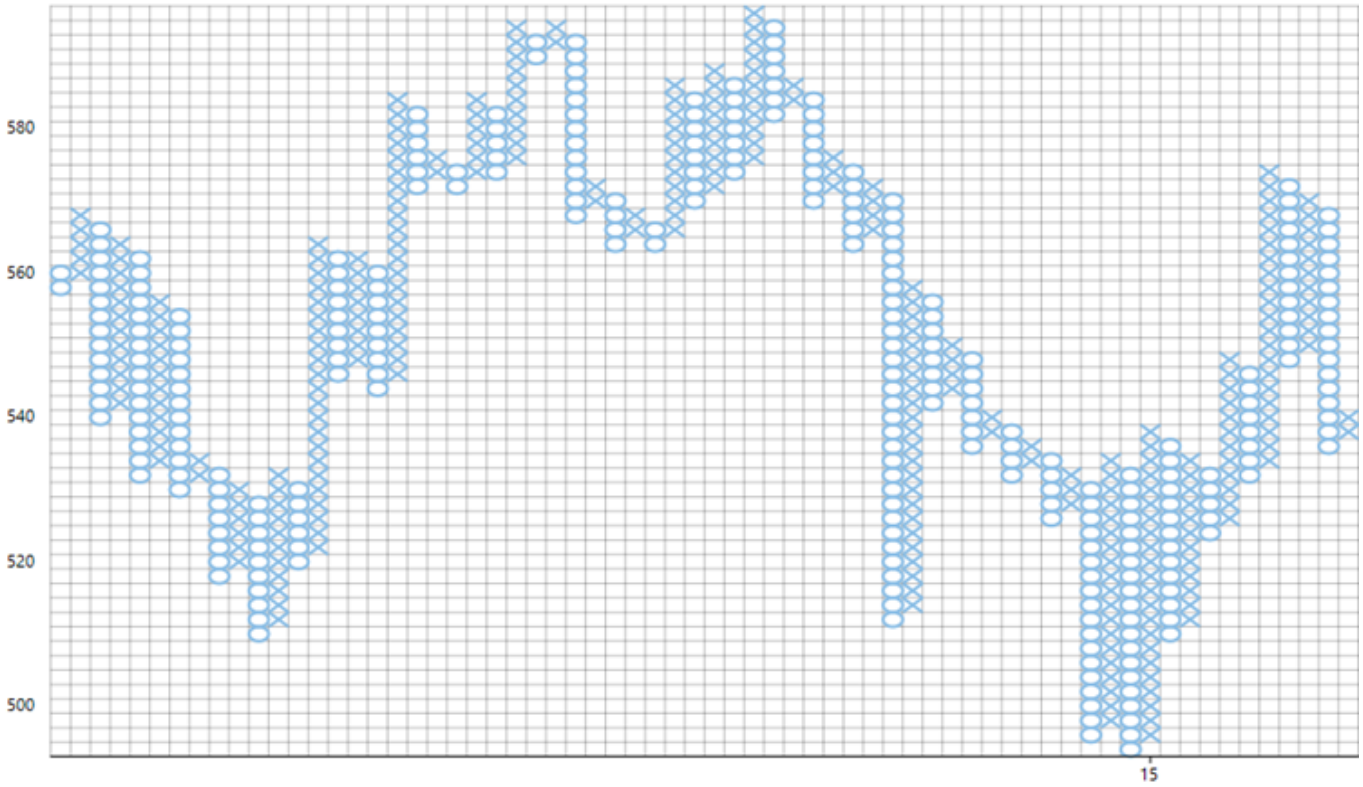
先頭に移動

ポイントアンドフィギュアチャート

ポイントアンドフィギュアチャートは、積み重ねた X または O を列にして並べて株価のトレンドを明示します。時間ベースのチャートとは異なり、このチャートは、時間を考慮せずに株価の変動だけに注目します。X が株価の上昇を表し、O が株価の下落を表します。

ボックスサイズを指定できます。ボックスサイズは、1 つの X と O がそれぞれ表す額です。ボックスサイズの額だけ株価が上昇すると、X が記録されます。さらに上昇すると、その上にもう 1 つ X が積み重ねて記録されます。一方、逆の傾向を示すと、ボックスサイズの額だけ株価が下落したところで、次の列に O が記録されます。すべての株価の変動を記録しても冗長なだけなので、反転幅を設定して、株価の反転を記録する 1 列内の最小の X または 0 の数を指定できます。指定した反転幅になるまでは、株価の反転が発生しても無視されます。

FinancialChart は、ポイントアンドフィギュアチャートの標準スケーリング、固定スケーリング、動的スケーリングをサポートします。標準スケーリングは、ChartCraft よって事前定義された株価範囲テーブルを使用してボックスのサイズを定義します。固定スケーリングでは、エンドユーザーがボックスサイズを指定でき、動的スケーリングでは、計算された ATR 値をボックスサイズとして使用します。



先頭に移動

分析機能

FinancialChart for WPF は、金融データをシステマティックに効率よく分析できるさまざまな分析機能を提供します。分析機能には、傾向線、移動平均、各種テクニカルインジケータがあります。これらのツールを使用して、予測の問題点を分析したり、データ全体を検証したり、資産の市場方向性を予測することができます。

FinancialChart 分析機能の詳細については、以下のリンクをクリックしてください。

- 傾向線
- 移動平均
- インジケータ
- オーバーレイ
- フィボナッチツール

傾向線

傾向線は、テクニカル分析によってさまざまな傾向を特定および確認するために重要なツールです。2 つ以上の価格ポイントを直線で接続する傾向線は、将来の抵抗線にも支持線にもなります。基本的に傾向線は、データの傾向を表し、予測の問題点を検討するために使用されます。傾向線は株価チャートでよく使用されますが、株価のテクニカル分析で使用される傾向インジケータである MACD（移動平均収束発散法）や、金融市場の分析で使用されるテクニカルインジケータである RSI（相対力指数）などのテクニカル分析グラフと組み合わせて使用できます。

サポートされる FitType は、**FitType** プロパティを使用して設定できます。このプロパティは **FitType** 列挙に含まれる値を受け取ります。各トレンドタイプは、そのタイプの計算式に基づいて描画されます。

タイプ	説明
Average X	チャートデータから X の平均値を計算し、傾向線を描画します。
Average Y	チャートデータから Y の平均値を計算し、傾向線を描画します。
Exponential	データ値の増減が徐々に大きくなっていく場合に使用すると便利な曲線です。データにゼロまたは負の値がある場合は、指数傾向線は作成できません。
Fourier	波のような関数を単純な正弦波の組み合わせとして表示する方法です。フーリエ級数式を使用して作成されます。
Linear	線形傾向線は、最高の適合度を持つ直線です。データポイントパターンが直線状であれば、データは線形です。R-2 乗値が 1 またはそれに近い場合は線形傾向線がフィットします。
Logarithmic	最高の適合度を持つ曲線で、データの視覚効果を高めるために使用されます。データの変化率が急速に増加または減少してから平坦化する場合に使用されます。正の値も負の値も使用できます。
Max X	チャートから X の最大値を取り、それを使用して傾向線を描画します。
Max Y	チャートから Y の最大値を取り、それを使用して傾向線を描画します。
Min X	チャートから X の最小値を取り、それを使用して傾向線を描画します。
Min Y	チャートから Y の最小値を取り、それを使用して傾向線を描画します。
Polynomial	データが振幅の形状を表す場合に使用されるねじれた線です。大きなデータセットに対して損益を分析する際に便利です。

Power	一定の率で増加する計算式に沿ったデータセットでの使用に適しています。たとえば、1 秒ごとの車両の加速量などです。

Trendline インスタンスを作成し、**Trendline** クラスの**FitType**、**Order**、**SampleCount** などのプロパティを設定します。傾向線のインスタンスを**Series** コレクションに追加します。

XAML

```
<c1:C1FinancialChart Binding="Sales"
                     BindingX="Date"
                     x:Name="financialChart"
                     ChartType="LineSymbols"
                     ItemsSource="{Binding DataContext.Data}"
                     HorizontalAlignment="Left"
                     Height="321"
                     VerticalAlignment="Top"
                     Width="620"
                     Margin="79,85,0,0">
    <c1:FinancialSeries AxisX="{x:Null}"
                        AxisY="{x:Null}"
                        SeriesName="{x:Null}">
    </c1:FinancialSeries>
    <c1:TrendLine FitType="Fourier"
                  Order="10"
                  SampleCount="150"
                  x:Name="trendLine" />
</c1:C1FinancialChart>
```

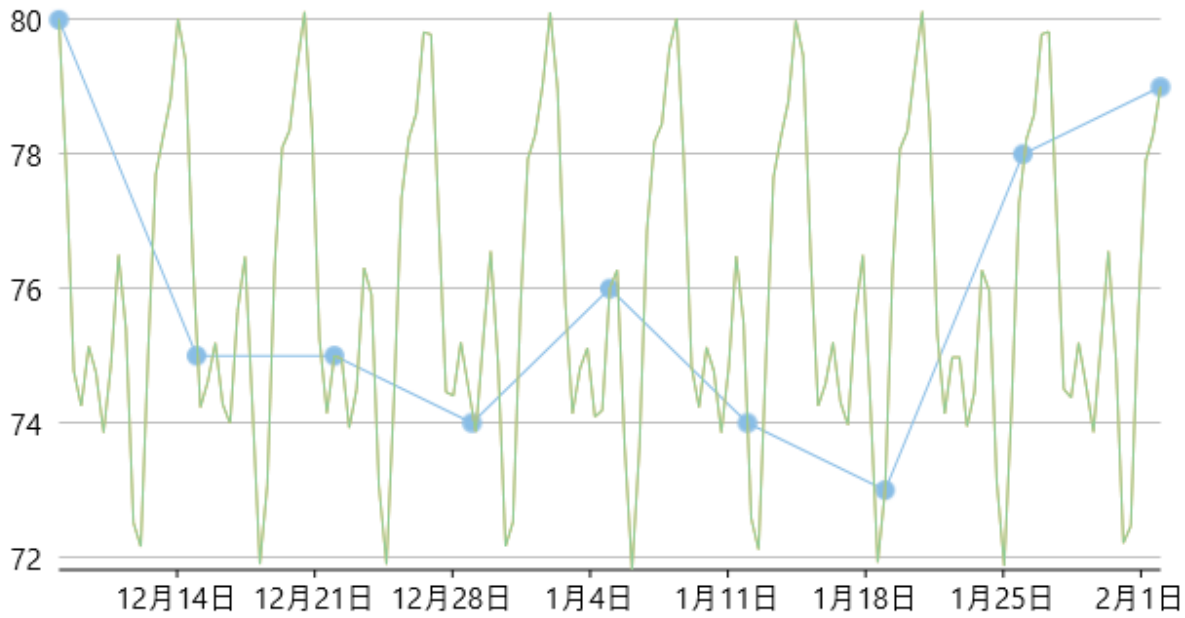
Code

Visual Basic copyCode

```
' TrendLineクラスのインスタンスを作成します
Dim trendline As New C1.WPF.Chart.TrendLine()

' Trendlineインスタンスのプロパティを設定します
trendline.FitType = C1.Chart.FitType.Fourier
trendline.SampleCount = 150
trendline.Order = 10

' 傾向線をFinancial Chart Seriesコレクションに追加します
financialChart.Series.Add(trendline)
```

C# copyCode

```
// TrendLineクラスのインスタンスを作成します
C1.WPF.Chart.TrendLine trendline = new C1.WPF.Chart.TrendLine();

// Trendlineインスタンスのプロパティを設定します
trendline.FitType = C1.Chart.FitType.Fourier;
trendline.SampleCount = 150;
trendline.Order = 10;

// 傾向線をFinancial Chart Seriesコレクションに追加します
financialChart.Series.Add(trendline);
```



移動平均

移動平均は、株価チャートで使用される、移動平均による傾向線です。データセット全体の一部の平均値から系列を作成することで、データポイントを分析します。

FinancialChart では、**MovingAverage** クラスのインスタンスを作成し、**Type** プロパティを **MovingAverageType** 列挙に含まれる次の値のいずれかに設定します。

- **Exponential**: 直近 n 個の値の加重平均で、重み付けは 1 つ前の値と比較して指数関数的に減っていきます。
- **Simple**: 直近 n 個の値の平均。
- **Triangular**: 直近 n 個の値の加重平均で、結果は 2 回の平滑化を行った単純移動平均と同じです。
- **Weighted**: 直近 n 個の値の加重平均で、重み付けは 1 つ前の値と比較して 1 ずつ減っていきます。

ChartType プロパティを設定して、移動平均のチャートタイプを指定できます。このプロパティは、**FinancialChartType** 列挙に含まれる値を受け取ります。チャートタイプの詳細については、「株価チャートタイプ」を参照してください。

さらに、**Period** プロパティを使用して、移動平均の期間を指定できます。これらのプロパティを設定したら、移動平均を**Series** コレクションに追加します。

XAML

```
<c1:C1FinancialChart Binding="Sales"
                     BindingX="Date"
                     x:Name="financialChart"
                     ItemsSource="{Binding DataContext.Data}"
                     HorizontalAlignment="Left"
                     Height="321"
                     VerticalAlignment="Top"
                     Width="620"
                     Margin="79,85,0,0">
    <c1:FinancialSeries ChartType="LineSymbols"
```

```
                        AxisX="{x:Null}"
                        AxisY="{x:Null}"
                        SeriesName="{x:Null}">
    </c1:FinancialSeries>
    <c1:MovingAverage x:Name="ma"
                      Type="Weighted"
                      ChartType="Line"
                      Period="2"/>
</c1:C1FinancialChart>
```

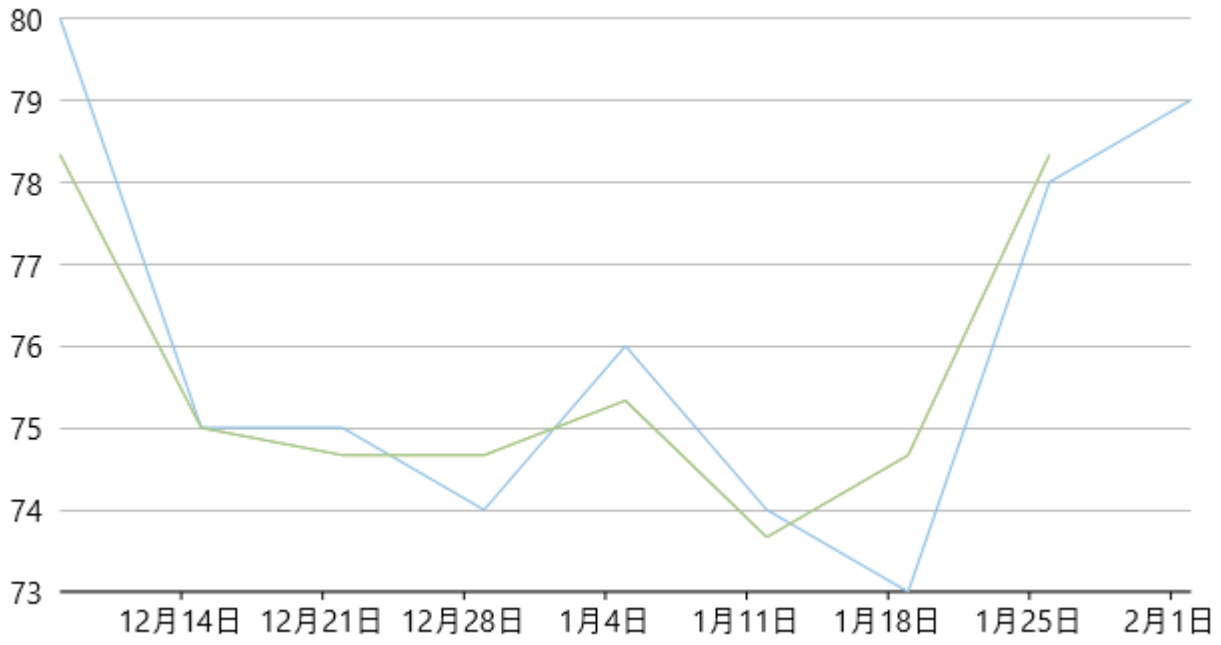
Code

Visual Basic copyCode

```
' MovingAverageクラスのインスタンスを作成します
Dim ma As New C1.WPF.Chart.Finance.MovingAverage()

' 移動平均インスタンスのプロパティを設定します
ma.Type = C1.Chart.MovingAverageType.Weighted
ma.ChartType = C1.Chart.Finance.FinancialChartType.Line
ma.Period = 2

' Seriesコレクションに移動平均インスタンスを追加します
financialChart.Series.Add(ma)
```

C# copyCode

```
// MovingAverageクラスのインスタンスを作成します
C1.WPF.Chart.Finance.MovingAverage ma = new C1.WPF.Chart.Finance.MovingAverage();

// 移動平均インスタンスのプロパティを設定します
ma.Type = C1.Chart.MovingAverageType.Weighted;
ma.ChartType = C1.Chart.Finance.FinancialChartType.Line;
ma.Period = 2;

// Seriesコレクションに移動平均インスタンスを追加します
financialChart.Series.Add(ma);
```



インジケータ

テクニカルインジケータは、元のデータセットに 1 つ以上の数式を適用して計算することにより派生したデータのセットです。テクニカルインジケータは、一般に資産の市場の方向を予測するために使用され、通常は元のデータと Y 軸のスケールが異なるため、別にプロットされます。

WPF 版 は、財務アプリケーションで簡単に使用できる、FinancialChart コントロール用のテクニカルインジケータをサポートしています。これらの株価インジケータは、チャートパターンとしてプロットされ、テクニカル分析の基礎を形成します。

テクニカルインジケータの Y 軸のスケールは株価または出来高チャートのデータのスケールとは異なるため、一般にインジケータは、元の株価または出来高のデータとは別にプロットされることに注意してください。

以下のセクションでは、FinanicalChart for WPF でサポートされているさまざまな株価チャートインジケータについて説明します。

- ATR
- RSI
- CCI
- Williams %R
- Stochastic
- MACD

ATR

ATR（Average True Range）は、資産のボラティリティを測定するテクニカルインジケータです。価格の動向を示すのではなく、価格のボラティリティの程度を示します。ATR は、通常、14 期間に基づいて、日中、日、週、または月単位で計算できます。ボラティリティが高い株は ATR が高くなる一方、ボラティリティが低い株は ATR が低くなります。

また、FinancialChart では、実行時に **GetValues()** メソッドを使用して、計算された ATR 値を取得できます。これにより、アプリケーションでアラートを作成したり、動的データを使用する際にログを取ることができます。

次のコードスニペットは、**ATR** クラスのインスタンスを作成して、Average True Indicator を使用します。また、このサンプルはクラス **DataService** を使用して、株価チャートのデータを取得します。

- **DataService.vb**

```
Public Class DataService
    Private _companies As New List(Of Company)()
    Private _cache As New Dictionary(Of String, List(Of Quote))()

    Private Sub New()
        _companies.Add(New Company() With {
            Key.Symbol = "box",
            Key.Name = "Box Inc"
        })
        _companies.Add(New Company() With {
            Key.Symbol = "fb",
            Key.Name = "Facebook"
        })
    End Sub

    Public Function GetCompanies() As List(Of Company)
        Return _companies
    End Function

    Public Function GetSymbolData(symbol As String) As List(Of Quote)
        If Not _cache.Keys.Contains(symbol) Then
            Dim path As String = String.Format("FinancialChartExplorer.Resources.{0}.json", symbol)
            Dim stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path)
            Dim ser = New DataContractJsonSerializer(GetType(Quote()))
            Dim data = DirectCast(ser.ReadObject(stream), Quote())
            _cache.Add(symbol, data.ToList())
        End If

        Return _cache(symbol)
    End Function

    Shared _ds As DataService
    Public Shared Function GetService() As DataService
        If _ds Is Nothing Then
            _ds = New DataService()
        End If
        Return _ds
    End Function
End Class
```

- **DataService.cs**

```
public class DataService
{
    List<Company> _companies = new List<Company>();
    Dictionary<string, List<Quote>> _cache = new Dictionary<string, List<Quote>>();

    private DataService()
    {
        _companies.Add(new Company() { Symbol = "box", Name = "Box Inc" });
        _companies.Add(new Company() { Symbol = "fb", Name = "Facebook" });
    }

    public List<Company> GetCompanies()
    {
        return _companies;
    }

    public List<Quote> GetSymbolData(string symbol)
    {
        if (!_cache.Keys.Contains(symbol))
        {
            string path = string.Format("FinancialChartExplorer.Resources.{0}.json", symbol);
            var stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path);
            var ser = new DataContractJsonSerializer(typeof(Quote[]));
            var data = (Quote[])ser.ReadObject(stream);
            _cache.Add(symbol, data.ToList());
        }

        return _cache[symbol];
    }
```

```
    static DataService _ds;
    public static DataService GetService()
    {
        if (_ds == null)
            _ds = new DataService();
        return _ds;
    }
}
```

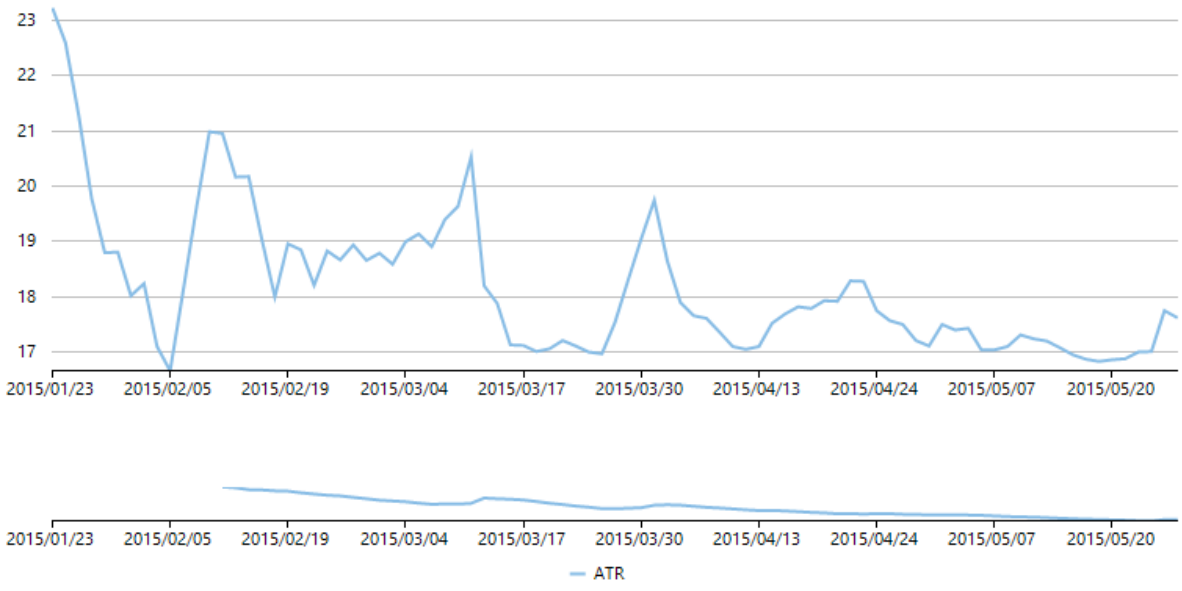
- **Visual Basic**

```
Partial Public Class Indicators
    Inherits UserControl

    Private dataService As DataService = DataService.GetService()
    Private atr As New ATR() With {
        Key.SeriesName = "ATR"
    }

    Public Sub New()
        InitializeComponent()
    End Sub

    Public ReadOnly Property Data() As List(Of Quote)
        Get
            Return dataService.GetSymbolData("box")
        End Get
    End Property

    Public ReadOnly Property IndicatorType() As List(Of String)
        Get
            Return New List(Of String)() From {
                "Average True Range"
            }
        End Get
    End Property

    Private Sub OnIndicatorTypeSelectionChanged(sender As Object, e As SelectionChangedEventArgs)
        Dim ser As FinancialSeries = Nothing
        If cbIndicatorType.SelectedIndex = 0 Then
            ser = atr
        End If


        If ser IsNot Nothing AndAlso Not indicatorChart.Series.Contains(ser) Then
            indicatorChart.BeginUpdate()
            indicatorChart.Series.Clear()
            indicatorChart.Series.Add(ser)
            indicatorChart.EndUpdate()
        End If
    End Sub

    Private Sub OnFinancialChartRendered(sender As Object, e As C1.WPF.Chart.RenderEventArgs)
        If indicatorChart IsNot Nothing Then
            indicatorChart.AxisX.Min = DirectCast(financialChart.AxisX, IAxis).GetMin()
            indicatorChart.AxisX.Max = DirectCast(financialChart.AxisX, IAxis).GetMax()
        End If
    End Sub
End Class
```

- **C#**

```
public partial class Indicators : UserControl
{

    DataService dataService = DataService.GetService();
    ATR atr = new ATR() { SeriesName = "ATR" };

    public Indicators()
    {
        InitializeComponent();
    }
```

```
    public List<Quote> Data
    {
        get
        {
            return dataService.GetSymbolData("box");
        }
    }

    public List<string> IndicatorType
    {
        get
        {
            return new List<string>()
            {
                "Average True Range",
            };
        }
    }

    void OnIndicatorTypeSelectionChanged(object sender, SelectionChangedEventArgs e)
    {
        FinancialSeries ser = null;
        if (cbIndicatorType.SelectedIndex == 0)
            ser = atr;


        if (ser != null && !indicatorChart.Series.Contains(ser))
        {
            indicatorChart.BeginUpdate();
            indicatorChart.Series.Clear();
            indicatorChart.Series.Add(ser);
            indicatorChart.EndUpdate();
        }
    }

    void OnFinancialChartRendered(object sender, C1.WPF.Chart.RenderEventArgs e)
    {
        if (indicatorChart != null)
        {
            indicatorChart.AxisX.Min = ((IAxis)financialChart.AxisX).GetMin();
            indicatorChart.AxisX.Max = ((IAxis)financialChart.AxisX).GetMax();
        }
    }
}
```



RSI

FinancialChart の RSI（相対力指数）インジケータは、株価の動きの速さと大きさを測定するモメンタムオシレータです。資産の終値の上向きの動きを取引期間中の下向きの動きと比較して、株の強さまたは弱さを判断します。値は 0 ～ 100 の間で変動します。プラスの変化が強い株の方が、マイナスの変化が強い株より RSI が高くなります。

最新の評価益の大きさを最新の評価損と比較する際に RSI を応用することで、資産の買われ過ぎ/売られ過ぎの状況を判断できます。株は、RSI が 70 を超えると買われ過ぎ、30 を下回ると売られ過ぎと見なされます。

また、FinancialChart では、実行時に**GetValues()** メソッドを使用して、計算された RSI 値を取得できます。これにより、アプリケーションでアラートを作成したり、動的データを使用する際にログを取ることができます。

コードスニペットは、クラス DataService を使用します。コードを確認するには、ATR を参照してください。さらに、このサンプルは**RSI** クラスのインスタンスを作成して、RSI を使用します。

- **DataService.vb**

```
Public Class DataService
    Private _companies As New List(Of Company)()
    Private _cache As New Dictionary(Of String, List(Of Quote))()

    Private Sub New()
        _companies.Add(New Company() With {
            Key.Symbol = "box",
            Key.Name = "Box Inc"
        })
        _companies.Add(New Company() With {
            Key.Symbol = "fb",
            Key.Name = "Facebook"
        })
    End Sub

    Public Function GetCompanies() As List(Of Company)
        Return _companies
    End Function

    Public Function GetSymbolData(symbol As String) As List(Of Quote)
        If Not _cache.Keys.Contains(symbol) Then
            Dim path As String = String.Format("FinancialChartExplorer.Resources.{0}.json", symbol)
            Dim stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path)
            Dim ser = New DataContractJsonSerializer(GetType(Quote()))
            Dim data = DirectCast(ser.ReadObject(stream), Quote())
            _cache.Add(symbol, data.ToList())
        End If

        Return _cache(symbol)
    End Function

    Shared _ds As DataService
    Public Shared Function GetService() As DataService
        If _ds Is Nothing Then
            _ds = New DataService()
        End If
        Return _ds
    End Function
End Class
```

