BINDABLE CANDLESTICK CHART COMPONENT

Victor Utomo

Program Studi Teknik Informatika STMIK PROVISI, Semarang
victor.utomo@gmail.com

Abstract
In the world of stock trading, charting is commonly believed to provide a better view to the stock prices rather than a table. While there are many chart types exist, candlestick is at the top of the heap. Candlestick chart considered gives a quick visual analysis of price action. But despite of its popularity, it is so hard to find a development tools that provide candlestick chart type.
On the other hand, there is Windows Presentation Foundation (WPF). Built based upon the newest Microsoft technology, the WPF differs from the older Windows Forms. By using an Extendable Application Markup Language (XAML) file as user interface definition, WPF use DirectX to render the definition to graphics on screen. Using DirectX also means an ability to create animation and 3D object, things can't be found on Window Forms. Not only improvement in graphics, WPF also introduces a new way to bind data to its interface. Though it's not complicated, it requires a little work to implement.
This article is going to use WPF to build a candlestick chart component. This component will able to display a candlestick chart on screen only by binding data to the component.

Keywords: candlestick chart, binding, WPF

1. Introduction

Chart reading has fascinated stock traders for hundreds of years, even though it has never been totally accepted by academics and other market purist. Determining the trend or direction of prices is one of the main objectives of chart analysis. Pattern recognition leading up to a chart setup is another purpose of chart work.

There are three main popular charting techniques: bar charts, point-and-figure charts, and candlestick charts. But candlestick charts is one of the most popular charting techniques.

Unfortunately, it is quite hard to find a candlestick chart type in a chart tool/component. When application developers need to show a candlestick chart in their application, they tend to create it as an exclusive control for the application.

Meanwhile, there is a new technology from Microsoft called WPF. This technology gives developer an easy way to build a more interesting and interactive UI, e.g. chart.

This article gives a brief description on how WPF could be used to create component that serve basic purpose of candlestick charting.

2. Candlestick Chart

2.1 Introduction to Candlestick Chart

Candlestick charting methods have been around for hundreds of years, but candlesticks have caught on over the past decade or so as a charting standard in the United States.

With advancements in technology and the growing availability of trading and investing resources available to traders, many options exist for the charting of securities. Why candlestick charting is at the top of the heap? Here are the top reasons:

- One of the best features of candlestick charting in general is the visual appeal and readability. People can glance at a candlestick chart and quickly gain an understanding of what's going on with the price of a security.
- Even after reading up on the most rudimentary of candlestick basics, people can easily spot the opening and closing price for a security on a candlestick chart. These price levels can be very important areas of support and resistance from day to day, and knowing where they are can be extremely helpful, especially for short-term traders.
- Candlestick charts also feature specific patterns that you can identify and use to decide when it's time to buy, sell, or wait on a trade or investment. These patterns can be a real boon to your work with securities, and you can combine them with other technical indicators for even more reliable results.

2.2 Candlestick Components

The following four pieces of information are combined to create a candlestick:

- Price on the open. The price at which a security opens on a given period is the first

piece of information used in creating a candlestick. Depending on whether the security's performance is bullish or bearish, the opening price corresponds to either the bottom edge of a candlestick's candle or the top edge.

- High price. The highest price that a security reaches during a given period corresponds to the top of a candlestick's wick. If a security opens at a certain price and then trades consistently lower than that price throughout the period, there won't be any wick at all above the candle.
- Low price. The lowest price that a security reaches during a period corresponds to the bottom of a candlestick's wick. If the price action for that period is extremely bullish and prices trade higher than the open, there won't be any wick below the candle.
- Price on the close. After a security finishes trading during a given period, its closing price is the last piece of information used to create a candle-stick. Depending on the security's performance during that period, the closing price can correspond to either the top edge of a candlestick's candle (if the period was bullish) or the bottom edge (if the period was bearish). (Rhoads, 2008)

Using the conventional way of displaying candles, a dark body indicates that the closing price was below the opening price, and a white or hollow body indicates that the closing price was higher than the opening price. Though nowadays, the color of the candles are a matter of preference. (Person, 2004)

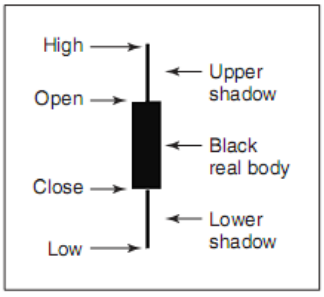


Figure 1. Candlestick Close Below Open (Person, 2004)

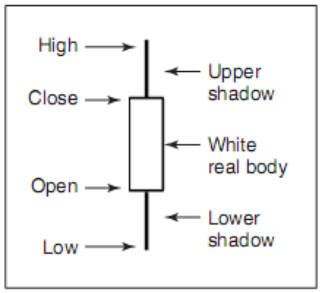


Figure 2. Candlestick Close Above Open (Person, 2004)

2.3 Chart Pattern

The components of a candlestick may be the bones of candlestick charting, but candlestick patterns are the heart and soul. Patterns appear on candlestick chart as simple, single-stick occurrences or complex, multi-stick formations, and many different types of patterns can tell you what may be in store for a security that you've had your eye on for trading or investing. And knowing what may lie ahead can be the difference between a profitable trade and a flop.