- **DataService.cs**

```
public class DataService
{
    List<Company> _companies = new List<Company>();
    Dictionary<string, List<Quote>> _cache = new Dictionary<string, List<Quote>>();

    private DataService()
    {
        _companies.Add(new Company() { Symbol = "box", Name = "Box Inc" });
        _companies.Add(new Company() { Symbol = "fb", Name = "Facebook" });
    }

    public List<Company> GetCompanies()
    {
```

```
        return _companies;
    }

    public List<Quote> GetSymbolData(string symbol)
    {
        if (!_cache.Keys.Contains(symbol))
        {
            string path = string.Format("FinancialChartExplorer.Resources.{0}.json", symbol);
            var stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path);
            var ser = new DataContractJsonSerializer(typeof(Quote[]));
            var data = (Quote[])ser.ReadObject(stream);
            _cache.Add(symbol, data.ToList());
        }

        return _cache[symbol];
    }

    static DataService _ds;
    public static DataService GetService()
    {
        if (_ds == null)
            _ds = new DataService();
        return _ds;
    }
}
```

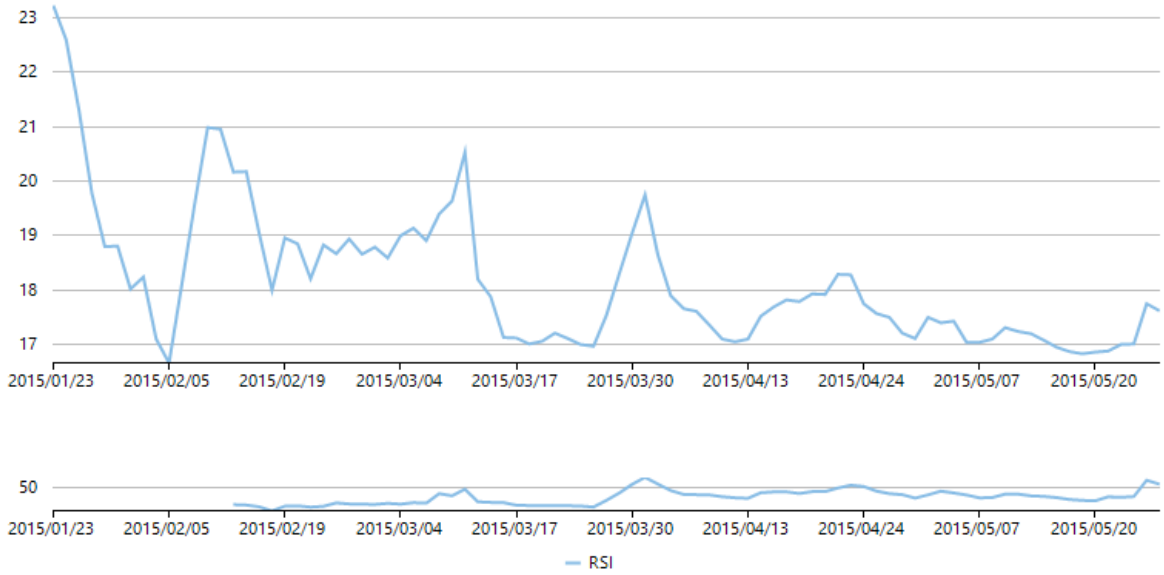
- **Visual Basic**

```
Partial Public Class Indicators
    Inherits UserControl

    Private dataService As DataService = DataService.GetService()
    Private rsi As New RSI() With {
        Key.SeriesName = "RSI"
    }

    Public Sub New()
        InitializeComponent()
    End Sub

    Public ReadOnly Property Data() As List(Of Quote)
        Get
            Return dataService.GetSymbolData("box")
        End Get
    End Property

    Public ReadOnly Property IndicatorType() As List(Of String)
        Get
            Return New List(Of String)() From {
                "Relative Strength Index"
            }
        End Get
    End Property

    Private Sub OnIndicatorTypeSelectionChanged(sender As Object, e As SelectionChangedEventArgs)
        Dim ser As FinancialSeries = Nothing
        If cbIndicatorType.SelectedIndex = 0 Then
            ser = rsi
        End If


        If ser IsNot Nothing AndAlso Not indicatorChart.Series.Contains(ser) Then
            indicatorChart.BeginUpdate()
            indicatorChart.Series.Clear()
            indicatorChart.Series.Add(ser)
            indicatorChart.EndUpdate()
        End If
    End Sub

    Private Sub OnFinancialChartRendered(sender As Object, e As C1.WPF.Chart.RenderEventArgs)
        If indicatorChart IsNot Nothing Then
            indicatorChart.AxisX.Min = DirectCast(financialChart.AxisX, IAxis).GetMin()
```

```
            indicatorChart.AxisX.Max = DirectCast(financialChart.AxisX, IAxis).GetMax()
        End If
    End Sub
End Class
```

- **C#**

```
public partial class Indicators : UserControl
{

    DataService dataService = DataService.GetService();
    RSI rsi = new RSI() { SeriesName = "RSI" };

    public Indicators()
    {
        InitializeComponent();
    }

    public List<Quote> Data
    {
        get
        {
            return dataService.GetSymbolData("box");
        }
    }

    public List<string> IndicatorType
    {
        get
        {
            return new List<string>()
            {
                "Relative Strength Index",
            };
        }
    }

    void OnIndicatorTypeSelectionChanged(object sender, SelectionChangedEventArgs e)
    {
        FinancialSeries ser = null;
        if (cbIndicatorType.SelectedIndex == 0)
            ser = rsi;


        if (ser != null && !indicatorChart.Series.Contains(ser))
        {
            indicatorChart.BeginUpdate();
            indicatorChart.Series.Clear();
            indicatorChart.Series.Add(ser);
            indicatorChart.EndUpdate();
        }
    }

    void OnFinancialChartRendered(object sender, C1.WPF.Chart.RenderEventArgs e)
    {
        if (indicatorChart != null)
        {
            indicatorChart.AxisX.Min = ((IAxis)financialChart.AxisX).GetMin();
            indicatorChart.AxisX.Max = ((IAxis)financialChart.AxisX).GetMax();
        }
    }
}
```



CCI

CCI（商品チャンネル指数）インジケータは、資産の現在の時価レベルを、指定された期間の平均時価と比較して測定するオシレータです。新しいトレンドを判断したり、極端な状況について警告するために使用されます。

FinancialChart では、**CCI** オブジェクトを使用して CCI を使用する必要があります。また、FinancialChart では、実行時に **GetValues()** メソッドを使用して、計算された CCI 値を取得できます。これにより、アプリケーションでアラートを作成したり、動的データを使用する際にログを取ることができます。

CCI インジケータの使用方法については、次のコードスニペットを参照してください。このコードスニペットは、クラス **DataService** を使用します。このコードは、ATR で確認できます。

- **Visual Basic**

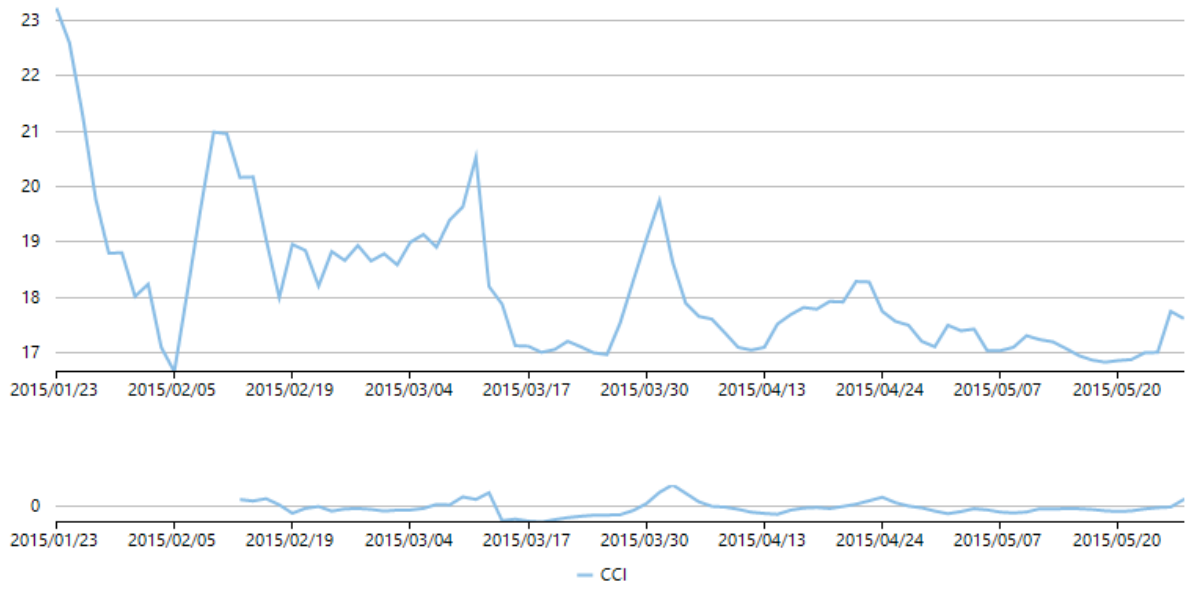
```
Partial Public Class Indicators
    Inherits UserControl

    Private dataService As DataService = dataService.GetService()
    Private cci As New CCI() With {
        Key.SeriesName = "CCI"
    }

    Public Sub New()
        InitializeComponent()
    End Sub

    Public ReadOnly Property Data() As List(Of Quote)
        Get
            Return dataService.GetSymbolData("box")
        End Get
    End Property

    Public ReadOnly Property IndicatorType() As List(Of String)
        Get
            Return New List(Of String)() From {
                "Commodity Channel Index"
            }
        End Get
    End Property

    Private Sub OnIndicatorTypeSelectionChanged(sender As Object, e As SelectionChangedEventArgs)
        Dim ser As FinancialSeries = Nothing
        If cbIndicatorType.SelectedIndex = 0 Then
            ser = cci
        End If
```

```
        If ser IsNot Nothing AndAlso Not indicatorChart.Series.Contains(ser) Then
            indicatorChart.BeginUpdate()
            indicatorChart.Series.Clear()
            indicatorChart.Series.Add(ser)
            indicatorChart.EndUpdate()
        End If
    End Sub

    Private Sub OnFinancialChartRendered(sender As Object, e As C1.WPF.Chart.RenderEventArgs)
        If indicatorChart IsNot Nothing Then
            indicatorChart.AxisX.Min = DirectCast(financialChart.AxisX, IAxis).GetMin()
            indicatorChart.AxisX.Max = DirectCast(financialChart.AxisX, IAxis).GetMax()
        End If
    End Sub
End Class
```

- **C#**

```
public partial class Indicators : UserControl
{

    DataService dataService = DataService.GetService();
    CCI cci = new CCI() { SeriesName = "CCI" };

    public Indicators()
    {
        InitializeComponent();
    }

    public List<Quote> Data
    {
        get
        {
            return dataService.GetSymbolData("box");
        }
    }

    public List<string> IndicatorType
    {
        get
        {
            return new List<string>()
            {
                "Commodity Channel Index",
            };
        }
    }

    void OnIndicatorTypeSelectionChanged(object sender, SelectionChangedEventArgs e)
    {
        FinancialSeries ser = null;
        if (cbIndicatorType.SelectedIndex == 0)
            ser = cci;


        if (ser != null && !indicatorChart.Series.Contains(ser))
        {
            indicatorChart.BeginUpdate();
            indicatorChart.Series.Clear();
            indicatorChart.Series.Add(ser);
            indicatorChart.EndUpdate();
        }
    }

    void OnFinancialChartRendered(object sender, C1.WPF.Chart.RenderEventArgs e)
    {
        if (indicatorChart != null)
        {
            indicatorChart.AxisX.Min = ((IAxis)financialChart.AxisX).GetMin();
```

```
            indicatorChart.AxisX.Max = ((IAxis)financialChart.AxisX).GetMax();
        }
    }
}
```



Williams %R

FinancialChart のWilliams %R（ウィリアムズ %R）インジケータは、現在の資産価格を過去の一定の期間中の最高値と比較するモメンタムインジケータです。通常、比較対象の過去の期間は 14 期間です。このインジケータは、0 ～ -100 の間で変動します。これは、短期ストキャスティクスの逆になります。Williams %Rが過去の期間の最高値と比較した株の終値のレベルを表示するのに対して、ストキャスティクスは最安値と比較した株の終値のレベルを表示します。どちらのインジケータも同じ線を示しますが、スケーリングが異なります。Williams %R を買われ過ぎ/売られ過ぎのレベルの判定に応用すると、買いと売りのシグナルを提供したり、モメンタムを確認することができます。

WilliamsR インジケータを使用するには、**WilliamsR** クラスのインスタンスを作成する必要があります。また、FinancialChart では、実行時に **GetValues()** メソッドを使用して、計算された WilliamsR 値を取得できます。これにより、アプリケーションでアラートを作成したり、動的データを使用する際にログを取ることができます。

次のコードスニペットは、**WilliamsR** クラスのインスタンスを作成して、このインジケータを使用します。

- **Visual Basic**

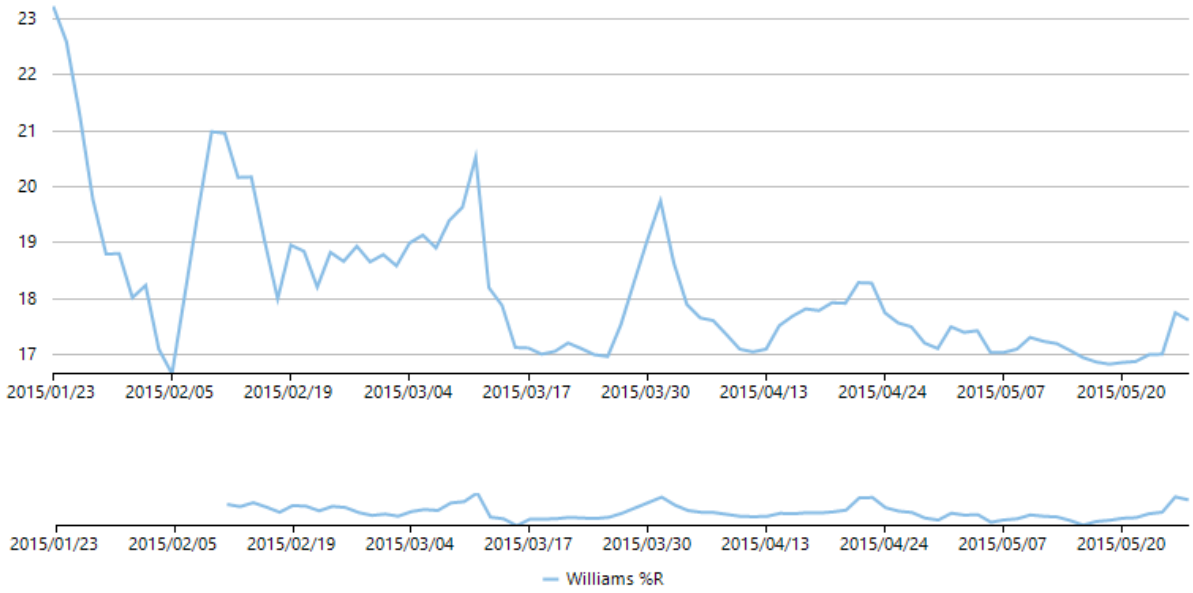
```
Partial Public Class Indicators
    Inherits UserControl

    Private dataService As DataService = dataService.GetService()
    Private wr As New WilliamsR() With {
        Key.SeriesName = "WilliamsR"
    }

    Public Sub New()
        InitializeComponent()
    End Sub

    Public ReadOnly Property Data() As List(Of Quote)
        Get
            Return dataService.GetSymbolData("box")
        End Get
    End Property

    Public ReadOnly Property IndicatorType() As List(Of String)
        Get
            Return New List(Of String)() From {
                "Williams %R"
```

```
            }
        End Get
    End Property

    Private Sub OnIndicatorTypeSelectionChanged(sender As Object, e As SelectionChangedEventArgs)
        Dim ser As FinancialSeries = Nothing
        If cbIndicatorType.SelectedIndex = 0 Then
            ser = wr
        End If


        If ser IsNot Nothing AndAlso Not indicatorChart.Series.Contains(ser) Then
            indicatorChart.BeginUpdate()
            indicatorChart.Series.Clear()
            indicatorChart.Series.Add(ser)
            indicatorChart.EndUpdate()
        End If
    End Sub

    Private Sub OnFinancialChartRendered(sender As Object, e As C1.WPF.Chart.RenderEventArgs)
        If indicatorChart IsNot Nothing Then
            indicatorChart.AxisX.Min = DirectCast(financialChart.AxisX, IAxis).GetMin()
            indicatorChart.AxisX.Max = DirectCast(financialChart.AxisX, IAxis).GetMax()
        End If
    End Sub
End Class
```

- **C#**

```
public partial class Indicators : UserControl
{

    DataService dataService = DataService.GetService();
    WilliamsR wr = new WilliamsR() { SeriesName = "WilliamsR" };

    public Indicators()
    {
        InitializeComponent();
    }

    public List<Quote> Data
    {
        get
        {
            return dataService.GetSymbolData("box");
        }
    }

    public List<string> IndicatorType
    {
        get
        {
            return new List<string>()
            {
                "Williams %R"
            };
        }
    }

    void OnIndicatorTypeSelectionChanged(object sender, SelectionChangedEventArgs e)
    {
        FinancialSeries ser = null;
        if (cbIndicatorType.SelectedIndex == 0)
            ser = wr;


        if (ser != null && !indicatorChart.Series.Contains(ser))
        {
            indicatorChart.BeginUpdate();
            indicatorChart.Series.Clear();
            indicatorChart.Series.Add(ser);
```

```
            indicatorChart.EndUpdate();
        }
    }

    void OnFinancialChartRendered(object sender, C1.WPF.Chart.RenderEventArgs e)
    {
        if (indicatorChart != null)
        {
            indicatorChart.AxisX.Min = ((IAxis)financialChart.AxisX).GetMin();
            indicatorChart.AxisX.Max = ((IAxis)financialChart.AxisX).GetMax();
        }
    }
}
```



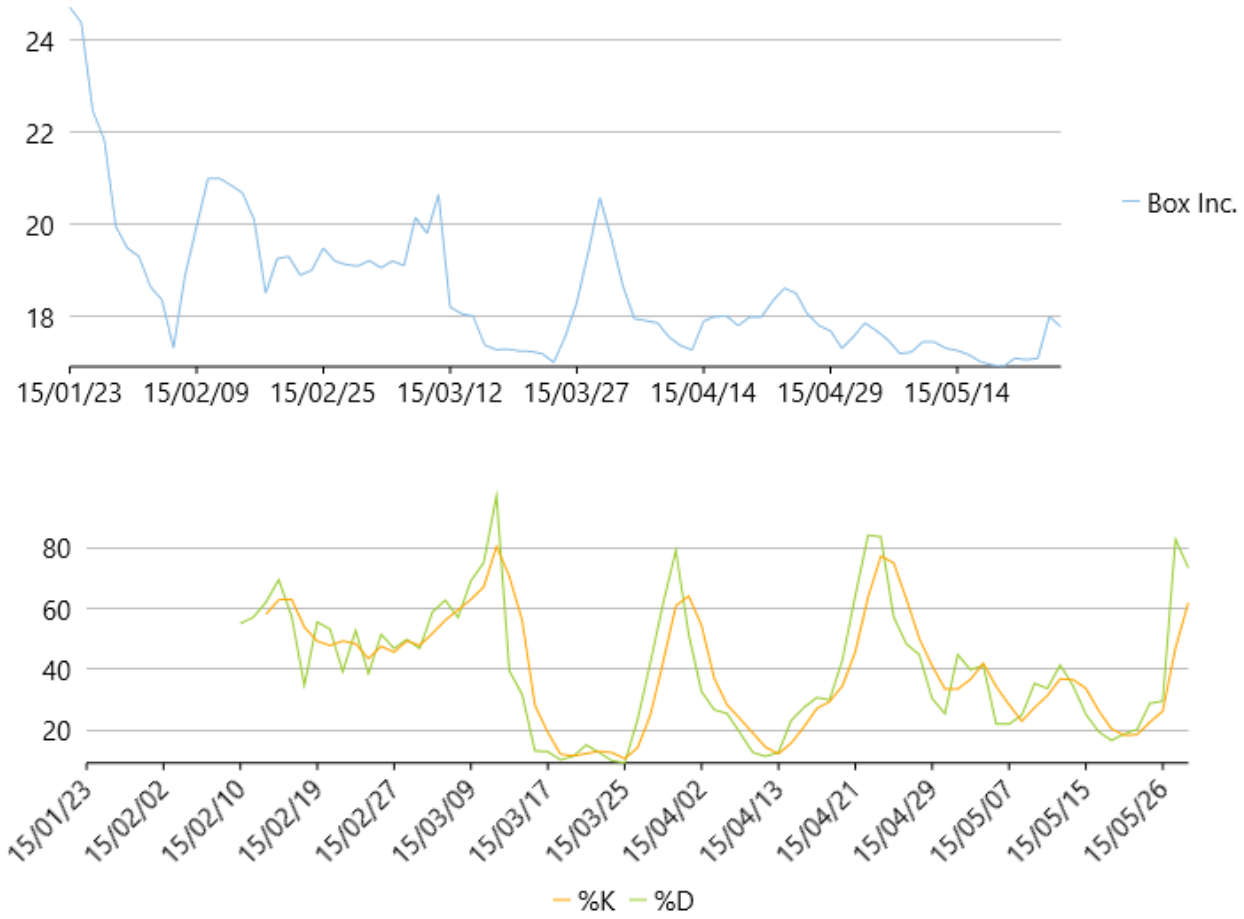
Stochastic

Stochastic（ストキャスティクス）インジケータは、株価のターニングポイントの前兆となるモメンタムインジケータです。金融商品の終値を、一定期間の価格の範囲と比較します。ブルベアセットアップを特定することで将来の反転を予測するために使用できます。

ストキャスティクスインジケータは、K 線と D 線で測定されます。D 線をよく観察することで、FinancialChart から重要なシグナルを発見します。スローストキャスティクスを作成するには、SmoothingPeriod を 3 に設定します。SmoothingPeriod の値を 2 に設定すると、フルストキャスティクスが作成されます。ファストストキャスティクスを作成するには、SmoothingPeriod を整数値 1 に設定します。

FinancialChart にストキャスティクスインジケータを追加するには、FinancialChart コントロールをアプリケーションに追加し、コントロールに適切なデータソースを連結するか、**Quote Collection** でコントロールにデータを追加します。FinancialChart にデータを連結または追加するには、ItemsSource オブジェクトを使用します。**Stochastic** クラスは、**KPeriod**（整数値を受け取り、指定期間の価格範囲を計算する）、**DPeriod**（整数値を受け取り、K 線の移動平均を計算する）、**SmoothingPeriod**（整数値を受け取り、ファスト/フル/スローストキャスティクスを作成する）プロパティを公開します。これらのプロパティの値に基づいて、ストキャスティクスインジケータが計算され、FinancialChart にプロットされます。**KLineStyle** および **DLineStyle** プロパティを利用して、系列の外観を変更できます。

また、FinancialChart では、アプリケーションでアラートを作成したり、動的データの使用中にログを取るために、計算された **D 値**、**Dx 値**、;**K 値**、**Kx 値**を実行時に取得できます。



次の例は、上の図で示すように、企業 Box Inc. の一定期間の株価を考察し、ボリュームチャートとは別にストキャスティクスインジケータをプロットします。この例では、json ファイルのデータを使用します。この json ファイルにアクセスするために、DataService.cs クラスを作成します。

⚠ json ファイルの[ビルドアクション]プロパティが**[埋め込まれたリソース]**に設定されていることを確認します。

PriceChart.xaml copyCode

```
<Window
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:StochasticInd"
        xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
        x:Class="StochasticInd.MainWindow"
        mc:Ignorable="d"
        Title="MainWindow" Height="350" Width="525"
        DataContext="{Binding RelativeSource={RelativeSource Mode=Self}}">
    <Grid>

        <c1:C1FinancialChart x:Name="financialChart"
                             BindingX="Date"
                             Binding="Close"
                             ChartType="Line"
                             ItemsSource="{Binding Data}"
                             ToolTipContent="{}{seriesName}&#x000A;{Date} {y}"
```

```
                            Margin="10,29,10,177"
                            Rendered="OnFinancialChartRendered">

            <c1:FinancialSeries Binding="High,Low,Open,Close"
                                SeriesName="Box Inc." />
            <c1:C1FinancialChart.AxisX>
                <c1:Axis LabelAngle="45" MajorUnit="3"/>
            </c1:C1FinancialChart.AxisX>
        </c1:C1FinancialChart>
```

IndicatorChart.xaml copyCode

```
        <c1:Stochastic x:Name="stochastic"
                       SeriesName="%K,%D"
                       DPeriod="3"
                       KPeriod="13"
                       SmoothingPeriod="1">
            <c1:Stochastic.DLineStyle>
                <c1:ChartStyle Stroke="Orange" />
            </c1:Stochastic.DLineStyle>
            <c1:Stochastic.KLineStyle>
                <c1:ChartStyle Stroke="YellowGreen" />
            </c1:Stochastic.KLineStyle>
        </c1:Stochastic>
        <c1:C1FinancialChart.AxisX>
            <c1:Axis LabelAngle="45" MajorUnit="3"/>
        </c1:C1FinancialChart.AxisX>
    </c1:C1FinancialChart>
</Grid>
```

DataService.cs に次の参照を追加します。

- System.Collections.Generic
- System.Linq
- System.Runtime.Serialization.Json
- System.Reflection

- **DataService.vb**

```
Public Class DataService
    Public Function GetData() As List(Of Quote)
        Dim path As String = "Indicator.Resources.box.json"
        'Indicatorをアプリケーション名で置き換えます。
        Dim stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path)
        Dim ser = New DataContractJsonSerializer(GetType(Quote()))
        Dim data = DirectCast(ser.ReadObject(stream), Quote())
        Return data.ToList()
    End Function
    Shared _ds As DataService
    Public Shared Function GetService() As DataService
        If _ds Is Nothing Then
            _ds = New DataService()
        End If
        Return _ds
    End Function
End Class
```

- **DataService.cs**

```
public class DataService
```

```
{
    public List<Quote> GetData()
    {
        string path = "StochasticInd.Resources.box.json";
        //StochasticIndをアプリケーション名で置き換えます。
        var stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path);
        var ser = new DataContractJsonSerializer(typeof(Quote[]));
        var data = (Quote[])ser.ReadObject(stream);
        return data.ToList();
    }
    static DataService _ds;
    public static DataService GetService()
    {
        if (_ds == null)
            _ds = new DataService();
        return _ds;
    }
}
```

Json Data


```
[
    {"date":"15/01/23","open":20.2,"high":24.73,"low":20.16,"close":23.23,"volume":42593223},
    {"date":"15/01/26","open":23.67,"high":24.39,"low":22.5,"close":22.6,"volume":8677164},
    {"date":"15/01/27","open":22,"high":22.47,"low":21.17,"close":21.3,"volume":3272512},
    {"date":"15/01/28","open":21.62,"high":21.84,"low":19.6,"close":19.78,"volume":5047364},
    {"date":"15/01/29","open":19.9,"high":19.95,"low":18.51,"close":18.8,"volume":3419482},
    {"date":"15/01/30","open":18.47,"high":19.48,"low":18.22,"close":18.81,"volume":2266439},
    {"date":"15/02/02","open":19.18,"high":19.3,"low":18.01,"close":18.02,"volume":2071168},
    {"date":"15/02/03","open":18.22,"high":18.64,"low":18.12,"close":18.24,"volume":1587435},
    {"date":"15/02/04","open":18.2,"high":18.35,"low":17,"close":17.1,"volume":2912224},
    {"date":"15/02/05","open":17.3,"high":17.31,"low":16.41,"close":16.66,"volume":2682187},
    {"date":"15/02/06","open":17.39,"high":18.88,"low":17.21,"close":18.12,"volume":3929164},
    {"date":"15/02/09","open":18.86,"high":19.95,"low":18.45,"close":19.6,"volume":3226650},
    {"date":"15/02/10","open":20.5,"high":21,"low":19.63,"close":20.99,"volume":2804409},
    {"date":"15/02/11","open":20.89,"high":21,"low":20.2,"close":20.96,"volume":1698365},
    {"date":"15/02/12","open":20.66,"high":20.85,"low":19.75,"close":20.17,"volume":1370320},
    {"date":"15/02/13","open":20.19,"high":20.68,"low":20,"close":20.18,"volume":711951},
    {"date":"15/02/17","open":19.5,"high":20.1,"low":18.8,"close":19.05,"volume":2093602},
    {"date":"15/02/18","open":18.31,"high":18.5,"low":17.96,"close":18,"volume":1849490},
    {"date":"15/02/19","open":18.33,"high":19.25,"low":17.91,"close":18.96,"volume":1311518},
    {"date":"15/02/20","open":18.68,"high":19.3,"low":18.65,"close":18.85,"volume":1001692},
    {"date":"15/02/23","open":18.8,"high":18.89,"low":18.11,"close":18.21,"volume":670087},
    {"date":"15/02/24","open":18.46,"high":19,"low":18.27,"close":18.83,"volume":759263},
    {"date":"15/02/25","open":18.83,"high":19.48,"low":18.47,"close":18.67,"volume":915580},
    {"date":"15/02/26","open":18.64,"high":19.2,"low":18.64,"close":18.94,"volume":461283},
    {"date":"15/02/27","open":18.8,"high":19.12,"low":18.55,"close":18.66,"volume":617199},
    {"date":"15/03/02","open":18.66,"high":19.09,"low":18.65,"close":18.79,"volume":519605},
    {"date":"15/03/03","open":18.79,"high":19.21,"low":18.45,"close":18.59,"volume":832415},
    {"date":"15/03/04","open":18.64,"high":19.05,"low":18.32,"close":19,"volume":539688},
    {"date":"15/03/05","open":19.2,"high":19.2,"low":18.8,"close":19.14,"volume":486149},
    {"date":"15/03/06","open":19.03,"high":19.1,"low":18.7,"close":18.91,"volume":685659},
    {"date":"15/03/09","open":18.98,"high":20.15,"low":18.96,"close":19.4,"volume":1321363},
    {"date":"15/03/10","open":19.3,"high":19.8,"low":18.85,"close":19.64,"volume":615743},
    {"date":"15/03/11","open":20.08,"high":20.65,"low":19.24,"close":20.53,"volume":2167167},
    {"date":"15/03/12","open":17.17,"high":18.2,"low":16.76,"close":18.2,"volume":6837638},
    {"date":"15/03/13","open":18.05,"high":18.05,"low":17.3,"close":17.88,"volume":1715629},
    {"date":"15/03/16","open":17.91,"high":18,"low":17.01,"close":17.13,"volume":1321313},
    {"date":"15/03/17","open":17.28,"high":17.37,"low":16.6,"close":17.12,"volume":1272242},
    {"date":"15/03/18","open":17.1,"high":17.27,"low":16.91,"close":17.01,"volume":530063},
    {"date":"15/03/19","open":17,"high":17.28,"low":17,"close":17.06,"volume":536427},
    {"date":"15/03/20","open":17.13,"high":17.24,"low":16.88,"close":17.21,"volume":1320237},
    {"date":"15/03/23","open":17.21,"high":17.23,"low":17.01,"close":17.11,"volume":509798},
    {"date":"15/03/24","open":17.02,"high":17.18,"low":16.82,"close":17,"volume":962149},
    {"date":"15/03/25","open":16.92,"high":16.99,"low":16.82,"close":16.97,"volume":565673},
    {"date":"15/03/26","open":16.83,"high":17.56,"low":16.83,"close":17.54,"volume":884523},
    {"date":"15/03/27","open":17.58,"high":18.3,"low":17.11,"close":18.3,"volume":705626},
    {"date":"15/03/30","open":18.5,"high":19.4,"low":18.4,"close":19.05,"volume":1151620},
```

```
    {"date":"15/03/31","open":19.08,"high":20.58,"low":18.4,"close":19.75,"volume":2020679},
    {"date":"15/04/01","open":19.69,"high":19.69,"low":18.55,"close":18.65,"volume":961078},
    {"date":"15/04/02","open":18.56,"high":18.66,"low":17.85,"close":17.9,"volume":884233},
    {"date":"15/04/06","open":17.78,"high":17.94,"low":17.51,"close":17.66,"volume":605252},
    {"date":"15/04/07","open":17.62,"high":17.9,"low":17.53,"close":17.61,"volume":591988},
    {"date":"15/04/08","open":17.64,"high":17.85,"low":17.32,"close":17.36,"volume":618855},
    {"date":"15/04/09","open":17.33,"high":17.54,"low":17.1,"close":17.1,"volume":761855},
    {"date":"15/04/10","open":17.08,"high":17.36,"low":17,"close":17.05,"volume":568373},
    {"date":"15/04/13","open":17.24,"high":17.26,"low":16.81,"close":17.1,"volume":667142},
    {"date":"15/04/14","open":17.1,"high":17.89,"low":17.02,"close":17.52,"volume":870138},
    {"date":"15/04/15","open":17.6,"high":17.99,"low":17.5,"close":17.69,"volume":530456},
    {"date":"15/04/16","open":17.95,"high":18,"low":17.6,"close":17.82,"volume":548730},
    {"date":"15/04/17","open":17.75,"high":17.79,"low":17.5,"close":17.79,"volume":446373},
    {"date":"15/04/20","open":17.63,"high":17.98,"low":17.52,"close":17.93,"volume":487017},
    {"date":"15/04/21","open":17.96,"high":17.98,"low":17.71,"close":17.92,"volume":320302},
    {"date":"15/04/22","open":17.88,"high":18.33,"low":17.57,"close":18.29,"volume":644812},
    {"date":"15/04/23","open":18.29,"high":18.61,"low":18.18,"close":18.28,"volume":563879},
    {"date":"15/04/24","open":18.5,"high":18.5,"low":17.61,"close":17.75,"volume":650762},
    {"date":"15/04/27","open":17.97,"high":18.05,"low":17.45,"close":17.57,"volume":437294},
    {"date":"15/04/28","open":17.65,"high":17.79,"low":17.39,"close":17.5,"volume":224519},
    {"date":"15/04/29","open":17.68,"high":17.68,"low":17.1,"close":17.21,"volume":495706},
    {"date":"15/04/30","open":17.22,"high":17.3,"low":17,"close":17.11,"volume":391040},
    {"date":"15/05/01","open":17.11,"high":17.55,"low":16.85,"close":17.5,"volume":563075},
    {"date":"15/05/02","open":17.56,"high":17.85,"low":17.3,"close":17.4,"volume":253138},
    {"date":"15/05/05","open":17.68,"high":17.68,"low":17.09,"close":17.43,"volume":290935},
    {"date":"15/05/06","open":17.48,"high":17.48,"low":17,"close":17.04,"volume":313662},
    {"date":"15/05/07","open":17.05,"high":17.19,"low":16.92,"close":17.04,"volume":360284},
    {"date":"15/05/08","open":17.13,"high":17.21,"low":16.91,"close":17.1,"volume":297653},
    {"date":"15/05/11","open":17.16,"high":17.44,"low":17.13,"close":17.31,"volume":268504},
    {"date":"15/05/12","open":17.28,"high":17.44,"low":16.99,"close":17.24,"volume":376961},
    {"date":"15/05/13","open":17.24,"high":17.3,"low":17.06,"close":17.2,"volume":244617},
    {"date":"15/05/14","open":17.24,"high":17.25,"low":17.02,"close":17.08,"volume":252526},
    {"date":"15/05/15","open":17.06,"high":17.16,"low":16.95,"close":16.95,"volume":274783},
    {"date":"15/05/18","open":16.95,"high":17.01,"low":16.76,"close":16.87,"volume":418513},
    {"date":"15/05/19","open":16.93,"high":16.94,"low":16.6,"close":16.83,"volume":367660},
    {"date":"15/05/20","open":16.8,"high":16.9,"low":16.65,"close":16.86,"volume":297914},
    {"date":"15/05/21","open":16.9,"high":17.08,"low":16.79,"close":16.88,"volume":229346},
    {"date":"15/05/22","open":16.9,"high":17.05,"low":16.85,"close":17,"volume":253279},
    {"date":"15/05/26","open":17.03,"high":17.08,"low":16.86,"close":17.01,"volume":212640},
    {"date":"15/05/27","open":17.01,"high":17.99,"low":16.87,"close":17.75,"volume":857109},
    {"date":"15/05/28","open":17.77,"high":17.77,"low":17.44,"close":17.62,"volume":338482}
]
```


コードビューで次の参照を追加します。

- System.Collections.Generic
- System.Windows
- System.Runtime.Serialization

- **Visual Basic**

```
<DataContract>
Public Class Quote
    <DataMember(Name:="date")>
    Public Property [Date]() As String
        Get
            Return m_Date
        End Get
        Set
            m_Date = Value
        End Set
    End Property
    Private m_Date As String

    <DataMember(Name:="high")>
    Public Property High() As Double
        Get
```

```
            Return m_High
        End Get
        Set
            m_High = Value
        End Set
    End Property
    Private m_High As Double

    <DataMember(Name:="low")>
    Public Property Low() As Double
        Get
            Return m_Low
        End Get
        Set
            m_Low = Value
        End Set
    End Property
    Private m_Low As Double

    <DataMember(Name:="open")>
    Public Property Open() As Double
        Get
            Return m_Open
        End Get
        Set
            m_Open = Value
        End Set
    End Property
    Private m_Open As Double

    <DataMember(Name:="close")>
    Public Property Close() As Double
        Get
            Return m_Close
        End Get
        Set
            m_Close = Value
        End Set
    End Property
    Private m_Close As Double

    <DataMember(Name:="volume")>
    Public Property Volume() As Double
        Get
            Return m_Volume
        End Get
        Set
            m_Volume = Value
        End Set
    End Property
    Private m_Volume As Double
End Class
''' MainWindow.xamlの相互作用ロジック
Partial Public Class MainWindow
    Inherits Window
    Private dataService As DataService = dataService.GetService()
    Public Sub New()
        InitializeComponent()
    End Sub
    Public ReadOnly Property Data() As List(Of Quote)
        Get
            Return dataService.GetData()
        End Get
    End Property

    Private Sub OnFinancialChartRendered(sender As Object, e As C1.WPF.Chart.RenderEventArgs)
        indicatorChart.AxisX.Min = DirectCast(financialChart.AxisX, IAxis).GetMin()
        indicatorChart.AxisX.Max = DirectCast(financialChart.AxisX, IAxis).GetMax()
```

```
    End Sub
End Class
```

- **C#**

```
[DataContract]
public class Quote
{
    [DataMember(Name = "date")]
    public string Date { get; set; }

    [DataMember(Name = "high")]
    public double High { get; set; }

    [DataMember(Name = "low")]
    public double Low { get; set; }

    [DataMember(Name = "open")]
    public double Open { get; set; }

    [DataMember(Name = "close")]
    public double Close { get; set; }

    [DataMember(Name = "volume")]
    public double Volume { get; set; }
}
/// MainWindow.xamlの相互作用ロジック
public partial class MainWindow : Window
{
    DataService dataService = DataService.GetService();
    public MainWindow()
    {
        InitializeComponent();
    }
    public List<Quote> Data
    {
        get
        {
            return dataService.GetData();
        }
    }

    private void OnFinancialChartRendered(object sender, C1.WPF.Chart.RenderEventArgs e)
    {
        indicatorChart.AxisX.Min = ((IAxis)financialChart.AxisX).GetMin();
        indicatorChart.AxisX.Max = ((IAxis)financialChart.AxisX).GetMax();
    }
 }
```

先頭に移動

MACD

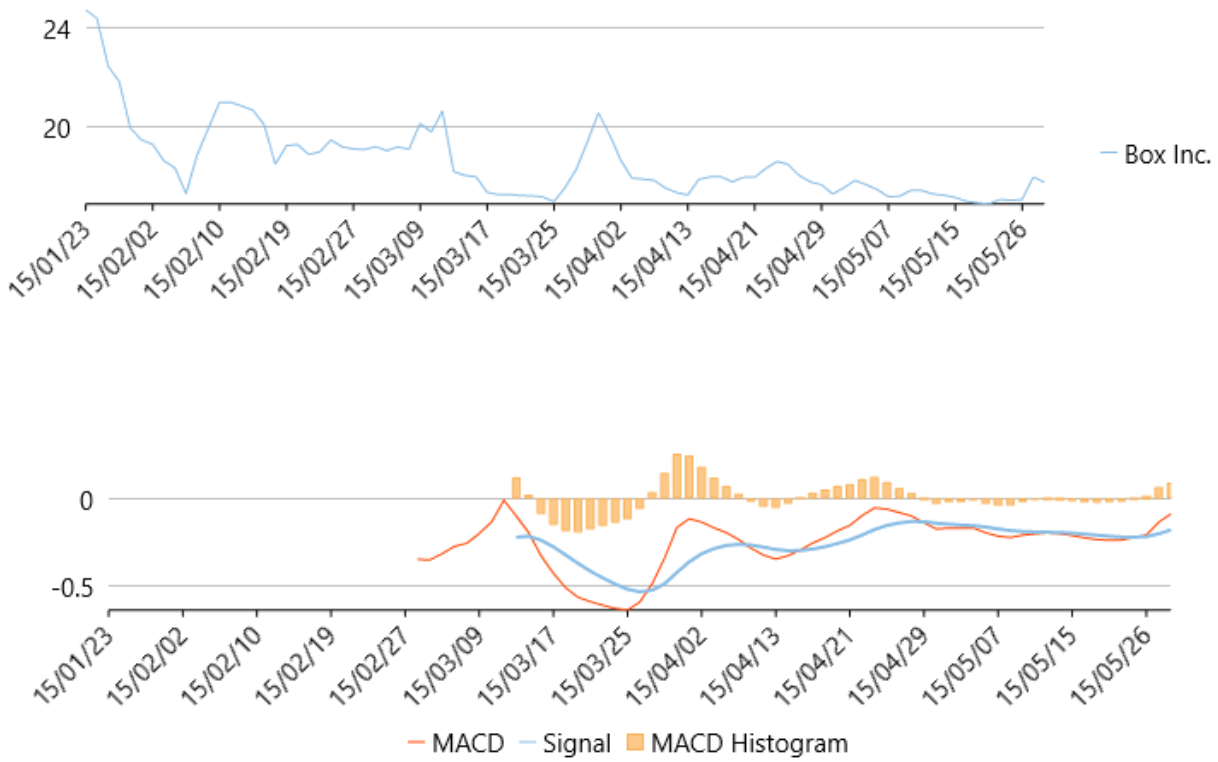
FinancialChart の MACD（移動平均収束発散法）インジケータは、トレンドフォロー型のモメンタムインジケータです。資産価格の強さ、方向、期間、およびモメンタムの変化を明らかにします。このインジケータを使用すると、短期的な価格モメンタムを効率よく見極めることができます。

MACD モメンタムオシレータは、26 日間の指数移動平均と 12 日間の指数移動平均の関係を表示します。2 つの移動平均が収束、交差、発散するに伴って、MACD オシレータはゼロラインの上下に変動します。オシレータに重ねて「シグナルライン」がプロットされます。このラインは MACD の 9 日間の指数移動平均を示し、買いシグナルと売りシグナルのトリガとして機能します。MACD がゼロラインより下になると、売りシグナルの発生です。

MACD ヒストグラムは、短期 MACD ラインとシグナルラインの差を示すオシレータです。MACD インジケータと同様に、ヒストグラムもゼロラインの上下に変動します。正のヒストグラムは、MACD がシグナルラインより上にあることを示し、MACD がシグナルラインより下になると、負のヒストグラムになります。負の MACD ヒストグラムは売りシグナルです。

FinancialChart に MACD インジケータと MACD ヒストグラムを追加するには、FinancialChart コントロールをアプリケーションに追加し、コントロールに適切なデータソースを連結するか、**Quote Collection** でコントロールにデータを追加します。FinancialChart にデータを連結または追加するには、**ItemsSource** オブジェクトを使用します。**MacdBase** クラスは、**FastPeriod**、**SlowPeriod**、および **SmoothingPeriod** プロパティを公開します。これらのプロパティの値に基づいて、MACD インジケータとヒストグラムが計算され、FinancialChart にプロットされます。系列の外観は、**MacdLineStyle** および **SignalLineStyle** プロパティで操作できます。

また、FinancialChart では、アプリケーションでアラートを作成したり、動的データの使用中にログを取るために、計算された **Macd 値**、**Macd x 値**、**シグナル値**、**シグナル x 値**を実行時に取得できます。



次の例は、上の図で示すように、企業 Box Inc. の一定期間の株価を考察し、ボリュームチャートとは別に MACD インジケータと MACD ヒストグラムをプロットします。この例では、json ファイルのデータを使用します。この json ファイルにアクセスするために、DataService.cs クラスを作成します。

⚠ json ファイルの[ビルドアクション]プロパティが**[埋め込まれたリソース]**に設定されていることを確認します。

PriceChart.xaml copyCode

```
<Window xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
        x:Class="MACDInd.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:MACDInd"
        mc:Ignorable="d"
        Title="MainWindow" Height="350" Width="525"
        DataContext="{Binding RelativeSource={RelativeSource Mode=Self}}">
<Grid>
    <c1:C1FinancialChart x:Name="financialChart"
                         ItemsSource="{Binding Data}"
                         BindingX="Date"
                         ChartType="Line"
                         ToolTipContent="{}{seriesName}&#x000A;{Date} {y}"
```

```
                        Margin="0,0,0,146"
                        Rendered="OnFinancialChartRendered">

        <c1:FinancialSeries Binding="High,Low,Open,Close" SeriesName="Box Inc." />
        <c1:C1FinancialChart.AxisX>
            <c1:Axis LabelAngle="45" MajorUnit="3"/>
        </c1:C1FinancialChart.AxisX>
    </c1:C1FinancialChart>
```

IndicatorChart.xaml copyCode

```
    <c1:C1FinancialChart x:Name="indicatorChart"
                         BindingX="Date"
                         Binding="High,Low,Close"
                         LegendPosition="Bottom"
                         ItemsSource="{Binding Data}"
                         Background="White"
                         ToolTipContent="{}{seriesName}&#x000A;Date: {Date}&#x000A;Y:
                         {y:n2}&#x000A;Volume: {Volume:n0}"
                         Margin="0,178,0,0">

        <c1:Macd x:Name="Macd" SeriesName="MACD,Signal">
            <c1:Macd.MacdLineStyle>
                <c1:ChartStyle Stroke="OrangeRed" />
            </c1:Macd.MacdLineStyle>
        </c1:Macd>

        <c1:MacdHistogram x:Name="MACDHistogram"
                          SeriesName= "MACD Histogram"
                          FastPeriod="12"
                          SlowPeriod="26"
                          SmoothingPeriod="9" />

        <c1:C1FinancialChart.AxisX>
        <c1:Axis LabelAngle="45" MajorUnit="3"/>
    </c1:C1FinancialChart.AxisX>
    </c1:C1FinancialChart>
</Grid>
```

DataService.cs に次の参照を追加します。

- System.Collections.Generic
- System.Linq
- System.Runtime.Serialization.Json
- System.Reflection

- **DataService.vb**

```
Public Class DataService
    Public Function GetData() As List(Of Quote)
        Dim path As String = "Indicator.Resources.box.json"
        'Indicatorをアプリケーション名で置き換えます。
        Dim stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path)
        Dim ser = New DataContractJsonSerializer(GetType(Quote()))
        Dim data = DirectCast(ser.ReadObject(stream), Quote())
        Return data.ToList()
    End Function
    Shared _ds As DataService
```

```
    Public Shared Function GetService() As DataService
        If _ds Is Nothing Then
            _ds = New DataService()
        End If
        Return _ds
    End Function
End Class
```

- **DataService.cs**

```
public class DataService
{
    public List<Quote> GetData()
    {
        string path = "MACDInd.Resources.box.json";
        //MACDIndをアプリケーション名で置き換えます。
        var stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path);
        var ser = new DataContractJsonSerializer(typeof(Quote[]));
        var data = (Quote[])ser.ReadObject(stream);
        return data.ToList();
    }
    static DataService _ds;
    public static DataService GetService()
    {
        if (_ds == null)
            _ds = new DataService();
        return _ds;
    }
}
```

Json Data


```
[
    {"date":"15/01/23","open":20.2,"high":24.73,"low":20.16,"close":23.23,"volume":42593223},
    {"date":"15/01/26","open":23.67,"high":24.39,"low":22.5,"close":22.6,"volume":8677164},
    {"date":"15/01/27","open":22,"high":22.47,"low":21.17,"close":21.3,"volume":3272512},
    {"date":"15/01/28","open":21.62,"high":21.84,"low":19.6,"close":19.78,"volume":5047364},
    {"date":"15/01/29","open":19.9,"high":19.95,"low":18.51,"close":18.8,"volume":3419482},
    {"date":"15/01/30","open":18.47,"high":19.48,"low":18.22,"close":18.81,"volume":2266439},
    {"date":"15/02/02","open":19.18,"high":19.3,"low":18.01,"close":18.02,"volume":2071168},
    {"date":"15/02/03","open":18.22,"high":18.64,"low":18.12,"close":18.24,"volume":1587435},
    {"date":"15/02/04","open":18.2,"high":18.35,"low":17,"close":17.1,"volume":2912224},
    {"date":"15/02/05","open":17.3,"high":17.31,"low":16.41,"close":16.66,"volume":2682187},
    {"date":"15/02/06","open":17.39,"high":18.88,"low":17.21,"close":18.12,"volume":3929164},
    {"date":"15/02/09","open":18.86,"high":19.95,"low":18.45,"close":19.6,"volume":3226650},
    {"date":"15/02/10","open":20.5,"high":21,"low":19.63,"close":20.99,"volume":2804409},
    {"date":"15/02/11","open":20.89,"high":21,"low":20.2,"close":20.96,"volume":1698365},
    {"date":"15/02/12","open":20.66,"high":20.85,"low":19.75,"close":20.17,"volume":1370320},
    {"date":"15/02/13","open":20.19,"high":20.68,"low":20,"close":20.18,"volume":711951},
    {"date":"15/02/17","open":19.5,"high":20.1,"low":18.8,"close":19.05,"volume":2093602},
    {"date":"15/02/18","open":18.31,"high":18.5,"low":17.96,"close":18,"volume":1849490},
    {"date":"15/02/19","open":18.33,"high":19.25,"low":17.91,"close":18.96,"volume":1311518},
    {"date":"15/02/20","open":18.68,"high":19.3,"low":18.65,"close":18.85,"volume":1001692},
    {"date":"15/02/23","open":18.8,"high":18.89,"low":18.11,"close":18.21,"volume":670087},
    {"date":"15/02/24","open":18.46,"high":19,"low":18.27,"close":18.83,"volume":759263},
    {"date":"15/02/25","open":18.83,"high":19.48,"low":18.47,"close":18.67,"volume":915580},
    {"date":"15/02/26","open":18.64,"high":19.2,"low":18.64,"close":18.94,"volume":461283},
    {"date":"15/02/27","open":18.8,"high":19.12,"low":18.55,"close":18.66,"volume":617199},
    {"date":"15/03/02","open":18.66,"high":19.09,"low":18.65,"close":18.79,"volume":519605},
    {"date":"15/03/03","open":18.79,"high":19.21,"low":18.45,"close":18.59,"volume":832415},
    {"date":"15/03/04","open":18.64,"high":19.05,"low":18.32,"close":19,"volume":539688},
    {"date":"15/03/05","open":19.2,"high":19.2,"low":18.8,"close":19.14,"volume":486149},
    {"date":"15/03/06","open":19.03,"high":19.1,"low":18.7,"close":18.91,"volume":685659},
    {"date":"15/03/09","open":18.98,"high":20.15,"low":18.96,"close":19.4,"volume":1321363},
    {"date":"15/03/10","open":19.3,"high":19.8,"low":18.85,"close":19.64,"volume":615743},
    {"date":"15/03/11","open":20.08,"high":20.65,"low":19.24,"close":20.53,"volume":2167167},
    {"date":"15/03/12","open":17.17,"high":18.2,"low":16.76,"close":18.2,"volume":6837638},
    {"date":"15/03/13","open":18.05,"high":18.05,"low":17.3,"close":17.88,"volume":1715629},
    {"date":"15/03/16","open":17.91,"high":18,"low":17.01,"close":17.13,"volume":1321313},
    {"date":"15/03/17","open":17.28,"high":17.37,"low":16.6,"close":17.12,"volume":1272242},
```

```
    {"date":"15/03/18","open":17.1,"high":17.27,"low":16.91,"close":17.01,"volume":530063},
    {"date":"15/03/19","open":17,"high":17.28,"low":17,"close":17.06,"volume":536427},
    {"date":"15/03/20","open":17.13,"high":17.24,"low":16.88,"close":17.21,"volume":1320237},
    {"date":"15/03/23","open":17.21,"high":17.23,"low":17.01,"close":17.11,"volume":509798},
    {"date":"15/03/24","open":17.02,"high":17.18,"low":16.82,"close":17,"volume":962149},
    {"date":"15/03/25","open":16.92,"high":16.99,"low":16.82,"close":16.97,"volume":565673},
    {"date":"15/03/26","open":16.83,"high":17.56,"low":16.83,"close":17.54,"volume":884523},
    {"date":"15/03/27","open":17.58,"high":18.3,"low":17.11,"close":18.3,"volume":705626},
    {"date":"15/03/30","open":18.5,"high":19.4,"low":18.4,"close":19.05,"volume":1151620},
    {"date":"15/03/31","open":19.08,"high":20.58,"low":18.4,"close":19.75,"volume":2020679},
    {"date":"15/04/01","open":19.69,"high":19.69,"low":18.55,"close":18.65,"volume":961078},
    {"date":"15/04/02","open":18.56,"high":18.66,"low":17.85,"close":17.9,"volume":884233},
    {"date":"15/04/06","open":17.78,"high":17.94,"low":17.51,"close":17.66,"volume":605252},
    {"date":"15/04/07","open":17.62,"high":17.9,"low":17.53,"close":17.61,"volume":591988},
    {"date":"15/04/08","open":17.64,"high":17.85,"low":17.32,"close":17.36,"volume":618855},
    {"date":"15/04/09","open":17.33,"high":17.54,"low":17.1,"close":17.1,"volume":761855},
    {"date":"15/04/10","open":17.08,"high":17.36,"low":17,"close":17.05,"volume":568373},
    {"date":"15/04/13","open":17.24,"high":17.26,"low":16.81,"close":17.1,"volume":667142},
    {"date":"15/04/14","open":17.1,"high":17.89,"low":17.02,"close":17.52,"volume":870138},
    {"date":"15/04/15","open":17.6,"high":17.99,"low":17.5,"close":17.69,"volume":530456},
    {"date":"15/04/16","open":17.95,"high":18,"low":17.6,"close":17.82,"volume":548730},
    {"date":"15/04/17","open":17.75,"high":17.79,"low":17.5,"close":17.79,"volume":446373},
    {"date":"15/04/20","open":17.63,"high":17.98,"low":17.52,"close":17.93,"volume":487017},
    {"date":"15/04/21","open":17.96,"high":17.98,"low":17.71,"close":17.92,"volume":320302},
    {"date":"15/04/22","open":17.88,"high":18.33,"low":17.57,"close":18.29,"volume":644812},
    {"date":"15/04/23","open":18.29,"high":18.61,"low":18.18,"close":18.28,"volume":563879},
    {"date":"15/04/24","open":18.5,"high":18.5,"low":17.61,"close":17.75,"volume":650762},
    {"date":"15/04/27","open":17.97,"high":18.05,"low":17.45,"close":17.57,"volume":437294},
    {"date":"15/04/28","open":17.65,"high":17.79,"low":17.39,"close":17.5,"volume":224519},
    {"date":"15/04/29","open":17.68,"high":17.68,"low":17.1,"close":17.21,"volume":495706},
    {"date":"15/04/30","open":17.22,"high":17.3,"low":17,"close":17.11,"volume":391040},
    {"date":"15/05/01","open":17.11,"high":17.55,"low":16.85,"close":17.5,"volume":563075},
    {"date":"15/05/02","open":17.56,"high":17.85,"low":17.3,"close":17.4,"volume":253138},
    {"date":"15/05/05","open":17.68,"high":17.68,"low":17.09,"close":17.43,"volume":290935},
    {"date":"15/05/06","open":17.48,"high":17.48,"low":17,"close":17.04,"volume":313662},
    {"date":"15/05/07","open":17.05,"high":17.19,"low":16.92,"close":17.04,"volume":360284},
    {"date":"15/05/08","open":17.13,"high":17.21,"low":16.91,"close":17.1,"volume":297653},
    {"date":"15/05/11","open":17.16,"high":17.44,"low":17.13,"close":17.31,"volume":268504},
    {"date":"15/05/12","open":17.28,"high":17.44,"low":16.99,"close":17.24,"volume":376961},
    {"date":"15/05/13","open":17.24,"high":17.3,"low":17.06,"close":17.2,"volume":244617},
    {"date":"15/05/14","open":17.24,"high":17.25,"low":17.02,"close":17.08,"volume":252526},
    {"date":"15/05/15","open":17.06,"high":17.16,"low":16.95,"close":16.95,"volume":274783},
    {"date":"15/05/18","open":16.95,"high":17.01,"low":16.76,"close":16.87,"volume":418513},
    {"date":"15/05/19","open":16.93,"high":16.94,"low":16.6,"close":16.83,"volume":367660},
    {"date":"15/05/20","open":16.8,"high":16.9,"low":16.65,"close":16.86,"volume":297914},
    {"date":"15/05/21","open":16.9,"high":17.08,"low":16.79,"close":16.88,"volume":229346},
    {"date":"15/05/22","open":16.9,"high":17.05,"low":16.85,"close":17,"volume":253279},
    {"date":"15/05/26","open":17.03,"high":17.08,"low":16.86,"close":17.01,"volume":212640},
    {"date":"15/05/27","open":17.01,"high":17.99,"low":16.87,"close":17.75,"volume":857109},
    {"date":"15/05/28","open":17.77,"high":17.77,"low":17.44,"close":17.62,"volume":338482}
]
```