Though the pattern reading beyond the scope of this article, it is clear that multi candlestick formation is important to create some of the pattern.

3. Windows Presentation Foundation (WPF)

3.1 Motivation Behind WPF

WPF (introduced with .NET 3.0) was purposely created to merge previously unrelated programming tasks into a single unified object model. Thus, if you need to author a 3D animation, you have no need to manually program against DirectX API (although you could), as 3D functionality is baked directly into WPF. These are the functionality that WPF support

- Building forms with controls
- 2D graphics support
- 3D graphics support
- Support for streaming video
- Support for flow-style documents

3.2 Shapes

Shapes are the core elements of WPF. With shapes you can draw 2-dimensional graphics using rectangles, lines, ellipses, paths, polygons, and polylines that are represented by classes derived from the abstract base class *Shape*. Shapes are defined in the namespace System.Windows.Shapes namespace.

3.3 Brushes

There are various brushes that WPF offers for drawing backgrounds and foregrounds.

- **SolidColorBrush**. This brush uses a solid color. The complete area is drawn with the same color.
- **LinearGradientBrush**. For a smooth color change, you can use this brush. This brush defines the StartPoint and EndPoint properties. With this, you can assign 2-dimensional coordinates for the linear gradient.

- **RadialGradientBrush**. With this brush you can smooth the color in a radiant way. This brush defines the color start with the GradientOrigin point.

3.4 Content Controls

A ContentControl has a content property, with which you can add any content to the control. These are some of content controls

- **Label**. The class represents the text label for a control. This class also has support for access keys, for example, a menu command.
- **UserControl**. Using this class as a base class provides a simple way to create custom controls.
- **Window**. This class allows you to create windows and dialog boxes. With this class, you get a frame with minimize/maximize/close buttons and a system menu.

3.5 Layout

To define the layout of the application, you can use a class that derives from the *Panel* base class. Several layout containers are available that are discussed here. A layout container needs to do two main tasks, measure and arrange. With measuring, the container asks its children for the preferred size. Because the complete size answered by the controls might not be available, the container next decides and arranges the size and positions of its children.

Here are several those layout containers

- **StackPanel**. It is a simple container control that just shows one element after the other. The orientation of the StackPanel can be horizontal or vertical.
- **WrapPanel**. It positions the children from left to right, one after the other, as long as they fit into the line, and then continues with the next line. The orientation of the panel can be horizontal or vertical.
- **Canvas**. This class is a panel that allows you to explicitly position controls. This defines the attached properties Left, Right, Top, and Bottom.
- **DockPanel**. This class is very similar to the Windows Forms docking functionality. Here, you can specify the area where child controls should be arranged.
- **Grid**. Using this class, you can arrange your controls with rows and columns. The grid also allows star sizing, whereby the space for the rows and columns is calculated according to the available space and relative to other rows and columns. (Nagel, 2010)

3.6 Understanding the Role of Dependency Properties

WPF supports a unique programming concept termed a dependency property. Dependency property is very similar to “normal” property, but different on how a dependency property is implemented within the class. However, from a high level, all dependency properties are created in the following manner:

- First, the class that defined a dependency property must have DependenyObjects in its inheritance chain.
- A single dependency property is represented as a public, static, read-only field in the class of type DependencyProperty. By convention, this field is named by suffixing the word Property to the name of the CLR wrapper.
- Finally, the class will define a XAML-friendly CLR property, which makes calls to method provided by DependencyObject to get and set the value.

In a nutshell, the motivation of dependency properties is to provide a way to compute the value of a property based on the value of other inputs. Here is a list of some of these key benefits, which go well beyond those of the simple data encapsulation found with a CLR property:

- Dependency properties can inherit their values from a parent element’s XAML definition. For example, if you defined a value for the FontSize attribute in the opening tag of a <Window>, all controls in that Window would have the same font size by default.
- Dependency properties support the ability to have values set by elements contained within their XAML scope, such as a Button setting the Dock property of a DockPanel parent.
- Dependency properties allow WPF to compute a value based on multiple external values, which can be very important for animation and data binding services.
- Dependency properties provide infrastructure support for WPF triggers, which is used quite often when working with animation and data binding. (Troelsen, 2010)

4. The Component

The component is a WPF User Control Library assembly. It contains 4 classes and those are

- DailyStockClass
- ChartHelper
- CandleStickNode
- CandleStickChart

4.1 DailyStockClass

DailyStockClass provides basic properties, such as

- `HighestValue`
- `LowestValue`
- `OpenValue`
- `CloseValue`
- `MarketTime`

DailyStockClass also contains a static method that acts as converter from any type of data source to generic list of DailyStockClass. The generic list is preferred since it will provide a strong type.