コードビューで次の参照を追加します。

- System.Collections.Generic
- System.Windows
- System.Runtime.Serialization

- **Visual Basic**

```
<DataContract>
Public Class Quote
    <DataMember(Name:="date")>
    Public Property [Date]() As String
        Get
            Return m_Date
        End Get
        Set
            m_Date = Value
```

```
        End Set
    End Property
    Private m_Date As String

    <DataMember(Name:="high")>
    Public Property High() As Double
        Get
            Return m_High
        End Get
        Set
            m_High = Value
        End Set
    End Property
    Private m_High As Double

    <DataMember(Name:="low")>
    Public Property Low() As Double
        Get
            Return m_Low
        End Get
        Set
            m_Low = Value
        End Set
    End Property
    Private m_Low As Double

    <DataMember(Name:="open")>
    Public Property Open() As Double
        Get
            Return m_Open
        End Get
        Set
            m_Open = Value
        End Set
    End Property
    Private m_Open As Double

    <DataMember(Name:="close")>
    Public Property Close() As Double
        Get
            Return m_Close
        End Get
        Set
            m_Close = Value
        End Set
    End Property
    Private m_Close As Double

    <DataMember(Name:="volume")>
    Public Property Volume() As Double
        Get
            Return m_Volume
        End Get
        Set
            m_Volume = Value
        End Set
    End Property
    Private m_Volume As Double
End Class
''' Macd.xamlの相互作用ロジック
Partial Public Class Macd
    Inherits Window
    Private dataService As DataService = dataService.GetService()
    Public Sub New()
        InitializeComponent()
    End Sub
    Public ReadOnly Property Data() As List(Of Quote)
        Get
            Return dataService.GetData()
        End Get
```

```
    End Property

    Private Sub OnFinancialChartRendered(sender As Object, e As C1.WPF.Chart.RenderEventArgs)
        indicatorChart.AxisX.Min = DirectCast(financialChart.AxisX, IAxis).GetMin()
        indicatorChart.AxisX.Max = DirectCast(financialChart.AxisX, IAxis).GetMax()
    End Sub
End Class
```

- **C#**

```
[DataContract]
public class Quote
{
    [DataMember(Name = "date")]
    public string Date { get; set; }

    [DataMember(Name = "high")]
    public double High { get; set; }

    [DataMember(Name = "low")]
    public double Low { get; set; }

    [DataMember(Name = "open")]
    public double Open { get; set; }

    [DataMember(Name = "close")]
    public double Close { get; set; }

    [DataMember(Name = "volume")]
    public double Volume { get; set; }
}
/// MainWindow.xamlの相互作用ロジック
public partial class MainWindow : Window
{
    DataService dataService = DataService.GetService();
    public MainWindow()
    {
        InitializeComponent();
    }
    public List<Quote> Data
    {
        get
        {
            return dataService.GetData();
        }
    }

    private void OnFinancialChartRendered(object sender, C1.WPF.Chart.RenderEventArgs e)
    {
        indicatorChart.AxisX.Min = ((IAxis)financialChart.AxisX).GetMin();
        indicatorChart.AxisX.Max = ((IAxis)financialChart.AxisX).GetMax();
    }
}
```

先頭に移動

オーバーレイ

インジケータなどのテクニカルオーバーレイは、金融商品の過去から現在までの株価に数式を適用して計算することによって派生された一連のデータポイントです。これらは、資産の市場動向を予測するために使用されます。インジケータとは異なり、オーバーレイの Y 軸スケールは同じなので、元の価格データまたはボリュームデータと共にプロットされます。

以下のセクションでは、FinanicalChart がサポートするテクニカルオーバーレイについて説明します。

Bollinger Bands

ボリンジャーバンドオーバーレイについて学習します。

Envelopes
エンベロープオーバーレイについて学習します。

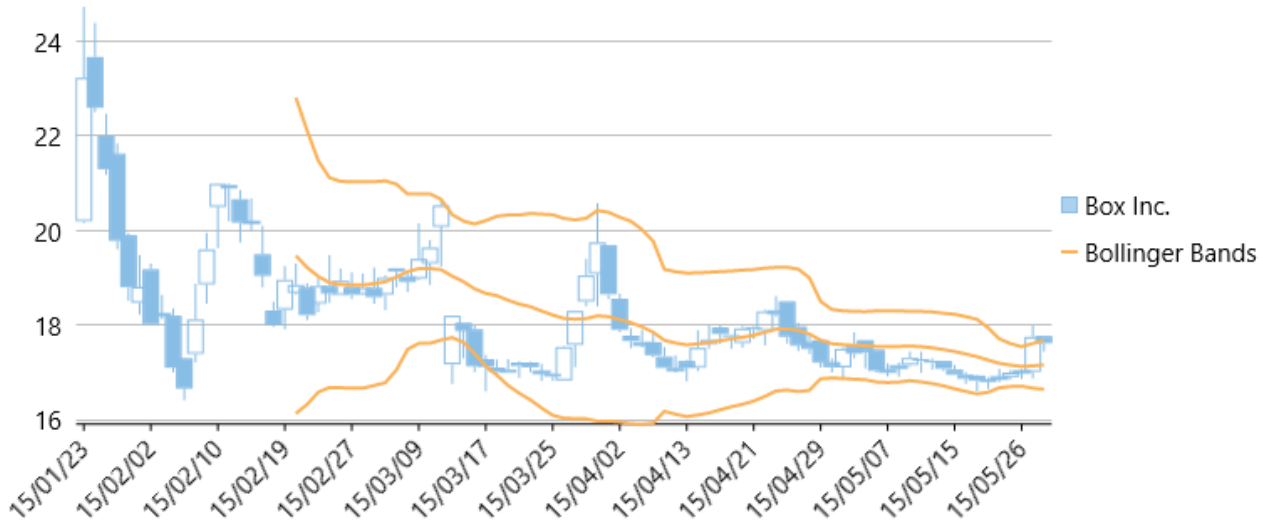
Ichimoku Clouds
Learn about Ichimoku clouds overlay.

Bollinger Bands

Bollinger Bands（ボリンジャーバンド）オーバーレイは、一定期間の株価の動きの上限と下限を示します。これは、標準偏差に基づくボラティリティバンドで、移動平均の上下に置かれます。ボリンジャーバンドは、株価の高い安いを相対的に定義するために役立ちます。上のバンド近くにプロットされた株価は高いと見なされ、下のバンド近くにプロットされた価格は低いと見なされます。ハンドの幅はボラティリティの程度を示します。ボリンジャーバンドの天上に届く株価は、中間トレンドの上方に伸びすぎであると見なされ、ボリンジャーバンドの底に届く株価は、中間トレンドの下方に伸びすぎであると見なされます。

FinancialChart にボリンジャーバンドオーバーレイを追加するには、FinancialChart コントロールをアプリケーションに追加し、コントロールに適切なデータソースを連結するか、**Quote Collection** でコントロールにデータを追加します。FinancialChart にデータを連結または追加するには、ItemsSource オブジェクトを使用します。**BollingerBands** クラスは、上下バンドの標準偏差を指定する **Multiplier** プロパティを公開し、**IndicatorBase** クラスは、**Period** プロパティを公開します。これはミドルバンドの単純移動平均を計算するための整数値を受け取ります。これらのプロパティの値に基づいて、ボリンジャーバンドオーバーレイが計算され、FinancialChart にプロットされます。

また、FinancialChart では、アプリケーションでアラートを作成したり、動的データの使用中にログを取るために、計算された**下限 y 値、中間 y 値、上限 y 値、x 値**を実行時に取得できます。



次の例は、上の図で示すように、企業 Box Inc. の一定期間の株価を考察し、同じ株価チャートにボリンジャーバンドオーバーレイをプロットします。この例では、json ファイルのデータを使用します。この json ファイルにアクセスするために、DataService.cs クラスを作成します。

⚠ json ファイルの[ビルドアクション]プロパティが**[埋め込まれたリソース]**に設定されていることを確認します。

XAML copyCode

```
<Window
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:BollingerBOverlay"
    xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
    x:Class="BollingerBOverlay.MainWindow"
    DataContext="{Binding RelativeSource={RelativeSource Mode=Self}}"
```

```
        mc:Ignorable="d"
        Title="MainWindow" Height="350" Width="525">
    <Grid>
<c1:C1FinancialChart x:Name="financialChart"
                     ItemsSource="{Binding Data}"
                     Background="White"
                     ChartType="Candlestick"
                     BindingX="Date"
                     ToolTipContent="{}{seriesName}&#x000A;{Date} {y}"
                     Margin="6,37,4,71">
            <c1:FinancialSeries Binding="High,Low,Open,Close"
                                SeriesName="Box Inc." />
            <c1:BollingerBands x:Name="bollinger"
                               Multiplier="2"
                               Period="20"
                               Binding="High,Low,Close"
                               SeriesName="Bollinger Bands" />

            <c1:C1FinancialChart.AxisX>
                <c1:Axis LabelAngle="45" MajorUnit="3"/>
            </c1:C1FinancialChart.AxisX>
        </c1:C1FinancialChart>
    </Grid>
</Window>
```

DataService.cs に次の参照を追加します。

- System.Collections.Generic
- System.Linq
- System.Runtime.Serialization.Json
- System.Reflection

- **DataService.vb**

```
Public Class DataService
    Public Function GetData() As List(Of Quote)
        Dim path As String = "OverlayVB.Resources.box.json"
        'OverlayVBをアプリケーション名で置き換えます。
        Dim stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path)
        Dim ser = New DataContractJsonSerializer(GetType(Quote()))
        Dim data = DirectCast(ser.ReadObject(stream), Quote())
        Return data.ToList()
    End Function
    Shared _ds As DataService
    Public Shared Function GetService() As DataService
        If _ds Is Nothing Then
            _ds = New DataService()
        End If
        Return _ds
    End Function
End Class
```

- **DataService.cs**

```
public class DataService
{
    public List<Quote> GetData()
    {
        string path = "BollingerBOverlay.Resources.box.json";
        //BollingerBOverlayをアプリケーション名で置き換えます。
        var stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path);
```

```
            var ser = new DataContractJsonSerializer(typeof(Quote[]));
            var data = (Quote[])ser.ReadObject(stream);
            return data.ToList();
        }
        static DataService _ds;
        public static DataService GetService()
        {
            if (_ds == null)
                _ds = new DataService();
            return _ds;
        }
}
```

Json Data

```
[
    {"date":"15/01/23","open":20.2,"high":24.73,"low":20.16,"close":23.23,"volume":42593223},
    {"date":"15/01/26","open":23.67,"high":24.39,"low":22.5,"close":22.6,"volume":8677164},
    {"date":"15/01/27","open":22,"high":22.47,"low":21.17,"close":21.3,"volume":3272512},
    {"date":"15/01/28","open":21.62,"high":21.84,"low":19.6,"close":19.78,"volume":5047364},
    {"date":"15/01/29","open":19.9,"high":19.95,"low":18.51,"close":18.8,"volume":3419482},
    {"date":"15/01/30","open":18.47,"high":19.48,"low":18.22,"close":18.81,"volume":2266439},
    {"date":"15/02/02","open":19.18,"high":19.3,"low":18.01,"close":18.02,"volume":2071168},
    {"date":"15/02/03","open":18.22,"high":18.64,"low":18.12,"close":18.24,"volume":1587435},
    {"date":"15/02/04","open":18.2,"high":18.35,"low":17,"close":17.1,"volume":2912224},
    {"date":"15/02/05","open":17.3,"high":17.31,"low":16.41,"close":16.66,"volume":2682187},
    {"date":"15/02/06","open":17.39,"high":18.88,"low":17.21,"close":18.12,"volume":3929164},
    {"date":"15/02/09","open":18.86,"high":19.95,"low":18.45,"close":19.6,"volume":3226650},
    {"date":"15/02/10","open":20.5,"high":21,"low":19.63,"close":20.99,"volume":2804409},
    {"date":"15/02/11","open":20.89,"high":21,"low":20.2,"close":20.96,"volume":1698365},
    {"date":"15/02/12","open":20.66,"high":20.85,"low":19.75,"close":20.17,"volume":1370320},
    {"date":"15/02/13","open":20.19,"high":20.68,"low":20,"close":20.18,"volume":711951},
    {"date":"15/02/17","open":19.5,"high":20.1,"low":18.8,"close":19.05,"volume":2093602},
    {"date":"15/02/18","open":18.31,"high":18.5,"low":17.96,"close":18,"volume":1849490},
    {"date":"15/02/19","open":18.33,"high":19.25,"low":17.91,"close":18.96,"volume":1311518},
    {"date":"15/02/20","open":18.68,"high":19.3,"low":18.65,"close":18.85,"volume":1001692},
    {"date":"15/02/23","open":18.8,"high":18.89,"low":18.11,"close":18.21,"volume":670087},
    {"date":"15/02/24","open":18.46,"high":19,"low":18.27,"close":18.83,"volume":759263},
    {"date":"15/02/25","open":18.83,"high":19.48,"low":18.47,"close":18.67,"volume":915580},
    {"date":"15/02/26","open":18.64,"high":19.2,"low":18.64,"close":18.94,"volume":461283},
    {"date":"15/02/27","open":18.8,"high":19.12,"low":18.55,"close":18.66,"volume":617199},
    {"date":"15/03/02","open":18.66,"high":19.09,"low":18.65,"close":18.79,"volume":519605},
    {"date":"15/03/03","open":18.79,"high":19.21,"low":18.45,"close":18.59,"volume":832415},
    {"date":"15/03/04","open":18.64,"high":19.05,"low":18.32,"close":19,"volume":539688},
    {"date":"15/03/05","open":19.2,"high":19.2,"low":18.8,"close":19.14,"volume":486149},
    {"date":"15/03/06","open":19.03,"high":19.1,"low":18.7,"close":18.91,"volume":685659},
    {"date":"15/03/09","open":18.98,"high":20.15,"low":18.96,"close":19.4,"volume":1321363},
    {"date":"15/03/10","open":19.3,"high":19.8,"low":18.85,"close":19.64,"volume":615743},
    {"date":"15/03/11","open":20.08,"high":20.65,"low":19.24,"close":20.53,"volume":2167167},
    {"date":"15/03/12","open":17.17,"high":18.2,"low":16.76,"close":18.2,"volume":6837638},
    {"date":"15/03/13","open":18.05,"high":18.05,"low":17.3,"close":17.88,"volume":1715629},
    {"date":"15/03/16","open":17.91,"high":18,"low":17.01,"close":17.13,"volume":1321313},
    {"date":"15/03/17","open":17.28,"high":17.37,"low":16.6,"close":17.12,"volume":1272242},
    {"date":"15/03/18","open":17.1,"high":17.27,"low":16.91,"close":17.01,"volume":530063},
    {"date":"15/03/19","open":17,"high":17.28,"low":17,"close":17.06,"volume":536427},
    {"date":"15/03/20","open":17.13,"high":17.24,"low":16.88,"close":17.21,"volume":1320237},
    {"date":"15/03/23","open":17.21,"high":17.23,"low":17.01,"close":17.11,"volume":509798},
    {"date":"15/03/24","open":17.02,"high":17.18,"low":16.82,"close":17,"volume":962149},
    {"date":"15/03/25","open":16.92,"high":16.99,"low":16.82,"close":16.97,"volume":565673},
    {"date":"15/03/26","open":16.83,"high":17.56,"low":16.83,"close":17.54,"volume":884523},
    {"date":"15/03/27","open":17.58,"high":18.3,"low":17.11,"close":18.3,"volume":705626},
    {"date":"15/03/30","open":18.5,"high":19.4,"low":18.4,"close":19.05,"volume":1151620},
    {"date":"15/03/31","open":19.08,"high":20.58,"low":18.4,"close":19.75,"volume":2020679},
    {"date":"15/04/01","open":19.69,"high":19.69,"low":18.55,"close":18.65,"volume":961078},
    {"date":"15/04/02","open":18.56,"high":18.66,"low":17.85,"close":17.9,"volume":884233},
    {"date":"15/04/06","open":17.78,"high":17.94,"low":17.51,"close":17.66,"volume":605252},
    {"date":"15/04/07","open":17.62,"high":17.9,"low":17.53,"close":17.61,"volume":591988},
    {"date":"15/04/08","open":17.64,"high":17.85,"low":17.32,"close":17.36,"volume":618855},
```

```
    {"date":"15/04/09","open":17.33,"high":17.54,"low":17.1,"close":17.1,"volume":761855},
    {"date":"15/04/10","open":17.08,"high":17.36,"low":17,"close":17.05,"volume":568373},
    {"date":"15/04/13","open":17.24,"high":17.26,"low":16.81,"close":17.1,"volume":667142},
    {"date":"15/04/14","open":17.1,"high":17.89,"low":17.02,"close":17.52,"volume":870138},
    {"date":"15/04/15","open":17.6,"high":17.99,"low":17.5,"close":17.69,"volume":530456},
    {"date":"15/04/16","open":17.95,"high":18,"low":17.6,"close":17.82,"volume":548730},
    {"date":"15/04/17","open":17.75,"high":17.79,"low":17.5,"close":17.79,"volume":446373},
    {"date":"15/04/20","open":17.63,"high":17.98,"low":17.52,"close":17.93,"volume":487017},
    {"date":"15/04/21","open":17.96,"high":17.98,"low":17.71,"close":17.92,"volume":320302},
    {"date":"15/04/22","open":17.88,"high":18.33,"low":17.57,"close":18.29,"volume":644812},
    {"date":"15/04/23","open":18.29,"high":18.61,"low":18.18,"close":18.28,"volume":563879},
    {"date":"15/04/24","open":18.5,"high":18.5,"low":17.61,"close":17.75,"volume":650762},
    {"date":"15/04/27","open":17.97,"high":18.05,"low":17.45,"close":17.57,"volume":437294},
    {"date":"15/04/28","open":17.65,"high":17.79,"low":17.39,"close":17.5,"volume":224519},
    {"date":"15/04/29","open":17.68,"high":17.68,"low":17.1,"close":17.21,"volume":495706},
    {"date":"15/04/30","open":17.22,"high":17.3,"low":17,"close":17.11,"volume":391040},
    {"date":"15/05/01","open":17.11,"high":17.55,"low":16.85,"close":17.5,"volume":563075},
    {"date":"15/05/02","open":17.56,"high":17.85,"low":17.3,"close":17.4,"volume":253138},
    {"date":"15/05/05","open":17.68,"high":17.68,"low":17.09,"close":17.43,"volume":290935},
    {"date":"15/05/06","open":17.48,"high":17.48,"low":17,"close":17.04,"volume":313662},
    {"date":"15/05/07","open":17.05,"high":17.19,"low":16.92,"close":17.04,"volume":360284},
    {"date":"15/05/08","open":17.13,"high":17.21,"low":16.91,"close":17.1,"volume":297653},
    {"date":"15/05/11","open":17.16,"high":17.44,"low":17.13,"close":17.31,"volume":268504},
    {"date":"15/05/12","open":17.28,"high":17.44,"low":16.99,"close":17.24,"volume":376961},
    {"date":"15/05/13","open":17.24,"high":17.3,"low":17.06,"close":17.2,"volume":244617},
    {"date":"15/05/14","open":17.24,"high":17.25,"low":17.02,"close":17.08,"volume":252526},
    {"date":"15/05/15","open":17.06,"high":17.16,"low":16.95,"close":16.95,"volume":274783},
    {"date":"15/05/18","open":16.95,"high":17.01,"low":16.76,"close":16.87,"volume":418513},
    {"date":"15/05/19","open":16.93,"high":16.94,"low":16.6,"close":16.83,"volume":367660},
    {"date":"15/05/20","open":16.8,"high":16.9,"low":16.65,"close":16.86,"volume":297914},
    {"date":"15/05/21","open":16.9,"high":17.08,"low":16.79,"close":16.88,"volume":229346},
    {"date":"15/05/22","open":16.9,"high":17.05,"low":16.85,"close":17,"volume":253279},
    {"date":"15/05/26","open":17.03,"high":17.08,"low":16.86,"close":17.01,"volume":212640},
    {"date":"15/05/27","open":17.01,"high":17.99,"low":16.87,"close":17.75,"volume":857109},
    {"date":"15/05/28","open":17.77,"high":17.77,"low":17.44,"close":17.62,"volume":338482}
]
```


コードビューで次の参照を追加します。

- System.Collections.Generic
- System.Windows
- System.Runtime.Serialization

- **Visual Basic**

```
<DataContract>
Public Class Quote
    <DataMember(Name:="date")>
    Public Property [Date]() As String
        Get
            Return m_Date
        End Get
        Set
            m_Date = Value
        End Set
    End Property
    Private m_Date As String

    <DataMember(Name:="high")>
    Public Property High() As Double
        Get
            Return m_High
        End Get
        Set
            m_High = Value
        End Set
    End Property
```

```
    Private m_High As Double

    <DataMember(Name:="low")>
    Public Property Low() As Double
        Get
            Return m_Low
        End Get
        Set
            m_Low = Value
        End Set
    End Property
    Private m_Low As Double

    <DataMember(Name:="open")>
    Public Property Open() As Double
        Get
            Return m_Open
        End Get
        Set
            m_Open = Value
        End Set
    End Property
    Private m_Open As Double

    <DataMember(Name:="close")>
    Public Property Close() As Double
        Get
            Return m_Close
        End Get
        Set
            m_Close = Value
        End Set
    End Property
    Private m_Close As Double

    <DataMember(Name:="volume")>
    Public Property Volume() As Double
        Get
            Return m_Volume
        End Get
        Set
            m_Volume = Value
        End Set
    End Property
    Private m_Volume As Double
End Class
''' MWBollinger.xamlの相互作用ロジック
Partial Public Class MWBollinger
    Inherits Window
    Private dataService As DataService = DataService.GetService()
    Public Sub New()
        InitializeComponent()
    End Sub
    Public ReadOnly Property Data() As List(Of Quote)
        Get
            Return dataService.GetData()
        End Get
    End Property
End Class
```

- **C#**

```
    [DataContract]
    public class Quote
    {
        [DataMember(Name = "date")]
        public string Date { get; set; }
```

```
        [DataMember(Name = "high")]
        public double High { get; set; }

        [DataMember(Name = "low")]
        public double Low { get; set; }

        [DataMember(Name = "open")]
        public double Open { get; set; }

        [DataMember(Name = "close")]
        public double Close { get; set; }

        [DataMember(Name = "volume")]
        public double Volume { get; set; }
    }
    /// <summary>
    /// MainWindow.xamlの相互作用ロジック
    /// </summary>
    public partial class MainWindow : Window
    {
        DataService dataService = DataService.GetService();
        public MainWindow()
        {
            InitializeComponent();
        }
        public List<Quote> Data
        {
            get
            {
                return dataService.GetData();
            }
        }
    }
}
```

先頭に移動

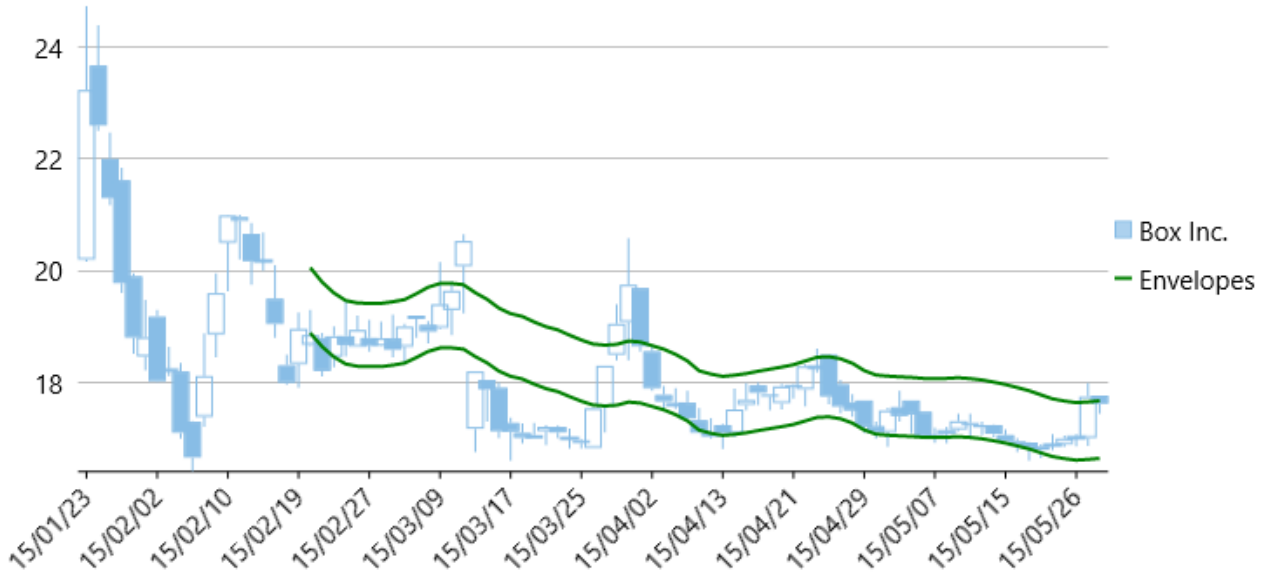
Envelopes

Envelopes（エンベロープ）オーバーレイは、FinancialChartの移動平均エンベロープオーバーレイ系列を表します。この移動平均エンベロープは、標準移動平均の上および下に設定されるパーセントベースのエンベロープです。移動平均には、単純移動平均または指数移動平均を使用できます。強気配トレンドの間は、上部エンベロープを越えたブレークスルーが強さと上昇トレンドの継続を示します。弱気配トレンドの間は、下部エンベロープを越えたブレークスルーが強さと下降トレンドの継続を示します。

FinancialChart にエンベロープオーバーレイを追加するには、FinancialChart コントロールをアプリケーションに追加し、コントロールに適切なデータソースを連結するか、**Quote Collection** でコントロールにデータを追加します。FinancialChart にデータを連結または追加するには、ItemsSource オブジェクトを使用します。

Envelopes クラスは、上下エンベロープをレンダリングするためのパーセンテージ値を受け取る **Size** プロパティと、単純移動平均または指数移動平均を指定する **Type** プロパティを公開します。**IndicatorBase** クラスは **Period** プロパティを提供します。これは、単純または指数移動平均を計算するための基準期間を指定する整数値を受け取ります。これらのプロパティの値に基づいて、エンベロープオーバーレイが計算され、FinancialChartにプロットされます。

また、FinancialChart では、アプリケーションでアラートを作成したり、動的データの使用中にログを取るために、計算された**下限 y 値**、**上限 y 値**を実行時に取得できます。



次の例は、上の図で示すように、企業 Box Inc. の一定期間の株価を考察し、同じ株価チャートにエンベロープオーバーレイをプロットします。この例では、json ファイルのデータを使用します。この json ファイルにアクセスするために、DataService.cs クラスを作成します。

⚠ json ファイルの[ビルドアクション]プロパティが**[埋め込まれたリソース]**に設定されていることを確認します。

XAML copyCode

```
<Window
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:EnvelopeOverlay"
    xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
    x:Class="EnvelopeOverlay.MainWindow"
    DataContext="{Binding RelativeSource={RelativeSource Mode=Self}}"
    mc:Ignorable="d"
    Title="MainWindow" Height="350" Width="525">
<Grid>
    <c1:C1FinancialChart x:Name="financialChart"
                         ItemsSource="{Binding Data}"
                         Background="White"
                         ChartType="Candlestick"
                         BindingX="Date"
                         ToolTipContent="{}{seriesName}&#x000A;{Date} {y}"
                         Margin="0,37,-1,71">
        <c1:FinancialSeries Binding="High,Low,Open,Close"
                            SeriesName="Box Inc." />
      <c1:Envelopes Binding="High,Low,Close"
                    Period="20"
                    Size="0.03"
                    Type="Simple"
                    SeriesName="Envelopes">
        <c1:Envelopes.Style>
          <c1:ChartStyle Stroke="Green" StrokeThickness="2" />
        </c1:Envelopes.Style>
      </c1:Envelopes>
            <c1:C1FinancialChart.AxisX>
```

```
                <c1:Axis LabelAngle="45" MajorUnit="3"/>
            </c1:C1FinancialChart.AxisX>
        </c1:C1FinancialChart>
    </Grid>
</Window>
```

DataService.cs に次の参照を追加します。

- System.Collections.Generic
- System.Linq
- System.Runtime.Serialization.Json
- System.Reflection

- **DataService.vb**

```
Public Class DataService
    Public Function GetData() As List(Of Quote)
        Dim path As String = "OverlayVB.Resources.box.json"
        'OverlayVBをアプリケーション名で置き換えます。
        Dim stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path)
        Dim ser = New DataContractJsonSerializer(GetType(Quote()))
        Dim data = DirectCast(ser.ReadObject(stream), Quote())
        Return data.ToList()
    End Function
    Shared _ds As DataService
    Public Shared Function GetService() As DataService
        If _ds Is Nothing Then
            _ds = New DataService()
        End If
        Return _ds
    End Function
End Class
```

- **DataService.cs**

```
public class DataService
{
    public List<Quote> GetData()
    {
        string path = "EnvelopeOverlay.Resources.box.json";
        //EnvelopeOverlayをアプリケーション名で置き換えます。
        var stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path);
        var ser = new DataContractJsonSerializer(typeof(Quote[]));
        var data = (Quote[])ser.ReadObject(stream);
        return data.ToList();
    }
    static DataService _ds;
    public static DataService GetService()
    {
        if (_ds == null)
            _ds = new DataService();
        return _ds;
    }
}
```

Json Data

```
[
    {"date":"15/01/23","open":20.2,"high":24.73,"low":20.16,"close":23.23,"volume":42593223},
    {"date":"15/01/26","open":23.67,"high":24.39,"low":22.5,"close":22.6,"volume":8677164},
    {"date":"15/01/27","open":22,"high":22.47,"low":21.17,"close":21.3,"volume":3272512},
    {"date":"15/01/28","open":21.62,"high":21.84,"low":19.6,"close":19.78,"volume":5047364},
    {"date":"15/01/29","open":19.9,"high":19.95,"low":18.51,"close":18.8,"volume":3419482},
    {"date":"15/01/30","open":18.47,"high":19.48,"low":18.22,"close":18.81,"volume":2266439},
    {"date":"15/02/02","open":19.18,"high":19.3,"low":18.01,"close":18.02,"volume":2071168},
```

```
{"date":"15/02/03","open":18.22,"high":18.64,"low":18.12,"close":18.24,"volume":1587435},
{"date":"15/02/04","open":18.2,"high":18.35,"low":17,"close":17.1,"volume":2912224},
{"date":"15/02/05","open":17.3,"high":17.31,"low":16.41,"close":16.66,"volume":2682187},
{"date":"15/02/06","open":17.39,"high":18.88,"low":17.21,"close":18.12,"volume":3929164},
{"date":"15/02/09","open":18.86,"high":19.95,"low":18.45,"close":19.6,"volume":3226650},
{"date":"15/02/10","open":20.5,"high":21,"low":19.63,"close":20.99,"volume":2804409},
{"date":"15/02/11","open":20.89,"high":21,"low":20.2,"close":20.96,"volume":1698365},
{"date":"15/02/12","open":20.66,"high":20.85,"low":19.75,"close":20.17,"volume":1370320},
{"date":"15/02/13","open":20.19,"high":20.68,"low":20,"close":20.18,"volume":711951},
{"date":"15/02/17","open":19.5,"high":20.1,"low":18.8,"close":19.05,"volume":2093602},
{"date":"15/02/18","open":18.31,"high":18.5,"low":17.96,"close":18,"volume":1849490},
{"date":"15/02/19","open":18.33,"high":19.25,"low":17.91,"close":18.96,"volume":1311518},
{"date":"15/02/20","open":18.68,"high":19.3,"low":18.65,"close":18.85,"volume":1001692},
{"date":"15/02/23","open":18.8,"high":18.89,"low":18.11,"close":18.21,"volume":670087},
{"date":"15/02/24","open":18.46,"high":19,"low":18.27,"close":18.83,"volume":759263},
{"date":"15/02/25","open":18.83,"high":19.48,"low":18.47,"close":18.67,"volume":915580},
{"date":"15/02/26","open":18.64,"high":19.2,"low":18.64,"close":18.94,"volume":461283},
{"date":"15/02/27","open":18.8,"high":19.12,"low":18.55,"close":18.66,"volume":617199},
{"date":"15/03/02","open":18.66,"high":19.09,"low":18.65,"close":18.79,"volume":519605},
{"date":"15/03/03","open":18.79,"high":19.21,"low":18.45,"close":18.59,"volume":832415},
{"date":"15/03/04","open":18.64,"high":19.05,"low":18.32,"close":19,"volume":539688},
{"date":"15/03/05","open":19.2,"high":19.2,"low":18.8,"close":19.14,"volume":486149},
{"date":"15/03/06","open":19.03,"high":19.1,"low":18.7,"close":18.91,"volume":685659},
{"date":"15/03/09","open":18.98,"high":20.15,"low":18.96,"close":19.4,"volume":1321363},
{"date":"15/03/10","open":19.3,"high":19.8,"low":18.85,"close":19.64,"volume":615743},
{"date":"15/03/11","open":20.08,"high":20.65,"low":19.24,"close":20.53,"volume":2167167},
{"date":"15/03/12","open":17.17,"high":18.2,"low":16.76,"close":18.2,"volume":6837638},
{"date":"15/03/13","open":18.05,"high":18.05,"low":17.3,"close":17.88,"volume":1715629},
{"date":"15/03/16","open":17.91,"high":18,"low":17.01,"close":17.13,"volume":1321313},
{"date":"15/03/17","open":17.28,"high":17.37,"low":16.6,"close":17.12,"volume":1272242},
{"date":"15/03/18","open":17.1,"high":17.27,"low":16.91,"close":17.01,"volume":530063},
{"date":"15/03/19","open":17,"high":17.28,"low":17,"close":17.06,"volume":536427},
{"date":"15/03/20","open":17.13,"high":17.24,"low":16.88,"close":17.21,"volume":1320237},
{"date":"15/03/23","open":17.21,"high":17.23,"low":17.01,"close":17.11,"volume":509798},
{"date":"15/03/24","open":17.02,"high":17.18,"low":16.82,"close":17,"volume":962149},
{"date":"15/03/25","open":16.92,"high":16.99,"low":16.82,"close":16.97,"volume":565673},
{"date":"15/03/26","open":16.83,"high":17.56,"low":16.83,"close":17.54,"volume":884523},
{"date":"15/03/27","open":17.58,"high":18.3,"low":17.11,"close":18.3,"volume":705626},
{"date":"15/03/30","open":18.5,"high":19.4,"low":18.4,"close":19.05,"volume":1151620},
{"date":"15/03/31","open":19.08,"high":20.58,"low":18.4,"close":19.75,"volume":2020679},
{"date":"15/04/01","open":19.69,"high":19.69,"low":18.55,"close":18.65,"volume":961078},
{"date":"15/04/02","open":18.56,"high":18.66,"low":17.85,"close":17.9,"volume":884233},
{"date":"15/04/06","open":17.78,"high":17.94,"low":17.51,"close":17.66,"volume":605252},
{"date":"15/04/07","open":17.62,"high":17.9,"low":17.53,"close":17.61,"volume":591988},
{"date":"15/04/08","open":17.64,"high":17.85,"low":17.32,"close":17.36,"volume":618855},
{"date":"15/04/09","open":17.33,"high":17.54,"low":17.1,"close":17.1,"volume":761855},
{"date":"15/04/10","open":17.08,"high":17.36,"low":17,"close":17.05,"volume":568373},
{"date":"15/04/13","open":17.24,"high":17.26,"low":16.81,"close":17.1,"volume":667142},
{"date":"15/04/14","open":17.1,"high":17.89,"low":17.02,"close":17.52,"volume":870138},
{"date":"15/04/15","open":17.6,"high":17.99,"low":17.5,"close":17.69,"volume":530456},
{"date":"15/04/16","open":17.95,"high":18,"low":17.6,"close":17.82,"volume":548730},
{"date":"15/04/17","open":17.75,"high":17.79,"low":17.5,"close":17.79,"volume":446373},
{"date":"15/04/20","open":17.63,"high":17.98,"low":17.52,"close":17.93,"volume":487017},
{"date":"15/04/21","open":17.96,"high":17.98,"low":17.71,"close":17.92,"volume":320302},
{"date":"15/04/22","open":17.88,"high":18.33,"low":17.57,"close":18.29,"volume":644812},
{"date":"15/04/23","open":18.29,"high":18.61,"low":18.18,"close":18.28,"volume":563879},
{"date":"15/04/24","open":18.5,"high":18.5,"low":17.61,"close":17.75,"volume":650762},
{"date":"15/04/27","open":17.97,"high":18.05,"low":17.45,"close":17.57,"volume":437294},
{"date":"15/04/28","open":17.65,"high":17.79,"low":17.39,"close":17.5,"volume":224519},
{"date":"15/04/29","open":17.68,"high":17.68,"low":17.1,"close":17.21,"volume":495706},
{"date":"15/04/30","open":17.22,"high":17.3,"low":17,"close":17.11,"volume":391040},
{"date":"15/05/01","open":17.11,"high":17.55,"low":16.85,"close":17.5,"volume":563075},
{"date":"15/05/02","open":17.56,"high":17.85,"low":17.3,"close":17.4,"volume":253138},
{"date":"15/05/05","open":17.68,"high":17.68,"low":17.09,"close":17.43,"volume":290935},
{"date":"15/05/06","open":17.48,"high":17.48,"low":17,"close":17.04,"volume":313662},
{"date":"15/05/07","open":17.05,"high":17.19,"low":16.92,"close":17.04,"volume":360284},
{"date":"15/05/08","open":17.13,"high":17.21,"low":16.91,"close":17.1,"volume":297653},
{"date":"15/05/11","open":17.16,"high":17.44,"low":17.13,"close":17.31,"volume":268504},
```

```
    {"date":"15/05/12","open":17.28,"high":17.44,"low":16.99,"close":17.24,"volume":376961},
    {"date":"15/05/13","open":17.24,"high":17.3,"low":17.06,"close":17.2,"volume":244617},
    {"date":"15/05/14","open":17.24,"high":17.25,"low":17.02,"close":17.08,"volume":252526},
    {"date":"15/05/15","open":17.06,"high":17.16,"low":16.95,"close":16.95,"volume":274783},
    {"date":"15/05/18","open":16.95,"high":17.01,"low":16.76,"close":16.87,"volume":418513},
    {"date":"15/05/19","open":16.93,"high":16.94,"low":16.6,"close":16.83,"volume":367660},
    {"date":"15/05/20","open":16.8,"high":16.9,"low":16.65,"close":16.86,"volume":297914},
    {"date":"15/05/21","open":16.9,"high":17.08,"low":16.79,"close":16.88,"volume":229346},
    {"date":"15/05/22","open":16.9,"high":17.05,"low":16.85,"close":17,"volume":253279},
    {"date":"15/05/26","open":17.03,"high":17.08,"low":16.86,"close":17.01,"volume":212640},
    {"date":"15/05/27","open":17.01,"high":17.99,"low":16.87,"close":17.75,"volume":857109},
    {"date":"15/05/28","open":17.77,"high":17.77,"low":17.44,"close":17.62,"volume":338482}
]
```

コードビューで次の参照を追加します。

- System.Collections.Generic
- System.Windows
- System.Runtime.Serialization

- **Visual Basic**

```
<DataContract>
Public Class Quote
    <DataMember(Name:="date")>
    Public Property [Date]() As String
        Get
            Return m_Date
        End Get
        Set
            m_Date = Value
        End Set
    End Property
    Private m_Date As String

    <DataMember(Name:="high")>
    Public Property High() As Double
        Get
            Return m_High
        End Get
        Set
            m_High = Value
        End Set
    End Property
    Private m_High As Double

    <DataMember(Name:="low")>
    Public Property Low() As Double
        Get
            Return m_Low
        End Get
        Set
            m_Low = Value
        End Set
    End Property
    Private m_Low As Double

    <DataMember(Name:="open")>
    Public Property Open() As Double
        Get
            Return m_Open
        End Get
        Set
            m_Open = Value
        End Set
    End Property
    Private m_Open As Double
```

```
    <DataMember(Name:="close")>
    Public Property Close() As Double
        Get
            Return m_Close
        End Get
        Set
            m_Close = Value
        End Set
    End Property
    Private m_Close As Double

    <DataMember(Name:="volume")>
    Public Property Volume() As Double
        Get
            Return m_Volume
        End Get
        Set
            m_Volume = Value
        End Set
    End Property
    Private m_Volume As Double
End Class
''' MWEnvelopes.xamlの相互作用ロジック
Partial Public Class MWEnvelopes
    Inherits Window
    Private dataService As DataService = DataService.GetService()
    Public Sub New()
        InitializeComponent()
    End Sub
    Public ReadOnly Property Data() As List(Of Quote)
        Get
            Return dataService.GetData()
        End Get
    End Property
End Class
```

- **C#**

```
[DataContract]
    public class Quote
    {
        [DataMember(Name = "date")]
        public string Date { get; set; }

        [DataMember(Name = "high")]
        public double High { get; set; }

        [DataMember(Name = "low")]
        public double Low { get; set; }

        [DataMember(Name = "open")]
        public double Open { get; set; }

        [DataMember(Name = "close")]
        public double Close { get; set; }

        [DataMember(Name = "volume")]
        public double Volume { get; set; }
    }

/// MainWindow.xamlの相互作用ロジック
public partial class MainWindow : Window
    {
        DataService dataService = DataService.GetService();
        public MainWindow()
        {
```

```
            InitializeComponent();
        }
        public List<Quote> Data
        {
            get
            {
                return dataService.GetData();
            }
        }
    }
```

先頭に移動

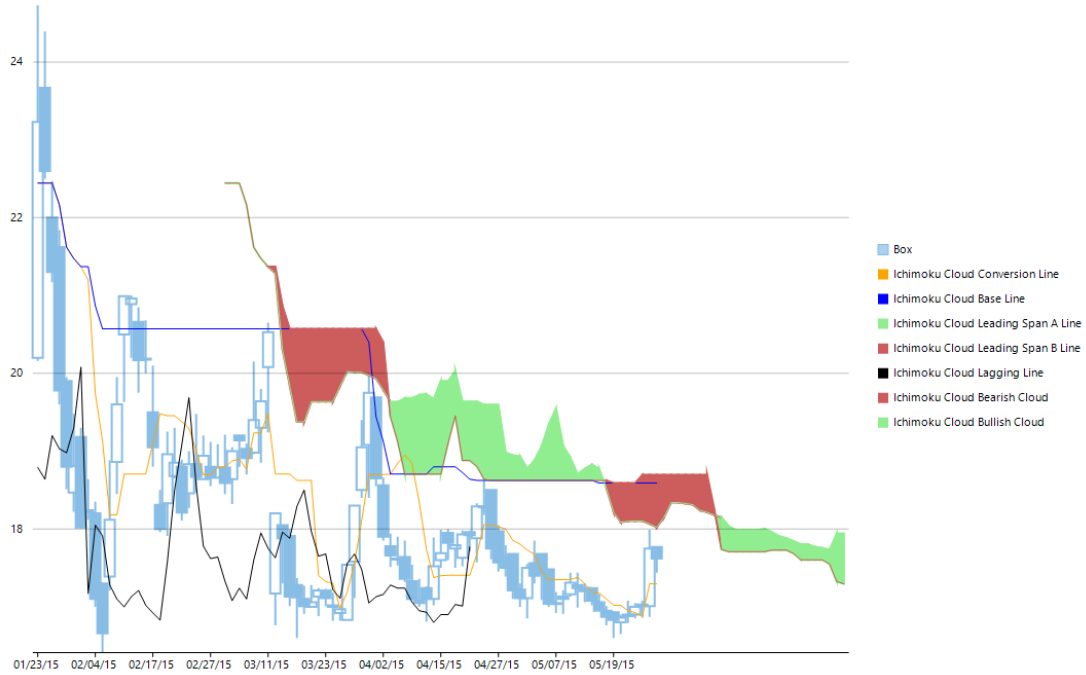
Ichimoku Clouds

Ichimoku cloud is an overlay that provides all the information about the price trend at a glance, including its direction, momentum, dynamic support and resistance levels and even trade signals. It uses moving averages to show bullish and bearish crossover points.The ichimoku cloud consists of five plots.

1. Conversion Line (9 period high+ 9 period low)/2))
2. Base Line (26 period high + 26 period low)/2))
3. Leading Span A (ConversionLine + Base Line)/2))
4. Leading Span B (52 period high + 52 period low)/2))
5. Lagging Span (Plotted 26 days in past)

Cloud, which is the most important feature of the overlay is formed between Leading Span A and the Leading span B plots and helps in identifying the trends. The trend is upwards when prices go above the cloud. However, it goes downwards when prices go below the cloud. When the Leading span A rises above the Leading Span B, it indicates strengthening of the upward trend. However, the downward trend strengthens when the Leading span B goes above the Leading Span A.

To use ichimoku cloud overlay in **FinancialChart**, add a FinancialChart control to your application and bind it to an appropriate data source using the **DataSource** property. The **IchimokuCloud** class allows you to modify the period values for each of the plotting lines, for example, conversion line has a default period of 9 days, however, you can modify the period value for conversion line using the **ConversionPeriod** property. Similarly, you can use **BasePeriod**, **LeadingPeriod**, and **LaggingPeriod** properties to modify the period values for base line, leading span, and lagging span respectively. Also, it provides styling properties for each of the five plots mentioned above, i.e. **ConversionLineStyle**, **BaseLineStyle**, **LaggingLineStyle**, **LeadingSpanALineStyle**, and **LeadingSpanBLineStyle** property. You can also set the color of bullish and bearish cloud using **BullishCloudColor** and **BearishCloudColor** properties respectively.



The following example considers stock data for a company Box Inc. over a period of time and plots ichimoku cloud on the same financial chart, as shown in the image above. The example uses data from a JSON file, and **DataService.cs** class is created to access the JSON file.

⚠ Make sure that Build Action property of the JSON file is set to **Embedded Resource**.

Make sure to add the following references in DataService.cs:

- System.Collections.Generic
- System.Linq
- System.Runtime.Serialization.Json
- System.Reflection

XAML

```
<UserControl x:Class="IchimokuOverlay.Overlays"
             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:local="clr-namespace:IchimokuOverlay"
             xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
             DataContext="{Binding RelativeSource={RelativeSource Mode=Self}}"
             mc:Ignorable="d"
             d:DesignHeight="300" d:DesignWidth="300">
    <Grid>
        <c1:C1FinancialChart
            x:Name="overlayChart"
            ItemsSource="{Binding Data}"
            Background="White"
            ChartType="Candlestick"
            BindingX="Date"
            ToolTipContent="{}{seriesName}{Date} {y}">
            <c1:FinancialSeries Binding="High,Low,Open,Close" SeriesName="Box"/>

            <c1:IchimokuCloud x:Name="ichimoku" Binding="High,Low,Close" SeriesName="Ichimoku Cloud">
                <c1:IchimokuCloud.BearishCloudColor>
                    <SolidColorBrush Color="Tomato"/>
                </c1:IchimokuCloud.BearishCloudColor>
                <c1:IchimokuCloud.BullishCloudColor>
                    <SolidColorBrush Color="LightGreen"/>
                </c1:IchimokuCloud.BullishCloudColor>
                <c1:IchimokuCloud.ConversionLineStyle>
                    <c1:ChartStyle StrokeThickness="0.7" Stroke="Orange"/>
                </c1:IchimokuCloud.ConversionLineStyle>
                <c1:IchimokuCloud.BaseLineStyle>
                    <c1:ChartStyle StrokeThickness="0.7" Stroke="Blue"/>
                </c1:IchimokuCloud.BaseLineStyle>
                <c1:IchimokuCloud.LeadingSpanALineStyle>
                    <c1:ChartStyle StrokeThickness="0.8" Stroke="LightGreen"/>
                </c1:IchimokuCloud.LeadingSpanALineStyle>
                <c1:IchimokuCloud.LeadingSpanBLineStyle>
                    <c1:ChartStyle  StrokeThickness="0.8" Stroke="Tomato"/>
                </c1:IchimokuCloud.LeadingSpanBLineStyle>
                <c1:IchimokuCloud.LaggingLineStyle>
                    <c1:ChartStyle StrokeThickness="0.7" Stroke="Black"/>
                </c1:IchimokuCloud.LaggingLineStyle>
            </c1:IchimokuCloud>
            <c1:C1FinancialChart.AxisX>
                <c1:Axis LabelAngle="45" MajorUnit="3"/>
            </c1:C1FinancialChart.AxisX>
        </c1:C1FinancialChart>
    </Grid>
</UserControl>
```

DataService.cs

```
public class DataService
    {
        public List<Quote> GetSymbolData(string symbol, int nitems = 0)
        {
            string path = string.Format("IchimokuOverlay.Resources.{0}.json", symbol);
            var stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path);
```

```
            var ser = new System.Runtime.Serialization.Json.DataContractJsonSerializer(typeof(Quote[]));
            var data = (Quote[])ser.ReadObject(stream);
            return data.ToList();
        }

        static DataService _ds;
        public static DataService GetService()
        {
            if (_ds == null)
                _ds = new DataService();
            return _ds;
        }
    }
    public class Quote
    {
        public string date { get; set; }
        public double high { get; set; }
        public double low { get; set; }
        public double open { get; set; }
        public double close { get; set; }
        public double volume { get; set; }

        public DateTime Date
        {
            get { return DateTime.ParseExact(date.ToString(), "MM/dd/yy",
System.Globalization.CultureInfo.InvariantCulture); }
        }
    }
    public class Company
    {
        public string Symbol {get; set;}
        public string Name {get; set;}
    }
```

JSON Data


```
[
    { "date": "01/23/15", "open": 20.2, "high": 24.73, "low": 20.16, "close": 23.23, "volume": 42593223 },
    { "date": "01/26/15", "open": 23.67, "high": 24.39, "low": 22.5, "close": 22.6, "volume": 8677164 },
    { "date": "01/27/15", "open": 22, "high": 22.47, "low": 21.17, "close": 21.3, "volume": 3272512 },
    { "date": "01/28/15", "open": 21.62, "high": 21.84, "low": 19.6, "close": 19.78, "volume": 5047364 },
    { "date": "01/29/15", "open": 19.9, "high": 19.95, "low": 18.51, "close": 18.8, "volume": 3419482 },
    { "date": "01/30/15", "open": 18.47, "high": 19.48, "low": 18.22, "close": 18.81, "volume": 2266439 },
    { "date": "02/02/15", "open": 19.18, "high": 19.3, "low": 18.01, "close": 18.02, "volume": 2071168 },
    { "date": "02/03/15", "open": 18.22, "high": 18.64, "low": 18.12, "close": 18.24, "volume": 1587435 },
    { "date": "02/04/15", "open": 18.2, "high": 18.35, "low": 17, "close": 17.1, "volume": 2912224 },
    { "date": "02/05/15", "open": 17.3, "high": 17.31, "low": 16.41, "close": 16.66, "volume": 2682187 },
    { "date": "02/06/15", "open": 17.39, "high": 18.88, "low": 17.21, "close": 18.12, "volume": 3929164 },
    { "date": "02/09/15", "open": 18.86, "high": 19.95, "low": 18.45, "close": 19.6, "volume": 3226650 },
    { "date": "02/10/15", "open": 20.5, "high": 21, "low": 19.63, "close": 20.99, "volume": 2804409 },
    { "date": "02/11/15", "open": 20.89, "high": 21, "low": 20.2, "close": 20.96, "volume": 1698365 },
    { "date": "02/12/15", "open": 20.66, "high": 20.85, "low": 19.75, "close": 20.17, "volume": 1370320 },
    { "date": "02/13/15", "open": 20.19, "high": 20.68, "low": 20, "close": 20.18, "volume": 711951 },
    { "date": "02/17/15", "open": 19.5, "high": 20.1, "low": 18.8, "close": 19.05, "volume": 2093602 },
    { "date": "02/18/15", "open": 18.31, "high": 18.5, "low": 17.96, "close": 18, "volume": 1849490 },
    { "date": "02/19/15", "open": 18.33, "high": 19.25, "low": 17.91, "close": 18.96, "volume": 1311518 },
    { "date": "02/20/15", "open": 18.68, "high": 19.3, "low": 18.65, "close": 18.85, "volume": 1001692 },
    { "date": "02/23/15", "open": 18.8, "high": 18.89, "low": 18.11, "close": 18.21, "volume": 670087 },
    { "date": "02/24/15", "open": 18.46, "high": 19, "low": 18.27, "close": 18.83, "volume": 759263 },
    { "date": "02/25/15", "open": 18.83, "high": 19.48, "low": 18.47, "close": 18.67, "volume": 915580 },
    { "date": "02/26/15", "open": 18.64, "high": 19.2, "low": 18.64, "close": 18.94, "volume": 461283 },
    { "date": "02/27/15", "open": 18.8, "high": 19.12, "low": 18.55, "close": 18.66, "volume": 617199 },
    { "date": "03/02/15", "open": 18.66, "high": 19.09, "low": 18.65, "close": 18.79, "volume": 519605 },
    { "date": "03/03/15", "open": 18.79, "high": 19.21, "low": 18.45, "close": 18.59, "volume": 832415 },
    { "date": "03/04/15", "open": 18.64, "high": 19.05, "low": 18.32, "close": 19, "volume": 539688 },
    { "date": "03/05/15", "open": 19.2, "high": 19.2, "low": 18.8, "close": 19.14, "volume": 486149 },
    { "date": "03/06/15", "open": 19.03, "high": 19.1, "low": 18.7, "close": 18.91, "volume": 685659 },
    { "date": "03/09/15", "open": 18.98, "high": 20.15, "low": 18.96, "close": 19.4, "volume": 1321363 },
    { "date": "03/10/15", "open": 19.3, "high": 19.8, "low": 18.85, "close": 19.64, "volume": 615743 },
    { "date": "03/11/15", "open": 20.08, "high": 20.65, "low": 19.24, "close": 20.53, "volume": 2167167 },
    { "date": "03/12/15", "open": 17.17, "high": 18.2, "low": 16.76, "close": 18.2, "volume": 6837638 },
```

```
{ "date": "03/13/15", "open": 18.05, "high": 18.05, "low": 17.3, "close": 17.88, "volume": 1715629 },
{ "date": "03/16/15", "open": 17.91, "high": 18, "low": 17.01, "close": 17.13, "volume": 1321313 },
{ "date": "03/17/15", "open": 17.28, "high": 17.37, "low": 16.6, "close": 17.12, "volume": 1272242 },
{ "date": "03/18/15", "open": 17.1, "high": 17.27, "low": 16.91, "close": 17.01, "volume": 530063 },
{ "date": "03/19/15", "open": 17, "high": 17.28, "low": 17, "close": 17.06, "volume": 536427 },
{ "date": "03/20/15", "open": 17.13, "high": 17.24, "low": 16.88, "close": 17.21, "volume": 1320237 },
{ "date": "03/23/15", "open": 17.21, "high": 17.23, "low": 17.01, "close": 17.11, "volume": 509798 },
{ "date": "03/24/15", "open": 17.02, "high": 17.18, "low": 16.82, "close": 17, "volume": 962149 },
{ "date": "03/25/15", "open": 16.92, "high": 16.99, "low": 16.82, "close": 16.97, "volume": 565673 },
{ "date": "03/26/15", "open": 16.83, "high": 17.56, "low": 16.83, "close": 17.54, "volume": 884523 },
{ "date": "03/27/15", "open": 17.58, "high": 18.3, "low": 17.11, "close": 18.3, "volume": 705626 },
{ "date": "03/30/15", "open": 18.5, "high": 19.4, "low": 18.4, "close": 19.05, "volume": 1151620 },
{ "date": "03/31/15", "open": 19.08, "high": 20.58, "low": 18.4, "close": 19.75, "volume": 2020679 },
{ "date": "04/01/15", "open": 19.69, "high": 19.69, "low": 18.55, "close": 18.65, "volume": 961078 },
{ "date": "04/02/15", "open": 18.56, "high": 18.66, "low": 17.85, "close": 17.9, "volume": 884233 },
{ "date": "04/06/15", "open": 17.78, "high": 17.94, "low": 17.51, "close": 17.66, "volume": 605252 },
{ "date": "04/07/15", "open": 17.62, "high": 17.9, "low": 17.53, "close": 17.61, "volume": 591988 },
{ "date": "04/08/15", "open": 17.64, "high": 17.85, "low": 17.32, "close": 17.36, "volume": 618855 },
{ "date": "04/09/15", "open": 17.33, "high": 17.54, "low": 17.1, "close": 17.1, "volume": 761855 },
{ "date": "04/10/15", "open": 17.08, "high": 17.36, "low": 17, "close": 17.05, "volume": 568373 },
{ "date": "04/13/15", "open": 17.24, "high": 17.26, "low": 16.81, "close": 17.1, "volume": 667142 },
{ "date": "04/14/15", "open": 17.1, "high": 17.89, "low": 17.02, "close": 17.52, "volume": 870138 },
{ "date": "04/15/15", "open": 17.6, "high": 17.99, "low": 17.5, "close": 17.69, "volume": 530456 },
{ "date": "04/16/15", "open": 17.95, "high": 18, "low": 17.6, "close": 17.82, "volume": 548730 },
{ "date": "04/17/15", "open": 17.75, "high": 17.79, "low": 17.5, "close": 17.79, "volume": 446373 },
{ "date": "04/20/15", "open": 17.63, "high": 17.98, "low": 17.52, "close": 17.93, "volume": 487017 },
{ "date": "04/21/15", "open": 17.96, "high": 17.98, "low": 17.71, "close": 17.92, "volume": 320302 },
{ "date": "04/22/15", "open": 17.88, "high": 18.33, "low": 17.57, "close": 18.29, "volume": 644812 },
{ "date": "04/23/15", "open": 18.29, "high": 18.61, "low": 18.18, "close": 18.28, "volume": 563879 },
{ "date": "04/24/15", "open": 18.5, "high": 18.5, "low": 17.61, "close": 17.75, "volume": 650762 },
{ "date": "04/27/15", "open": 17.97, "high": 18.05, "low": 17.45, "close": 17.57, "volume": 437294 },
{ "date": "04/28/15", "open": 17.65, "high": 17.79, "low": 17.39, "close": 17.5, "volume": 224519 },
{ "date": "04/29/15", "open": 17.68, "high": 17.68, "low": 17.1, "close": 17.21, "volume": 495706 },
{ "date": "04/30/15", "open": 17.22, "high": 17.3, "low": 17, "close": 17.11, "volume": 391040 },
{ "date": "05/01/15", "open": 17.11, "high": 17.55, "low": 16.85, "close": 17.5, "volume": 563075 },
{ "date": "05/04/15", "open": 17.56, "high": 17.85, "low": 17.3, "close": 17.4, "volume": 253138 },
{ "date": "05/05/15", "open": 17.68, "high": 17.68, "low": 17.09, "close": 17.43, "volume": 290935 },
{ "date": "05/06/15", "open": 17.48, "high": 17.48, "low": 17, "close": 17.04, "volume": 313662 },
{ "date": "05/07/15", "open": 17.05, "high": 17.19, "low": 16.92, "close": 17.04, "volume": 360284 },
{ "date": "05/08/15", "open": 17.13, "high": 17.21, "low": 16.91, "close": 17.1, "volume": 297653 },
{ "date": "05/11/15", "open": 17.16, "high": 17.44, "low": 17.13, "close": 17.31, "volume": 268504 },
{ "date": "05/12/15", "open": 17.28, "high": 17.44, "low": 16.99, "close": 17.24, "volume": 376961 },
{ "date": "05/13/15", "open": 17.24, "high": 17.3, "low": 17.06, "close": 17.2, "volume": 244617 },
{ "date": "05/14/15", "open": 17.24, "high": 17.25, "low": 17.02, "close": 17.08, "volume": 252526 },
{ "date": "05/15/15", "open": 17.06, "high": 17.16, "low": 16.95, "close": 16.95, "volume": 274783 },
{ "date": "05/18/15", "open": 16.95, "high": 17.01, "low": 16.76, "close": 16.87, "volume": 418513 },
{ "date": "05/19/15", "open": 16.93, "high": 16.94, "low": 16.6, "close": 16.83, "volume": 367660 },
{ "date": "05/20/15", "open": 16.8, "high": 16.9, "low": 16.65, "close": 16.86, "volume": 297914 },
{ "date": "05/21/15", "open": 16.9, "high": 17.08, "low": 16.79, "close": 16.88, "volume": 229346 },
{ "date": "05/22/15", "open": 16.9, "high": 17.05, "low": 16.85, "close": 17, "volume": 253279 },
{ "date": "05/26/15", "open": 17.03, "high": 17.08, "low": 16.86, "close": 17.01, "volume": 212640 },
{ "date": "05/27/15", "open": 17.01, "high": 17.99, "low": 16.87, "close": 17.75, "volume": 857109 },
{ "date": "05/28/15", "open": 17.77, "high": 17.77, "low": 17.44, "close": 17.62, "volume": 338482 }
        ]
```