4.2 ChartHelper

ChartHelper only provides a single property that gives margin to CandleStickChartNode objects. The margin used to arrange the node objects in the chart object.

4.3 CandleStickNode

This class is a WPF User Control that will draw a single candlestick chart. This is the mark up inside the XAML file

```
<StackPanel>
   <Rectangle Fill="Red"
              Height="30"
              Width="2"
              Name="CandleNeck"/>
   <Rectangle Stroke="Red"
              Fill="Red"
              Height="60"
              Width="30"
              Name="CandleBody"/>
   <Rectangle Fill="Red"
              Height="30"
              Width="2"
              Name="CandleLeg"/>
</StackPanel>
```

Code 1. CandleStickNode in Markup Language

Figure 3. CandleStickNode in Design View

While CandleBody will act as the body, the CandleNeck and CandleLeg will act as the upper and lower shadow. Stackpanel is chosen as layout since it will arrange the CandleNeck, CandleBody and CandleLeg in a flow manner.

DailyStockClass also has four dependency properties. Those dependency properties are

- HighestValue
- OpenValue
- CloseValue
- LowestValue

These are the codes on how to create the dependency property using HighestValue dependency property as the example

```
public static readonly DependencyProperty
    HighestValueProperty =
    DependencyProperty.Register(
    "HighestValue",
    typeof(double),
    typeof(CandleStickNode),
    new FrameworkPropertyMetadata((double)4,
        FrameworkPropertyMetadataOptions.
        AffectsRender,
        new PropertyChangedCallback(
            OnHighestValueChanged))
    );

public double HighestValue
{
    get { return (double)
        GetValue(HighestValueProperty); }
    set { SetValue(HighestValueProperty,
        value); }
}

public static void OnHighestValueChanged(
    DependencyObject sender,
    DependencyPropertyChangedEventArgs e)
{
...
}
```

Code 2. HighValue Dependency Property

These dependency properties will hold an important role as any value change on any of these properties will trigger a method that will change the candlestick chart appearance.

The method itself will adjust the chart appearance to meet the candlestick behavior by doing these

- Determine body, neck and leg's length.
- Determine what color the candlestick is since color is important in candlestick. This class gives green color for a bullish market and red color for a bearish market.

4.4 CandleStickChart

Since a single candlestick is not enough, a container for a group of candlestick is needed. CandleStickChart is the container class. Basically this class acts as the data source interface by having a dependency property of DataSource. The DataSource dependency property must use Object as its type, since DataSource should be able to receive object with many types. This property will trigger a method that will adjust the appearance.

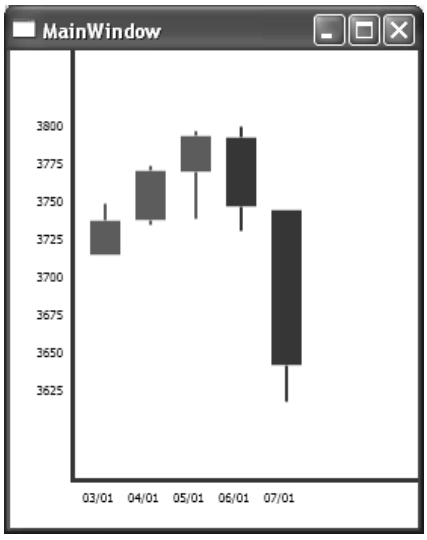
While DataSource use Object as its type, the internal implementation will use generic list of CandleStickNode as its type. This is where the conversion method in DailyStockClass comes in. In case of conversion then the chart will show nothing. But if the conversion is succeed it will do the following

- Add CandleStickNode object to the chart and binds the corresponding data.
- Arrange CandleStickNode objects with the help of the ChartHelper class.
- Arrange the label of the chart.

4.5 Implementation

The implementation of the component is quite straightforward. After adding the assembly as reference, drag the CandleStickChart object to WPF windows and bind the data to it.

Here is the result of binding data from Indonesia stock index (IHSG) from January 3rd 2011 until January 7th 2011



5. Conclusion

WPF gives the developer ability to create a more interesting and interactive UI easily. By using XAML standard, WPF is also easier to maintain. WPF also introduce the new concept of dependency property that will serve functionality as interface for data binding.

The candlestick chart is one example on how WPF could easily create an interface which would be a rather difficult to create using the older Windows Forms technology.

Like the rest of the objects in .NET, WPF also supports component that increase its portability and scalability. For instance, it is possible to use only the CandleStickNode in this article and implementing your own full functionality charting class.

Reference:

Person , John L., 2004, *A Complete Guide to Technical Trading Tactics*, Hoboken, John Wiley & Sons, Inc.

Rhoads, Russell, 2008,*Candlestick Charting For Dummies*, Indianapolis, Wiley Publishing, Inc.

Troelsen, Andrew, 2010*, Pro C# 2010 and the .NET 4 Platform*, New York, Apress.

Nagel, Christian et al, 2010*, Professional C# 4 and .NET 4*, Indianapolis, Wiley Publishing, Inc.