MainWindow.xaml.cs

```
public partial class MainWindow : Window
    {
        DataService dataService = DataService.GetService();
        public MainWindow()
        {
            InitializeComponent();
        }
        public List<Quote> Data
        {
            get
            {
                return dataService.GetSymbolData("box");
            }
        }
     }
```

フィボナッチツール

フィボナッチツールを使用すると、株価チャートで警告レベルを計算してプロットでき、テクニカル分析に便利です。金融商品トレーダーのテクニカル分析で商品の株価とボリュームの変化を予想するために、フィボナッチ数列に含まれる数の間の数学的な関係（比）が使用されます。

以下のセクションでは、FinancialChart で使用できるフィボナッチツールについて説明します。

フィボナッチリトレースメント
: FinancialChart でフィボナッチリトレースメントを実装する方法について説明します。

フィボナッチアーク
: FinancialChart でフィボナッチアークを実装する方法について説明します。

フィボナッチファン
: FinancialChart でフィボナッチファンを実装する方法について説明します。

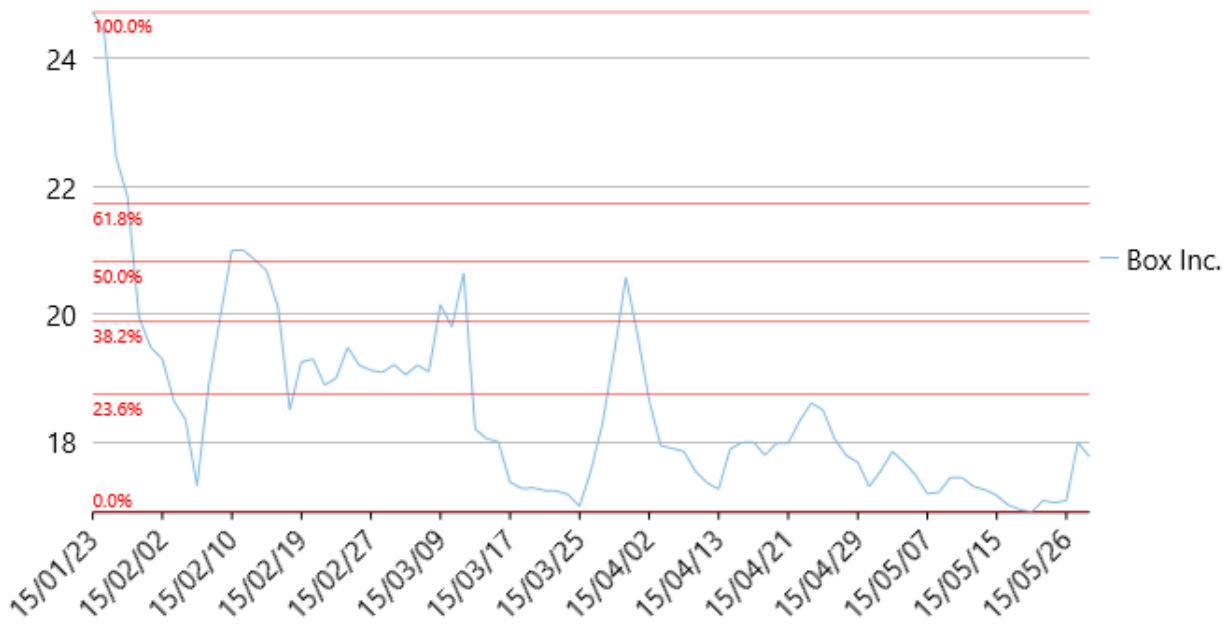
フィボナッチタイムゾーン
: FinancialChart でフィボナッチタイムゾーンを実装する方法について説明します。

フィボナッチリトレースメント

フィボナッチリトレースメントは、FinancialChart内の何本かの水平線（リトレースメントレベル）で構成されるテクニカル分析ツールです。これらのレベルは、トレンドが元の方向で継続するまでのサポート（株価が下がらない）またはレジスタンス（株価が上がらない）の幅を明示するために使用されます。これらのフィボナッチレベルは、最初に高低（両極値）間の傾向線を描画し、次のそれらの垂直距離を主要なフィボナッチ比率23.6%、38.2%、50%、61.8%、100% で分割することによって作成されます。重要なフィボナッチリトレースメントレベルは、最大プルバックゾーンの 61.8% で、明確な買いまたは売りシグナルを示します。

FinancialChart でフィボナッチリトレースメントを使用するには、コントロールに適切なデータソースを連結するか、**Quote Collection** でコントロールにデータを追加します。FinancialChart にデータを連結または追加するには、ItemsSource オブジェクトを使用します。

Fibonacci クラスは、**Uptrend** および **Levels** プロパティを公開します。これらのプロパティで設定した値に基づいて、FinancialChart に警告レベルがプロットされます。



次の例は、上の図で示すように、企業 Box Inc. の一定期間の株価を考察し、同じ株価チャートに警告レベル（リトレースメント）をプロットします。この例では、json ファイルのデータを使用します。この json ファイルにアクセスするために、DataService.cs クラスを作成します。

⚠ json ファイルの[ビルドアクション]プロパティが**[埋め込まれたリソース]**に設定されていることを確認します。

XAML copyCode

```
    <Window
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Fibonacci"
        xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
        x:Class="Fibonacci.MainWindow"
        mc:Ignorable="d"
        Title="MainWindow"
        DataContext="{Binding RelativeSource={RelativeSource Mode=Self}}">
    <Grid>

        <c1:C1FinancialChart x:Name="financialChart"
                             ItemsSource="{Binding Data}"
                             BindingX="Date"
                             ChartType="Line"
                             ToolTipContent="{}{seriesName}&#x000A;{Date} {y}">
        <c1:FinancialSeries Binding="High,Low,Open,Close"
                            ChartType="Line"
                            SeriesName="Box Inc."/>
          <c1:Fibonacci Binding="Close">
            <c1:Fibonacci.Style>
              <c1:ChartStyle Fill="Red"
                             Stroke="Red"
                             StrokeThickness="0.5"
                             FontSize="10"/>
            </c1:Fibonacci.Style>
          </c1:Fibonacci>
          <c1:C1FinancialChart.AxisX>
                <c1:Axis LabelAngle="45" MajorUnit="3"/>
          </c1:C1FinancialChart.AxisX>

        </c1:C1FinancialChart>

    </Grid>
</Window>
```

DataService.cs に次の参照を追加します。

- System.Collections.Generic
- System.Linq
- System.Runtime.Serialization.Json
- System.Reflection

- **DataService.vb**

```
Public Class DataService
    Public Function GetData() As List(Of Quote)
        Dim path As String = "FibonacciVB.Resources.box.json"
        'FibonacciVBをアプリケーション名で置き換えます。
        Dim stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path)
        Dim ser = New DataContractJsonSerializer(GetType(Quote()))
        Dim data = DirectCast(ser.ReadObject(stream), Quote())
        Return data.ToList()
```

```
    End Function
    Shared _ds As DataService
    Public Shared Function GetService() As DataService
        If _ds Is Nothing Then
            _ds = New DataService()
        End If
        Return _ds
    End Function
End Class
```

- **DataService.cs**

```
public class DataService
{
    public List<Quote> GetData()
    {
        string path = "Fibonacci.Resources.box.json";
        //Fibonacciをアプリケーション名で置き換えます。
        var stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path);
        var ser = new DataContractJsonSerializer(typeof(Quote[]));
        var data = (Quote[])ser.ReadObject(stream);
        return data.ToList();
    }
    static DataService _ds;
    public static DataService GetService()
    {
        if (_ds == null)
            _ds = new DataService();
        return _ds;
    }
}
```

Json Data

```
[
    {"date":"15/01/23","open":20.2,"high":24.73,"low":20.16,"close":23.23,"volume":42593223},
    {"date":"15/01/26","open":23.67,"high":24.39,"low":22.5,"close":22.6,"volume":8677164},
    {"date":"15/01/27","open":22,"high":22.47,"low":21.17,"close":21.3,"volume":3272512},
    {"date":"15/01/28","open":21.62,"high":21.84,"low":19.6,"close":19.78,"volume":5047364},
    {"date":"15/01/29","open":19.9,"high":19.95,"low":18.51,"close":18.8,"volume":3419482},
    {"date":"15/01/30","open":18.47,"high":19.48,"low":18.22,"close":18.81,"volume":2266439},
    {"date":"15/02/02","open":19.18,"high":19.3,"low":18.01,"close":18.02,"volume":2071168},
    {"date":"15/02/03","open":18.22,"high":18.64,"low":18.12,"close":18.24,"volume":1587435},
    {"date":"15/02/04","open":18.2,"high":18.35,"low":17,"close":17.1,"volume":2912224},
    {"date":"15/02/05","open":17.3,"high":17.31,"low":16.41,"close":16.66,"volume":2682187},
    {"date":"15/02/06","open":17.39,"high":18.88,"low":17.21,"close":18.12,"volume":3929164},
    {"date":"15/02/09","open":18.86,"high":19.95,"low":18.45,"close":19.6,"volume":3226650},
    {"date":"15/02/10","open":20.5,"high":21,"low":19.63,"close":20.99,"volume":2804409},
    {"date":"15/02/11","open":20.89,"high":21,"low":20.2,"close":20.96,"volume":1698365},
    {"date":"15/02/12","open":20.66,"high":20.85,"low":19.75,"close":20.17,"volume":1370320},
    {"date":"15/02/13","open":20.19,"high":20.68,"low":20,"close":20.18,"volume":711951},
    {"date":"15/02/17","open":19.5,"high":20.1,"low":18.8,"close":19.05,"volume":2093602},
    {"date":"15/02/18","open":18.31,"high":18.5,"low":17.96,"close":18,"volume":1849490},
    {"date":"15/02/19","open":18.33,"high":19.25,"low":17.91,"close":18.96,"volume":1311518},
    {"date":"15/02/20","open":18.68,"high":19.3,"low":18.65,"close":18.85,"volume":1001692},
    {"date":"15/02/23","open":18.8,"high":18.89,"low":18.11,"close":18.21,"volume":670087},
    {"date":"15/02/24","open":18.46,"high":19,"low":18.27,"close":18.83,"volume":759263},
    {"date":"15/02/25","open":18.83,"high":19.48,"low":18.47,"close":18.67,"volume":915580},
    {"date":"15/02/26","open":18.64,"high":19.2,"low":18.64,"close":18.94,"volume":461283},
    {"date":"15/02/27","open":18.8,"high":19.12,"low":18.55,"close":18.66,"volume":617199},
    {"date":"15/03/02","open":18.66,"high":19.09,"low":18.65,"close":18.79,"volume":519605},
    {"date":"15/03/03","open":18.79,"high":19.21,"low":18.45,"close":18.59,"volume":832415},
    {"date":"15/03/04","open":18.64,"high":19.05,"low":18.32,"close":19,"volume":539688},
    {"date":"15/03/05","open":19.2,"high":19.2,"low":18.8,"close":19.14,"volume":486149},
    {"date":"15/03/06","open":19.03,"high":19.1,"low":18.7,"close":18.91,"volume":685659},
    {"date":"15/03/09","open":18.98,"high":20.15,"low":18.96,"close":19.4,"volume":1321363},
    {"date":"15/03/10","open":19.3,"high":19.8,"low":18.85,"close":19.64,"volume":615743},
    {"date":"15/03/11","open":20.08,"high":20.65,"low":19.24,"close":20.53,"volume":2167167},
```

```
    {"date":"15/03/12","open":17.17,"high":18.2,"low":16.76,"close":18.2,"volume":6837638},
    {"date":"15/03/13","open":18.05,"high":18.05,"low":17.3,"close":17.88,"volume":1715629},
    {"date":"15/03/16","open":17.91,"high":18,"low":17.01,"close":17.13,"volume":1321313},
    {"date":"15/03/17","open":17.28,"high":17.37,"low":16.6,"close":17.12,"volume":1272242},
    {"date":"15/03/18","open":17.1,"high":17.27,"low":16.91,"close":17.01,"volume":530063},
    {"date":"15/03/19","open":17,"high":17.28,"low":17,"close":17.06,"volume":536427},
    {"date":"15/03/20","open":17.13,"high":17.24,"low":16.88,"close":17.21,"volume":1320237},
    {"date":"15/03/23","open":17.21,"high":17.23,"low":17.01,"close":17.11,"volume":509798},
    {"date":"15/03/24","open":17.02,"high":17.18,"low":16.82,"close":17,"volume":962149},
    {"date":"15/03/25","open":16.92,"high":16.99,"low":16.82,"close":16.97,"volume":565673},
    {"date":"15/03/26","open":16.83,"high":17.56,"low":16.83,"close":17.54,"volume":884523},
    {"date":"15/03/27","open":17.58,"high":18.3,"low":17.11,"close":18.3,"volume":705626},
    {"date":"15/03/30","open":18.5,"high":19.4,"low":18.4,"close":19.05,"volume":1151620},
    {"date":"15/03/31","open":19.08,"high":20.58,"low":18.4,"close":19.75,"volume":2020679},
    {"date":"15/04/01","open":19.69,"high":19.69,"low":18.55,"close":18.65,"volume":961078},
    {"date":"15/04/02","open":18.56,"high":18.66,"low":17.85,"close":17.9,"volume":884233},
    {"date":"15/04/06","open":17.78,"high":17.94,"low":17.51,"close":17.66,"volume":605252},
    {"date":"15/04/07","open":17.62,"high":17.9,"low":17.53,"close":17.61,"volume":591988},
    {"date":"15/04/08","open":17.64,"high":17.85,"low":17.32,"close":17.36,"volume":618855},
    {"date":"15/04/09","open":17.33,"high":17.54,"low":17.1,"close":17.1,"volume":761855},
    {"date":"15/04/10","open":17.08,"high":17.36,"low":17,"close":17.05,"volume":568373},
    {"date":"15/04/13","open":17.24,"high":17.26,"low":16.81,"close":17.1,"volume":667142},
    {"date":"15/04/14","open":17.1,"high":17.89,"low":17.02,"close":17.52,"volume":870138},
    {"date":"15/04/15","open":17.6,"high":17.99,"low":17.5,"close":17.69,"volume":530456},
    {"date":"15/04/16","open":17.95,"high":18,"low":17.6,"close":17.82,"volume":548730},
    {"date":"15/04/17","open":17.75,"high":17.79,"low":17.5,"close":17.79,"volume":446373},
    {"date":"15/04/20","open":17.63,"high":17.98,"low":17.52,"close":17.93,"volume":487017},
    {"date":"15/04/21","open":17.96,"high":17.98,"low":17.71,"close":17.92,"volume":320302},
    {"date":"15/04/22","open":17.88,"high":18.33,"low":17.57,"close":18.29,"volume":644812},
    {"date":"15/04/23","open":18.29,"high":18.61,"low":18.18,"close":18.28,"volume":563879},
    {"date":"15/04/24","open":18.5,"high":18.5,"low":17.61,"close":17.75,"volume":650762},
    {"date":"15/04/27","open":17.97,"high":18.05,"low":17.45,"close":17.57,"volume":437294},
    {"date":"15/04/28","open":17.65,"high":17.79,"low":17.39,"close":17.5,"volume":224519},
    {"date":"15/04/29","open":17.68,"high":17.68,"low":17.1,"close":17.21,"volume":495706},
    {"date":"15/04/30","open":17.22,"high":17.3,"low":17,"close":17.11,"volume":391040},
    {"date":"15/05/01","open":17.11,"high":17.55,"low":16.85,"close":17.5,"volume":563075},
    {"date":"15/05/02","open":17.56,"high":17.85,"low":17.3,"close":17.4,"volume":253138},
    {"date":"15/05/05","open":17.68,"high":17.68,"low":17.09,"close":17.43,"volume":290935},
    {"date":"15/05/06","open":17.48,"high":17.48,"low":17,"close":17.04,"volume":313662},
    {"date":"15/05/07","open":17.05,"high":17.19,"low":16.92,"close":17.04,"volume":360284},
    {"date":"15/05/08","open":17.13,"high":17.21,"low":16.91,"close":17.1,"volume":297653},
    {"date":"15/05/11","open":17.16,"high":17.44,"low":17.13,"close":17.31,"volume":268504},
    {"date":"15/05/12","open":17.28,"high":17.44,"low":16.99,"close":17.24,"volume":376961},
    {"date":"15/05/13","open":17.24,"high":17.3,"low":17.06,"close":17.2,"volume":244617},
    {"date":"15/05/14","open":17.24,"high":17.25,"low":17.02,"close":17.08,"volume":252526},
    {"date":"15/05/15","open":17.06,"high":17.16,"low":16.95,"close":16.95,"volume":274783},
    {"date":"15/05/18","open":16.95,"high":17.01,"low":16.76,"close":16.87,"volume":418513},
    {"date":"15/05/19","open":16.93,"high":16.94,"low":16.6,"close":16.83,"volume":367660},
    {"date":"15/05/20","open":16.8,"high":16.9,"low":16.65,"close":16.86,"volume":297914},
    {"date":"15/05/21","open":16.9,"high":17.08,"low":16.79,"close":16.88,"volume":229346},
    {"date":"15/05/22","open":16.9,"high":17.05,"low":16.85,"close":17,"volume":253279},
    {"date":"15/05/26","open":17.03,"high":17.08,"low":16.86,"close":17.01,"volume":212640},
    {"date":"15/05/27","open":17.01,"high":17.99,"low":16.87,"close":17.75,"volume":857109},
    {"date":"15/05/28","open":17.77,"high":17.77,"low":17.44,"close":17.62,"volume":338482}
]
```


コードビューで次の参照を追加します。

- System.Collections.Generic
- System.Windows
- System.Runtime.Serialization

- **Visual Basic**

```
<DataContract>
Public Class Quote
    <DataMember(Name:="date")>
```

```
    Public Property [Date]() As String
        Get
            Return m_Date
        End Get
        Set
            m_Date = Value
        End Set
    End Property
    Private m_Date As String

    <DataMember(Name:="high")>
    Public Property High() As Double
        Get
            Return m_High
        End Get
        Set
            m_High = Value
        End Set
    End Property
    Private m_High As Double

    <DataMember(Name:="low")>
    Public Property Low() As Double
        Get
            Return m_Low
        End Get
        Set
            m_Low = Value
        End Set
    End Property
    Private m_Low As Double

    <DataMember(Name:="open")>
    Public Property Open() As Double
        Get
            Return m_Open
        End Get
        Set
            m_Open = Value
        End Set
    End Property
    Private m_Open As Double

    <DataMember(Name:="close")>
    Public Property Close() As Double
        Get
            Return m_Close
        End Get
        Set
            m_Close = Value
        End Set
    End Property
    Private m_Close As Double

    <DataMember(Name:="volume")>
    Public Property Volume() As Double
        Get
            Return m_Volume
        End Get
        Set
            m_Volume = Value
        End Set
    End Property
    Private m_Volume As Double
End Class
''' MainWindow.xamlの相互作用ロジック
Partial Public Class MainWindow
    Inherits Window
```

```
    Private dataService As DataService = dataService.GetService()
    Public Sub New()
        InitializeComponent()
    End Sub
    Public ReadOnly Property Data() As List(Of Quote)
        Get
            Return dataService.GetData()
        End Get
    End Property
End Class
```

- **C#**

```
[DataContract]
public class Quote
{
    [DataMember(Name = "date")]
    public string Date { get; set; }

    [DataMember(Name = "high")]
    public double High { get; set; }

    [DataMember(Name = "low")]
    public double Low { get; set; }

    [DataMember(Name = "open")]
    public double Open { get; set; }

    [DataMember(Name = "close")]
    public double Close { get; set; }

    [DataMember(Name = "volume")]
    public double Volume { get; set; }
}
/// MainWindow.xamlの相互作用ロジック
public partial class MainWindow : Window
{
    DataService dataService = DataService.GetService();
    public MainWindow()
    {
        InitializeComponent();
    }
    public List<Quote> Data
    {
        get
        {
            return dataService.GetData();
        }
    }
}
```

先頭に移動

フィボナッチアーク

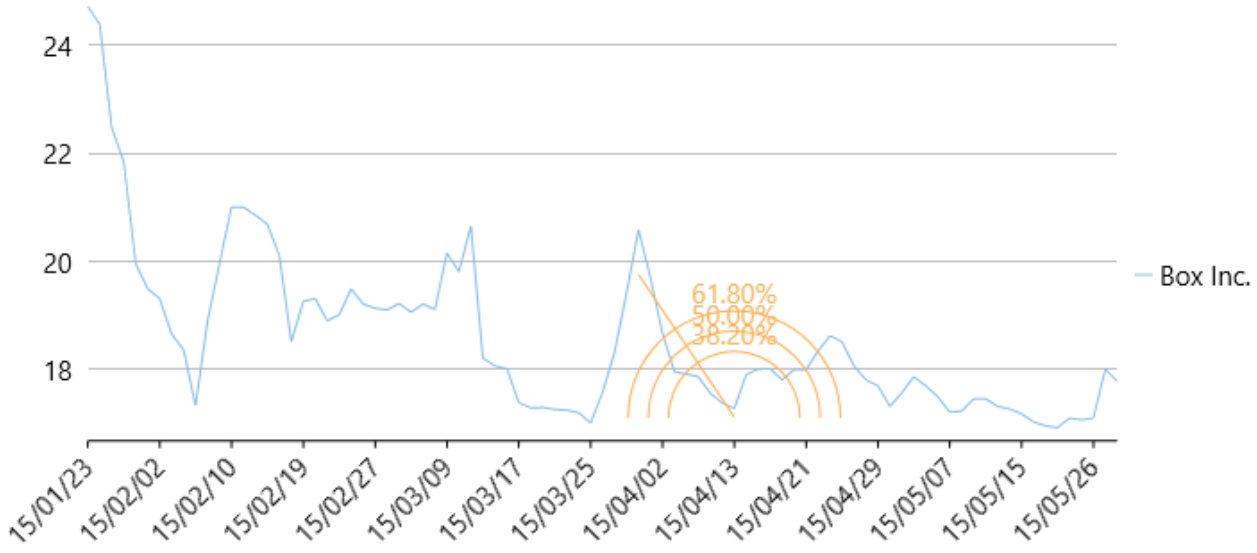
フィボナッチアークは、FinancialChart 内の傾向線から広がる 3本の曲線とベースラインで構成されるテクニカル分析ツールです。これらのアークを使用して、トレーダーは、下降後のカウンタートレンドバウンドのレジスタンスゾーンまたはリバーサルゾーンを予想します。短期間の小さな株価変化に対しては、短いベースラインと幅が狭いアークが作成されます。長期間の大きな価格変化に対しては、長いベースラインと幅が広いアークが作成されます。

フィボナッチアークを作成するには、指定された期間の高（ピーク）と低（谷）の 2 点間にベースラインを描画します。このベースラインと主要なフィボナッチレベル 38.2%、50%、61.8% で交差するようにアークが描画されます。株価の変動だけが考慮されるフィボナッチリトレースメントとは異なり、フィボナッチアークでは時間要素も考慮されます。

FinancialChart で、フィボナッチアークを使用するには、コントロールをアプリケーションに追加し、コントロールに適切なデータソースを連結するか、**Quote Collection** でコントロールにデータを追加します。FinancialChartにデータを連結または追加するには、

ItemsSource オブジェクトを使用します。

Fibonacci クラスは、**Uptrend** プロパティを公開します。**FibonacciArcs** クラスのオブジェクトを作成すると、フィボナッチアークがチャートで有効になります。さらに、**FibonacciArcs** クラスは、**StartX**、**EndX**、**StartY**、**EndY** の各プロパティを公開します。これらのプロパティに基づいて、FinancialChart にカーブがプロットされます。



次の例は、上の図で示すように、企業 Box Inc. の一定期間の株価を考察し、同じ株価チャートに曲線をプロットします。この例では、json ファイルのデータを使用します。この json ファイルにアクセスするために、DataService.cs クラスを作成します。

⚠ json ファイルの[ビルドアクション]プロパティが**[埋め込まれたリソース]**に設定されていることを確認します。

XAML copyCode

```
<Window
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Fibonacci"
        xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
        x:Class="Fibonacci.MainWindowArcs"
        mc:Ignorable="d"
        Title="MainWindowArcs"
        DataContext="{Binding RelativeSource={RelativeSource Mode=Self}}"
        Height="300" Width="300">

    <Grid>
        <c1:C1FinancialChart x:Name="financialChart"
                             ItemsSource="{Binding Data}"
                             BindingX="Date"
                             ChartType="Line"
                             ToolTipContent="{}{seriesName}&#x000A;{Date} {y}">
            <c1:FinancialSeries Binding="High,Low,Open,Close"
                                ChartType="Line"
                                SeriesName="Box Inc."/>

          <c1:FibonacciArcs x:Name="Arcs"
                            Binding="Close"
                            StartX="46"
```

```
                            StartY="19.75"
                            EndX="54"
                            EndY="17.1">
            <c1:FibonacciArcs.Style>
              <c1:ChartStyle Stroke="Green" />
            </c1:FibonacciArcs.Style>
          </c1:FibonacciArcs>

            <c1:C1FinancialChart.AxisX>
                <c1:Axis LabelAngle="45" MajorUnit="3"/>
            </c1:C1FinancialChart.AxisX>

        </c1:C1FinancialChart>
    </Grid>
</Window>
```

DataService.cs に次の参照を追加します。

- System.Collections.Generic
- System.Linq
- System.Runtime.Serialization.Json
- System.Reflection

- **DataService.vb**

```
Public Class DataService
    Public Function GetData() As List(Of Quote)
        Dim path As String = "FibonacciVB.Resources.box.json"
        'FibonacciVBをアプリケーション名で置き換えます。
        Dim stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path)
        Dim ser = New DataContractJsonSerializer(GetType(Quote()))
        Dim data = DirectCast(ser.ReadObject(stream), Quote())
        Return data.ToList()
    End Function
    Shared _ds As DataService
    Public Shared Function GetService() As DataService
        If _ds Is Nothing Then
            _ds = New DataService()
        End If
        Return _ds
    End Function
End Class
```

- **DataService.cs**

```
public class DataService
{
    public List<Quote> GetData()
    {
        string path = "Fibonacci.Resources.box.json";
        //Fibonacciをアプリケーション名で置き換えます。
        var stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path);
        var ser = new DataContractJsonSerializer(typeof(Quote[]));
        var data = (Quote[])ser.ReadObject(stream);
        return data.ToList();
    }
    static DataService _ds;
    public static DataService GetService()
    {
        if (_ds == null)
            _ds = new DataService();
        return _ds;
    }
```

```
}
```

Json Data


```
[
    {"date":"15/01/23","open":20.2,"high":24.73,"low":20.16,"close":23.23,"volume":42593223},
    {"date":"15/01/26","open":23.67,"high":24.39,"low":22.5,"close":22.6,"volume":8677164},
    {"date":"15/01/27","open":22,"high":22.47,"low":21.17,"close":21.3,"volume":3272512},
    {"date":"15/01/28","open":21.62,"high":21.84,"low":19.6,"close":19.78,"volume":5047364},
    {"date":"15/01/29","open":19.9,"high":19.95,"low":18.51,"close":18.8,"volume":3419482},
    {"date":"15/01/30","open":18.47,"high":19.48,"low":18.22,"close":18.81,"volume":2266439},
    {"date":"15/02/02","open":19.18,"high":19.3,"low":18.01,"close":18.02,"volume":2071168},
    {"date":"15/02/03","open":18.22,"high":18.64,"low":18.12,"close":18.24,"volume":1587435},
    {"date":"15/02/04","open":18.2,"high":18.35,"low":17,"close":17.1,"volume":2912224},
    {"date":"15/02/05","open":17.3,"high":17.31,"low":16.41,"close":16.66,"volume":2682187},
    {"date":"15/02/06","open":17.39,"high":18.88,"low":17.21,"close":18.12,"volume":3929164},
    {"date":"15/02/09","open":18.86,"high":19.95,"low":18.45,"close":19.6,"volume":3226650},
    {"date":"15/02/10","open":20.5,"high":21,"low":19.63,"close":20.99,"volume":2804409},
    {"date":"15/02/11","open":20.89,"high":21,"low":20.2,"close":20.96,"volume":1698365},
    {"date":"15/02/12","open":20.66,"high":20.85,"low":19.75,"close":20.17,"volume":1370320},
    {"date":"15/02/13","open":20.19,"high":20.68,"low":20,"close":20.18,"volume":711951},
    {"date":"15/02/17","open":19.5,"high":20.1,"low":18.8,"close":19.05,"volume":2093602},
    {"date":"15/02/18","open":18.31,"high":18.5,"low":17.96,"close":18,"volume":1849490},
    {"date":"15/02/19","open":18.33,"high":19.25,"low":17.91,"close":18.96,"volume":1311518},
    {"date":"15/02/20","open":18.68,"high":19.3,"low":18.65,"close":18.85,"volume":1001692},
    {"date":"15/02/23","open":18.8,"high":18.89,"low":18.11,"close":18.21,"volume":670087},
    {"date":"15/02/24","open":18.46,"high":19,"low":18.27,"close":18.83,"volume":759263},
    {"date":"15/02/25","open":18.83,"high":19.48,"low":18.47,"close":18.67,"volume":915580},
    {"date":"15/02/26","open":18.64,"high":19.2,"low":18.64,"close":18.94,"volume":461283},
    {"date":"15/02/27","open":18.8,"high":19.12,"low":18.55,"close":18.66,"volume":617199},
    {"date":"15/03/02","open":18.66,"high":19.09,"low":18.65,"close":18.79,"volume":519605},
    {"date":"15/03/03","open":18.79,"high":19.21,"low":18.45,"close":18.59,"volume":832415},
    {"date":"15/03/04","open":18.64,"high":19.05,"low":18.32,"close":19,"volume":539688},
    {"date":"15/03/05","open":19.2,"high":19.2,"low":18.8,"close":19.14,"volume":486149},
    {"date":"15/03/06","open":19.03,"high":19.1,"low":18.7,"close":18.91,"volume":685659},
    {"date":"15/03/09","open":18.98,"high":20.15,"low":18.96,"close":19.4,"volume":1321363},
    {"date":"15/03/10","open":19.3,"high":19.8,"low":18.85,"close":19.64,"volume":615743},
    {"date":"15/03/11","open":20.08,"high":20.65,"low":19.24,"close":20.53,"volume":2167167},
    {"date":"15/03/12","open":17.17,"high":18.2,"low":16.76,"close":18.2,"volume":6837638},
    {"date":"15/03/13","open":18.05,"high":18.05,"low":17.3,"close":17.88,"volume":1715629},
    {"date":"15/03/16","open":17.91,"high":18,"low":17.01,"close":17.13,"volume":1321313},
    {"date":"15/03/17","open":17.28,"high":17.37,"low":16.6,"close":17.12,"volume":1272242},
    {"date":"15/03/18","open":17.1,"high":17.27,"low":16.91,"close":17.01,"volume":530063},
    {"date":"15/03/19","open":17,"high":17.28,"low":17,"close":17.06,"volume":536427},
    {"date":"15/03/20","open":17.13,"high":17.24,"low":16.88,"close":17.21,"volume":1320237},
    {"date":"15/03/23","open":17.21,"high":17.23,"low":17.01,"close":17.11,"volume":509798},
    {"date":"15/03/24","open":17.02,"high":17.18,"low":16.82,"close":17,"volume":962149},
    {"date":"15/03/25","open":16.92,"high":16.99,"low":16.82,"close":16.97,"volume":565673},
    {"date":"15/03/26","open":16.83,"high":17.56,"low":16.83,"close":17.54,"volume":884523},
    {"date":"15/03/27","open":17.58,"high":18.3,"low":17.11,"close":18.3,"volume":705626},
    {"date":"15/03/30","open":18.5,"high":19.4,"low":18.4,"close":19.05,"volume":1151620},
    {"date":"15/03/31","open":19.08,"high":20.58,"low":18.4,"close":19.75,"volume":2020679},
    {"date":"15/04/01","open":19.69,"high":19.69,"low":18.55,"close":18.65,"volume":961078},
    {"date":"15/04/02","open":18.56,"high":18.66,"low":17.85,"close":17.9,"volume":884233},
    {"date":"15/04/06","open":17.78,"high":17.94,"low":17.51,"close":17.66,"volume":605252},
    {"date":"15/04/07","open":17.62,"high":17.9,"low":17.53,"close":17.61,"volume":591988},
    {"date":"15/04/08","open":17.64,"high":17.85,"low":17.32,"close":17.36,"volume":618855},
    {"date":"15/04/09","open":17.33,"high":17.54,"low":17.1,"close":17.1,"volume":761855},
    {"date":"15/04/10","open":17.08,"high":17.36,"low":17,"close":17.05,"volume":568373},
    {"date":"15/04/13","open":17.24,"high":17.26,"low":16.81,"close":17.1,"volume":667142},
    {"date":"15/04/14","open":17.1,"high":17.89,"low":17.02,"close":17.52,"volume":870138},
    {"date":"15/04/15","open":17.6,"high":17.99,"low":17.5,"close":17.69,"volume":530456},
    {"date":"15/04/16","open":17.95,"high":18,"low":17.6,"close":17.82,"volume":548730},
    {"date":"15/04/17","open":17.75,"high":17.79,"low":17.5,"close":17.79,"volume":446373},
    {"date":"15/04/20","open":17.63,"high":17.98,"low":17.52,"close":17.93,"volume":487017},
    {"date":"15/04/21","open":17.96,"high":17.98,"low":17.71,"close":17.92,"volume":320302},
    {"date":"15/04/22","open":17.88,"high":18.33,"low":17.57,"close":18.29,"volume":644812},
    {"date":"15/04/23","open":18.29,"high":18.61,"low":18.18,"close":18.28,"volume":563879},
```

```
    {"date":"15/04/24","open":18.5,"high":18.5,"low":17.61,"close":17.75,"volume":650762},
    {"date":"15/04/27","open":17.97,"high":18.05,"low":17.45,"close":17.57,"volume":437294},
    {"date":"15/04/28","open":17.65,"high":17.79,"low":17.39,"close":17.5,"volume":224519},
    {"date":"15/04/29","open":17.68,"high":17.68,"low":17.1,"close":17.21,"volume":495706},
    {"date":"15/04/30","open":17.22,"high":17.3,"low":17,"close":17.11,"volume":391040},
    {"date":"15/05/01","open":17.11,"high":17.55,"low":16.85,"close":17.5,"volume":563075},
    {"date":"15/05/02","open":17.56,"high":17.85,"low":17.3,"close":17.4,"volume":253138},
    {"date":"15/05/05","open":17.68,"high":17.68,"low":17.09,"close":17.43,"volume":290935},
    {"date":"15/05/06","open":17.48,"high":17.48,"low":17,"close":17.04,"volume":313662},
    {"date":"15/05/07","open":17.05,"high":17.19,"low":16.92,"close":17.04,"volume":360284},
    {"date":"15/05/08","open":17.13,"high":17.21,"low":16.91,"close":17.1,"volume":297653},
    {"date":"15/05/11","open":17.16,"high":17.44,"low":17.13,"close":17.31,"volume":268504},
    {"date":"15/05/12","open":17.28,"high":17.44,"low":16.99,"close":17.24,"volume":376961},
    {"date":"15/05/13","open":17.24,"high":17.3,"low":17.06,"close":17.2,"volume":244617},
    {"date":"15/05/14","open":17.24,"high":17.25,"low":17.02,"close":17.08,"volume":252526},
    {"date":"15/05/15","open":17.06,"high":17.16,"low":16.95,"close":16.95,"volume":274783},
    {"date":"15/05/18","open":16.95,"high":17.01,"low":16.76,"close":16.87,"volume":418513},
    {"date":"15/05/19","open":16.93,"high":16.94,"low":16.6,"close":16.83,"volume":367660},
    {"date":"15/05/20","open":16.8,"high":16.9,"low":16.65,"close":16.86,"volume":297914},
    {"date":"15/05/21","open":16.9,"high":17.08,"low":16.79,"close":16.88,"volume":229346},
    {"date":"15/05/22","open":16.9,"high":17.05,"low":16.85,"close":17,"volume":253279},
    {"date":"15/05/26","open":17.03,"high":17.08,"low":16.86,"close":17.01,"volume":212640},
    {"date":"15/05/27","open":17.01,"high":17.99,"low":16.87,"close":17.75,"volume":857109},
    {"date":"15/05/28","open":17.77,"high":17.77,"low":17.44,"close":17.62,"volume":338482}
]
```


コードビューで次の参照を追加します。

- System.Collections.Generic
- System.Windows
- System.Runtime.Serialization

- **Visual Basic**

```
<DataContract>
Public Class Quote
    <DataMember(Name:="date")>
    Public Property [Date]() As String
        Get
            Return m_Date
        End Get
        Set
            m_Date = Value
        End Set
    End Property
    Private m_Date As String

    <DataMember(Name:="high")>
    Public Property High() As Double
        Get
            Return m_High
        End Get
        Set
            m_High = Value
        End Set
    End Property
    Private m_High As Double

    <DataMember(Name:="low")>
    Public Property Low() As Double
        Get
            Return m_Low
        End Get
        Set
            m_Low = Value
        End Set
    End Property
```

```
    Private m_Low As Double

    <DataMember(Name:="open")>
    Public Property Open() As Double
        Get
            Return m_Open
        End Get
        Set
            m_Open = Value
        End Set
    End Property
    Private m_Open As Double

    <DataMember(Name:="close")>
    Public Property Close() As Double
        Get
            Return m_Close
        End Get
        Set
            m_Close = Value
        End Set
    End Property
    Private m_Close As Double

    <DataMember(Name:="volume")>
    Public Property Volume() As Double
        Get
            Return m_Volume
        End Get
        Set
            m_Volume = Value
        End Set
    End Property
    Private m_Volume As Double
End Class
''' Arcs.xamlの相互作用ロジック
Partial Public Class Arcs

    Inherits Window
    Private dataService As DataService = dataService.GetService()
    Public Sub New()
        InitializeComponent()
    End Sub
    Public ReadOnly Property Data() As List(Of Quote)
        Get
            Return dataService.GetData()
        End Get
    End Property
End Class
```

- **C#**

```
[DataContract]
public class Quote
{
    [DataMember(Name = "date")]
    public string Date { get; set; }

    [DataMember(Name = "high")]
    public double High { get; set; }

    [DataMember(Name = "low")]
    public double Low { get; set; }

    [DataMember(Name = "open")]
    public double Open { get; set; }

    [DataMember(Name = "close")]
```

```
    public double Close { get; set; }

    [DataMember(Name = "volume")]
    public double Volume { get; set; }
}
/// MainWindowArcs.xamlの相互作用ロジック
public partial class MainWindowArcs : Window
{
    DataService dataService = DataService.GetService();
    public MainWindowArcs()
    {
        InitializeComponent();
    }
    public List<Quote> Data
    {
        get
        {
            return dataService.GetData();
        }
    }
}
```

先頭に移動

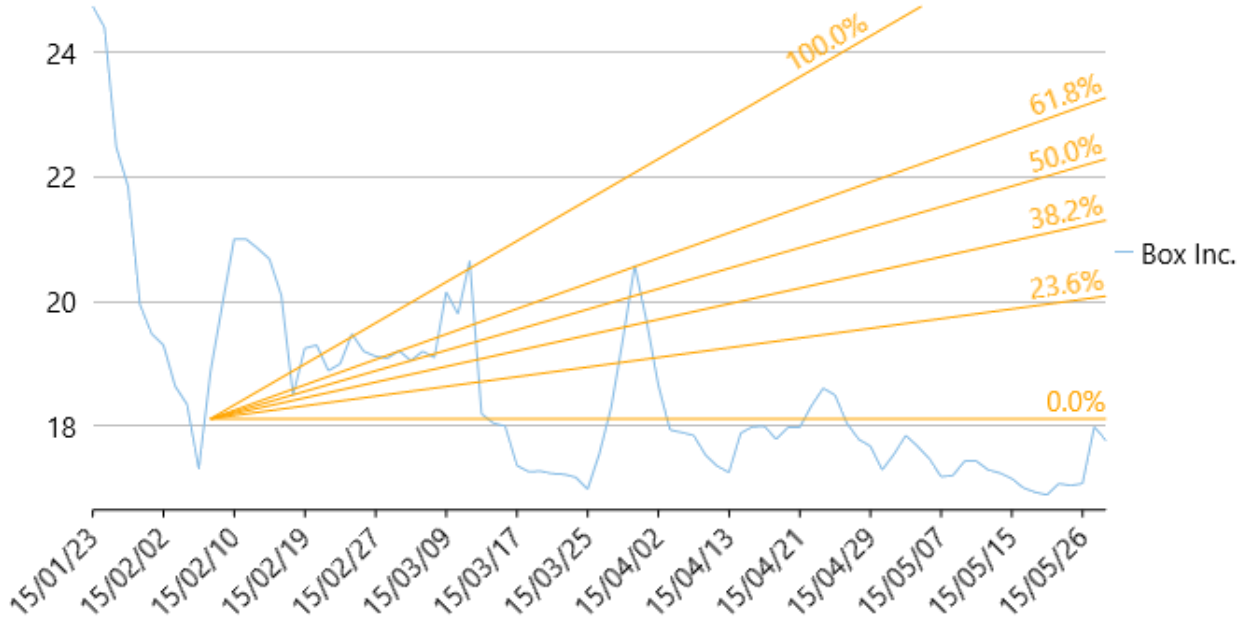
フィボナッチファン

フィボナッチファンは、フィボナッチリトレースメントポイントに基づく上昇トレンドおよび下降傾向線です。このような技術インジケータは、トレンドの移動速度の測定や、サポートおよびレジスタンス傾向線の構成にも役立ちます。上昇ファンラインはサポートレベルの予想に使用され、下降ファンラインはレジスタンスレベルの予想に役立ちます。上昇トレンドでは、株価が1 つのフィボナッチファン傾向線より下がると、次のフィボナッチファン傾向線までさらに下がることが予想されます。このような場合、フィボナッチファンラインはサポートとして機能します。一方、株価が 1 つのフィボナッチファン傾向線まで上がると、この傾向線がレジスタンスになると期待されます。

フィボナッチファンを作成するには、最初に指定された期間の高低 2 点間の傾向線を描画し、この 2 点間の垂直距離を主要なフィボナッチ比率 38.2%、50%、61.8% で分割します。これらの各分割点の結果として、この垂直距離内のポイントが取得されます。次に、左端のポイントから、これらのフィボナッチ比率を表す3 点にそれぞれラインを描画して、3 本の「ファン」ラインが作成されます。これらが、フィボナッチリトレースメントポイントに基づく傾向線になります。

FinancialChart でフィボナッチファンを使用するには、コントロールに適切なデータソースを連結するか、**Quote Collection** でコントロールにデータを追加します。FinancialChart にデータを連結または追加するには、ItemsSource オブジェクトを使用します。

Fibonacci クラスは、**Uptrend** プロパティを公開します。**FibonacciFans** クラスのオブジェクトを作成すると、フィボナッチファンがチャートで有効になります。さらに、**FibonacciFans** クラスは、**StartX**、**EndX**、**StartY**、**EndY** の各プロパティを公開します。これらのプロパティに基づいて、FinancialChart にフィボナッチファンラインがプロットされます。



次の例は、上の図で示すように、企業 Box Inc. の一定期間の株価を考察し、同じ株価チャートにファンラインをプロットします。この例では、json ファイルのデータを使用します。この json ファイルにアクセスするために、DataService.cs クラスを作成します。

⚠ json ファイルの[ビルドアクション]プロパティが**[埋め込まれたリソース]**に設定されていることを確認します。

XAML copyCode

```
<Window
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Fibonacci"
        xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
        x:Class="Fibonacci.MainWindowFans"
        mc:Ignorable="d"
        Title="MainWindowFans"
        DataContext="{Binding RelativeSource={RelativeSource Mode=Self}}"
        Height="300" Width="300">
    <Grid>

        <c1:C1FinancialChart x:Name="financialChart"
                             ItemsSource="{Binding Data}"
                             BindingX="Date"
                             ChartType="Line"
                             ToolTipContent="{}{seriesName}&#x000A;{Date} {y}">
            <c1:FinancialSeries Binding="High,Low,Open,Close"
                                ChartType="Line"
                                SeriesName="Box Inc."/>
          <c1:FibonacciFans x:Name="fans"
                            Binding="Close"
                            StartX="10"
                            StartY="18.12"
                            EndX="32"
                            EndY="20.53">
```

```
            <c1:FibonacciFans.Style>
              <c1:ChartStyle Stroke="Pink" />
            </c1:FibonacciFans.Style>
          </c1:FibonacciFans>

            <c1:C1FinancialChart.AxisX>
                <c1:Axis LabelAngle="45" MajorUnit="3"/>
            </c1:C1FinancialChart.AxisX>
        </c1:C1FinancialChart>
    </Grid>
</Window>
```

DataService.cs に次の参照を追加します。

- System.Collections.Generic
- System.Linq
- System.Runtime.Serialization.Json
- System.Reflection

- **DataService.vb**

```
Public Class DataService
    Public Function GetData() As List(Of Quote)
        Dim path As String = "FibonacciVB.Resources.box.json"
        'FibonacciVBをアプリケーション名で置き換えます。
        Dim stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path)
        Dim ser = New DataContractJsonSerializer(GetType(Quote()))
        Dim data = DirectCast(ser.ReadObject(stream), Quote())
        Return data.ToList()
    End Function
    Shared _ds As DataService
    Public Shared Function GetService() As DataService
        If _ds Is Nothing Then
            _ds = New DataService()
        End If
        Return _ds
    End Function
End Class
```

- **DataService.cs**

```
public class DataService
{
    public List<Quote> GetData()
    {
        string path = "Fibonacci.Resources.box.json";
        //Fibonacciをアプリケーション名で置き換えます。
        var stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path);
        var ser = new DataContractJsonSerializer(typeof(Quote[]));
        var data = (Quote[])ser.ReadObject(stream);
        return data.ToList();
    }
    static DataService _ds;
    public static DataService GetService()
    {
        if (_ds == null)
            _ds = new DataService();
        return _ds;
    }
}
```

Json Data

```
[
    {"date":"15/01/23","open":20.2,"high":24.73,"low":20.16,"close":23.23,"volume":42593223},
    {"date":"15/01/26","open":23.67,"high":24.39,"low":22.5,"close":22.6,"volume":8677164},
    {"date":"15/01/27","open":22,"high":22.47,"low":21.17,"close":21.3,"volume":3272512},
    {"date":"15/01/28","open":21.62,"high":21.84,"low":19.6,"close":19.78,"volume":5047364},
    {"date":"15/01/29","open":19.9,"high":19.95,"low":18.51,"close":18.8,"volume":3419482},
    {"date":"15/01/30","open":18.47,"high":19.48,"low":18.22,"close":18.81,"volume":2266439},
    {"date":"15/02/02","open":19.18,"high":19.3,"low":18.01,"close":18.02,"volume":2071168},
    {"date":"15/02/03","open":18.22,"high":18.64,"low":18.12,"close":18.24,"volume":1587435},
    {"date":"15/02/04","open":18.2,"high":18.35,"low":17,"close":17.1,"volume":2912224},
    {"date":"15/02/05","open":17.3,"high":17.31,"low":16.41,"close":16.66,"volume":2682187},
    {"date":"15/02/06","open":17.39,"high":18.88,"low":17.21,"close":18.12,"volume":3929164},
    {"date":"15/02/09","open":18.86,"high":19.95,"low":18.45,"close":19.6,"volume":3226650},
    {"date":"15/02/10","open":20.5,"high":21,"low":19.63,"close":20.99,"volume":2804409},
    {"date":"15/02/11","open":20.89,"high":21,"low":20.2,"close":20.96,"volume":1698365},
    {"date":"15/02/12","open":20.66,"high":20.85,"low":19.75,"close":20.17,"volume":1370320},
    {"date":"15/02/13","open":20.19,"high":20.68,"low":20,"close":20.18,"volume":711951},
    {"date":"15/02/17","open":19.5,"high":20.1,"low":18.8,"close":19.05,"volume":2093602},
    {"date":"15/02/18","open":18.31,"high":18.5,"low":17.96,"close":18,"volume":1849490},
    {"date":"15/02/19","open":18.33,"high":19.25,"low":17.91,"close":18.96,"volume":1311518},
    {"date":"15/02/20","open":18.68,"high":19.3,"low":18.65,"close":18.85,"volume":1001692},
    {"date":"15/02/23","open":18.8,"high":18.89,"low":18.11,"close":18.21,"volume":670087},
    {"date":"15/02/24","open":18.46,"high":19,"low":18.27,"close":18.83,"volume":759263},
    {"date":"15/02/25","open":18.83,"high":19.48,"low":18.47,"close":18.67,"volume":915580},
    {"date":"15/02/26","open":18.64,"high":19.2,"low":18.64,"close":18.94,"volume":461283},
    {"date":"15/02/27","open":18.8,"high":19.12,"low":18.55,"close":18.66,"volume":617199},
    {"date":"15/03/02","open":18.66,"high":19.09,"low":18.65,"close":18.79,"volume":519605},
    {"date":"15/03/03","open":18.79,"high":19.21,"low":18.45,"close":18.59,"volume":832415},
    {"date":"15/03/04","open":18.64,"high":19.05,"low":18.32,"close":19,"volume":539688},
    {"date":"15/03/05","open":19.2,"high":19.2,"low":18.8,"close":19.14,"volume":486149},
    {"date":"15/03/06","open":19.03,"high":19.1,"low":18.7,"close":18.91,"volume":685659},
    {"date":"15/03/09","open":18.98,"high":20.15,"low":18.96,"close":19.4,"volume":1321363},
    {"date":"15/03/10","open":19.3,"high":19.8,"low":18.85,"close":19.64,"volume":615743},
    {"date":"15/03/11","open":20.08,"high":20.65,"low":19.24,"close":20.53,"volume":2167167},
    {"date":"15/03/12","open":17.17,"high":18.2,"low":16.76,"close":18.2,"volume":6837638},
    {"date":"15/03/13","open":18.05,"high":18.05,"low":17.3,"close":17.88,"volume":1715629},
    {"date":"15/03/16","open":17.91,"high":18,"low":17.01,"close":17.13,"volume":1321313},
    {"date":"15/03/17","open":17.28,"high":17.37,"low":16.6,"close":17.12,"volume":1272242},
    {"date":"15/03/18","open":17.1,"high":17.27,"low":16.91,"close":17.01,"volume":530063},
    {"date":"15/03/19","open":17,"high":17.28,"low":17,"close":17.06,"volume":536427},
    {"date":"15/03/20","open":17.13,"high":17.24,"low":16.88,"close":17.21,"volume":1320237},
    {"date":"15/03/23","open":17.21,"high":17.23,"low":17.01,"close":17.11,"volume":509798},
    {"date":"15/03/24","open":17.02,"high":17.18,"low":16.82,"close":17,"volume":962149},
    {"date":"15/03/25","open":16.92,"high":16.99,"low":16.82,"close":16.97,"volume":565673},
    {"date":"15/03/26","open":16.83,"high":17.56,"low":16.83,"close":17.54,"volume":884523},
    {"date":"15/03/27","open":17.58,"high":18.3,"low":17.11,"close":18.3,"volume":705626},
    {"date":"15/03/30","open":18.5,"high":19.4,"low":18.4,"close":19.05,"volume":1151620},
    {"date":"15/03/31","open":19.08,"high":20.58,"low":18.4,"close":19.75,"volume":2020679},
    {"date":"15/04/01","open":19.69,"high":19.69,"low":18.55,"close":18.65,"volume":961078},
    {"date":"15/04/02","open":18.56,"high":18.66,"low":17.85,"close":17.9,"volume":884233},
    {"date":"15/04/06","open":17.78,"high":17.94,"low":17.51,"close":17.66,"volume":605252},
    {"date":"15/04/07","open":17.62,"high":17.9,"low":17.53,"close":17.61,"volume":591988},
    {"date":"15/04/08","open":17.64,"high":17.85,"low":17.32,"close":17.36,"volume":618855},
    {"date":"15/04/09","open":17.33,"high":17.54,"low":17.1,"close":17.1,"volume":761855},
    {"date":"15/04/10","open":17.08,"high":17.36,"low":17,"close":17.05,"volume":568373},
    {"date":"15/04/13","open":17.24,"high":17.26,"low":16.81,"close":17.1,"volume":667142},
    {"date":"15/04/14","open":17.1,"high":17.89,"low":17.02,"close":17.52,"volume":870138},
    {"date":"15/04/15","open":17.6,"high":17.99,"low":17.5,"close":17.69,"volume":530456},
    {"date":"15/04/16","open":17.95,"high":18,"low":17.6,"close":17.82,"volume":548730},
    {"date":"15/04/17","open":17.75,"high":17.79,"low":17.5,"close":17.79,"volume":446373},
    {"date":"15/04/20","open":17.63,"high":17.98,"low":17.52,"close":17.93,"volume":487017},
    {"date":"15/04/21","open":17.96,"high":17.98,"low":17.71,"close":17.92,"volume":320302},
    {"date":"15/04/22","open":17.88,"high":18.33,"low":17.57,"close":18.29,"volume":644812},
    {"date":"15/04/23","open":18.29,"high":18.61,"low":18.18,"close":18.28,"volume":563879},
    {"date":"15/04/24","open":18.5,"high":18.5,"low":17.61,"close":17.75,"volume":650762},
    {"date":"15/04/27","open":17.97,"high":18.05,"low":17.45,"close":17.57,"volume":437294},
    {"date":"15/04/28","open":17.65,"high":17.79,"low":17.39,"close":17.5,"volume":224519},
    {"date":"15/04/29","open":17.68,"high":17.68,"low":17.1,"close":17.21,"volume":495706},
```

```
    {"date":"15/04/30","open":17.22,"high":17.3,"low":17,"close":17.11,"volume":391040},
    {"date":"15/05/01","open":17.11,"high":17.55,"low":16.85,"close":17.5,"volume":563075},
    {"date":"15/05/02","open":17.56,"high":17.85,"low":17.3,"close":17.4,"volume":253138},
    {"date":"15/05/05","open":17.68,"high":17.68,"low":17.09,"close":17.43,"volume":290935},
    {"date":"15/05/06","open":17.48,"high":17.48,"low":17,"close":17.04,"volume":313662},
    {"date":"15/05/07","open":17.05,"high":17.19,"low":16.92,"close":17.04,"volume":360284},
    {"date":"15/05/08","open":17.13,"high":17.21,"low":16.91,"close":17.1,"volume":297653},
    {"date":"15/05/11","open":17.16,"high":17.44,"low":17.13,"close":17.31,"volume":268504},
    {"date":"15/05/12","open":17.28,"high":17.44,"low":16.99,"close":17.24,"volume":376961},
    {"date":"15/05/13","open":17.24,"high":17.3,"low":17.06,"close":17.2,"volume":244617},
    {"date":"15/05/14","open":17.24,"high":17.25,"low":17.02,"close":17.08,"volume":252526},
    {"date":"15/05/15","open":17.06,"high":17.16,"low":16.95,"close":16.95,"volume":274783},
    {"date":"15/05/18","open":16.95,"high":17.01,"low":16.76,"close":16.87,"volume":418513},
    {"date":"15/05/19","open":16.93,"high":16.94,"low":16.6,"close":16.83,"volume":367660},
    {"date":"15/05/20","open":16.8,"high":16.9,"low":16.65,"close":16.86,"volume":297914},
    {"date":"15/05/21","open":16.9,"high":17.08,"low":16.79,"close":16.88,"volume":229346},
    {"date":"15/05/22","open":16.9,"high":17.05,"low":16.85,"close":17,"volume":253279},
    {"date":"15/05/26","open":17.03,"high":17.08,"low":16.86,"close":17.01,"volume":212640},
    {"date":"15/05/27","open":17.01,"high":17.99,"low":16.87,"close":17.75,"volume":857109},
    {"date":"15/05/28","open":17.77,"high":17.77,"low":17.44,"close":17.62,"volume":338482}
]
```


コードビューで次の参照を追加します。

- System.Collections.Generic
- System.Windows
- System.Runtime.Serialization

- **Visual Basic**

```
<DataContract>
Public Class Quote
    <DataMember(Name:="date")>
    Public Property [Date]() As String
        Get
            Return m_Date
        End Get
        Set
            m_Date = Value
        End Set
    End Property
    Private m_Date As String

    <DataMember(Name:="high")>
    Public Property High() As Double
        Get
            Return m_High
        End Get
        Set
            m_High = Value
        End Set
    End Property
    Private m_High As Double

    <DataMember(Name:="low")>
    Public Property Low() As Double
        Get
            Return m_Low
        End Get
        Set
            m_Low = Value
        End Set
    End Property
    Private m_Low As Double

    <DataMember(Name:="open")>
    Public Property Open() As Double
```

```
        Get
            Return m_Open
        End Get
        Set
            m_Open = Value
        End Set
    End Property
    Private m_Open As Double

    <DataMember(Name:="close")>
    Public Property Close() As Double
        Get
            Return m_Close
        End Get
        Set
            m_Close = Value
        End Set
    End Property
    Private m_Close As Double

    <DataMember(Name:="volume")>
    Public Property Volume() As Double
        Get
            Return m_Volume
        End Get
        Set
            m_Volume = Value
        End Set
    End Property
    Private m_Volume As Double
End Class
''' Fans.xamlの相互作用ロジック
Partial Public Class Fans
    Inherits Window
    Private dataService As DataService = dataService.GetService()
    Public Sub New()
        InitializeComponent()
    End Sub
    Public ReadOnly Property Data() As List(Of Quote)
        Get
            Return dataService.GetData()
        End Get
    End Property
End Class
```

- **C#**

```
[DataContract]
public class Quote
{
    [DataMember(Name = "date")]
    public string Date { get; set; }

    [DataMember(Name = "high")]
    public double High { get; set; }

    [DataMember(Name = "low")]
    public double Low { get; set; }

    [DataMember(Name = "open")]
    public double Open { get; set; }

    [DataMember(Name = "close")]
    public double Close { get; set; }

    [DataMember(Name = "volume")]
    public double Volume { get; set; }
}
```

```
/// MainWindowFans.xamlの相互作用ロジック
public partial class MainWindowFans : Window
{
    DataService dataService = DataService.GetService();
    public MainWindowFans()
    {
        InitializeComponent();
    }
    public List<Quote> Data
    {
        get
        {
            return dataService.GetData();
        }
    }
}
```

先頭に移動

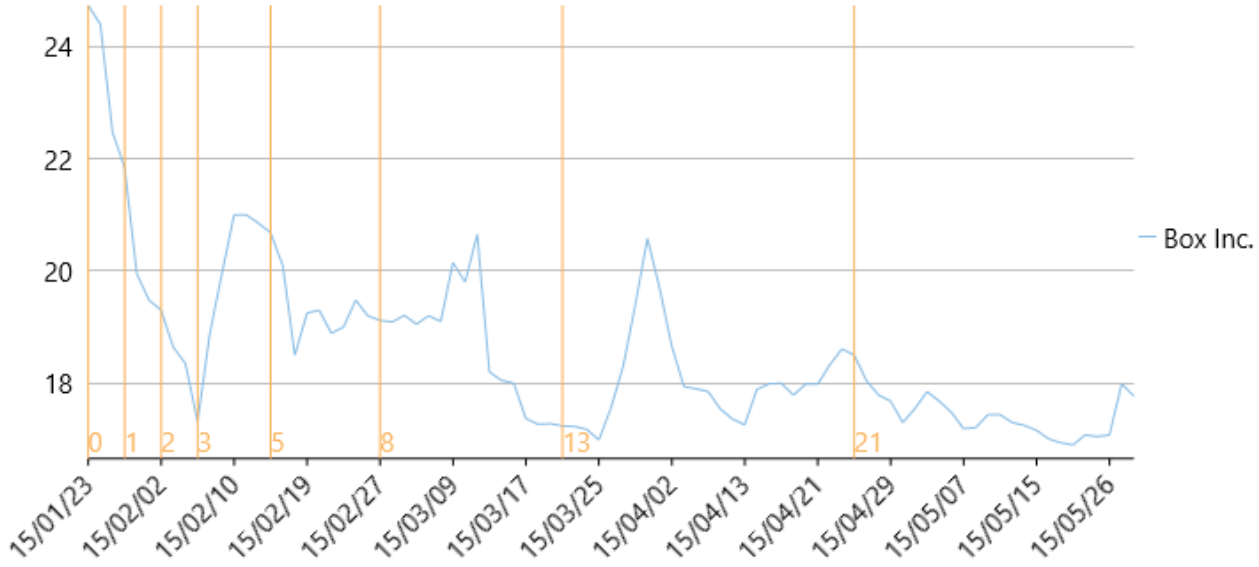
フィボナッチタイムゾーン

フィボナッチタイムゾーンは、トレーダーが金融商品の価格が重要な動きを示す時期を予想するために使用されるテクニカルインジケータです。タイムゾーンは、フィボナッチ数と呼ばれる数列（1、2、3、5、8、13、21、34...）に対応する一連の垂直ラインです。

株価チャートの（大きな価格変動の後の）タイムゾーンの開始位置はトレーダーが決定し、以降のフィボナッチ数列の位置に対応する日にそれぞれ垂直ラインが引かれます。フィボナッチタイムゾーンを使用して、テクニカルトレーダーは、これらの垂直タイムゾーンライン付近に将来の株価変化を予想し、売買の決定を行います。理想的には、21、34、55、89、144日後に価格のリバーサルポイントの可能性があるため、最初の 7 ～ 8 個のタイムゾーンは無視するように推奨されます。これらのポイントは、それぞれ 8、9、10、11、12 番目のタイムゾーンに対応します。

FinancialChart で、フィボナッチタイムゾーンを使用するには、コントロールをアプリケーションに追加し、コントロールに適切なデータソースを連結するか、**Quote Collection** でコントロールにデータを追加します。FinancialChart にデータを連結または追加するには、ItemsSource オブジェクトを使用します。

Fibonacci クラスは、**Uptrend** プロパティを公開します。**FibonacciTimeZones** クラスのオブジェクトを作成すると、フィボナッチタイムゾーンがチャートで有効になります。さらに、**FibonacciExtension** クラスは、**StartX**、**EndX** の各プロパティを公開します。これらのプロパティに基づいて、FinancialChart にタイムゾーンがプロットされます。



次の例は、上の図で示すように、企業 Box Inc. の一定期間の株価を考察し、同じ株価チャートにタイムゾーンをプロットします。この例では、json ファイルのデータを使用します。この json ファイルにアクセスするために、DataService.cs クラスを作成します。

⚠ json ファイルの[ビルドアクション]プロパティが**[埋め込まれたリソース]**に設定されていることを確認します。

XAML copyCode

```
<Window
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Fibonacci"
        xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
        x:Class="Fibonacci.MainWindowTZ"
        mc:Ignorable="d"
        Title="MainWindowTZ"
        DataContext="{Binding RelativeSource={RelativeSource Mode=Self}}"
        Height="300" Width="300">
    <Grid>

        <c1:C1FinancialChart x:Name="financialChart"
                             ItemsSource="{Binding Data}"
                             BindingX="Date"
                             ChartType="Candlestick"
                             ToolTipContent="{}{seriesName}&#x000A;{Date} {y}">
            <c1:FinancialSeries Binding="High,Low,Open,Close"
                                ChartType="Line"
                                SeriesName="Box Inc."/>
            <c1:FibonacciTimeZones x:Name="timeZones" Binding="Close" StartX="0" EndX="3">
                <c1:FibonacciTimeZones.Style>
                    <c1:ChartStyle Stroke="Brown" />
                </c1:FibonacciTimeZones.Style>
            </c1:FibonacciTimeZones>
            <c1:C1FinancialChart.AxisX>
                <c1:Axis LabelAngle="45" MajorUnit="3"/>
            </c1:C1FinancialChart.AxisX>
        </c1:C1FinancialChart>
    </Grid>
</Window>
```

DataService.cs に次の参照を追加します。

- System.Collections.Generic
- System.Linq
- System.Runtime.Serialization.Json
- System.Reflection

- **DataService.vb**

```
Public Class DataService
    Public Function GetData() As List(Of Quote)
        Dim path As String = "FibonacciVB.Resources.box.json"
        'FibonacciVBをアプリケーション名で置き換えます。
        Dim stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path)
        Dim ser = New DataContractJsonSerializer(GetType(Quote()))
        Dim data = DirectCast(ser.ReadObject(stream), Quote())
        Return data.ToList()
    End Function
    Shared _ds As DataService
    Public Shared Function GetService() As DataService
        If _ds Is Nothing Then
```

```
            _ds = New DataService()
        End If
        Return _ds
    End Function
End Class
```

- **DataService.cs**

```
public class DataService
{
    public List<Quote> GetData()
    {
        string path = "Fibonacci.Resources.box.json";
        //Fibonacciをアプリケーション名で置き換えます。
        var stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path);
        var ser = new DataContractJsonSerializer(typeof(Quote[]));
        var data = (Quote[])ser.ReadObject(stream);
        return data.ToList();
    }
    static DataService _ds;
    public static DataService GetService()
    {
        if (_ds == null)
            _ds = new DataService();
        return _ds;
    }
}
```

Json Data

```
[
    {"date":"15/01/23","open":20.2,"high":24.73,"low":20.16,"close":23.23,"volume":42593223},
    {"date":"15/01/26","open":23.67,"high":24.39,"low":22.5,"close":22.6,"volume":8677164},
    {"date":"15/01/27","open":22,"high":22.47,"low":21.17,"close":21.3,"volume":3272512},
    {"date":"15/01/28","open":21.62,"high":21.84,"low":19.6,"close":19.78,"volume":5047364},
    {"date":"15/01/29","open":19.9,"high":19.95,"low":18.51,"close":18.8,"volume":3419482},
    {"date":"15/01/30","open":18.47,"high":19.48,"low":18.22,"close":18.81,"volume":2266439},
    {"date":"15/02/02","open":19.18,"high":19.3,"low":18.01,"close":18.02,"volume":2071168},
    {"date":"15/02/03","open":18.22,"high":18.64,"low":18.12,"close":18.24,"volume":1587435},
    {"date":"15/02/04","open":18.2,"high":18.35,"low":17,"close":17.1,"volume":2912224},
    {"date":"15/02/05","open":17.3,"high":17.31,"low":16.41,"close":16.66,"volume":2682187},
    {"date":"15/02/06","open":17.39,"high":18.88,"low":17.21,"close":18.12,"volume":3929164},
    {"date":"15/02/09","open":18.86,"high":19.95,"low":18.45,"close":19.6,"volume":3226650},
    {"date":"15/02/10","open":20.5,"high":21,"low":19.63,"close":20.99,"volume":2804409},
    {"date":"15/02/11","open":20.89,"high":21,"low":20.2,"close":20.96,"volume":1698365},
    {"date":"15/02/12","open":20.66,"high":20.85,"low":19.75,"close":20.17,"volume":1370320},
    {"date":"15/02/13","open":20.19,"high":20.68,"low":20,"close":20.18,"volume":711951},
    {"date":"15/02/17","open":19.5,"high":20.1,"low":18.8,"close":19.05,"volume":2093602},
    {"date":"15/02/18","open":18.31,"high":18.5,"low":17.96,"close":18,"volume":1849490},
    {"date":"15/02/19","open":18.33,"high":19.25,"low":17.91,"close":18.96,"volume":1311518},
    {"date":"15/02/20","open":18.68,"high":19.3,"low":18.65,"close":18.85,"volume":1001692},
    {"date":"15/02/23","open":18.8,"high":18.89,"low":18.11,"close":18.21,"volume":670087},
    {"date":"15/02/24","open":18.46,"high":19,"low":18.27,"close":18.83,"volume":759263},
    {"date":"15/02/25","open":18.83,"high":19.48,"low":18.47,"close":18.67,"volume":915580},
    {"date":"15/02/26","open":18.64,"high":19.2,"low":18.64,"close":18.94,"volume":461283},
    {"date":"15/02/27","open":18.8,"high":19.12,"low":18.55,"close":18.66,"volume":617199},
    {"date":"15/03/02","open":18.66,"high":19.09,"low":18.65,"close":18.79,"volume":519605},
    {"date":"15/03/03","open":18.79,"high":19.21,"low":18.45,"close":18.59,"volume":832415},
    {"date":"15/03/04","open":18.64,"high":19.05,"low":18.32,"close":19,"volume":539688},
    {"date":"15/03/05","open":19.2,"high":19.2,"low":18.8,"close":19.14,"volume":486149},
    {"date":"15/03/06","open":19.03,"high":19.1,"low":18.7,"close":18.91,"volume":685659},
    {"date":"15/03/09","open":18.98,"high":20.15,"low":18.96,"close":19.4,"volume":1321363},
    {"date":"15/03/10","open":19.3,"high":19.8,"low":18.85,"close":19.64,"volume":615743},
    {"date":"15/03/11","open":20.08,"high":20.65,"low":19.24,"close":20.53,"volume":2167167},
    {"date":"15/03/12","open":17.17,"high":18.2,"low":16.76,"close":18.2,"volume":6837638},
    {"date":"15/03/13","open":18.05,"high":18.05,"low":17.3,"close":17.88,"volume":1715629},
    {"date":"15/03/16","open":17.91,"high":18,"low":17.01,"close":17.13,"volume":1321313},
    {"date":"15/03/17","open":17.28,"high":17.37,"low":16.6,"close":17.12,"volume":1272242},
```

```
    {"date":"15/03/18","open":17.1,"high":17.27,"low":16.91,"close":17.01,"volume":530063},
    {"date":"15/03/19","open":17,"high":17.28,"low":17,"close":17.06,"volume":536427},
    {"date":"15/03/20","open":17.13,"high":17.24,"low":16.88,"close":17.21,"volume":1320237},
    {"date":"15/03/23","open":17.21,"high":17.23,"low":17.01,"close":17.11,"volume":509798},
    {"date":"15/03/24","open":17.02,"high":17.18,"low":16.82,"close":17,"volume":962149},
    {"date":"15/03/25","open":16.92,"high":16.99,"low":16.82,"close":16.97,"volume":565673},
    {"date":"15/03/26","open":16.83,"high":17.56,"low":16.83,"close":17.54,"volume":884523},
    {"date":"15/03/27","open":17.58,"high":18.3,"low":17.11,"close":18.3,"volume":705626},
    {"date":"15/03/30","open":18.5,"high":19.4,"low":18.4,"close":19.05,"volume":1151620},
    {"date":"15/03/31","open":19.08,"high":20.58,"low":18.4,"close":19.75,"volume":2020679},
    {"date":"15/04/01","open":19.69,"high":19.69,"low":18.55,"close":18.65,"volume":961078},
    {"date":"15/04/02","open":18.56,"high":18.66,"low":17.85,"close":17.9,"volume":884233},
    {"date":"15/04/06","open":17.78,"high":17.94,"low":17.51,"close":17.66,"volume":605252},
    {"date":"15/04/07","open":17.62,"high":17.9,"low":17.53,"close":17.61,"volume":591988},
    {"date":"15/04/08","open":17.64,"high":17.85,"low":17.32,"close":17.36,"volume":618855},
    {"date":"15/04/09","open":17.33,"high":17.54,"low":17.1,"close":17.1,"volume":761855},
    {"date":"15/04/10","open":17.08,"high":17.36,"low":17,"close":17.05,"volume":568373},
    {"date":"15/04/13","open":17.24,"high":17.26,"low":16.81,"close":17.1,"volume":667142},
    {"date":"15/04/14","open":17.1,"high":17.89,"low":17.02,"close":17.52,"volume":870138},
    {"date":"15/04/15","open":17.6,"high":17.99,"low":17.5,"close":17.69,"volume":530456},
    {"date":"15/04/16","open":17.95,"high":18,"low":17.6,"close":17.82,"volume":548730},
    {"date":"15/04/17","open":17.75,"high":17.79,"low":17.5,"close":17.79,"volume":446373},
    {"date":"15/04/20","open":17.63,"high":17.98,"low":17.52,"close":17.93,"volume":487017},
    {"date":"15/04/21","open":17.96,"high":17.98,"low":17.71,"close":17.92,"volume":320302},
    {"date":"15/04/22","open":17.88,"high":18.33,"low":17.57,"close":18.29,"volume":644812},
    {"date":"15/04/23","open":18.29,"high":18.61,"low":18.18,"close":18.28,"volume":563879},
    {"date":"15/04/24","open":18.5,"high":18.5,"low":17.61,"close":17.75,"volume":650762},
    {"date":"15/04/27","open":17.97,"high":18.05,"low":17.45,"close":17.57,"volume":437294},
    {"date":"15/04/28","open":17.65,"high":17.79,"low":17.39,"close":17.5,"volume":224519},
    {"date":"15/04/29","open":17.68,"high":17.68,"low":17.1,"close":17.21,"volume":495706},
    {"date":"15/04/30","open":17.22,"high":17.3,"low":17,"close":17.11,"volume":391040},
    {"date":"15/05/01","open":17.11,"high":17.55,"low":16.85,"close":17.5,"volume":563075},
    {"date":"15/05/02","open":17.56,"high":17.85,"low":17.3,"close":17.4,"volume":253138},
    {"date":"15/05/05","open":17.68,"high":17.68,"low":17.09,"close":17.43,"volume":290935},
    {"date":"15/05/06","open":17.48,"high":17.48,"low":17,"close":17.04,"volume":313662},
    {"date":"15/05/07","open":17.05,"high":17.19,"low":16.92,"close":17.04,"volume":360284},
    {"date":"15/05/08","open":17.13,"high":17.21,"low":16.91,"close":17.1,"volume":297653},
    {"date":"15/05/11","open":17.16,"high":17.44,"low":17.13,"close":17.31,"volume":268504},
    {"date":"15/05/12","open":17.28,"high":17.44,"low":16.99,"close":17.24,"volume":376961},
    {"date":"15/05/13","open":17.24,"high":17.3,"low":17.06,"close":17.2,"volume":244617},
    {"date":"15/05/14","open":17.24,"high":17.25,"low":17.02,"close":17.08,"volume":252526},
    {"date":"15/05/15","open":17.06,"high":17.16,"low":16.95,"close":16.95,"volume":274783},
    {"date":"15/05/18","open":16.95,"high":17.01,"low":16.76,"close":16.87,"volume":418513},
    {"date":"15/05/19","open":16.93,"high":16.94,"low":16.6,"close":16.83,"volume":367660},
    {"date":"15/05/20","open":16.8,"high":16.9,"low":16.65,"close":16.86,"volume":297914},
    {"date":"15/05/21","open":16.9,"high":17.08,"low":16.79,"close":16.88,"volume":229346},
    {"date":"15/05/22","open":16.9,"high":17.05,"low":16.85,"close":17,"volume":253279},
    {"date":"15/05/26","open":17.03,"high":17.08,"low":16.86,"close":17.01,"volume":212640},
    {"date":"15/05/27","open":17.01,"high":17.99,"low":16.87,"close":17.75,"volume":857109},
    {"date":"15/05/28","open":17.77,"high":17.77,"low":17.44,"close":17.62,"volume":338482}
]
```


コードビューで次の参照を追加します。

- System.Collections.Generic
- System.Windows
- System.Runtime.Serialization

- **Visual Basic**

```
<DataContract>
Public Class Quote
    <DataMember(Name:="date")>
    Public Property [Date]() As String
        Get
            Return m_Date
        End Get
```

```
        Set
            m_Date = Value
        End Set
    End Property
    Private m_Date As String

    <DataMember(Name:="high")>
    Public Property High() As Double
        Get
            Return m_High
        End Get
        Set
            m_High = Value
        End Set
    End Property
    Private m_High As Double

    <DataMember(Name:="low")>
    Public Property Low() As Double
        Get
            Return m_Low
        End Get
        Set
            m_Low = Value
        End Set
    End Property
    Private m_Low As Double

    <DataMember(Name:="open")>
    Public Property Open() As Double
        Get
            Return m_Open
        End Get
        Set
            m_Open = Value
        End Set
    End Property
    Private m_Open As Double

    <DataMember(Name:="close")>
    Public Property Close() As Double
        Get
            Return m_Close
        End Get
        Set
            m_Close = Value
        End Set
    End Property
    Private m_Close As Double

    <DataMember(Name:="volume")>
    Public Property Volume() As Double
        Get
            Return m_Volume
        End Get
        Set
            m_Volume = Value
        End Set
    End Property
    Private m_Volume As Double
End Class
''' TimeZones.xamlの相互作用ロジック
Partial Public Class TimeZones

    Inherits Window
    Private dataService As DataService = dataService.GetService()
    Public Sub New()
        InitializeComponent()
```

```
    End Sub
    Public ReadOnly Property Data() As List(Of Quote)
        Get
            Return dataService.GetData()
        End Get
    End Property
End Class
```

- **C#**

```
[DataContract]
public class Quote
{
    [DataMember(Name = "date")]
    public string Date { get; set; }

    [DataMember(Name = "high")]
    public double High { get; set; }

    [DataMember(Name = "low")]
    public double Low { get; set; }

    [DataMember(Name = "open")]
    public double Open { get; set; }

    [DataMember(Name = "close")]
    public double Close { get; set; }

    [DataMember(Name = "volume")]
    public double Volume { get; set; }
}
/// MainWindowTZ.xamlの相互作用ロジック
public partial class MainWindowTZ : Window
{
    DataService dataService = DataService.GetService();
    public MainWindowTZ()
    {
        InitializeComponent();
    }
    public List<Quote> Data
    {
        get
        {
            return dataService.GetData();
        }
    }
}
```

先頭に移動

ユーザー操作機能

FinancialChart for WPFには、アプリケーションのカスタマイズと高度な開発に役立つ対話式のツールが組み込まれています。範囲セレクタなどの機能を使用すると、エンドユーザーが実行時に FinancialChart のデータの表示範囲を調整できます。

範囲セレクタの詳細については、次のリンクをクリックしてください。

- 範囲セレクタ

範囲セレクタ

FinancialChart の RangeSelector を使用すると、ユーザーが特定の範囲のデータを選択してチャートに表示できます。さまざまなタイプの株価チャートに簡単に RangeSelector を連結できます。これは主に金融業界で、さまざまなデータ範囲に対して株価の分析を実行するために使用されています。

RangeSelector には左スクロールボックス（最小値用）と右スクロールボックス（最大値用）があり、チャートの特定の時間をスクロールできます。ユーザーは RangeSelector の最小値と最大値を変更し、これらのスクロールボックスを左側や右側にドラッグして、チャート内でのデータの表示範囲を調整できます。範囲バーでスクロールボックスを左にドラッグすると値が減少し、右にドラッグする範囲バーの値が増加します。

次のコードスニペットは、アプリケーションを作成するために、Range Selectorを使用する方法を具体的に示します。

- **DataService.cs**

```
public class DataService
{
    List<Company> _companies = new List<Company>();
    Dictionary<string, List<Quote>> _cache = new Dictionary<string, List<Quote>>();

    private DataService()
    {
        _companies.Add(new Company() { Symbol = "box", Name = "Box Inc" });
        _companies.Add(new Company() { Symbol = "fb", Name = "Facebook" });
    }

    public List<Company> GetCompanies()
    {
        return _companies;
    }

    public List<Quote> GetSymbolData(string symbol)
    {
        if (!_cache.Keys.Contains(symbol))
        {
            string path = string.Format("FinancialChartExplorer.Resources.{0}.json", symbol);
            var stream = Assembly.GetExecutingAssembly().GetManifestResourceStream(path);
            var ser = new DataContractJsonSerializer(typeof(Quote[]));
            var data = (Quote[])ser.ReadObject(stream);
            _cache.Add(symbol, data.ToList());
        }

        return _cache[symbol];
    }

    static DataService _ds;
    public static DataService GetService()
    {
        if (_ds == null)
            _ds = new DataService();
        return _ds;
    }
}
```

XAML

```
<c1:C1FinancialChart BindingX="date"
                     Binding="high,low,open,close,volume"
                     ChartType="Candlestick"
                     ItemsSource="{Binding Data}">
    <c1:FinancialSeries />
    <c1:C1FinancialChart.AxisX>
        <c1:Axis Min="{Binding Source={x:Reference Name=rangeSelector},
            Path=LowerValue}"
                 Max="{Binding Source={x:Reference Name=rangeSelector},
            Path=UpperValue}" />
    </c1:C1FinancialChart.AxisX>
</c1:C1FinancialChart>
```

Code

C# copyCode

```
public partial class RangeSelector : UserControl
{
    DataService dataService = DataService.GetService();

    public RangeSelector()
    {
        InitializeComponent();
    }

    public List<Quote> Data
    {
        get
        {
            return dataService.GetSymbolData("fb");
        }
    }
}
```